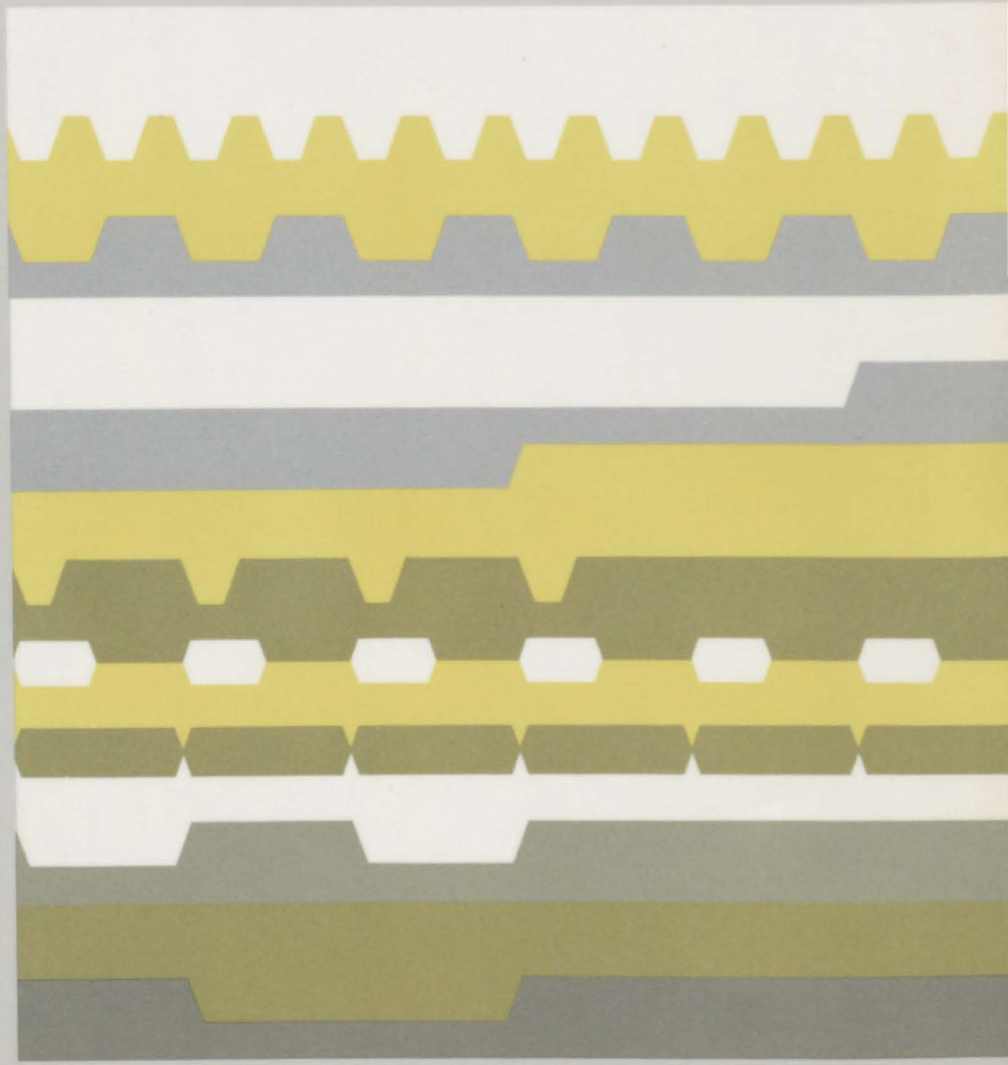
April, 1986

HITACHI

HD63484 ACRTC
ADVANCED CRT CONTROLLER
APPLICATION NOTE



#U90

HD63484 ACRTC ADVANCED CRT CONTROLLER

APPLICATION NOTE

Introduction to ACRTC Application. . . . . .Volume 1
Page 1

Hardware . . . . . .Volume 2
Page 123

Software . . . . . .Volume 3
Page 193

#U90

HITACHI

680-3-08
680-3-07

When using this manual, the reader should keep the following in mind:

1. This manual may, wholly or partially, be subject to change without notice.
2. All rights reserved: No one is permitted to reproduce or duplicate, in any form, the whole or part of this manual without Hitachi's permission.
3. Hitachi will not be responsible for any damage to the user that may result from accidents or any other reasons during operation of his unit according to this manual.
4. This manual neither ensures the enforcement of any industrial properties or other rights, nor sanctions the enforcement right thereof.
5. Circuitry and other examples described herein are meant merely to indicate characteristics and performance of Hitachi semiconductor-applied products. Hitachi assumes no responsibility for any patent infringements or other problems resulting from applications based on the examples described herein.

April 1986

Printed in U.S.A.

APPLICATION NOTE
Volume 1
Introduction to
ACRTC Application

Table of Contents

1. INTRODUCTION TO GRAPHICS SYSTEMS 1
 1.1 Bit Mapped and Vector Displays (Monochrome) 1
 1.2 Raster Scan 2
 1.3 Raster Display Screen 3
 1.4 Color System 4
 1.5 Displaying and Drawing 6
 1.6 Frame Buffer to Video Signals 10
2. DISPLAY INTERFACE 13
 2.1 The ACRTC Frame Buffer Considerations 13
 2.2 Graphic Address Increment Mode GAI 16
 2.3 Resolution of the Display 18
 2.4 Bits and Pixels 20
 2.5 Logical/Physical Mapping 22
 2.6 Screens 30
 2.7 Display Timing Signals 32
3. THE FRAME BUFFER INTERFACE 34
 3.1 The Access Modes 34
 3.2 Single Access Mode 35
 3.3 Interleaved Access Mode 35
 3.4 Superimposed Access Mode 36
 3.5 Graphics and Characters 37
 3.6 System Configuration 40
 3.7 System Examples 41
 3.8 A More Detailed Example of a Graphics System 44
 3.9 Display Memory Timing 46
 3.10 Display Memory Status Signals 48
 3.11 DRAM Refresh 50
 3.12 Video Attributes 51
 3.13 Dummy Cycles 55
4. SYSTEM DESIGN-DESIGN MEMORY 56
 4.1 Transactions 56
 4.2 Design Example 60

5. FRAME BUFFER HARDWARE DESIGN 63
5.1 Requirements Overview 63
5.2 Address Latch 64
5.3 Address Multiplexer 67
5.4 Write Decoder 67
5.5 Data Buffers 68
5.6 Data Multiplexers 68
5.7 Shift Register 69
5.8 DRAM Array 72
5.9 Attributes 73
6. OSCILLATOR TIMING AND CONTROL LOGIC 75
6.1 Overview 75
6.2 The ACRTC Clock and 3 The Pixel Clock 77
6.3 Pixel Clock 79
6.4 Write Enable: WRITE EN 79
6.5 Read Enable: READ EN 79
6.6 Attribute Latch Enable: ATT EN 79
6.7 Address Latch Enable: ADD EN 79
6.8 Row/Column Address Select: ROW/$\overline{\text{COL}}$ 79
6.9 The DRAM Row Address Strobe: $\overline{\text{RAS}}$ 80
6.10 The DRAM Column Address Strobe: $\overline{\text{CAS}}$ 81
6.11 Timing Generator 85
7. ACRTC PARAMETERS AND RESISTERS 89
7.1 Reset 89
7.2 Initialization 92
7.3 Timing Control RAM 93
7.4 Horizontal Timing 96
7.5 Vertical Timing 99
7.6 Display Control RAM 101
7.7 Control Register 109
VOLUME 2 - HARDWARE 123
VOLUME 3 - SOFTWARE 193
HITACHI SALES OFFICES 281

1. INTRODUCTION TO GRAPHICS SYSTEMS

1.1 Bit Mapped and Vector Displays (Monochrome)

In general images generated on a graphics display can be divided into two classes:

(i) Bit map (or raster scan)
(ii) Vector (stroke, or random scan)

For the bit mapped method, the image to be displayed is constructed from a two-dimensional array of picture elements called 'pixels'. Each of these can take on individual color and brightness values so that the image seen on such a display system is in effect a mosaic, formed from a matrix of small elements. The eye of the viewer then reconstructs the intended image, provided the elements are small enough to give the required resolution. It follows that in order to display fine detail there must be a large number of these pixels, each of which is 'mapped' from a screen position into a corresponding memory location within the frame buffer, and where each data value has attributes which specify the brightness of the associated pixel.

It is precisely because of this 'mapping' relationship between pixel screen position and frame buffer location that this technique is called the 'bit-mapped method'.

In the 'vector' method all drawings are constructed from straight lines (hence the term 'vector') where each line is specified by given end points. With this method curve construction can be realised by a piecewise approximation using many short lines joined together end to end. Similarly, lines can be placed close together to give the impression of a solid area. When used for simple lines and shapes, this technique offers economical usage of memory space. However, as the number and variety of displayed images increases, a situation is reached where the number of line sequences force the screen refresh rate to fall to a level where screen flicker becomes apparent.

Vector displays were common when the cost of memory was high and as a consequence, early bit-mapped displays were limited to low resolution text displays such as VDU's (even in these, the small amount of memory required was often further reduced by using built-in character generators).

Since the introduction of 64K, 256K DRAMS and with the impending introduction of the 1M DRAMs, the cost of implementing a high resolution bit-mapped display has reduced so dramatically that a point has been reached where this method is now the more popular for graphics applications.

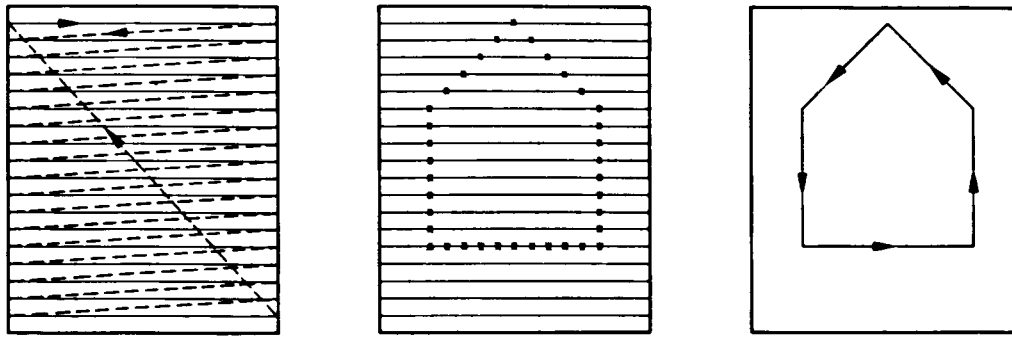
1.2 Raster Scan

In the display method known as 'Raster Scan' the display differs from the vector display technique principally in the way displayed data is represented on the screen. The term raster implies that the image to be represented on the screen is constructed by a succession of equidistant scan lines, of ''rasters'' where each of these scan lines is realized by moving an electron beam repeatedly across and down the screen. In this way the entire display area can be covered as shown in fig. 1-1 (a). To begin the display, the first raster, top of screen, is produced by moving the electron beam across the screen from the left to the righthand edge. The beam is then switched off and rapidly returned to the lefthand edge, and offset downwards, ready for the next raster line.

The whole sequence is then repeated until the bottom of the screen is reached. When this occurs the beam is then switched off and repositioned at the top of the screen. The periods when the beam current is switched off and the beam is returned to the start of a line or the beginning of a frame are known as 'flyback' periods.

To form an image on the screen, the intensity of the beam is varied at the appropriate points on each raster, so that when all of the rasters are displayed together, a total image will appear.

Each raster is effectively divided up into short sections which are represented in the display memory by a given number of binary bits, depending on whether a grey scale or color is required. These are known as pixels. Consequently as the electron beam transverses the raster the beam intensity at each pixel point is controlled by the data value held at the corresponding point in memory.



(a) Raster Scan (b) Display Example of Raster Scan (c) Display Example of Vector Display

Fig. 1-1 Raster Scan and Vector Display

Fig. 1-1(a) shows the basic raster scan, simplified by having only a few scan lines.

A simple drawing of a house using this system would produce the image shown in Fig. 1-1(b). The dots represent paints at which the electron beam intensity is increased.

By contrast, a vector display is shown in Fig. 1-1(c), whereby the image is constructed from 5 straight lines.

1.3 Raster Display Screen

There are three basic components to a raster display screen.

The output display device, (usually a monitor of TV standard), memory used for holding the data to be displayed (frame buffer), and a display controller for modifying the contents of the frame buffer and ensuring that the data held within is accurately displayed on the output display device.

Since output display devices are usually of the short persistence TV screen variety, images need to be repeatedly recreated on the screen in order to ensure that a continuous picture occurs. The display controller, therefore, needs to be able to repeatedly transfer (refresh) image information from the frame buffer out to the display device. For the typical TV type monitor, the repetition rate is in excess of 25 times per second in order to prevent flicker.

The frame buffer is a digital memory and must be of a size which is sufficient to hold the information representing the intensity of each pixel.

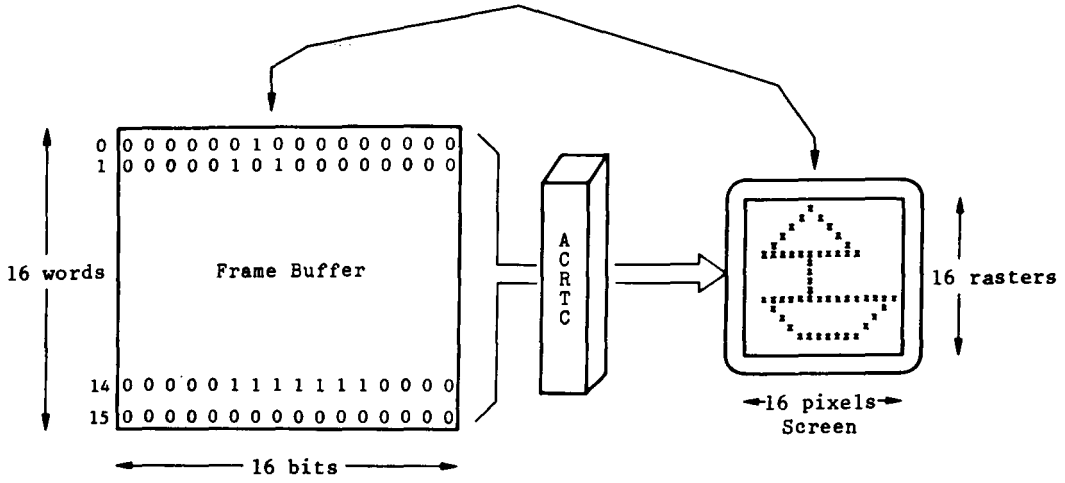


Fig. 1-2 Basic Components for Raster Display

1.4 Color System

Up until now we have considered only the simple black and white display-where a pixel had a 1 bit value or 0, i.e.: ON or OFF. With memory costs falling it is now reasonable to use a number of bits to specify a pixel's intensity and color.

The minimum requirements for achieving a gradual grey scale are 5 bits. For a color system with its three primary colors, we need 15 bits. The more bits per pixel the subtler the effect. These systems would allow for natural scene shading and coloring.

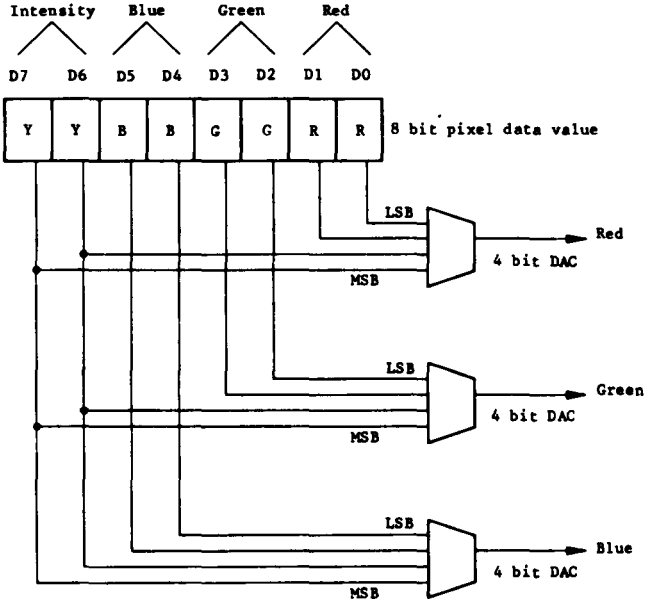
Most graphic systems, however, are not intended to reproduce the full range of natural colors and shades.

A common configuration is 4 bits per pixel, allowing one bit for each of the primary colors and one bit for intensity. Allowing 8 bits per pixel provides for two bits per primary color and for intensity. This arrangement-with 256 definable hues, can produce a very useful color display, eg: Fig.1-3(a).

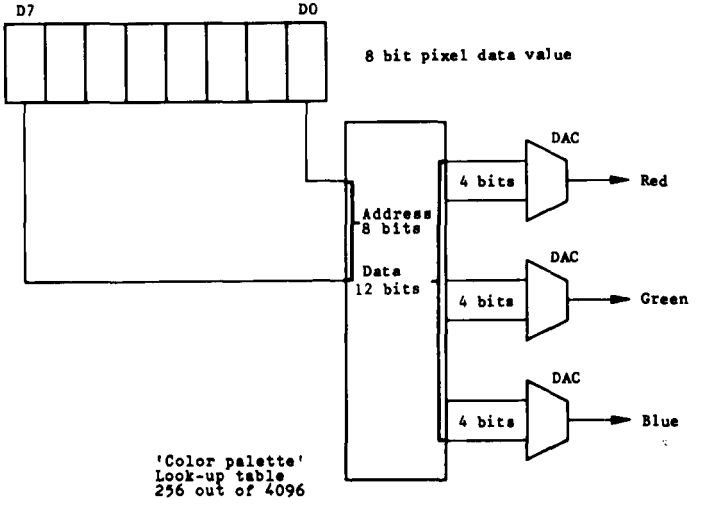
In order to offer a large range of colors without the penalty of many bits per pixel, a 'look-up table' can be used. The data value of a pixel is not used directly to control the color or intensity, but instead acts as an entry address within a 'color palette' look-up table. The value obtained from this table is used to define the pixel color and intensity. The trick is to obtain more bits from the table than were used as address. This means

that an 8 bits per pixel system could produce 12 bits of color data. The trade-off is that one can only choose from 256 out of the 4096 definable colors. As the table is accessed for each pixel, it must be operated at high speed.

Fig.1-3(b) shows this arrangement.



(a) 8 bits/pixel Color Display System



(b) Color Paleted Look-up Table

Fig. 1-3 Color Display System

1.5 Displaying and Drawing

We have considered how the displayed image is repeatedly reproduced from the frame buffer pixel values. This process keeps the display controller and frame buffer occupied for most of the time. During the horizontal and vertical flyback times no displaying takes place, so the frame buffer memory becomes briefly available for other purposes.

Typically 20% of the line period is used for horizontal flyback, and 7% of the field period for vertical flyback.

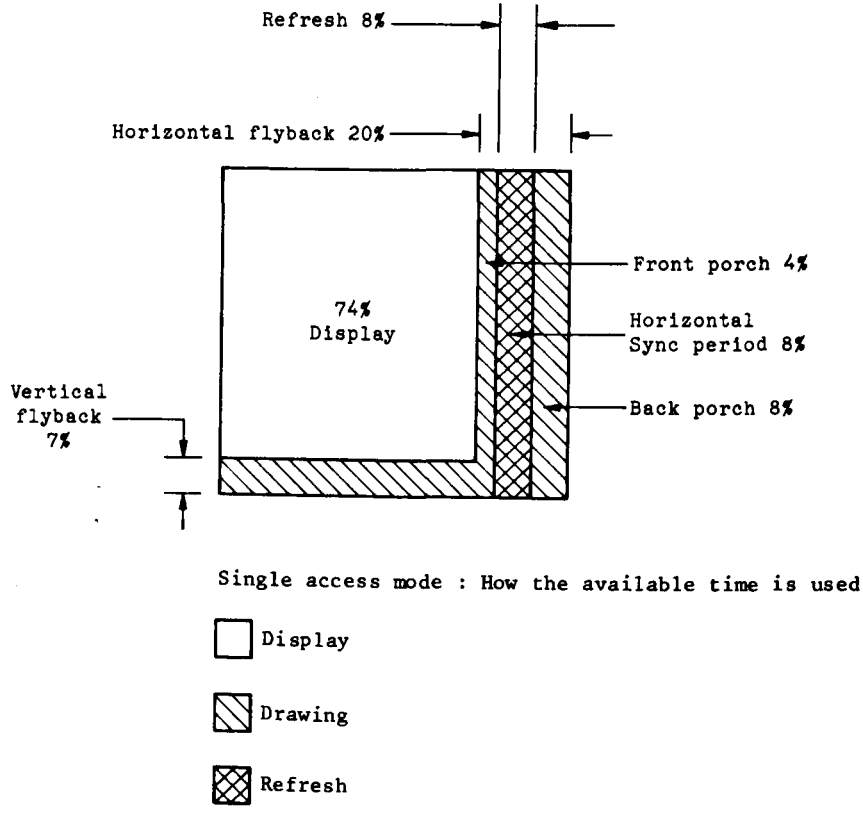


Fig. 1-4 Displaying and Drawing (Single Access Mode)

This leaves about 26% of the time available for non-display activities, that is DRAM refresh and drawing.

Dynamic RAMs require periodic refreshing so that their contents do not become corrupt. If these are used for the frame buffer, then refresh cycles must be performed during the horizontal flyback periods. This period is composed of 3 parts: the front porch, the sync period and the back porch representing 20%, 40%, and 40% of horizontal flyback period respectively. DRAM refresh takes place during the sync period leaving only 18% of the time for drawing.

Drawing is carried out by modifying the pixel data values within the frame buffer.

Drawing requires that the pixel data is first read, modified, and then written. As the frame buffer is occupied during the display periods, drawing must be done during the flyback periods. However, this makes the drawing slow. Additionally, if the display process is stopped while drawing operations are done, unpleasant visual 'flashes' appear on the screen. This method is not often used, though drawing is performed much faster.

A solution to this is to interleave drawing cycles with the display cycles. This requires that the pixel data obtained from a display cycle must be sufficient to last throughout the subsequent drawing cycle. This permits fast drawing with none of the visual 'flashing' problems.

It does impose the requirement that the frame buffer must either be accessed twice as fast, or it must provide twice as many pixel data values per access.

Single access mode gives typically (with display having priority over drawing):

74%	DISPLAY
0-18%	DRAWING as required
8%	DRAM REFRESH

This makes the drawing process rather slow.

If drawing has priority over displaying then

74-0%	DISPLAY
0-92%	DRAWING as required
8%	DRAM REFRESH

Clearly the drawing can be done faster, but when drawing the display process will be interrupted, disrupting the image.

Interleaved access provides for 50% drawing during the display period and 100% drawing during the flyback periods less any dram refresh time.

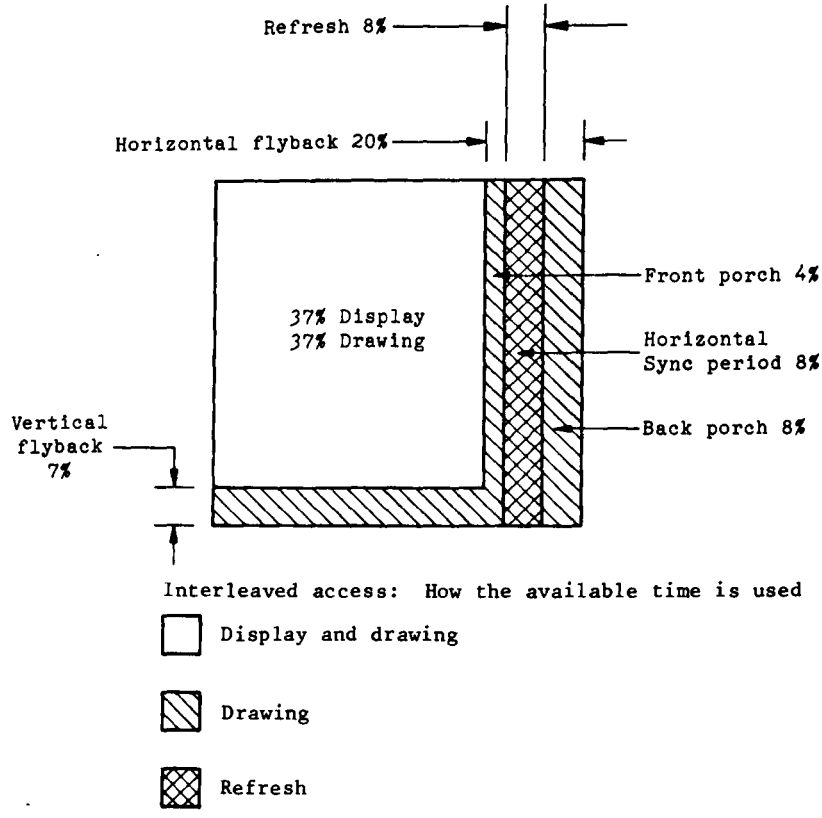


Fig. 1-5 Displaying and Drawing (Interleaved Access Mode)

Drawing time = (74% × 1/2) + 17% = 54%
Display time = 74%
DRAM refresh = 8%

The improvement of interleaved versus single access mode is 54% vs 18%; i.e. about 3 times.

As a further benefit of interleaving, the frame buffer and hence the ACRTC are cycled at a higher frequency, so the computation times for the drawing process will also be faster.

The drawing process involves three phases:

1. Reading a pixel data value from the frame buffer
2. Carrying out a computation and then
3. Writing the new pixel data value back to the location

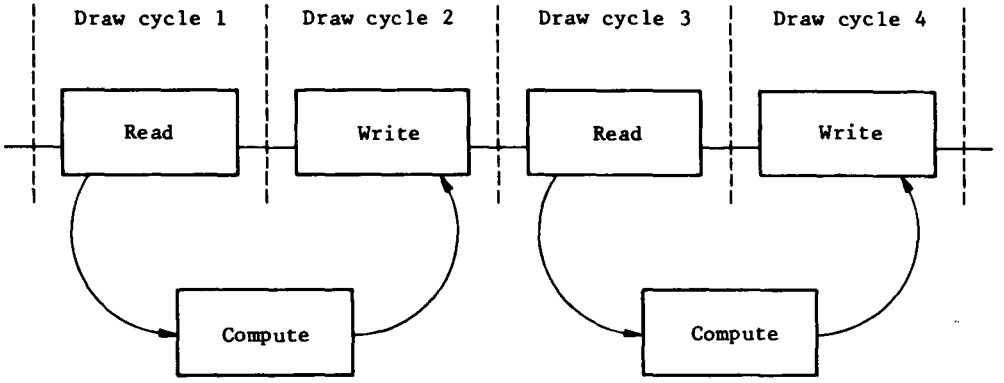


Fig. 1-6 Short pixel computation

All frame buffer transactions for drawing purposes must use drawing cycles.

When the computation time is very short, then the next read phase will be performed right after the write. If, however, the computation takes a while, then the result will not be available in time for the next draw cycle-hence the controller will skip cycles until the answer is available. This results in some waste when the pixel computation is complicated.

During the flyback periods 100% drawing cycles are available, so there will be cases of cycle skipping when drawing is done. When drawing and display cycles are interleaved then each draw cycle is separated by a display cycle and the pixel computation can be done in parallel with these cycles.

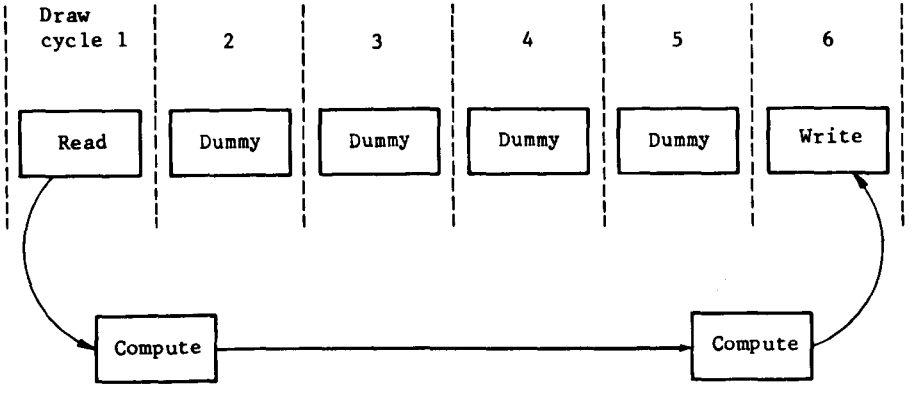


Fig. 1-7 Long Pixel Computation Waste

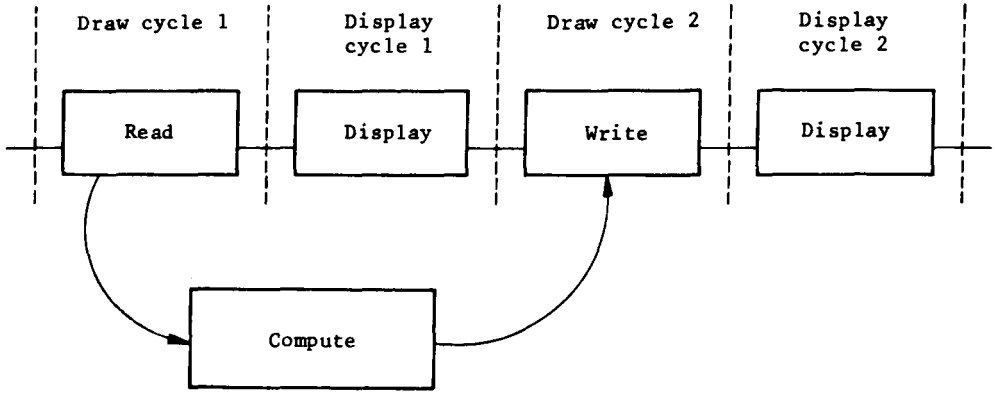


Fig. 1-8 Interleaved Long Pixel Computation, no Waste

This can often avoid any draw cycle skipping and hence improve the frame buffer usage.

Interleaved accessing can improve the drawing speed by considerably more than 3 times but the actual increase is dependent of the drawing operations undertaken and so is difficult to predict.

In a system, display cycles and drawing cycles contend for the frame buffer. The management of these functions is critical to a graphics system and deciding on which gets priority depends on the use to which the system will be put.

The ACRTC device can be configured to operate in any of the above modes. It is the interleaved mode that is the most powerful.

1.6 Frame Buffer to Video Signals

The basic raster scan display system described in Fig. 1-9 shows a display controller that transfers the pixel data from the frame buffer to the display monitor. During a display cycle, the frame buffer gives out parallel data. This information represents pixels and is loaded into a parallel to serial converter (shift register). A clock is applied to this shift register and the pixel values are presented sequentially to the display monitor as the video signal. The clock is called the pixel clock or dot clock and its frequency is that of the pixel rate for the system.

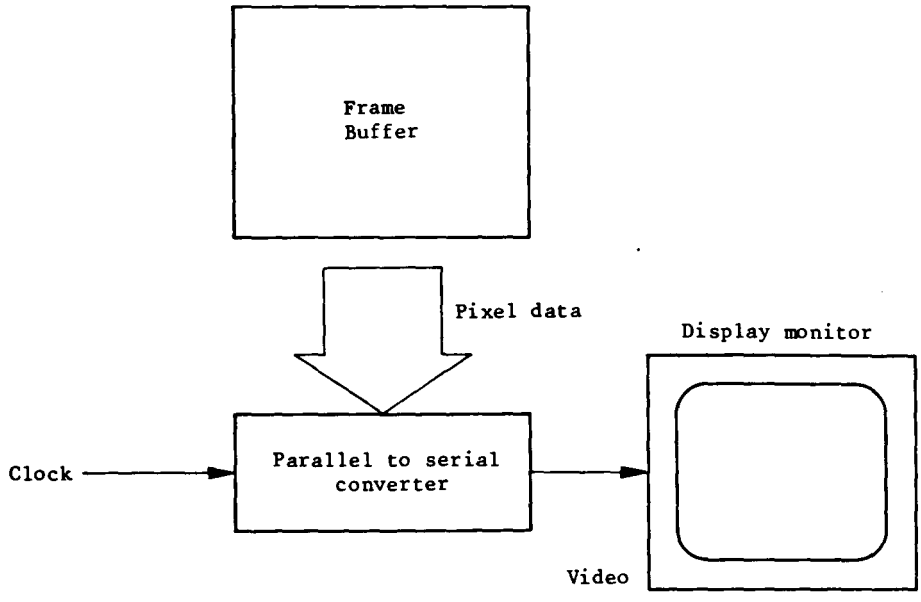


Fig. 1-9 Basic Raster Scan Display System

In a simple monochrome system, with either black or white pixels, each bit of the frame buffer would represent one pixel. A word would then hold 16 pixels and during a display cycle these would be parallel loaded into a 16 bit shift register. The applied pixel clock would shift the 1 bit values out of this register into the monitor, which would display the image-see Fig. 1-10.

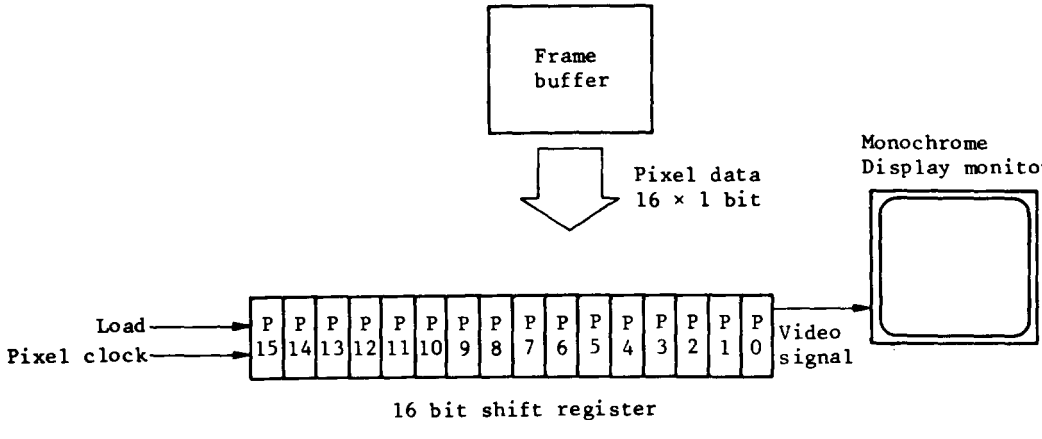


Fig. 1-10 Monochrome Display System

For a color system using 4 bits per pixel, a word would hold only 4 pixels. These would be loaded into four 4 bit shift registers and clocked out to the color display monitor. The 4 bits could drive the red, green, blue, and intensity signals of the monitor-see Fig. 1-11.

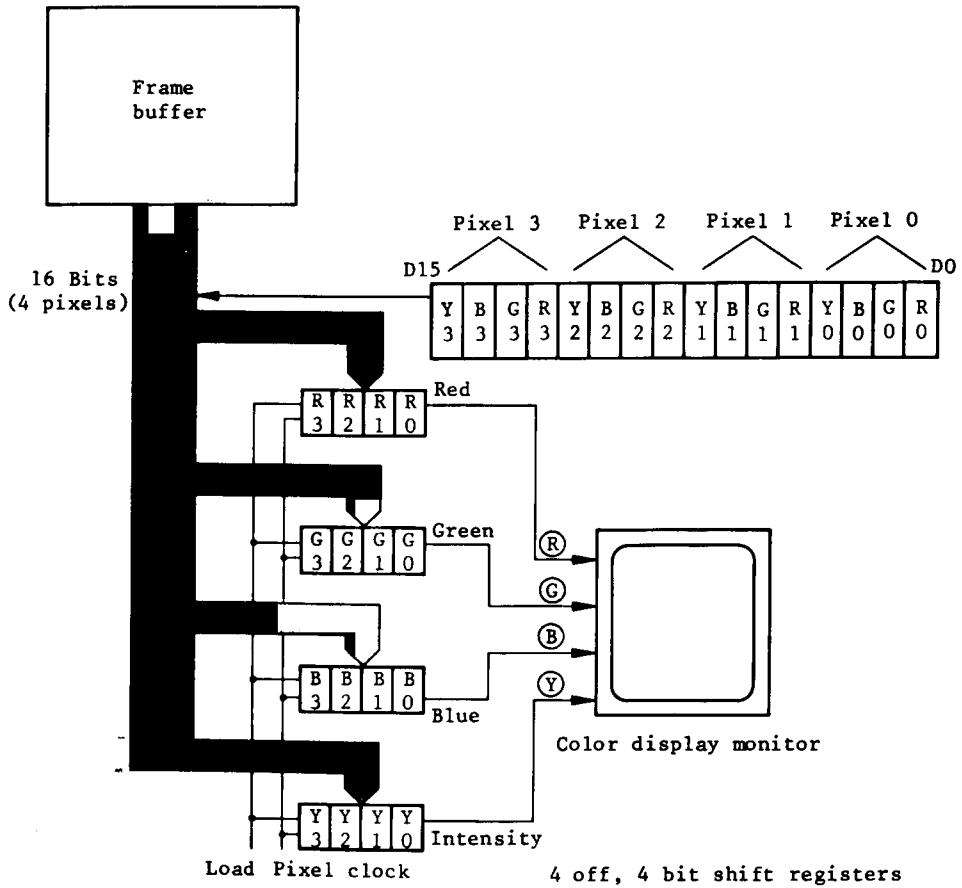


Fig. 1-11 Color Display System, 4 Bits per Pixel

2. DISPLAY INTERFACE

2.1 The ACRTC Frame Buffer Considerations

The ACRTC uses a word-oriented frame buffer to hold a bit map representation of the displayed image. Each screen pixel corresponds to a stored value within this memory. The number of bits stored for each pixel affects the range of the definable color hues or gray scales. 1 bit per pixel can provide a simple black and white display on a monochrome CRT. 16 bits per pixel can provide a display using 64K different colors. With the ACRTC it is possible to choose between 1, 2, 4, 8, or 16 bits per pixel. The more bits per pixel, the faster the frame buffer must be accessed for a given pixel rate. If 1 bit per pixel is specified, then the frame buffer need only be accessed once per 16 pixels, but if 8 bits per pixel is used then it needs to be accessed every 2 pixels. The chosen 'pixel rate' and 'bits per pixel' are factors that determine the frequency at which the frame buffer must be accessed for display purposes.

The pixel rate is a function of the resolution of the display. The active display time of one raster divided into the number of pixels horizontally gives the pixel rate.

$$\frac{\text{Horizontal pixel count}}{\text{Raster display time}} = \text{pixel rate}$$

Example: a 512-pixel display (horizontal resolution) using a UK TV line rate with an active display period of 51.2 μs. What is the pixel rate?

$$\frac{512}{51.2\ \mu s} = 10\ \text{MHz pixel rate}$$

The pixel data rate from the frame buffer is the product of the pixel rate and the 'bits per pixel' mode use. This represents the rate at which display data is passed to the display device.

Example:
If each pixel is specified by 8 bits of data and the pixel rate is 10 MHz. What is the pixel Data rate?

10 MHz × 8 bits = 80 M bits per second

Pixel data rate = 5 M words per second

The data obtained by such accesses are loaded into shift registers (parallel to serial converters) and shifted out by a pixel clock, becoming the video data stream that eventually goes to the CRT creating the image.

Our example requires that 5 words are obtained every microsecond from the frame buffer. This allows only 200 ns per access. As the cycle time of a typical 256K DRAM is about 250 ns, we have a problem. The solution is to obtain more than one word from the frame buffer during each display access. If we design the frame buffer to output 2 words (or 32 bits) in parallel, then we need only to access it every 400 ns, well within the DRAM specifications.

In order to do this, the frame buffer address must be incremented by 2 between each display access.

This function is supported by the ACRTC via the GAI (graphic address increment) value. This can be set to increment the frame buffer word address by values in the range 1/2 to 16. Setting GAI to +16 provides for 156 bits to be obtained per display access. For our example, we would set the GAI to +2, and arrange for 32 bits to be output from the frame buffer and the external shift register logic to accept this amount of data.

This graphic increment mode is useful where high pixel data rates are needed and the frame buffer memory chips would be too slow otherwise.

If only 16 bits (1 word) were obtained per display access, the maximum data rate for a frame buffer using DRAM devices with a cycle time of t_{cyc} is:

$$\text{Max Data Rate} = \frac{16}{t_{cyc}}$$

eg: with 250 ns cycle time DRAMS

$$\text{Max Data Rate} = \frac{16}{250\ \text{ns}} = 64\ \text{Mbits per second}$$

Which is clearly less than we needed for our example (80 Mbits per second).

The frame buffer is not only accessed for display purposes, but also for drawing. The ACRTC needs to be able to read and write the various pixels in order to create and modify images.

If we wish to interleave display and draw cycles, then only half the time that was available can be used for displaying. This puts a further burden on the frame buffer. The solution is to double the amount of data obtained during each display access, and increase the capacity of the shift registers and other external logic to suit.

Going back to our example system in which the frame buffer provided 2 words (32 bits) for each display access, then if we now wish to interleave drawing cycles with display cycles we will require 4 words (64 bits) to be output in parallel.

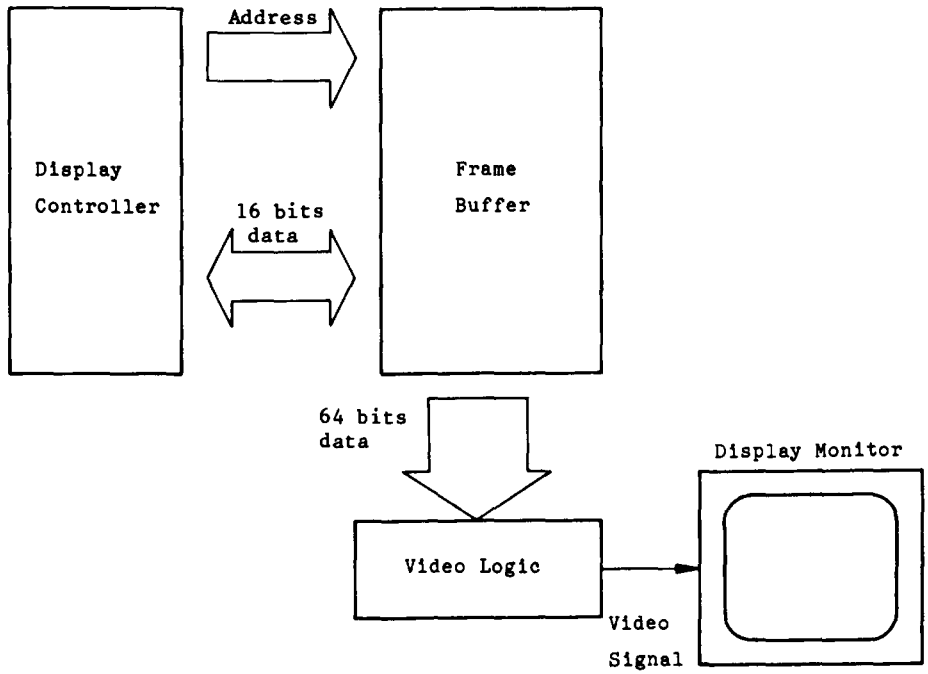


Fig. 2-1 Address Increments by +4 Words to Display
64 Bits of Pixel Data

The only alternative to this route is to further speed up the memory devices. Unfortunately as the frame buffer capacity is generally very large (1 Mbyte - 2 Mbytes), it is too costly to resort to fast but expensive memory devices using bipolar or ECL technology. Frame buffers generally therefore, utilize NMOS DRAMS with cycle times of between 200 and 300 ns.

This means that the GAI mode of the ACRTC must be set to +4 so that the addresses during display cycles will increment by four words.

2.2 Graphic Address Increment Mode - GAI

The facility provided by the ACRTC that supports obtaining a number of words of pixel data from each display access is the Graphic Address Increment mode, GAI. This can be set to increment the address used for display accesses in the range of +1/2 to +16, allowing incrementing of the address by 1 every other display access to incrementing by 16 every display access respectively.

The greater the number of bits obtained, the longer the allowable delay between accesses. When the frame memory is operated in the single access mode drawing must be slowly performed, as only the flyback period will be available.

Interleaved mode, also called dual access mode, time multiplexes the frame buffer between displaying and drawing on alternate access cycles. It requires, however, that twice as many pixels be obtained during each display access.

Table 2-1 relates the 4 factors involved in defining the frame buffer operation against the necessary GAI.

(i) Pixel rate (resolution)
(ii) Single or dual access mode
(iii) Bits per pixel (colors)
(iv) Memory device cycle time

For example, a system that needs a 32 MHz pixel rate, allocates 4 bits per pixel, uses a frame buffer with a 250 ns cycle time and needs to perform fast drawing via the dual access mode. What GAI should be used?.

Referring to the table gives the value of +4.

Proof:
The frame buffer will yield $4 \times 16 = 64$ bits

This is 64 bits
4 bits per pixel

So display duration = 16 pixels $\times \frac{1}{32\text{MHz}} = 500$ ns

= Two memory cycles at 250 ns each

That is one display and one draw cycle.

Table 2-1 shows the relationship between GAI, dot rate, access modes, bits/pixel, and memory cycle. The ACRTC compensates for the addressing scheme needed to implement GAI values other than +1 by offsetting the refresh address placed on the address lines.

Table 2-1 Graphic Address Increment Modes

Dot Rate		16MHz		32MHz		64MHz		128MHz	
Access Mode Color No. (bit/pixel)	Memory Cycle	S	D	S	D	S	D	S	D
1	250ns	–	+1/2	+1/2	+1	+1	+2	+2	+4
	500ns	+1/2	+1	+1	+2	+2	+4	+4	+8
2	250ns	+1/2	+1	+1	+2	+2	+4	+4	+8
	500ns	+1	+2	+2	+4	+4	+8	+8	+16
4	250ns	+1	+2	+2	+4	+4	+8	+8	+16
	500ns	+2	+4	+4	+8	+8	+16	+16	–
8	250ns	+2	+4	+4	+8	+8	+16	+16	–
	500ns	+4	+8	+8	+16	+16	–	–	–
16	250ns	+4	+8	+8	+16	+16	–	–	–
	500ns	+8	+16	+16	–	–	–	–	–

2.3 Resolution of the Display

Many factors must be manipulated to obtain a successful compromise between the requirements for high resolution and the practical limitations imposed by the frame buffer memory devices.

Starting with a system specification for the displayed resolution, we can relate this to the frame buffer cycle time, in the following way.

A color display monitor has its resolution limited by two major factors:

(i) The scan frequency at which it operates
(ii) The pitch of the phosphor dots on the face

For monochrome monitors, only the scan frequency limitation applies.

In the U.K. the T.V. scan frequency is 15.625 KHz, while in the U.S.A. it is 15.75 KHz.

The U.K. scan period is 64 μs, which includes both the active display time and the flyback time, with the flyback process usually requiring about 20% of the scan period. So typically, only 80% of the scan period is available for display purposes.

If the system horizontal resolution is HRES pixels and the system scan frequency is FSCAN, it follows that the pixel rate, Rpx = FSCAN × HRES × 1.25. High quality display monitors may require a flyback time of less than 20%.

Example A, display system with a 512 horizontal pixel resolution and a monitor running at 15.625 kHz scan frequency. What is the pixel rate (assuming a 20% flyback period)?

$R_{px} = 15.625 \times 10^3 \times 512 \times 1.25$

Pixel rate = 10 MHz

The pixel data rate from the frame buffer is given by the pixel rate multiplied by the number of bits per pixel. This data rate must be met by repeating the display access cycles sufficiently often bearing in mind both the amount of data obtained by each access and whether single or dual access (interleaved drawing), mode is used. These factors can be represented by the Frame Buffer cycle time (TFB), the Graphic Address Increment mode (GAI) and the Access mode (ACC). Relating these together gives a useful general equation for the frame buffer design limit against system resolution requirements.

$$TFB = \frac{128 \times 10^5 \times GAI}{NPX \times FS \times HRES \times ACC}$$

Where:

TFB = FRAME BUFFER CYCLE TIME; in ns
GAI = ACRTC ADDRESS INCREMENT VALUE; +1, +2, +4, +8, +16
NPX = NUMBER OF BITS PER PIXEL; 1, 2, 4, 8, 16
FS = DISPLAY MONITOR SCAN FREQUENCY, in kHz
HRES= HORIZONTAL RESOLUTION; in PIXELS
ACC = ACCESS MODE USED; SINGLE = 1
DUAL (interleaved) = 2

This assumes the flyback time is 20% of the scan time.

It is necessary to use memory devices (DRAM's) that have a cycle time less than TFB. The above equation is useful as a first approximation for a system design and will quickly indicate whether a particular scheme is achievable.

For example: (1)
A system is required that has a horizontal resolution of 512 pixels and uses a display monitor with a 31.25 kHz scan frequency. It will use 8 bits per pixel to obtain 256 colors and the frame buffer will be based on DRAMS with a cycle time of 300 ns.

In order to draw quickly, dual access mode will be used. How many bits must be obtained per display access?.

Allowing a 400 ns frame buffer cycle time will give some margin for logic delays around the DRAMS.

The equation becomes:

$$GAI = \frac{TFB \times NPX \times FS \times HRES \times ACC}{128 \times 10^5}$$

GAI = 8

Hence, by using the ACRTC Graphic Increment mode set to +8, 8 words will be obtained, that is 128 bits. The video shift register logic must be designed to have this capacity.

Example: (2)
A 1500 horizontal resolution black and white system uses a display scan frequency of 62.5 kHz. If the video shift registers can hold 32 bits and single access mode is to be used, what specification DRAMS are required?.

First, we need to solve for the frame buffer cycle time, TFB. Here the following values apply:

GAI = 2
NPX = 1
FS = 62.5
HRES= 1500
ACC = 1

$$TFB = \frac{128 \times 10^5 \times 2 \text{ ns}}{1 \times 62.5 \times 1500 \times 1}$$

TFB = 273 ns

If the frame buffer cycle time is 273 ns, it would be necessary to use 120 ns access time (eg: HM50256-12) DRAM's with a cycle time of 220 ns, with careful attention to the logic design to avoid excessive additional delays. If necessary, the GAI and video shift register capacity may need to be increased, allowing a greater frame buffer cycle time to be used. The high scan rate display monitor may not exhibit a horizontal flyback time of 20%, which is more typical of more standard monitors.

2.4 Bits and Pixels

The ACRTC supports a word orientated frame buffer. Each addressed location within it contains 16 bits. How these bits are used to represent image information is flexible in that the number of bits allocated to each pixel can be programmed. If only one bit is used then the result is a black and white display. Four bits per pixel provides for 16 grey levels on a monochrome display or the minimum signals for a color scheme using RGBY control (RED, GREEN, BLUE and INTENSITY). 16 bits per pixel can produce an image with realistic natural coloring and shading.

The ACRTC can operate on a range of bits per pixel values, specifically 1, 2, 4, 8 or 16. A major advance over previous graphic controllers lies in the approach taken to pixel processing for drawing. It provides no time penalty for drawing operations when more than one bit per pixel is specified, hence a

64K color scheme using 16 bits per pixel can be processed as quickly as a 1 bit per pixel black and white system.

A display access transfers one or more words from the frame buffer into a video shift register arrangement and a pixel clock shifts this data out as a stream of pixels that become part of the displayed image.

Pixels are stored within words in the frame buffer and packed continuously.

Consider a system that uses 4 bits per pixel and which outputs 32 bits (GAI = +2) per display access. Within the frame buffer there will be 4 pixels per word and two words will be read out during one access.

Fig. 2-2 shows the arrangement and Fig. 2-3 how the bits will be used to provide the video signals.

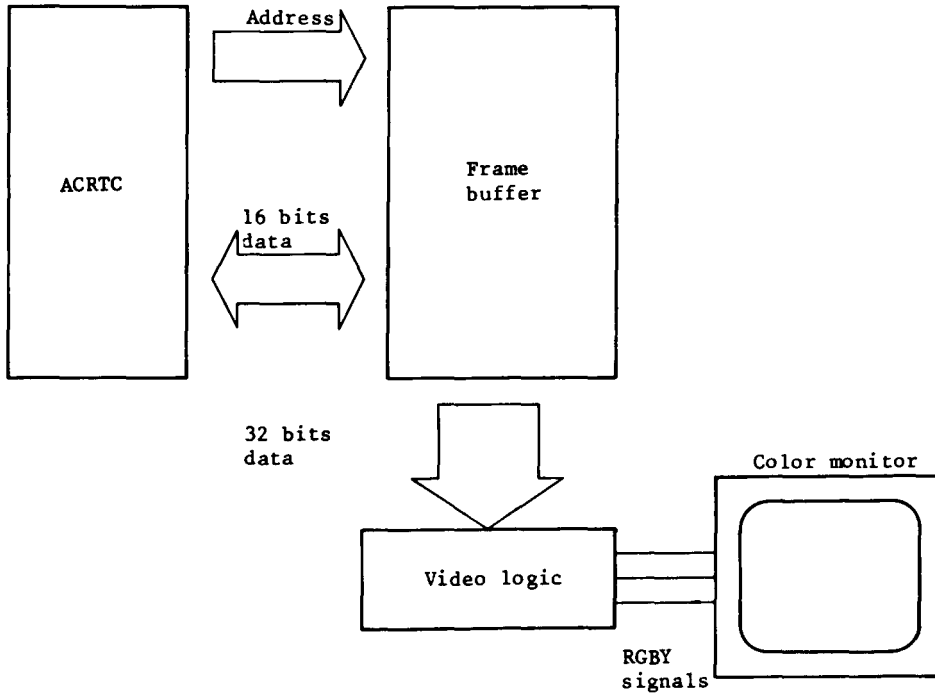


Fig. 2-2 Address Increments by +2 Words for Display Access to Obtain 32 Bits of Pixel Data

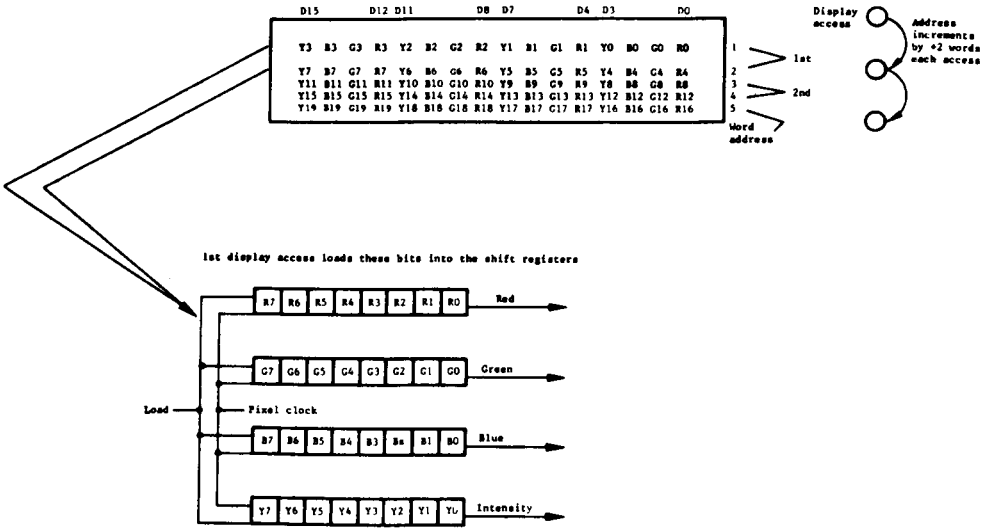


Fig. 2-3 4 of 8 bit Shift Register

2.5 Logical/Physical Mapping

The display screen is composed from an array of pixels, each mapped to a data value held in the frame buffer.

Each raster is a series of pixels and as such occupies a number of consecutive locations within the frame buffer, the amount being related to the number of pixels stored per word of memory and the number of pixels per raster, i.e. horizontal resolution.

Totalling up by the number of rasters used in the display, i.e. vertical resolution indicates the memory required for the display system.

Consider a display system with a 512 × 512 resolution and 4 bits per pixel.

The screen will look like Fig. 2-4.

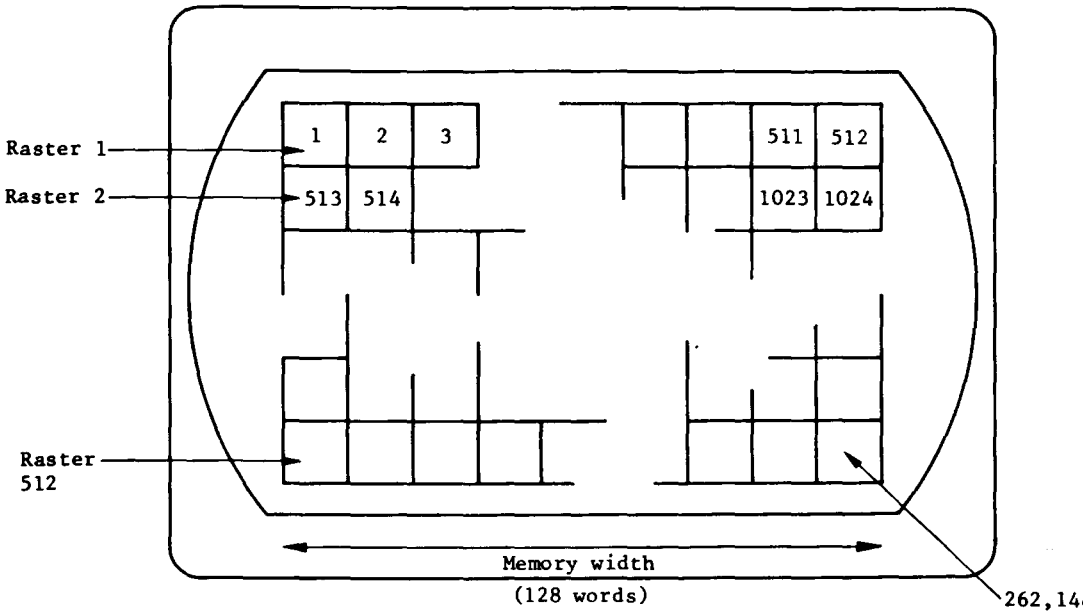


Fig. 2-4 Pixel Numbers and Screen Position

In all 256K pixels would be used. As it requires 1 word per 4 pixels the frame buffer capacity would be 64K words.

The frame buffer is word-oriented and the pixel storage would appear as in Fig. 2-5.

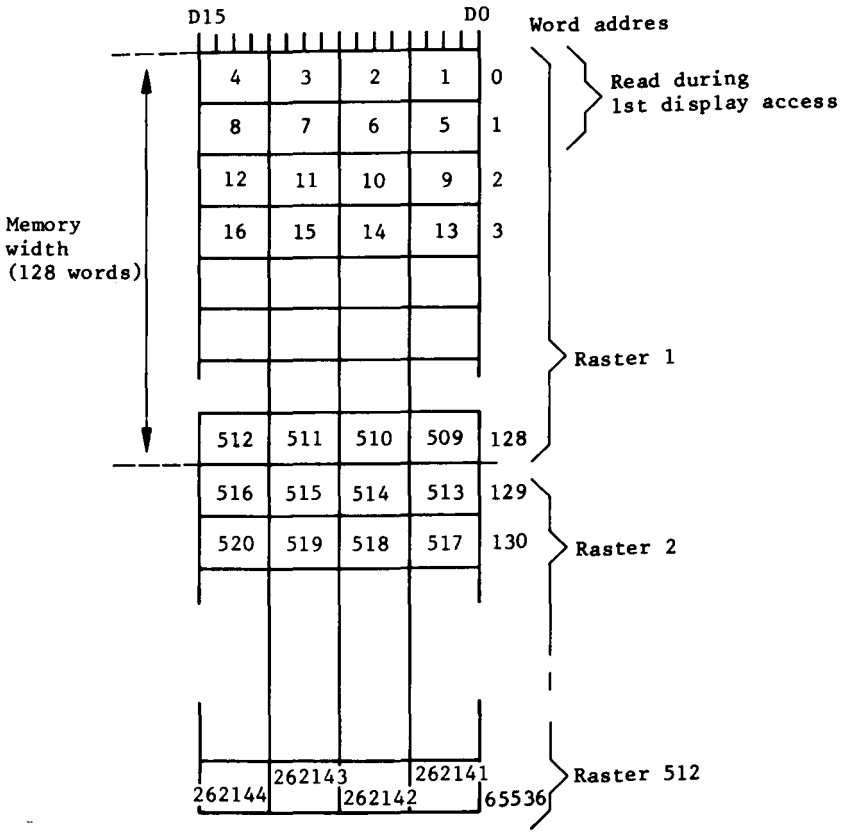


Fig. 2-5 Frame Buffer Contents as Pixel Numbers

If we were to use the GAI = +1 mode, pixels 1 through 4 would be accessed during the first display cycle, and then 5 thru 8 and so on.

Setting the GAI = +2 would allow 32 bits to be read during each display access. The first would obtain pixels 1 through 8 then 9 thru 16 and so on.

The memory width would be equivalent to 512 4 bit values or 128 words.

A 64K words frame buffer like this could be constructed from 16 64K × 1 bit DRAM devices. This scheme would work when using GAI = +1 as only one word would be read per access. In order to use GAI = +2, two words must be obtained and as only 16 devices are used this is clearly impossible, however if 32 devices are used then the frame buffer will be twice the size required, i.e. 128K words.

The extra frame buffer space will not be displayed on the screen. The resolution of the system could be increased to make more efficient use of the available frame buffer capacity. Alternatively the displayed screen could be moved about within the frame buffer to allow scrolling, giving the effect of moving the viewport. The ACRTC allows the screen width and starting point to be separately defined from the memory width.

Example in Fig. 2-4 had the same screen width and the memory width.

The memory width defines how many words are used to store a complete raster. The number of these words actually used for the displayed image is the screen width. If the screen width is less than the memory width, then horizontal scrolling is possible as the screen will show only part of each raster - see Fig. 2-6.

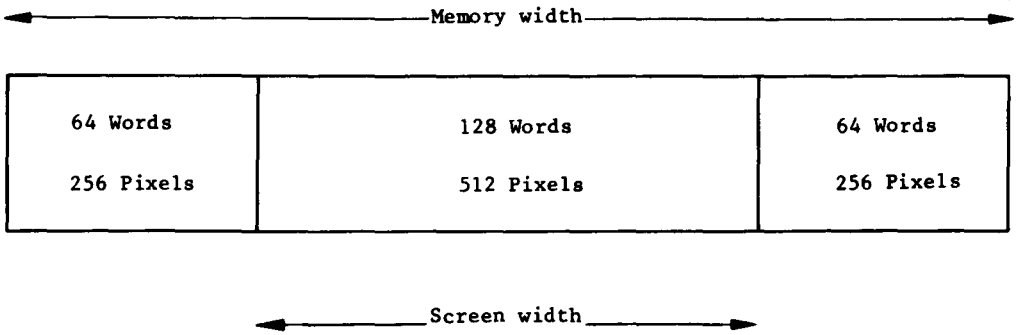
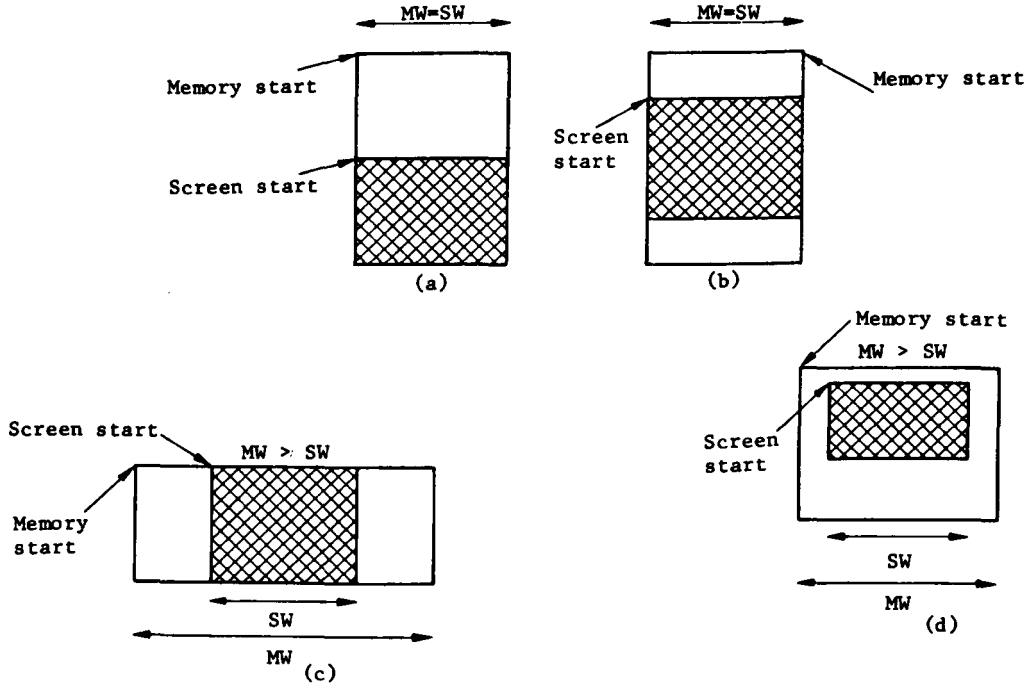


Fig. 2-6 Screen Width < Memory Width

The start address of the screen defines the point from which the screen data is obtained. In effect, it relates the top left corner of the display to a frame buffer location.

By manipulating these 3 factors, screen width, start address and memory width, many frame buffer configurations can be produced. Fig. 2-7 (a, b, c, d) gives some examples where the frame buffer capacity is twice that required for the display screen size.



Memory and screen configurations
Memory size = 2 × screen size
MW = Memory width
SW = Screen width

Fig. 2-7 Relation between Screen Width, Start Address and Memory Width

Taking Fig. 2-7(d) further, suppose the frame buffer was 128K words and the display was 512 × 512 4 bit/pixels and it started at 16th line down and centered in the frame buffer of memory width = 768 pixels horizontally, the detail would be as in Fig. 2-8 (a-e) and Fig. 2-9.

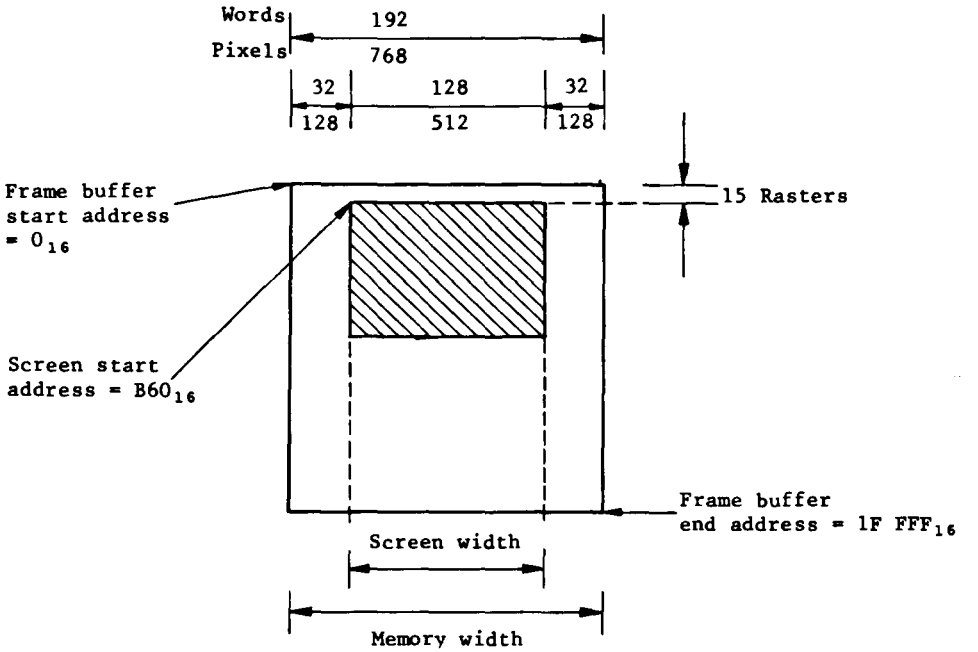


Fig. 2-8(a) Relation between Screen and Frame Buffer

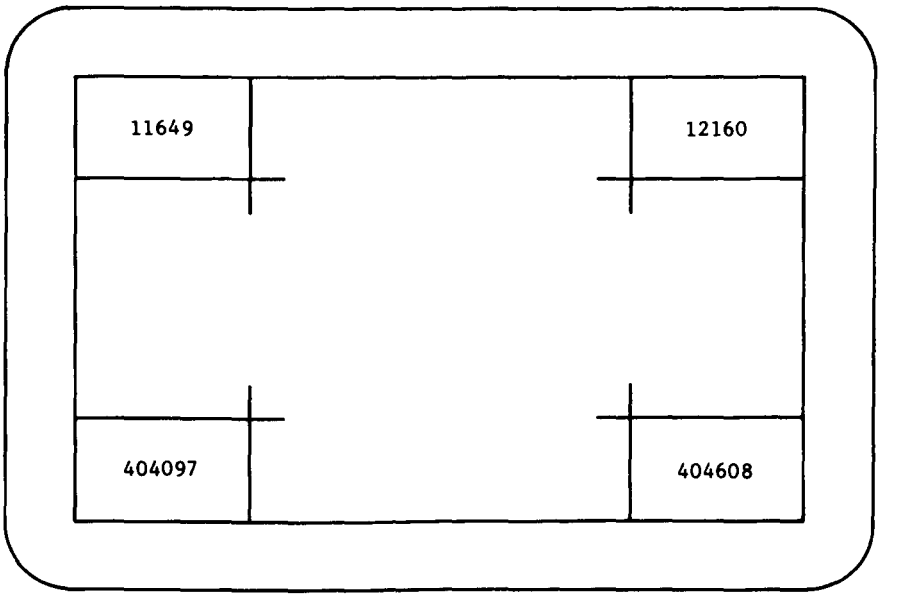


Fig. 2-8(b) Pixel Positions on the Screen

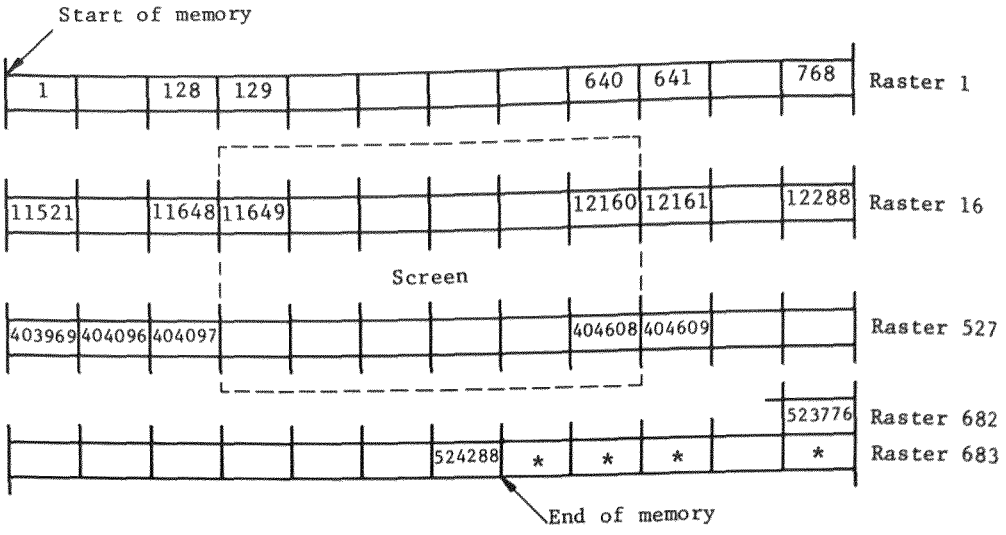
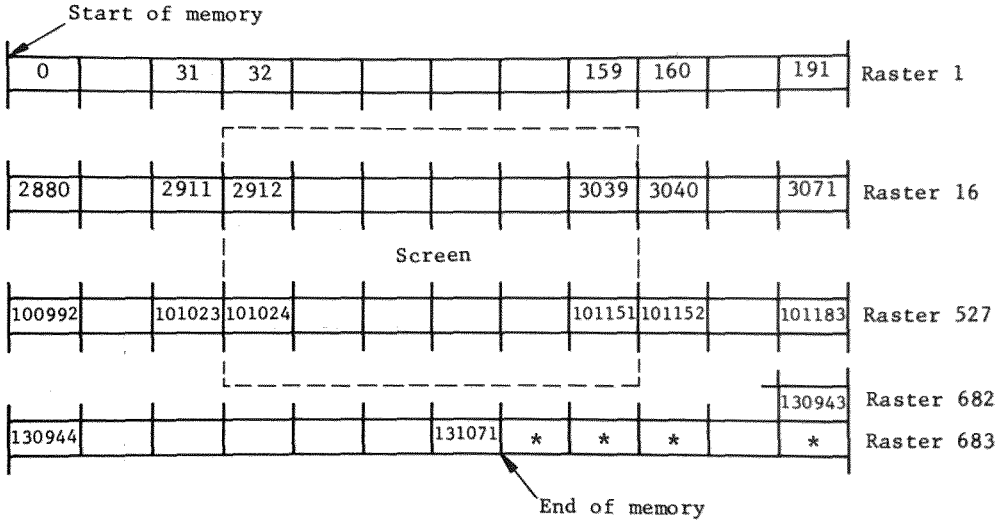


Fig. 2-8(c) Pixels within the Frame Buffer



Words within Frame Buffer, Decimal Address
4 Bits per Pixel, 4 Pixels per Word

Fig. 2-8(d) Pixels within the Frame Buffer

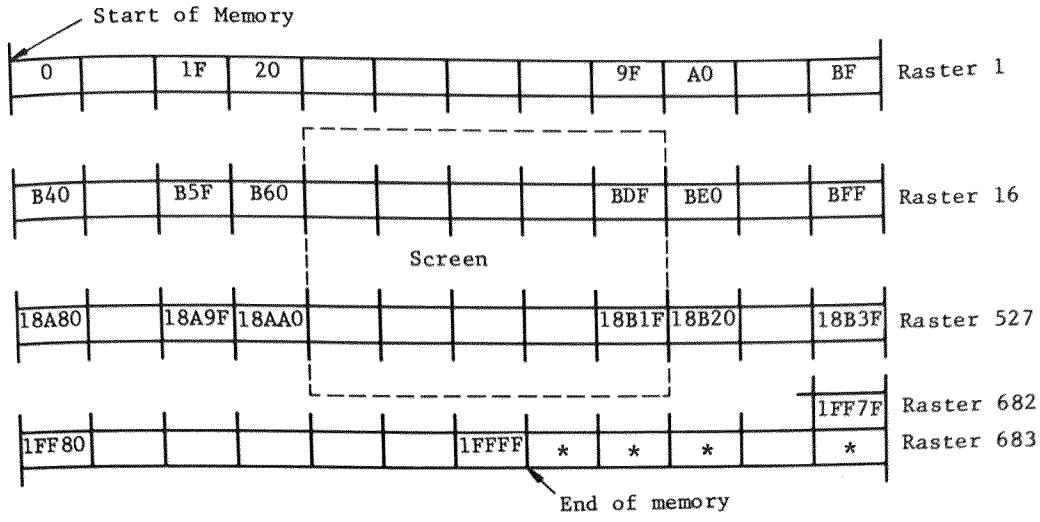


Fig. 2-8(e) Words within Frame Buffer, Hex Address
4 Bits per Pixel, 4 Pixels per Word

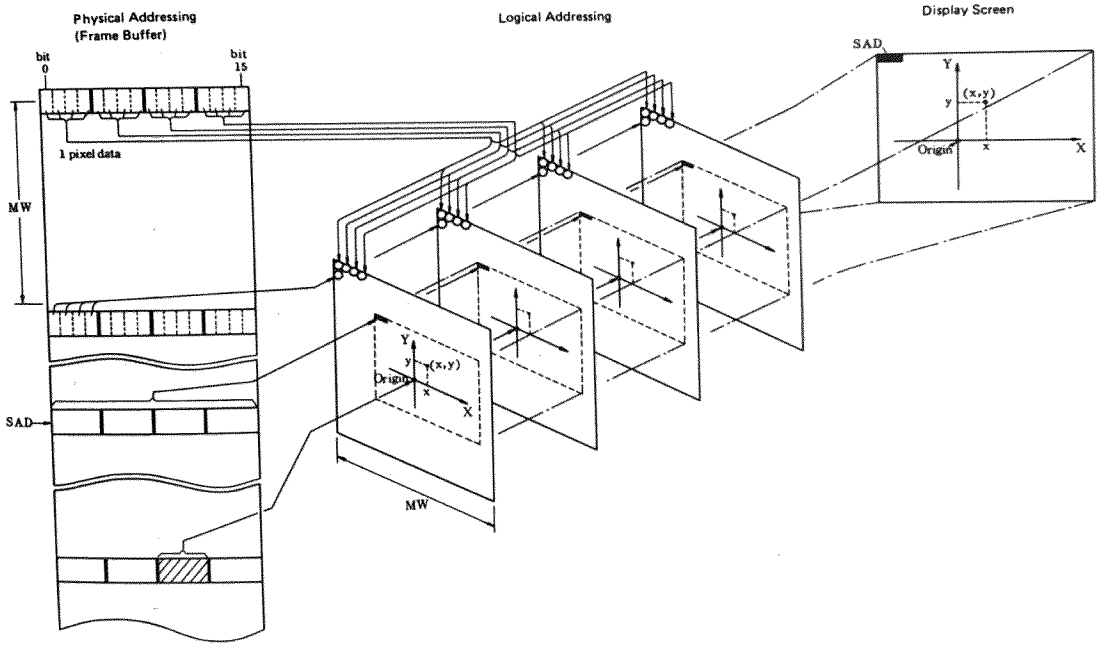


Fig. 2-9 Logical/Physical Addressing Example (4 bits per pixel)

2.6 Screens

The image produced on the display device need not necessarily come from one area of the frame buffer as the ACRTC supports multiple screens. The basic screen is called the BACKGROUND screen while an additional screen, called the WINDOW, can be defined that replaces it. The window screen size can be made equal or less than the background screen and its position is independent.

Further image flexibility is provided by allowing the background screen to be formed from up to three separete screens - split horizontally. These constituent parts are called the base screen, the upper screen and the lower screen. The width of these screens are all equal, however the vertical size and position are variable. This screen flexibility is demonstrated in Fig. 2-10 using pictorial representation.

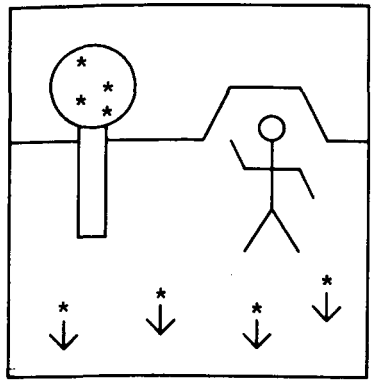


Fig. 2-10(a) Background

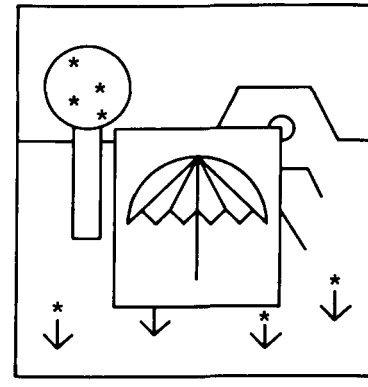


Fig. 2-10(b) Background with window

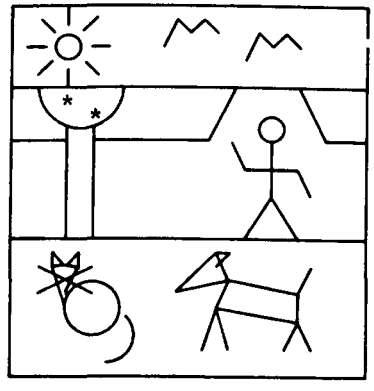


Fig. 2-10(c) Background split into upper, base and lower

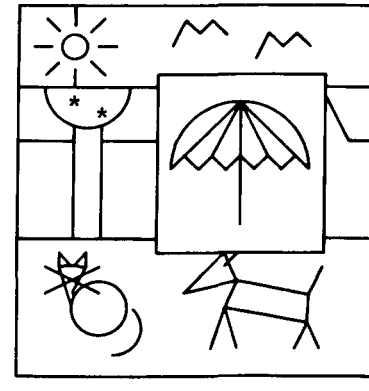
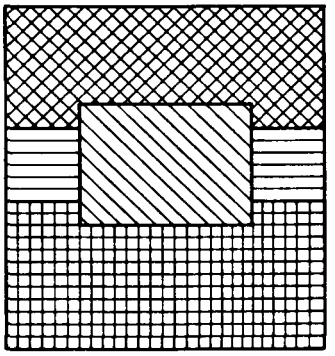


Fig. 2-10(d) Background split with window

Fig. 2-10 Screens

The data for each screen is held in separately defined areas within the frame buffer and the ACRTC handles the addressing of the appropriate locations to produce the correct complete image. Also the screen configuration can be dynamically altered.

The terminology relating to the various screens is set out below. Various terms are used and can lead to some confusion.



Base screen = Screen No. 1 = Split screen 0 (B)

Window screen = Screen No. 3 = Window (W)

Upper (split) screen = Screen No. 0 = Split screen 1 (B)

Lower (split) screen = Screen No. 2 = Split screen 2 (B)

UPPER (SPLIT) SCREEN = SCREEN No.0 = SPLIT SCREEN 1 (B)
BASE SCREEN = SCREEN No.1 = SPLIT SCREEN 0 (B)
LOWER (SPLIT) SCREEN = SCREEN No.2 = SPLIT SCREEN 2 (B)

WINDOW SCREEN = SCREEN No. 3 = WINDOW (W)

Key: B = Background screen
W = Window

Fig. 2-11 Split Screens and Window

A typical use for the background screens might be in graphics work station where the lower screen would show information about soft key functions. The upper screen would show a menu of facilities available and the base screen would be used as the main work area for the task in hand.

The window allows another area of additional information to be displayed over the top of the background. This could be used to provide a temporary HELP explanations or to show status information.

Rather than replace the background screens, the window can be superimposed. This additional mode adds flexibility as the window screen could hold text to be combined with graphics from the background screens.

2.7 Display Timing Signals

There are 3 signals involved with the timing of the raster display. A display monitor requires two pieces of timing information.

(1) Horizontal sync. - $\overline{\text{HSYNC}}$
(2) Vertical sync. - $\overline{\text{VSYNC}}$
(3) Display Timing Control - $\overline{\text{DISP1}}$ / $\overline{\text{DISP2}}$

$\overline{\text{HSYNC}}$ pulses on every raster and triggers the retrace of the electron beam from the righthand edge of the screen (the end of the previous raster) to the left-hand edge (the start of the next raster). It usually needs to last about 8% of the whole line period, dependent on the monitor in use.

$\overline{\text{VSYNC}}$ pulses on each field and triggers the retrace of the electron beam from the bottom of the screen (the end of the previous field). It usually needs to last about 6% of the field period again dependent on the particular monitor in use.

During both retrace periods the electron beam intensity must be kept to a minimum (blanked) to avoid spurious streaks on the screen.

In order to allow a margin for this the video signals are blanked just prior to the horizontal retrace process and unblanked a short time afterwards by $\overline{\text{DISP1}}$ or $\overline{\text{DISP2}}$ signal.

The margin before the HSYNC period is called the 'front porch' while that after HSYNC is the 'back porch'. Both expressions come from television terminology.

See Fig. 2-12

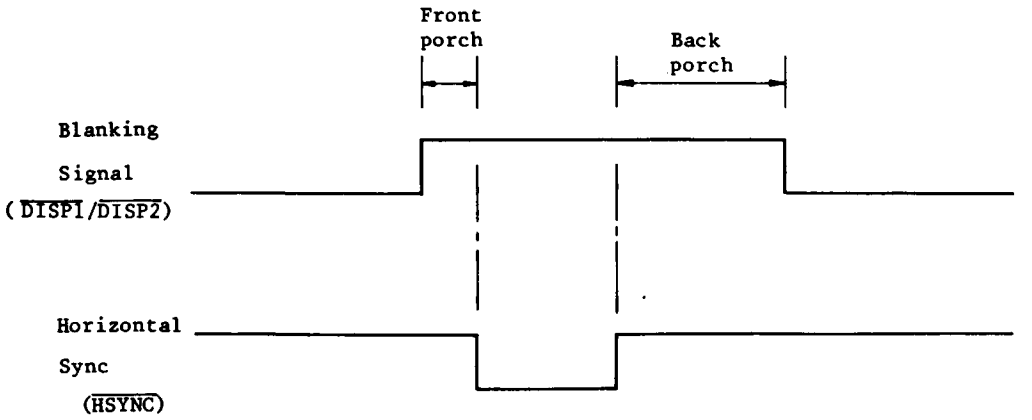


Fig. 2-12 Display Timing Signal

All the parameters that define when the active display starts and finishes are programmable.

3. THE FRAME BUFFER INTERFACE

3.1 The Access Modes

In order to create and modify images the frame buffer must not only be accessed for display purposes but also in order to read and write the various pixel locations.

These two types of cycles, DRAW and DISPLAY, compete for access to the frame buffer. The arbitration between these is a critical factor in the success of a system. The high data rates involved in generating a CRT raster image mean that even the fastest current NMOS DRAM devices are rather limiting as to what can be achieved. Faster device technologies are available e.g. bipolar and ECL, but these are prohibitively expensive.

Display Cycle (read only)

The ACRTC provides the address within the frame buffer at which the next pixel(s) to be displayed are located. The data value(s) are read from the frame buffer into the external parallel to serial converter (shift registers). The data is then shifted out to form the video signal(s) to the display monitor. One or more words of data may be obtained per access.

Draw Cycle (read or write)

(i) Read The ACRTC provides the word address, within the frame buffer, in which the pixel to be modified is located. The data word is read into the ACRTC where the relevant bits are modified according to the drawing operation used.

(ii) Write The ACRTC provides the word address, within the frame buffer, at which the data word is to be stored. The ACRTC outputs the data word, containing the new pixel information.

The ACRTC can support three access modes. These differ in their approach to achieving both a displayed image and a drawing facility. Drawing is always possible during the flyback periods (except during the refresh period) as no displaying takes place. The three modes are single, interleaved and superimposed. It is the manner in which they handle accesses during the display time that vary.

3.2 Single Access Mode

The basic access unit can be for either display or drawing purposes. It has two phases. During the first the address is output from the ACRTC. The activity in the second phase depends upon the purpose of the cycle. It a display access is being performed then the addressed data is read into the external video logic. When a drawing cycle is undertaken then data can either be read into the ACRTC or written from the ACRTC to the frame buffer.

The flyback periods are available for drawing as no displaying takes place. However, during the normal display time contention can exist. The relative priority between drawing and displaying will resolve these situations and is programmable. If displaying is the higher priority then the driving operation will be postponed until the flyback periods - reducing the drawing throughput. If drawing has a higher priority then the display process will be halted while the drawing is performed. The image will suffer from visual disturbances but the drawing will be faster. Fig. 3-1 shows frame buffer activity in single access mode.

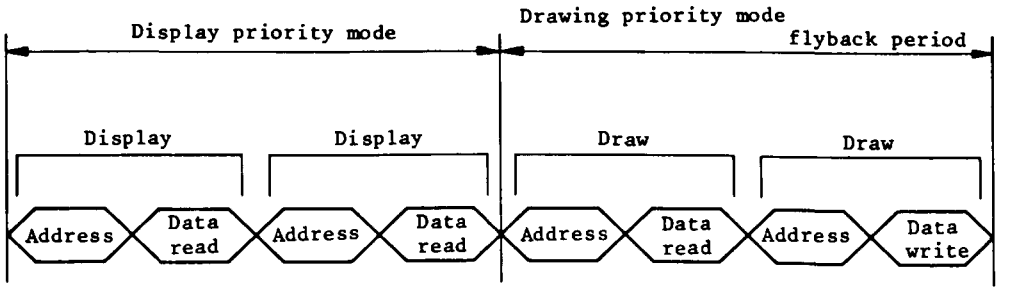


Fig. 3-1 Single Access Mode Frame Buffer Activity

3.3 Interleaved Access Mode

This is also called Dual Access Mode 0.

During the flyback periods no displaying takes place and continuous drawing capacity is available (less any DRAM refresh requirements).

The basic display unit is a display cycle composed of two accesses. The first is a display access and the second is a draw access. Each has two phases and is the same as for the single access mode. By alternating display accesses and drawing accesses the frame buffer is time multiplexed between these two function with the result that drawing throughput is increased considerably without the penalty of image disruption. This scheme however

does require that the frame buffer is accessed twice as often. Either the memory speed must be increased or the address increment size (GAI) doubled in order that twice the number of pixels are obtained during each display cycle, with repercussions as the amount of external logic needed.

Because it combines fast drawing capability with a stable image this mode is particularly useful and provides a clear improvement over earlier graphics devices which could only support single access mode operation. Fig. 3-2 shows frame buffer activity for interleaved access mode.

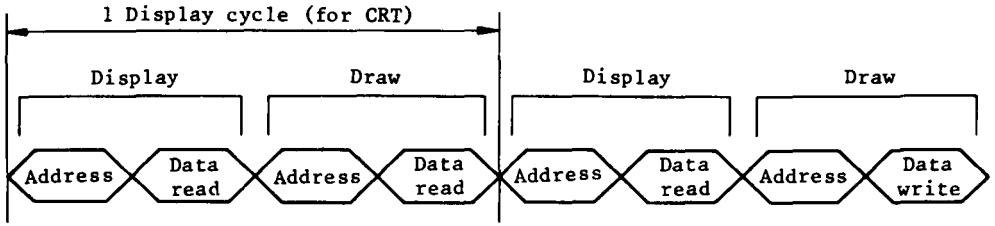


Fig. 3-2 Interleaved Access Mode Frame Buffer Activity

3.4 Superimposed Access Mode

This is also called Dual Access Mode 1. As before the flyback periods provide for drawing and DRAM refresh is necessary.

The mode is unusual in that it is used to generate two separate images that are combined together (superimposed) to produce the one image displayed. A typical use would be overlay text information on a graphical image.

The two images generated are the background screen and the window screen.

The basic display unit is a display cycle which has two accesses, both for display purposes. Each access has two phases - the same as for single mode.

The frame buffer is time multiplexed between the two screens. The first access obtains the data for the background screen and the second obtains it for the window screen.

The data for the screens needs to be temporarily held until it is ready for combining and sending to the display monitor. This requires additional external logic (latches) before the video logic. Combination can occur before or after the parallel to serial conversion.

If the window screen is defined as smaller than the background screen then there will be periods during the display process when window display accesses are not necessary. Rather than waste these opportunities they are available as additional drawing cycles. Under these conditions this mode acts like the Interleaved Mode. Fig. 3-3 shows frame buffer activity for superimposed access mode.

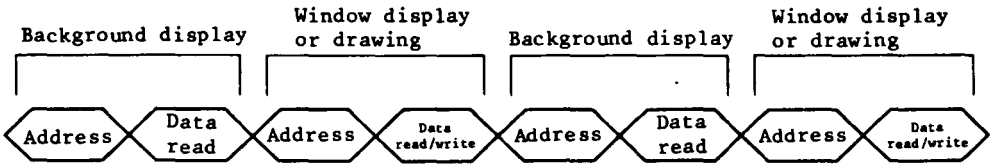


Fig. 3-3 Superimposed Access Mode Frame Buffer Activity

3.5 Graphics and Characters

With graphics display the data from the frame buffer represents pixel valves, which together form the image to be viewed. This data is passed through the video shift registers and then on to the display monitor.

An additional function supported by the ACRTC is for a Text display. Text is composed of characters that are a set of well-defined symbols. To reduce the storage requirements and simply reproduce symbols repetitively, a different approach is possible. Rather than store and manipulate pixels, character codes can be used. These are read when displaying and a look up table (character generator) used to obtain the pixel values which are then passed to a parallel to serial converter (P/S).

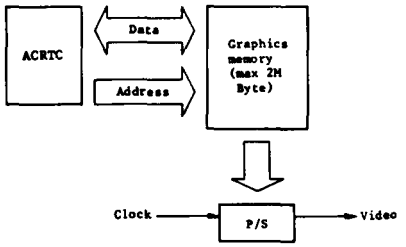


Fig. 3-4(a) Graphics

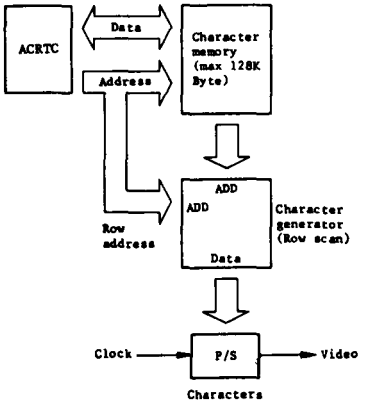


Fig. 3-4(b) Characters

Fig. 3-4 Graphics and Characters

The character generator is usually a ROM, though using a RAM allows the character symbols to be redefined.

The ACRTC can address 64K words of character memory by using 16 address lines. Five address lines are also provided that indicate the raster count. These are used as additional inputs to the character generator so that the pixels for the current raster of the character are output. The five lines allow characters composed of up to 32 rasters to be supported. This is sufficient for even the demanding KANJI (Japanese) characters.

A screen can be defined as sourced from either character or graphics memory. In fact the two memorys need not be physically separate as it is the routing of the display data that differs. The ACRTC provides a signal that indicates whether the current screen being accessed is defined as graphics or character. This may be used to enable the appropriate memory banks and/or modify the data routing so that when characters are being displayed the character generator is interposed between the memory and the video logic.

Attention must be paid to the amount of data produced during an access to the graphics memory and that for the character memory. The GAI facility only applies to graphics memory hence only one word is obtained per character display access. This could be used to provide one 7 bit ASCII character with 9 bits of attributes like color and intensity or two 7 bit ASCII characters with 2 bits of attributes each. If each character code provided 8 pixels an access would yield at most 16 pixels. This would only be compatable with the graphics function if it too generated 16 pixels per display access e.g. GAI =+4 and 4 bits per pixel.

If the two schemes are incompatible the video clock may be modified or more commonly separate parallel to serial converts would be used and the outputs combined to produce the video signal.

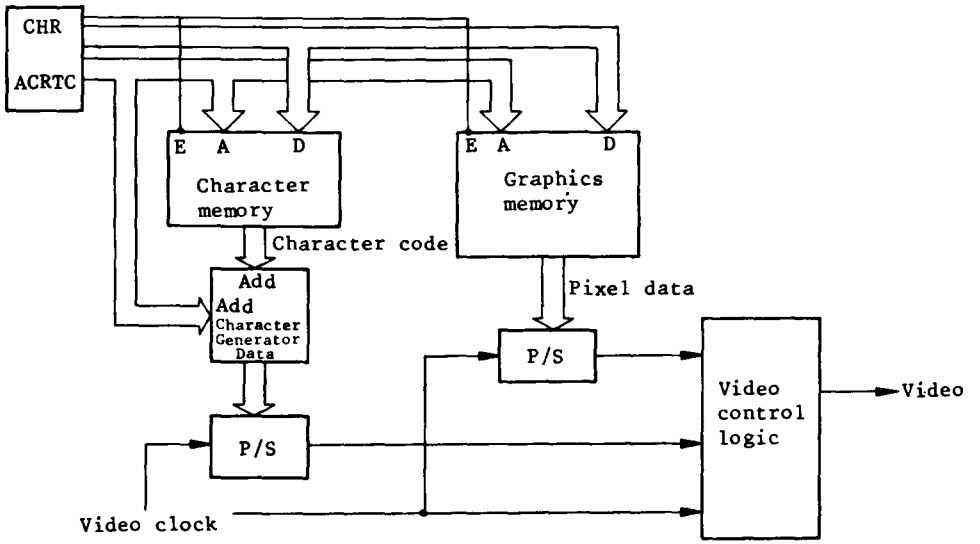


Fig. 3-5 Graphics and Characters

The host CPU must prepare the text information and transfer it to the ACRTC. The ACRTC passes word values from the host CPU to the character memory. It has no text manipulation facilities. It can however modify the displayed image by scrolling and zooming or repositioning.

Another approach to text displays is possible. This method does not use the character memory and character generator. Instead the graphics memory is used. The host CPU forms the character set needed via the ACRTC into the frame buffer in an area that will not be displayed. When text is needed, the ACRTC is used to copy the relevant characters from this original set into the necessary areas of the displayed screen, so building up the text. The small loss of usable frame buffer is more than off set by the greater flexibility and reduced external logic.

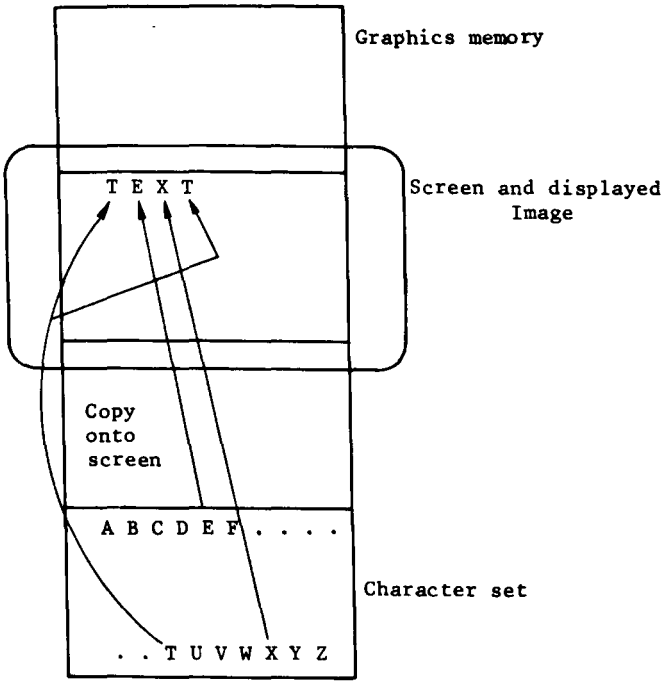


Fig. 3-6 Characters VIA Graphics

3.6 System Configuration

The ACRTC can support two memory areas, one is the graphics memory (also called the frame buffer) and the other is the character memory which is used with a character generator. Both of these are covered by the general term display memory.

Communication between the ACRTC and the display memory is via a 20 bits bus, of which 16 bits are time multiplexed address/data bus. Memory addresses are presented on these 20 signal lines and data is transferred on the low order 16 as the ACRTC reads and writes one word per access. Because of the multiplexing it is necessary to externally latch the low order 16 bits of address. The 20 bit address permits the ACRTC to manage a memory space of 1M words (2M bytes). This applies to the case of the graphics memory. For the character memory only a 16 bit address is used providing for a character memory of 64K words (128K Bytes). The additional 4 signal lines RS0 ~ RA3 carry the character row count value that is applied to the character generator in order that the correct pixel information is extracted. an extra row count signal (RA4) is provided to bring the total to 5, permitting characters with 32 rows to be formed.

A character memory/graphics memory select signal indicates which is presently in use.

The address/data signals are : MAD0 - MAD15
The address/row signals are : MA16/RA0 - MA19/RA3
The extra row signal are : RA4
The character/graphics signal is : CHR

3.7 System Examples

The external logic functions required vary according to system demands, however the three basic configurations will be considered:

(1) Graphics memory only
(2) Character memory only
(3) Combined graphics and character memory

Text can be produced using a character generator by methods (2) and (3). With configuration (1) characters can be formed using the method outlined in Section that is "bit mapped characters".

(1) Graphics Memory Only

The low order 16 bits of address are latched and together with the others form a 20 bit address bus that is applied to the graphics memory. Bi-directional data buffers transfer data between the ACRTC and the memory for drawing purposes. Display data obtained from the memory is loaded into the serial to parallel converter and clocked out to the display monitor. If more than 16 bits are obtained from the memory then a multiplexer is necessary in the data path to the ACRTC.

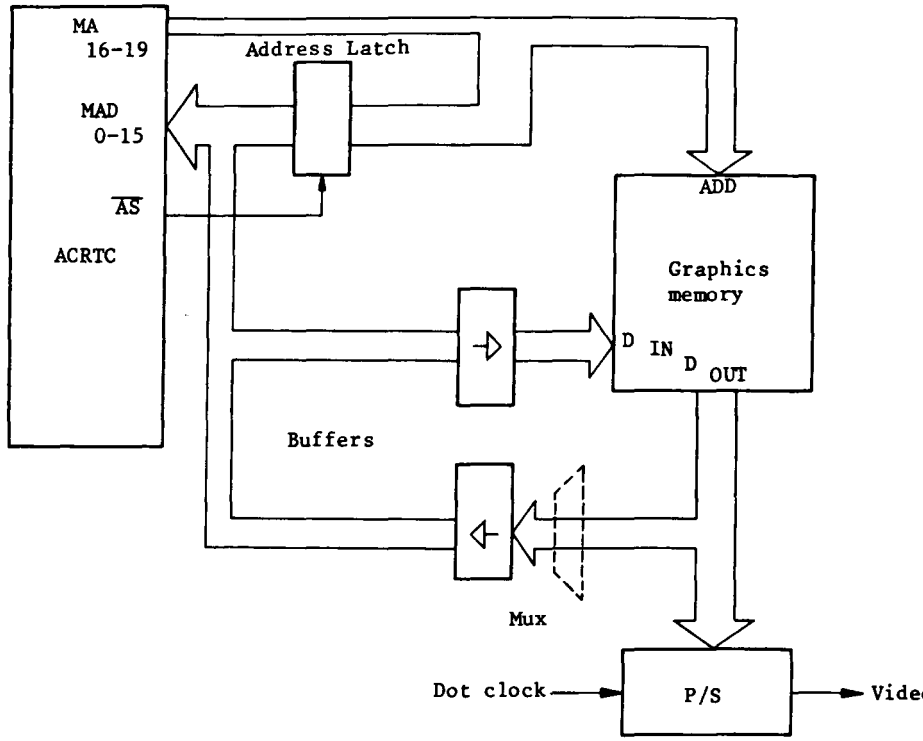


Fig. 3-7 Graphics Memory Only

(2) Character Memory Only

The 16 bit address is latched and applied to the character memory. The data obtained is applied to address inputs of a character generator and so are the 5 character row count signals. Some of the character memory data may represent character attributes and so be used to modify the video signals. The pixel data obtained from the character generator is loaded into the display monitor.

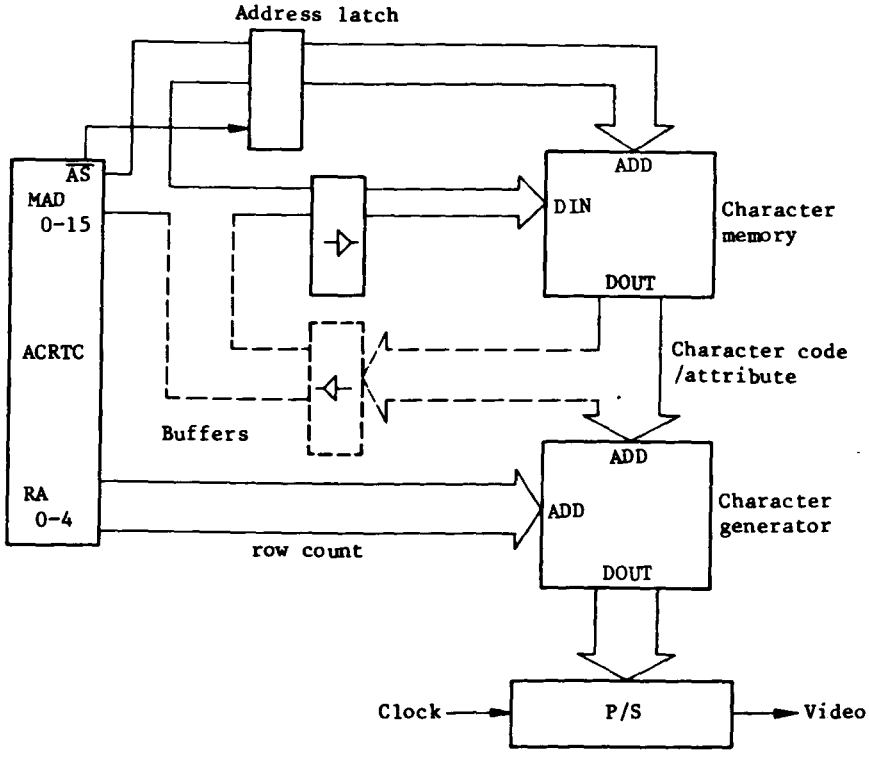


Fig. 3-8 Character Memory Only

(3) Graphics and Character Memory

Because some of the requirements are duplicated the external logic is reduced when both Graphics and Character memory are combined. However there is still additional logic over the graphics only system and the "bit mapped characters" approach can avoid this burden. See section 3.5, last paragraph.

If the pixel rates are not compatible then separate video logic may be required for graphics and characters. This can add considerably to external logic.

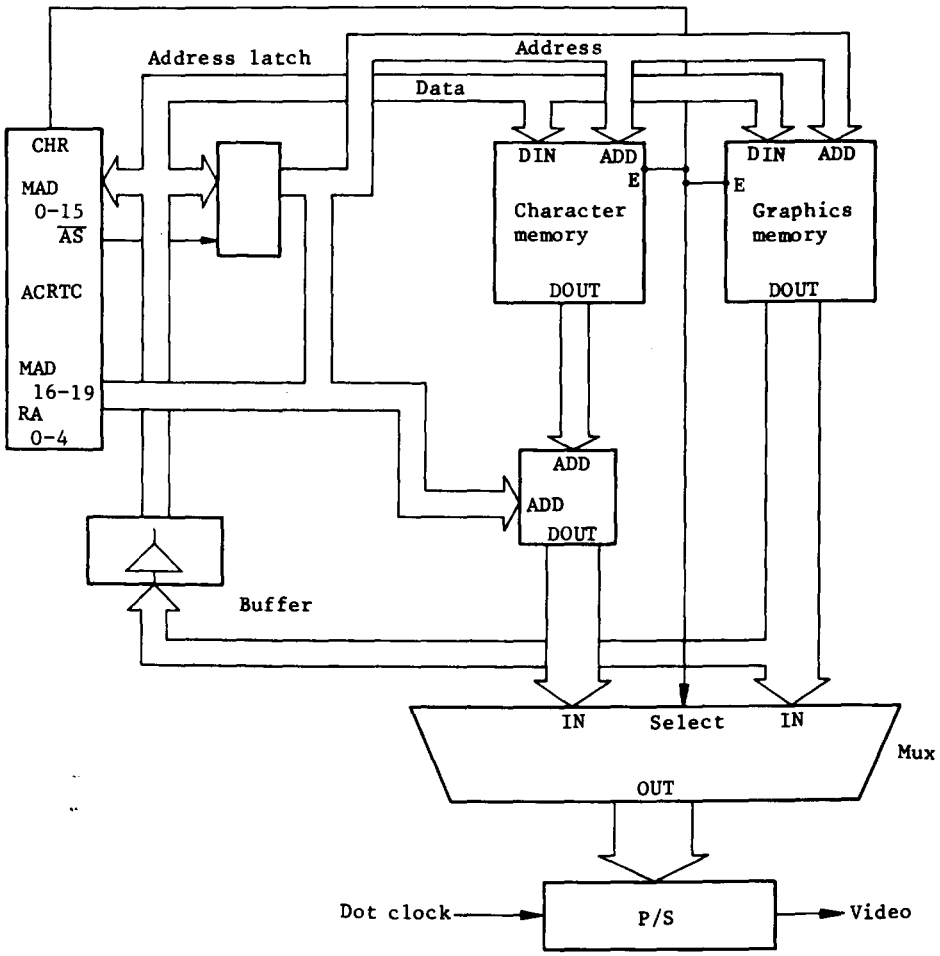


Fig. 3-9 Graphics and Character Memory

3.8 A More Detailed Example of a Graphics System

Consider an application that requires 128K words of frame buffer storage and that uses 4 bits per pixel in order to deriver red, green, blue, and intensity signals (RGBY) for a color monitor. Because of the DRAM cycle time it needs to output 8 pixels per display cycle i.e. 32 bits (hence GAI = +2) in order to meet the required pixel rate.

The system implementation of this would look like Fig. 3-10.

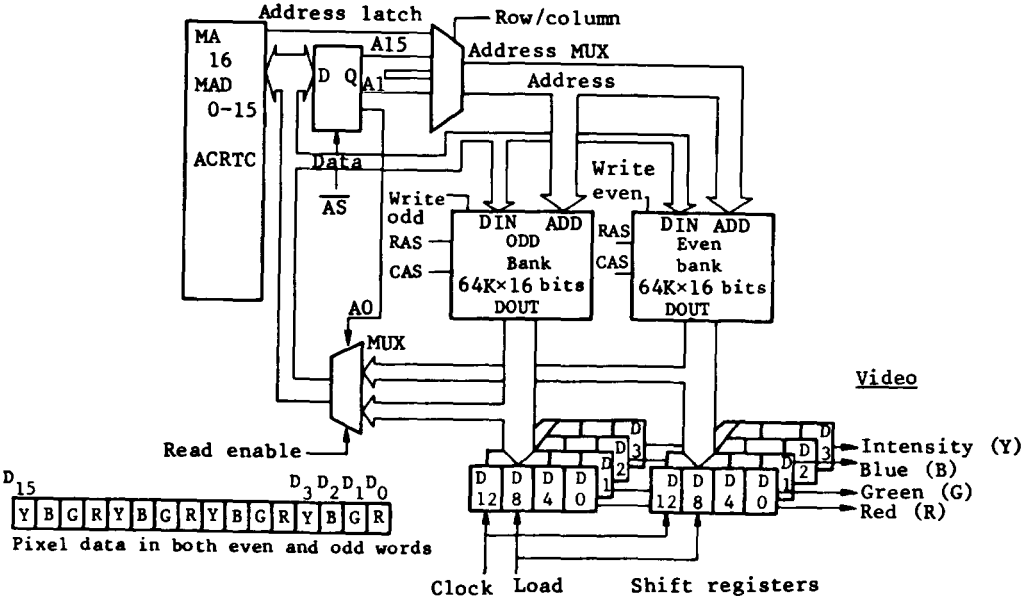


Fig. 3-10 A More Detailed Example of a Graphic System

The memory is constructed from two banks of 64K × 1 bit DRAMs, each of 16 devices. This arrangement permits 32 bits of data to be read during a display cycle, while 16 bit transfers are used for drawing. Any read access, display or draw produces 32 bits of data from the frame buffer. The LSB of address, A0 is not used within the memory array, instead it selects the odd or even word to be read by the ACRTC via the multiplexer. When writing it is used to write enable to the appropriate odd or even memory bank.

The pixels are packed 4 per word as nibbles with the red, green, blue, and intensity values as bits 0, 1, 2 and 3 respectively. During display, these control the electron guns of the monitor providing for 7 colors (including white each with two brightness levels and black.)

The 32 bits of pixel data are loaded into 4 of 8 bit shift registers, such that all the data bits relating to a given pixel attribute are together in one of these shift registers, e.g. shift register 0 has all the ''RED'' data bits, that is, bits 0, 4, 8 and 12 from both the odd and even words that have been accessed.

The pixel clock (or video clock or dot clock) runs at the pixel rate of the system and shifts the pixel values out of these registers. The outputs of the registers are the drive signals for the display monitor, i.e. the 'RGBY'.

3.9 Display Memory Timing

The ACRTC address and data bus communicate with the frame buffer with the aid of a number of control signals. These signals organize the timing of the transfers between the three major parts of a display subsystem.

(1) THE ACRTC
(ii) THE DISPLAY MEMORY
(iii) THE VIDEO LOGIC

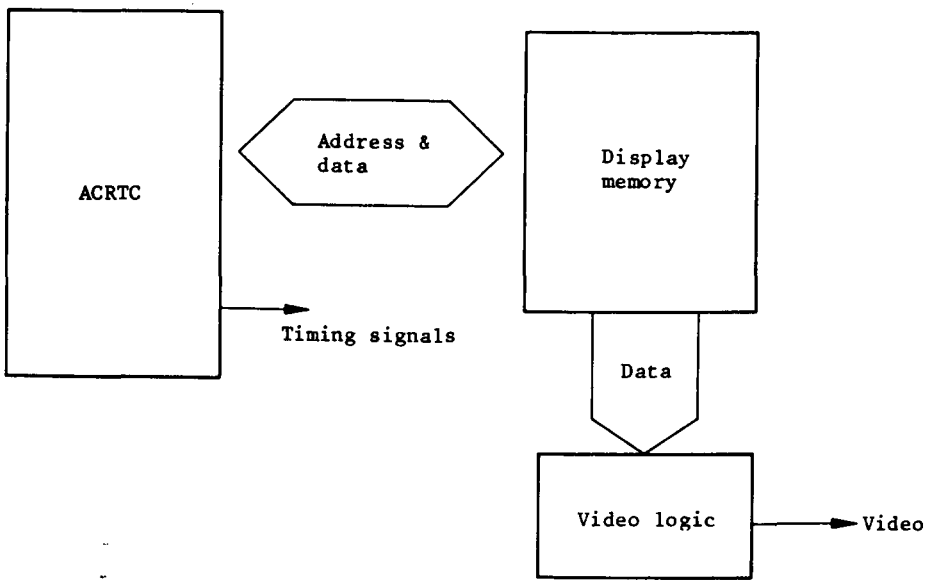


Fig. 3-11 Display Circuit

The fundamental timing is from the ACRTC clock input called 2CLK because it runs at twice the rate of the derived memory access timing signal MCYC. The address strobe $\overline{AS}$ forms the third timing signal. These are all related as shown in Fig. 3-12.

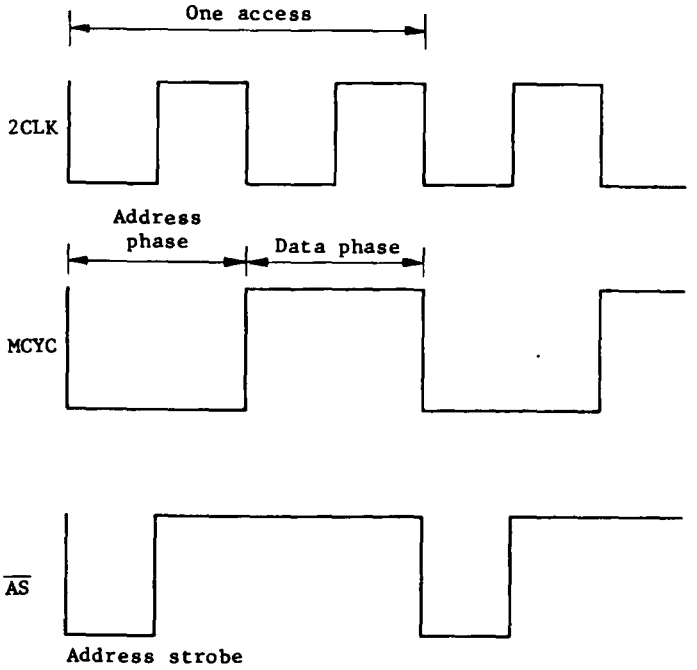


Fig. 3-12 Display Timing Signal

A frame buffer access lasts for two cycles of the 2CLK. During the first cycle the address is output and MCYC is low. $\overline{AS}$ becomes low at the start and returns high half way through this first cycle by which point the address is stable.

During the second cycle the data is transferred. If it is being written then the ACRTC provides the data on MAD 0-15. If it is being read these lines act as inputs. If a display read is in progress then MAD 0-15 are high impedance and ignored. The frame buffer data is loaded into the shift register.

In summary the fundamental clock is the 2CLK signal. From this the ACRTC generates all its timing. A half frequency version of this (MCYC) is available that indicates whether the present transaction is in the address or data phase. A strobe $\overline{AS}$ indicates when the address is stable.

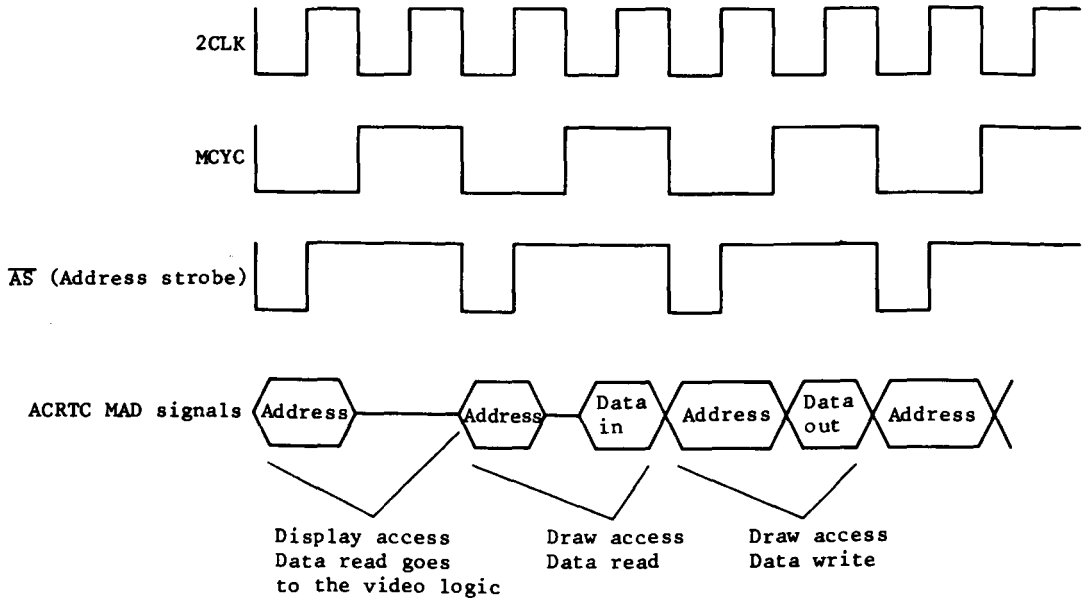


Fig. 3-13 Display Timing Signal

3.10 Display Memory Status Signals

The purpose of the current frame buffer access is indicated by 3 status signals from ACRTC:

(i) Memory Read : MRD
(ii) DRAW : $\overline{\text{DRAW}}$
(iii) CHARACTER : CHR

The names indicate the purpose to which they are generally put, however because of the many different types of cycle performed by the ACRTC the memory read signal assumes a low state during a window read access in superimpose mode. This ambiguity is due to encoding limitations.

Generally then the memory read signal indicates if the contents of the display memory will be read or written. Read cycles can be for both display and drawing purposes while writing will only apply to drawing.

Discrimination between drawing and displaying is provided by the DRAW signal.

Selection between the two display memory spaces supported, character memory and graphics memory, is done by the CHARACTER signal.

Table 3-1 indicates these states and the ACRTC PIN.

Table 3-1 Display Memory Status Signals

ACRTC PIN	SIGNAL	MNEMONIC	STATE	FUNCTION
PIN 55	MEMORY READ	MRD	LOW	DISPLAY MEMORY WRITE (OR WINDOW READ IN SUPERIMPOSED MODE)
			HIGH	DISPLAY MEMORY READ
PIN 54	DRAW	$\overline{\text{DRAW}}$	LOW	DRAWING ACCESS TO DISPLAY MEMORY
			HIGH	DISPLAY ACCESS TO DISPLAY MEMORY
PIN 56	CHARACTER	CHR	LOW	GRAPHIC MEMORY ACCESS
			HIGH	CHARACTER MEMORY ACCESS

3.11 DRAM Refresh

Dynamic RAM's offer very low 'cost per bit' storage and so they often form the basis of the display memory. Their main drawback is that they need periodically refreshing in order that their constants do not become corrupt. To support this function the ACRTC can be programmed to carry out DRAM refresh cycles during the HSYNC period. As no displaying takes place during the horizontal flyback period the effect is to reduce the time available for drawing.

The horizontal flyback period has 3 components.

(1) FRONT PORCH TYP. 20%
(2) HSYNC PERIOD TYP. 40%
(3) BACK PORCH TYP. 40%

Generally only 40% max is lost for HSYNC, the horizontal sync period. It is dependent upon the display monitor requirements and the screen configuration as when the display area is less than the potential CRT raster area, the porches will be extended to form a border.

During this period, successive refresh cycles are carried out on the Display Memory. It is necessary that sufficient cycles are performed to ensure that all the DRAM devices are adequately refreshed within their refresh period of either 128 cycles at 2mns or 256 cycles at 4mns.

The horizontal scan parameters programmed into the ACRTC use the units of memory cycles, that is half the ACRTC clock frequency 2CLK.

The HSYNC period is specified as the horizontal SYNC width HSW, a 5 bit field within the horizontal sync register. It can accept values between 2 and 31 cycles.

The number of refresh cycles performed per raster will equal HSW. If the scan frequency of the system is F_h then.

$$HSW \times F_h = \text{No. of refresh cycles per second performed}$$

However, this must equal or exceed the requirements of the DRAM devices used for the display memory. If these are N refresh cycles every t_r seconds then

$$\frac{N}{t_r} = \text{No. of refresh cycles per second performed}$$

Equating these gives the limit for the usable value for HSW.

$$HSW \geq \frac{N}{t_r}$$

Where HSW = Sync width in memory cycles
N = DRAM refresh requirements in cycles
F_h = Scan frequency of system in kHz

For example if a system has a scan frequency of 31.25 kHz (625 lines non-interlaced, 50 kHz frame rate) and uses 256K × 1 bit DRAM's type HITACHI HM50256 with 256 cycles/4 ms for refresh, what is the minimum HSW usable?

$$HSW \geq \frac{N}{t_r \times F_h}$$

here $N = 256$
$t_r = 4$
$F_h = 31.25$

$$HSW \geq \frac{256}{4 \times 31.25} = 2.05$$

Therefore the minimum HSW value that can be set is 3.

The factor that determines the setting of HSW is the horizontal sync period required by the monitor in use. This is often dominant. See Section 2.7

3.12 Video Attributes

During each horizontal flyback period, when no displaying is being done, the ACRTC uses the 20 address and data signal lines to output 20 bits of video attributes. This data needs to be latched externally. It provides status and control information that can be used to modify the next displayed raster. 8 of these bits are uncommitted and are completely free for any user defined purposes. They can be written via an internal ACRTC register and are output at the start of each raster.

Two four bit fields give the current values associated with the horizontal zoom and smooth scroll functions. These can be used by external timing circuits to condition the video signals such that the displayed image is manipulated.
A two-bit encoded field indicates which background screen(s) are presently being displayed if any.

A blink facility is provided on the remaining two bits. Each can be set to toggle periodically, the frequency being programmed via the Blink Control Register (BCR) of the ACRTC. External logic can make use of these to produce screen effects like blinking characters or cursors.

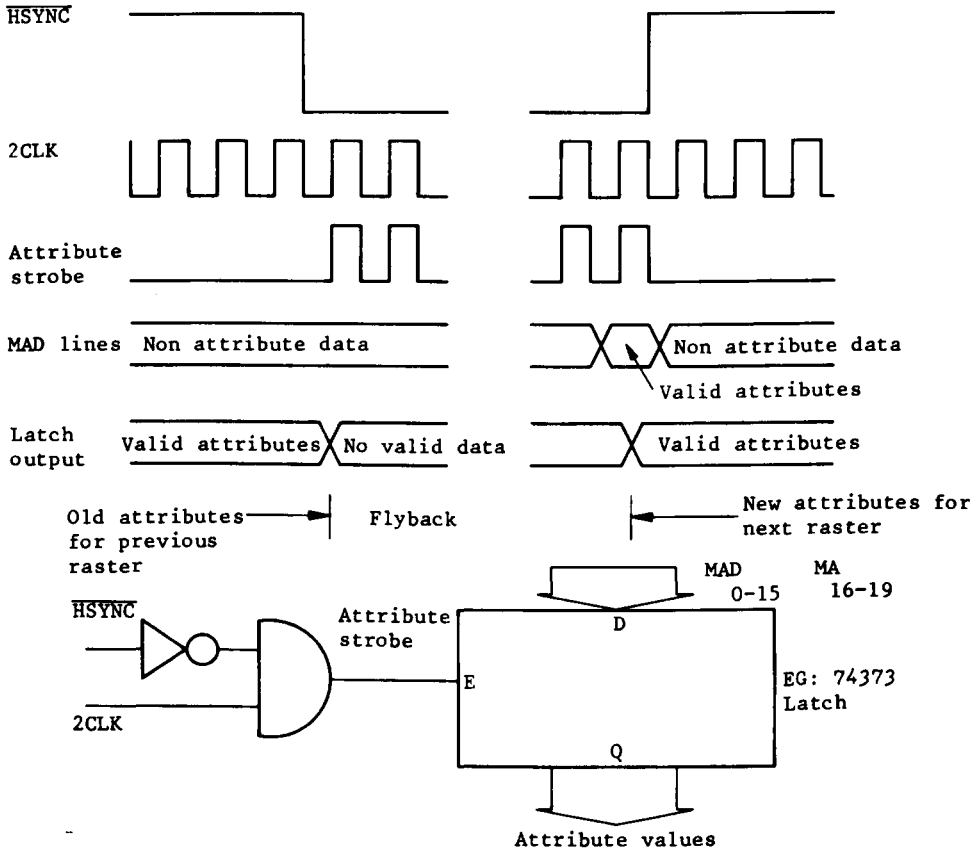


Fig. 3-14 Video Attributes Control

The video attributes are output as the last cycle prior to the $\overline{\text{HSYNC}}$ signal going high. The external latches should be strobed repeatedly by 2CLK during the low period of HSYNC but not when it is high. The valid attribute information will be captured and retained for use during the subsequent raster. Any invalid data temporarily latched during the earlier part of the HSYNC period will be of no consequence as it will be replaced.

A blanking signal is available from the ACRTC which is false during the active display time and true when the display should be blanked.

In fact the background and window screens can have separate blanking signals associated with them.

The two display signals are $\overline{\text{DISP1}}$ and $\overline{\text{DISP2}}$, both active low. They can be configured in two ways as Table 3-2 shows.

CONFIGURATION	$\overline{DISP1}$ Function	$\overline{DISP2}$ Function
0	Combined Horizontal and Vertical Display of Background screen	Combined Horizontal and Vertical Display of Window screen
1	Combined Horizontal Display of both Background and Window screens.	Vertical Display of both Background and Window screens

Table 3-2 Timing Control Ram

REG NO.	REGISTER	MNEMONIC	BITS	REG	MNEMONIC	NOTES
R80	Raster Count	RCR	11~0	Raster Cycle	RC	(1)
R82	Horizontal Sync	HSR	15~8 4~0	Horizontal Cycle Horizontal Sync. Width	HC HSW	(2)
R84	Horizontal Display	HDR	15~8 7~0	Horizontal Display Start Horizontal Display Width	HDS HDW	(2) (2)
R86	Vertical Sync.	VSR	11~0	Vertical Cycle	VC	
R88	Vertical Display	VDR	15~8 4~0	Vertical Display Start Vertical Sync. Width	VDS VSW	
R8A R8C R8E	Split Screen Width	SSW	11~0 11~0 11~0	Split Screen 1 Width Split Screen 0 Width Split Screen 2 Width	SP1 SP0 SP2	 (3) (3)
R90	Blink Control	BCR	15~12 11~8 7~4 3~0	Blink On 1 Blink Off 1 Blink On 2 Blink Off 2	BON1 BOFF1 BON2 BOFF2	(4) (4) (4) (4)
R92	Horizontal Width Display	HWR	15~8 7~0	Horizontal Window Start Horizontal Window Width	HWS HWW	(3)(2) (3)(2)
R94 R96	Vertical Window Display	VWR	11~0 11~0	Vertical Window Start Vertical Window Width	VWS VWW	(3) (3)
R98 R9A R9C	Graphic Cursor	GCR	15~8 7~0 11~0 11~0	Cursor X End Cursor X Start Cursor Y Start Cursor Y End	CXE CXS CYS CYE	(4) (4) (4) (4)

Notes:
(1) A Read Only Register
(2) The Load Value is one less than the Required Value
(3) Need only define if particular screen is enabled
(4) Need only define if function is to be used, otherwise = X

When low these signals indicate active display time while when high they indicate a blanking period.

The configuration mode (0 or 1) is set via the Display Control Register (Bit 15) of the ACRTC.

Mode 0 separates the active display periods of the Background and Window screens. The particular background screen being display is specified by the video attributes code on an individual raster basis, while $\overline{DISP1}$ and $\overline{DISP2}$ allow identification of the active screen (background or window) within a raster.

In Mode 1, $\overline{DISP1}$ can be used to blank the video signals to the monitor. $\overline{DISP2}$ indicates when the display memory is free for the relatively long period of the vertical retrace. This can be useful if another device is to gain direct access to the memory.

The duration of the sync period is set by the 5 bit horizontal sync width value programmed into ACRTC.

The selection of the value is determined by two factors

(1) DRAM REFRESH REQUIREMENTS
(2) DISPLAY MONITOR REQUIREMENTS

The DRAM considerations are dealt with in Section 3.11.

A typical display monitor may need a flyback period of about 20% of the whole period of which 40% will be for the HSYNC pulse and the rest will be the front and back porches. HSYNC therefore is about 8% of the line period.

Using this requirement we can obtain the following rule of thumb to satisfy the monitor requirements:

$$HSW \geq \frac{40 \times F_{2CLK}}{F_h}$$ where F_{2CLK} = The ACRTC clock frequency in MHz
F_h = system scan frequency in kHz

For example, if the line frequency is 31.25 kHz and the ACRTC is clocked at 5 MHz, what sync width value is required?

$$HSW \geq \frac{40 \times 5}{31.25} = 6.4$$

The nearest value would be 7.

The result can be compared with that produced by considering the DRAM refresh needs; i.e., typically about 2. The monitor requirements obviously dominate in the choice of HSW value in this system. This will often be the case unless the ACRTC is clocked at a low frequency and/or the scan frequency is very high.

By using the value of 7, the DRAM refresh need will be exceeded by a factor of approximately 3.

3.13 Dummy Cycle

There are occasions when the display memory becomes available for transactions, but none are required.

With a horizontally zoomed display the pixels are 'stretched' by reducing the frequency of the pixel clock. Because the pixel rate is reduced the memory need not be accessed so often for display purposes. The ACRTC skips the extra display cycles that are no longer needed by doing 'dummy cycles'. These are easily identified because there is no address strobe ($\overline{AS}$) at the start. No address is provided and the transaction should not affect the display process in particular the Parallel to Serial converter should not be reloaded. Apart from the running strobe the transaction appears like a display read cycle of address 0.

When drawing time is available but has not been requested, i.e. all drawing commands have been completed or the current drawing command involves complex pixel computation (e.g. arc, PAINT) and the result is not yet available then a dummy cycle is performed.

Again this has no address strobe ($\overline{AS}$) but otherwise appears as a drawing read cycle of address 0. These cycles can be ignored as any data obtained is disregarded by the ACRTC.

4. SYSTEM DESIGN - DISPLAY MEMORY

4.1 Transactions

The ACRTC communicates with the display memory using 20 address bits and 16 data bits.

A number of timing and status signals are available that indicate the progress and purpose of the current transfer. These are used with the external logic to route data between the elements of the system.

A display system can have 7 major elements:

(1) ACRTC
(2) ADDRESS LATCH
(3) ATTRIBUTES LATCH
(4) GRAPHICS MEMORY
(5) CHARACTER MEMORY
(6) CHARACTER GENERATOR
(7) PARALLEL TO SERIAL CONVERTOR

Elements (5) and (6) are only required for character generation other than by a bit-mapped graphics method. See Section 3.5

The block diagram for this system is shown below.

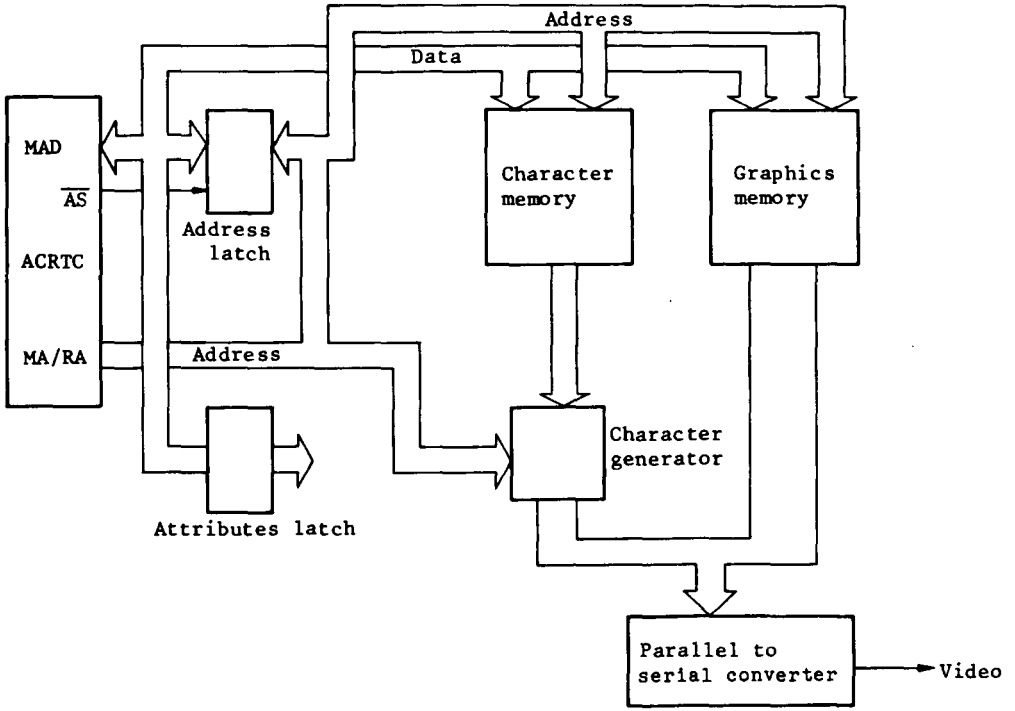


Fig. 4-1 Display Memory Control Circuit

Information is routed between these elements.

In all there are 24 different transaction types that can be performed.

The ACRTC provides the signals needed to manage these and Table 4-1 sets out the conditions that apply to each type.

Table 4-1 Display/Draw Transaction Types

	TRANSACTION				INFORMATION ROUTE	ACCESS MODE	RA0~4(7) MA16~19	MAD0~15	2 CLK	MCYC	$\overline{AS}$	MRD	$\overline{DRAW}$	CHR	$\overline{HSYNC}$	$\overline{DISP}$ 1	$\overline{DISP}$ 2
1	DISPLAY	ADDRESS	GRAPHICS		ACRTC>ADDLTH>GRCMEM	ALL	ADD16~19	ADD0~15	0	0	0	1	1	0	1	X	X
2	DISPLAY	DATA	GRAPHICS		GRCMEM>P/S CON	ALL	ADD16~19	Hi-Z	1	1	1	1	1	0	1	X	X
3	DISPLAY	ADDRESS	CHR		ACRTC>ADDLTH>CHRMEM	ALL	(2)	ADD0~15	0	0	0	1	1	1	1	X	X
4	DISPLAY	DATA	CHR		CHRMEM>CHRGEN>P/S CON	ALL	RA0~4	Hi-Z	1	1	1	1	1	1	1	X	X
5	DRAW	ADDRESS	GRAPHICS		ACRTC>ADDLTH>GRCMEM	ALL	ADD16~19	ADD0~15	0	0	0	X(1)	0	0	X	X	X
6	DRAW	DATA	GRAPHICS	READ	GRCMEM>ACRTC	ALL	ADD16~19	Din0~15	X	1	1	1	0	1	X	X	X
7	DRAW	DATA	GRAPHICS	WRITE	ACRTC>GRCMEM	ALL	ADD16~19	Dout~15	X	1	1	0	0	1	X	X	X
8	DRAW	ADDRESS	CHR		ACRTC>ADDLTH>CHRMEM	ALL	(2)	ADD0~15	0	0	0	X(1)	0	1	X	X	X
9	DRAW	DATA	CHR	READ	CHRMEM>ACRTC	ALL	(2)	Din0~15	X	1	1	1	0	1	X	X	X
10	DRAW	DATA	CHR	WRITE	ACRTC>CHRMEM	ALL	(2)	Dout0~15	X	1	1	0	0	1	X	X	X
11	DISPLAY	ADDRESS	GRAPHICS		ACRTC>ADDLTH>GRCMEM	DAI	ADD16~19	ADD0~15	0	0	0	1	1	0	1	X	X
12	DISPLAY	DATA	GRAPHICS	BKGND	GRCMEM>P/S CON (3)	DAI	ADD16~19	Hi-Z	1	1	1	1	1	0	1	X	1(3)
13	DISPLAY	DATA	GRAPHICS	WINDW	GRCMEM>P/S CON (3)	DAI	ADD16~19	Hi-Z	1	1	1	0	1	0	1	0	0(3)
14	DISPLAY	ADDRESS	CHR		ACRTC>ADDLTH>CHRMEM (3)	DAI	(2)	ADD0~15	0	0	0	0	1	1	1	X	X
15	DISPLAY	DATA	CHR	BKGND	CHRMEM>CHRGEN>P/S CON(3)	DAI	RA0~4	ADD0~15	1	1	1	1	1	1	1	X	1(3)
16	DISPLAY	DATA	CHR	WINDW	CHRMEM>CHRGEM>P/S CON(3)	DAI	RA0~4	ADD0~15	1	1	1	0	1	1	1	0	0(3)
17	DISPLAY	ADDRESS	GRAPHICS	HZOOM	DUMMY CYCLE (NO/AS) (4)	ALL	0	0	X	0	1	1	1	0	1	X	X
18	DISPLAY	DATA	GRAPHICS	HZOOM	DUMMY CYCLE	ALL	ADD16~19	Hi-Z	1	1	1	1	1	0	1	X	X
19	DISPLAY	ADDRESS	CHR	HZOOM	DUMMY CYCLE (NO /AS) (4)	ALL	0	0	X	0	1	1	1	1	1	X	X
20	DISPLAY	DATA	CHR	HZOOM	DUMMY CYCLE	ALL	RA0~4	Hi-Z	1	1	1	1	1	1	1	X	X
21	NO DRAW	ADDRESS	BOTH		DUMMY CYCLE (NO /AS)	ALL	0	0	X	0	1	1	1	0	X	X	X
22	NO DRAW	DATA	BOTH		DUMMY CYCLE	ALL	0	Hi-Z	X	1	1	1	1	0	X	X	X
23	REFRESH		BOTH		DRAM REFRESH	ALL	0	ADD0~8(5)	0	0	0	1	1	0	0	1	1
24	ATTRIBUTE		BOTH		ATTRIBUTE DATA (6)	ALL	ATT16~19	ATT0~15	1	1	1	1	1	0	0	1	1

NOTES:
(1) VALUE REFLECTS IF READ OR WRITE CYCLE PERFORMED
(2) VALUE HELD IN BITS RWP (PROC BITS 4,5,6,7)
(3) WINDOW SKEW WSS = 0
(4) FIRST CYCLE IS NORMAL, OTHERS ARE DUMMY CYCLES
(5) AS PER GAI SETTING
(6) LAST CYCLE PRIOR TO /HSYNC GOING HIGH
(7) RA4 = 0 WHEN NOT USED

MNEMONICS LIST:
1. ADDLTH - ADDRESS LATCH
2. BKGND - BACKGROUND (BASE, UPPER, LOWER) SCREEN
3. CHR - CHARACTER
4. CHRGEN - CHARACTER GENERATOR
5. CHRMEM - CHARACTER MEMORY
6. DAI - DUAL ACCESS MODE
7. GRCMEM - GRAPHICS MEMORY
8. HZOOM - HORIZONTAL ZOOM
9. P/S CON - PARALLEL TO SERIAL CONVERTOR
10. WINDW - WINDOW SCREEN
11. NO/AS - NO ADDRESS STROBE PRODUCED

This table shows, for each transaction:

(1) Information route
(2) The access mode(s) that it applies to
(3) The information available on the MA16/19 - RA0/4 signal lines
(4) The information on the MAD0-15 signal lines
(5) The state of the 3 timing signals 2CLK, MCYC, and $\overline{AS}$
(6) The state of the 3 status signals MRD, $\overline{DRAW}$, CHR
(7) The state of the 3 display control signals $\overline{HSYNC}$, $\overline{DISP1}$, $\overline{DISP2}$

For example transaction 5:

This represents the output of an address for a graphics drawing access.

The ACRTC provides the 20 bit address on MAD 0-15 and MA 16-19, which goes to the address latch, where it is captured and is then applied to the graphic memory.

The 2CLK, MCYC, and $\overline{AS}$ signals will be low as will the $\overline{DRAW}$ and CHR. MRD will assume a state that will indicate whether the access will be for reading or writing. The display control signals will not have a particular state as drawing can be done at any time.

Transaction 5 would be followed by either transaction 6 if reading, or transaction 7 if writing.

Consider mode graphics operation during display time; the display memory transaction would be Fig. 4-2.

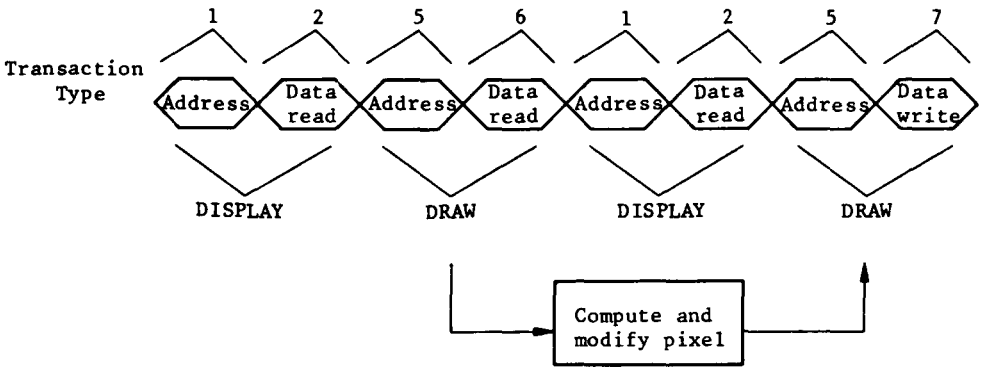


Fig. 4-2 Graphic Operation During Display time
(Interleaved Access Mode)

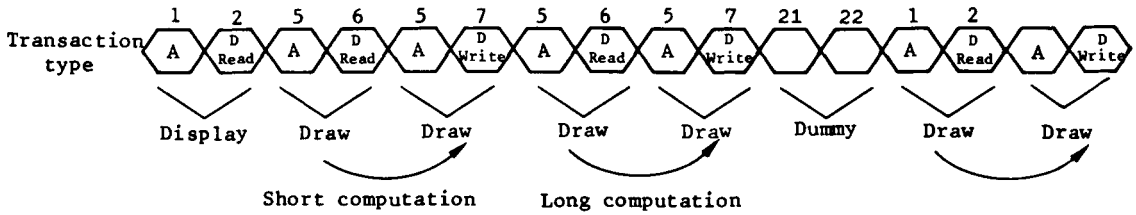


Fig. 4-3 Drawing Function on Single Access Mode

Fig. 4-3 uses single access mode, and shows both short and long pixel combinations. With a lengthy computation the new pixel data is not available in time for the next draw opportunity so a dummy cycle 21 and 22 is performed which does not affect the memory. The new pixel data is written during the subsequent draw cycle.

4.2 Design Example

Let us consider the design of a graphics display system using 256 clocks with a 512 × 512 pixel resolution and enough frame buffer capacity for two full size screens. Fast drawing is needed. The display monitor runs at 625 lines interlaced with a field frequency of 50 Hz.

While this scheme will produce noticable flicker due to the interlaced operation it will serve to demonstrate the principles.

To achieve fast drawing the ACRTC will be operated in interleaved mode.

For 256 colors, 8 bits per pixel will be required. These will be allocated: 2 bits per Red, Green, Blue, Intensity.

With 625 lines interlaced operation at a field rate of 50 Hz gives a scan frequency Fs of $\frac{625}{50} \times \frac{1}{2} = 15.625$ kHz.

The scan period for one raster is therefore 64 μs.
Only about 80% of this is for active display purposes i.e.
51.2 μs, the rest is for flyback.

If 512 pixels are to be displayed the

$$\text{Pixel rate} = \frac{512}{51.2} = 10 \text{ MHz}$$

The ACRTC can be clocked at half this frequency; i.e., 5 MHZ.

The frame buffer will be cycled at half of this frequency; i.e., 2.5 MHz. This gives a 400 ns cycle time.

Each screen will need 512 × 512 × 8 bits = 2 Mbits

Two screens therefore require 4 Mbits.

This could be provided by 64 or 64K x 1 bit DRAMs type Hitachi HM4864A.

Three speed version are available

HM4864A-12 cycle time = 220 ns
HM4864A-15 cycle time = 260 ns
HM4864A-20 cycle time = 330 ns

With a frame buffer cycle time of 400 ns careful logic design may allow the 330 ns devices to be used. However if the external logic delays are a problem then the faster DRAM parts may need to be used.

What GAI value will be needed?

$$GAI = \frac{TFB \times Npx \times Fs \times H.RES \times Acc}{128 \times 10^5}$$

Here TFB = 400
Npx = 8
Fs = 15.625
H.RES= 512
Acc = 2

$$GAI = \frac{400 \times 4 \times 31.25 \times 512 \times 2}{128 \times 10^5}$$

GAI = 4

Hence 4 × 16 bits = 64 bits will be read during each display cycle.

This requirement is satisfied by using the 64 of DRAM devices.

The frame buffer needs to be constructed from 4 banks of 64K words.

The parallel to serial conversion will be done by eight 8-bit shift registers, one for each of the bits of a pixel.

As the frame buffer provides 64 bits per read access and the ACRTC handles words, a 64 to 16 multiplexer scheme is needed.

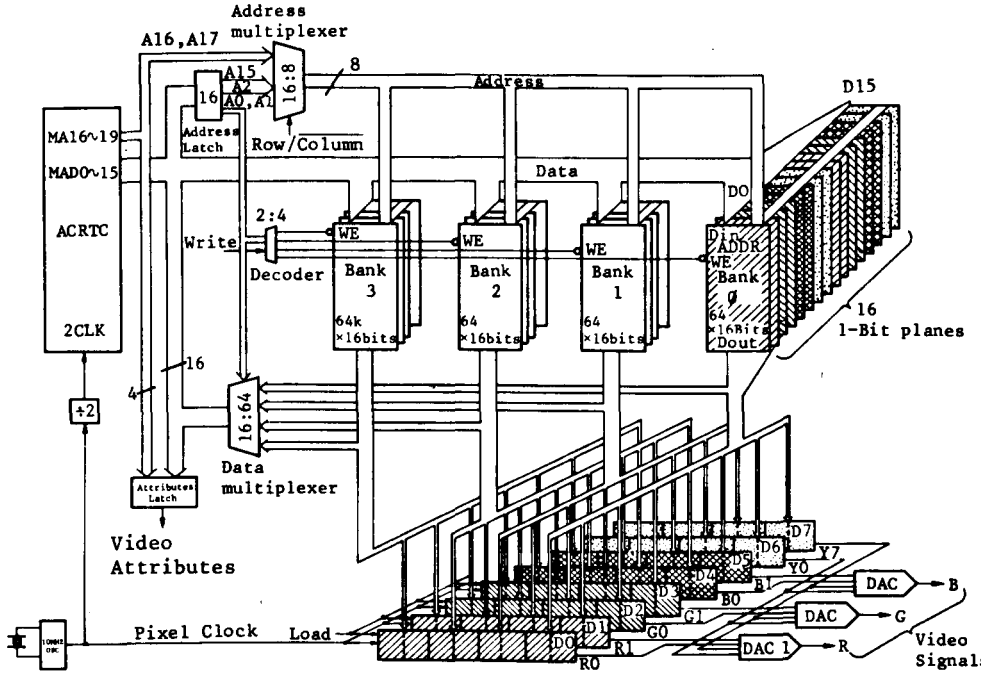


Fig. 4-4 Detailed System Diagram

5. FRAME BUFFER HARDWARE DESIGN

5.1 Requirements Overview

The detailed design of the frame buffer will now be considered. The system example (1) of Chapter 4 will be taken further, to act as a vehicle for these explanations.

By reviewing the system design (1), obtained in Chapter 4, we have the following scheme:

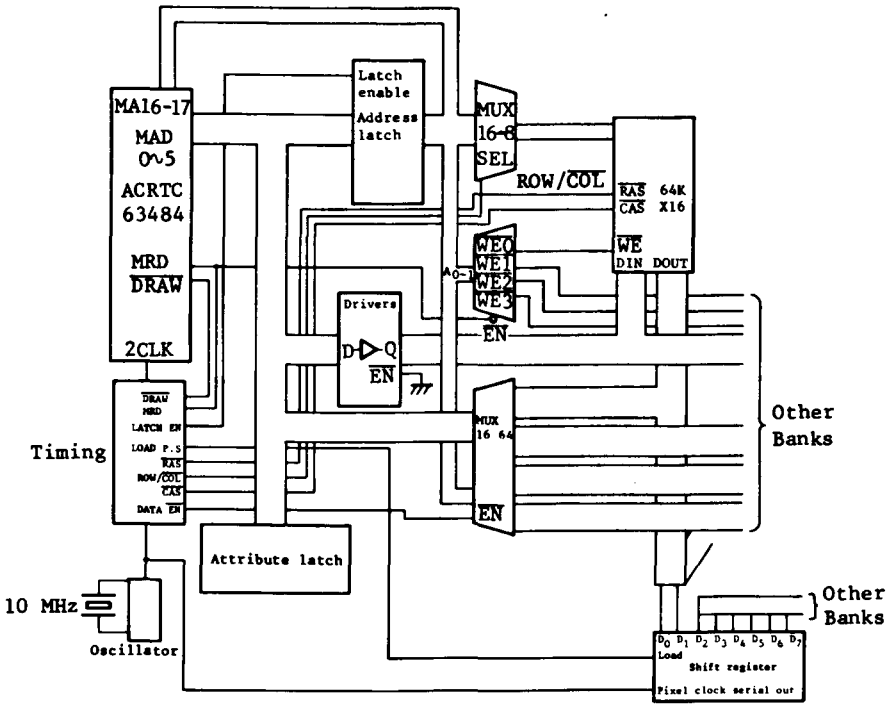


Fig. 5-1 Frame Buffer System Diagram

This shows in greater details the necessary external logic functions required to implement the frame buffer to satisfy the system specifications.

The memory is made up from 4 banks of 16 DRAMs, each bank providing 64K × 16 bits through the use of 64K × 1 bit DRAMs.

The main logic functions external to the ACRTC for frame buffer implementation.

(1) Address latch
(2) Address multiplexer and buffer
(3) Write decoder
(4) Data buffer
(5) Data multiplexer
(6) Shift registers
(7) DRAM arrays
(8) Attributes latch
(9) Oscillator, timing and control logic

Each of these will now be discussed further.

5.2 Address Latch

The address latch is used to capture the least significant 16 bits of address that are placed on the MAD_0 - MAD_{15} during the address phase of a frame buffer access cycle.

The address information is stable by (S2) after the falling edge of 2CLK. Timing signal $\overline{AS}$ can be used to enable a transparent latch formed from 2 of 74ALS373 octal latches.

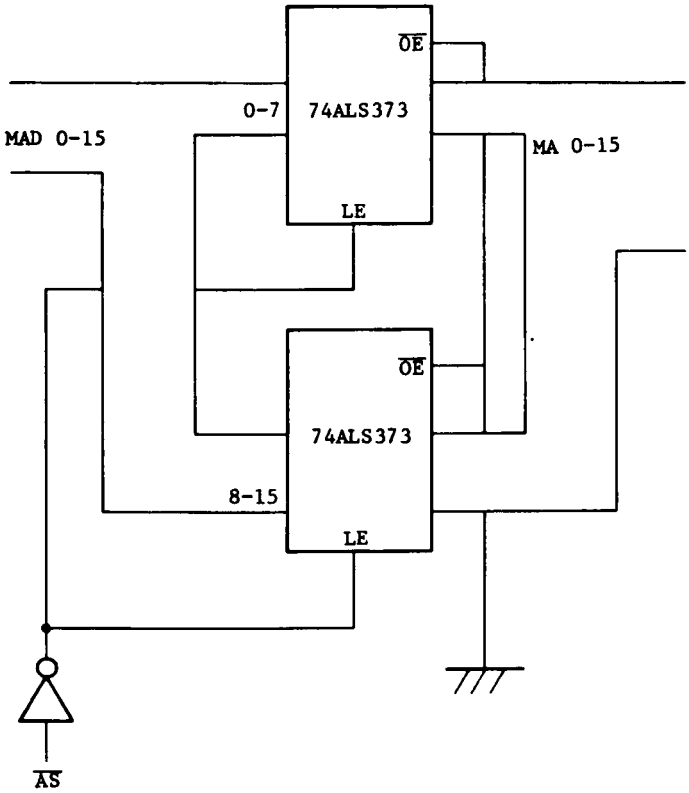


Fig. 5-2(a) Address Latch Circuit

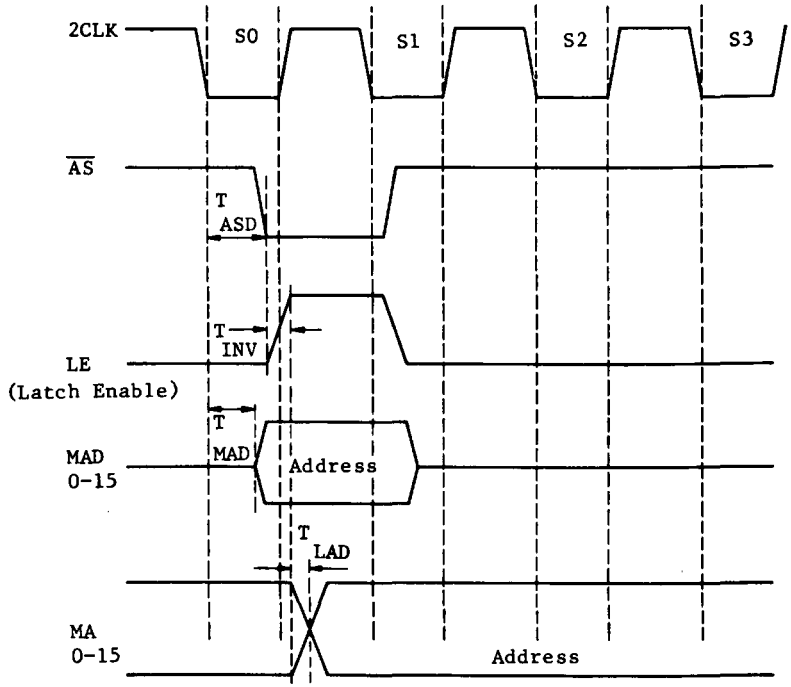


Fig. 5-2(b) Timing Chart of Address Latch

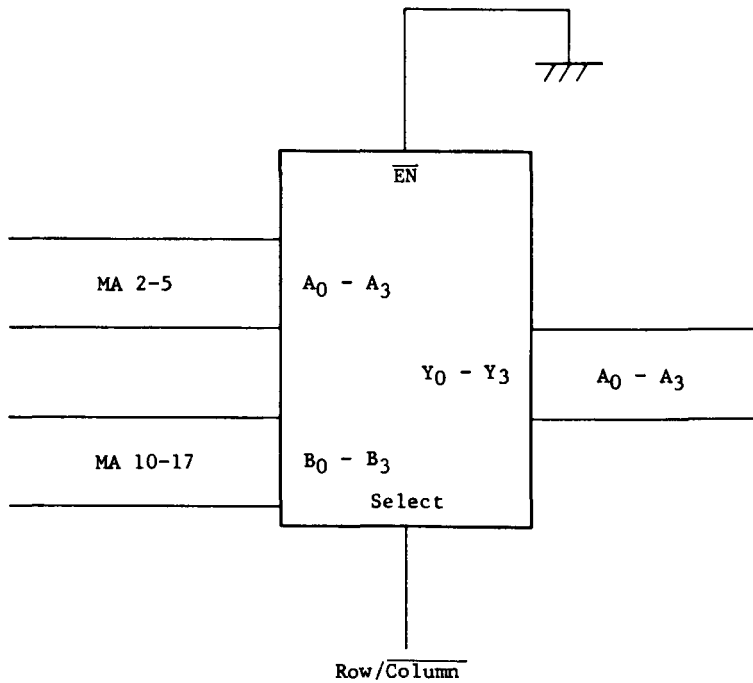


Fig. 5-2(c) Address Multiplexer

5.3 Address Multiplexer

The address must be multiplexed into an 8 bit row address and an 8 bit column address, for application to the DRAMs (64K x 1). This can be achieved by two 74ALS157 Quad 2:1 max. The address outputs then require buffering in order to drive to address signal lines of the frame buffer. Each signal line will have 64 DRAM devices. Care must be taken with the matching and terminating of these signal lines to keep noise and reflections to a minimum. Buffers are available with built-in termination/matching resistors.

The multiplexer is controlled by a signal row/$\overline{\text{column}}$ (ROW/$\overline{\text{COL}}$) which must be derived from the timing logic section described later.

5.4 Write Decoder

Write enables are generated by decoding the two least significant address bits A0 and A1.

As dual access mode 1 is not to be used then MRD can be used directly to produce the write enables. If dual access mode 1 is to be used, then we need to write enable only if both MRD and $\overline{\text{DRAW}}$ go low during display cycles, MRD indicating whether the window or background screen is active.

See Table 4-1 for signal states.

eg: Transactions 15, 16 & 7.

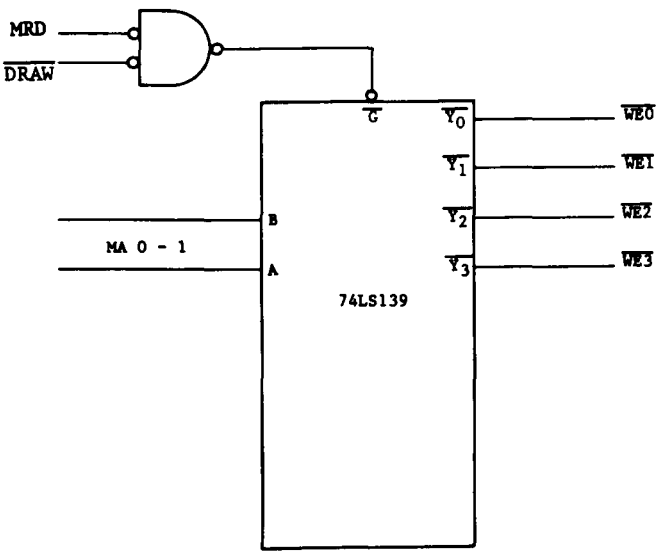


Fig. 5-3 Write Decorder

5.5 Data Buffers

The data lines from the ACRTC require buffering so as to drive the DRAM data I/O lines.

This requires typically 2 data drivers which have matching resistors built-in.

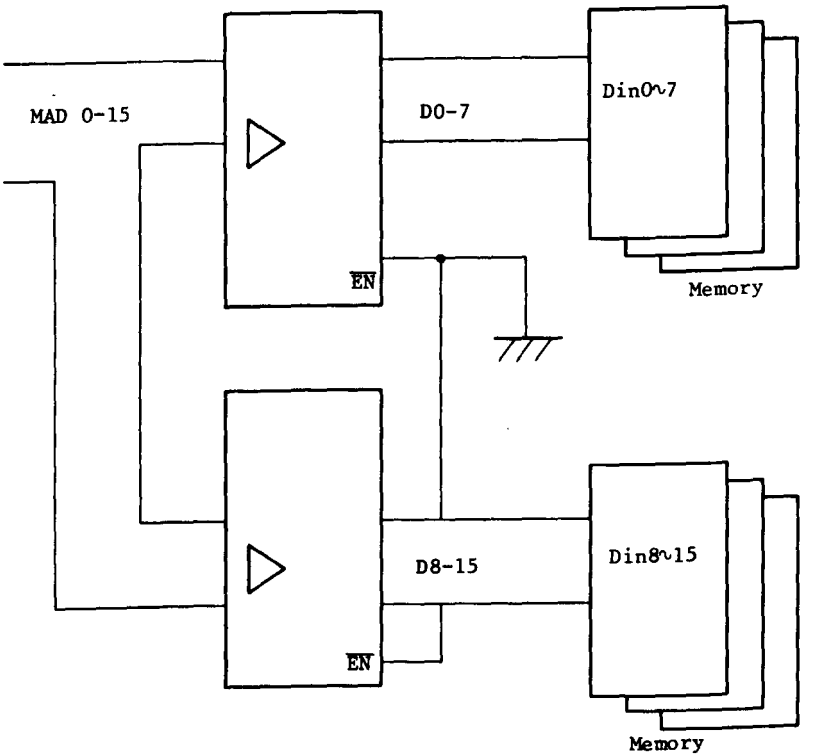


Fig. 5-4 Data Drivers Have Built-in Matching Resisters

5.6 Data Multiplexers

Each frame buffer read access generates 64 data bits. During drawing operations the ACRTC needs 16 bits, so a 64:16 data multiplexer is required. Selection is via the two least significant bits A0 and A1.

The multiplexer can be formed from eight 74ALS253 dual 4:1 multiplexers.

They are enabled when:

$\overline{DRAW}$ = 0 (draw)

MRD = 1 (read)

CHR = 0 (graphics)

If character memory is not used, ie: only graphics, the 'CHR' signal can be ignored.

A timing signal is needed, $\overline{CAS}$ is OK, from the timing and control logic section.

$\left\{\begin{array}{l}\overline{DRAW}\\ MRD\\ \overline{CAS}\end{array}\right.$

5.7 Shift Register

The 64 bits read from the frame buffer driving each display cycle are parallel, and loaded into 8 octal shift registers. These are clocked at the pixel rate and the outputs from these registers provide the video data signals. In this example the data is applied to three digital to logic converters (DACs). The analogue signals from these represent the RGB drives to the monitor.

The shift registers hold 8 pixels, each pixel of 8 bits. They could be 74ALS299. This can be clocked at up to 30 MHz, the S299 at up to 50 MHz, LS299 up to 35 MHz.

The load signal becomes active in time for the last pixel clock edge, reloading the registers.

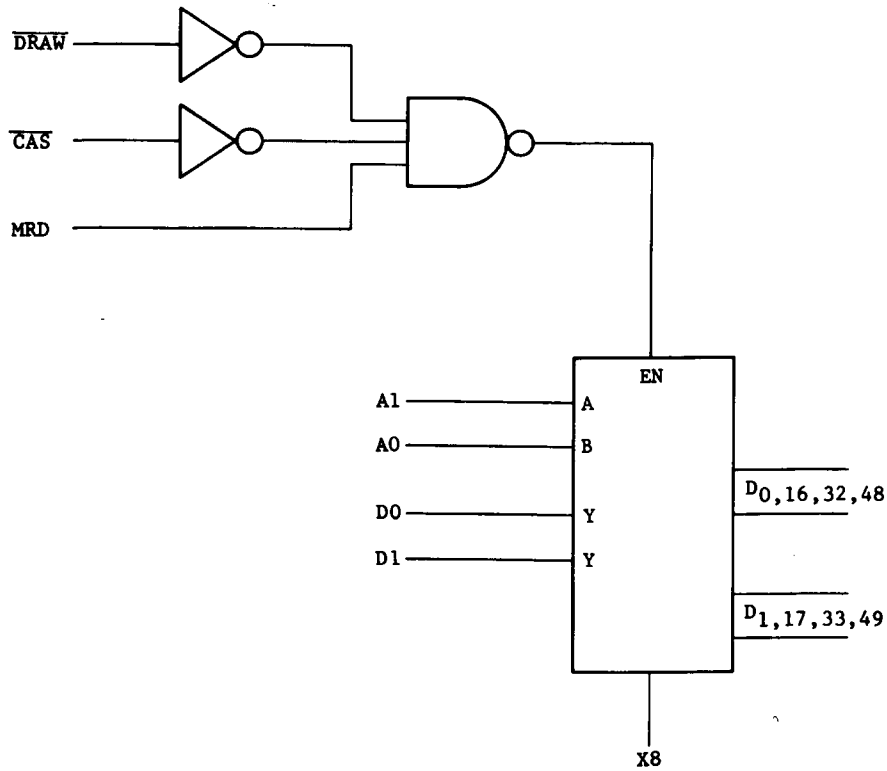


Fig. 5-5 Data Multiplexer

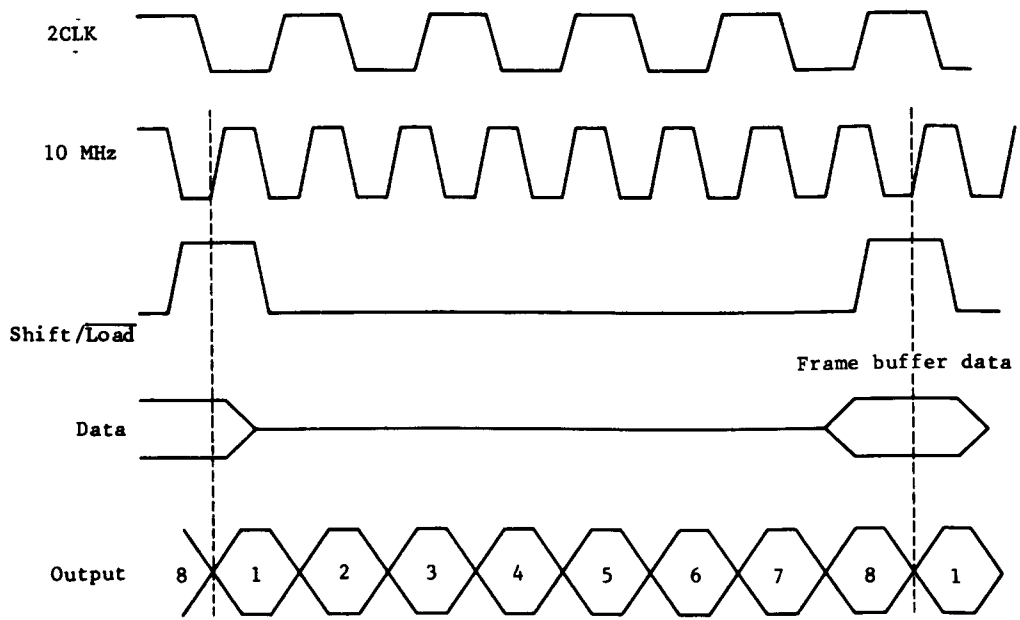


Fig. 5-6 Timing Chart of Shift Register

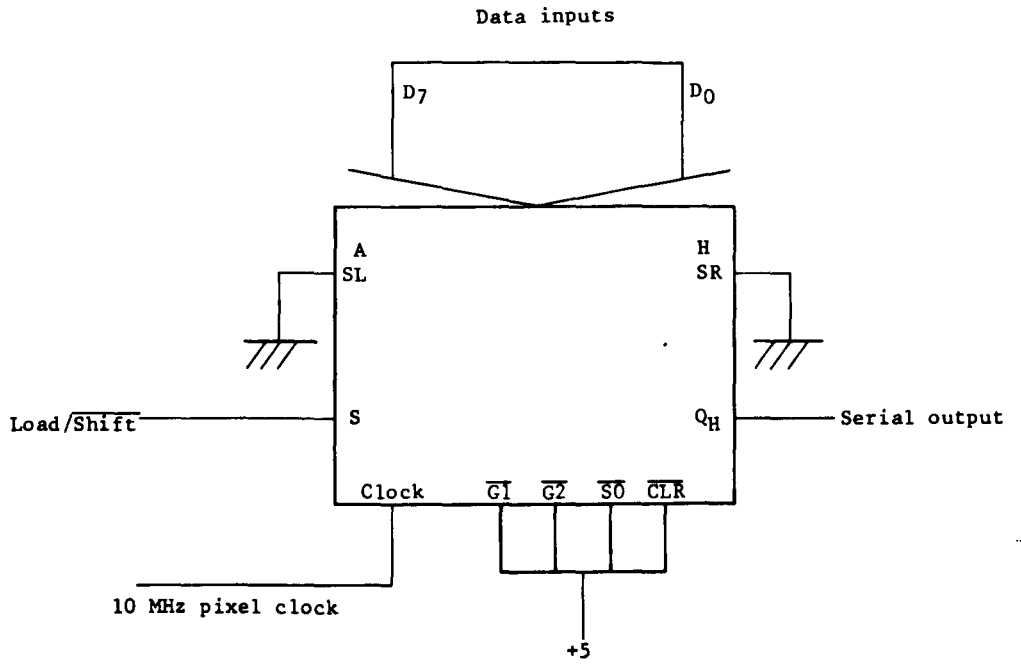


Fig. 5-7 Shift Register

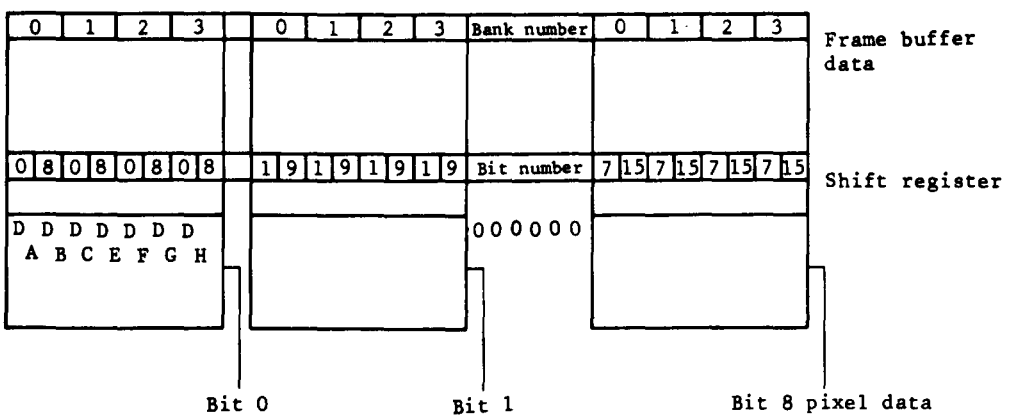


Fig. 5-8 Relation between Frame Buffer and Shift Register

5.8 DRAM Array

The DRAM devices used are 64K × 1 bit dynamics, arranged as four banks of 16 devices.

(150 ns DRAM's)

All banks are accessed during any read cycle, generating 64 bits of data (four words).

In the case of a display cycle, this data is loaded into the shift registers. For drawing, only one of these words is selected (according to the 2 LS address bits), and input to the ACRTC is through the data multiplexer.

When writing, only one bank is written. The data is applied to all the banks. The 2 LS address lines are decoded and used to write enable the appropriate bank.

Each bank is built from 16 DRAM's, one RAM for each data bit.

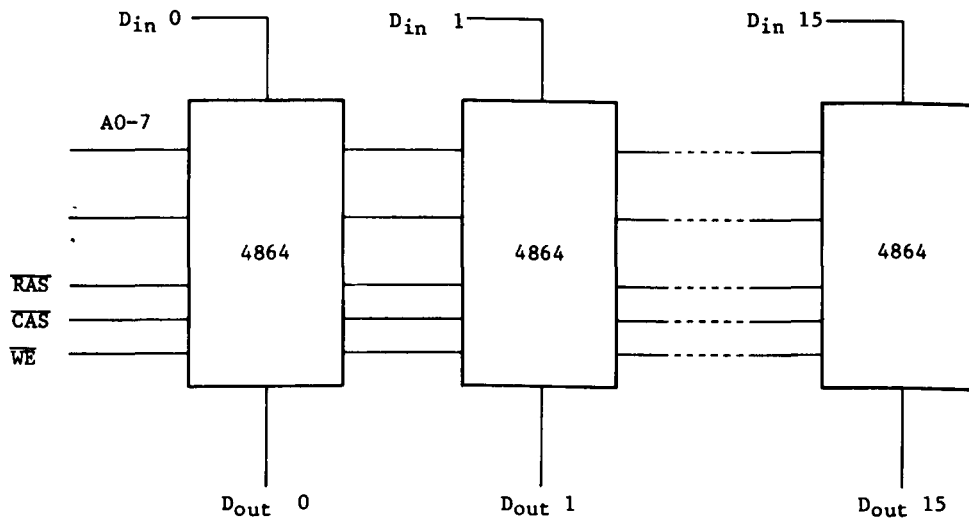


Fig. 5-9 DRAM Array

The cumulative capacitive loading of the DRAM's requires careful consideration in the driving of the signal lines to avoid noise and reflections.

Access time of DRAMs = 120 ns

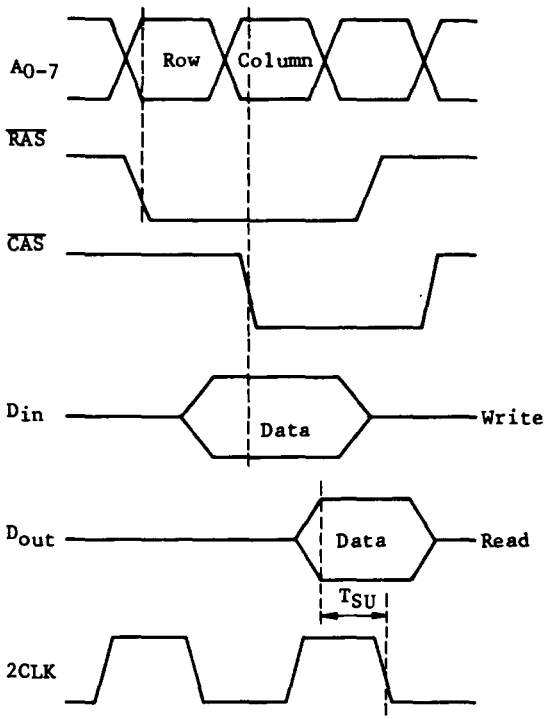


Fig. 5-10 Timing Chart of DRAM Read/Write

5.9 Attributes

The 20 bits attributes are output on the MAD 0-15 lines and MA 16-19 lines. They can be latched into 74LS373 octal latches. The speed of the devices is not so critical as in other areas of the external logic.

It is only necessary to latch those attributes that are needed by the system requirements, unused bits can be ignored. This could reduce the number of latches required.

The attributes are output on a raster by raster basis. The attributes are available at the end of the last cycle before $\overline{\text{HSYNC}}$ returns high. Because no unique condition indicates this directly, it is necessary to assume any cycle while $\overline{\text{HSYNC}}$ is low may end with an attribute transfer. Therefore the latches must be enabled at the end of each of these cycles. If it is the last cycle, then the captured data is valid attributes and will remain for use during the coming raster. However, if it was not the

last cycle, then the latch will hold spurious information which will be replaced on the next cycle, until good attributes are held.

6. OSCILLATOR, TIMING AND CONTROL LOGIC

6.1 Overview

The function of this logic is to control the flow of data between the system elements.

It is formed from a (1) Crystal oscillator, a (2) Timing generator and (3) Control logic.

It produces ten signals:

(1) The ACRTC clock : 2CLK
(2) The pixel clock : PXL CK
(3) The shift register load strobe : LOAD/$\overline{\text{SHIFT}}$
(4) The write enable : WRITE EN
(5) The read enable : READ EN
(6) The attribute latch enable : ATT EN
(7) The address latch enable : ADD EN
(8) The row/column address select : ROW/$\overline{\text{COL}}$
(9) The DRAM RAS strobe : $\overline{\text{RAS}}$
(10) The DRAM CAS strobe : $\overline{\text{CAS}}$

6.2 The ACRTC Clock and The Pixel Clock

A 10 MHz crystal oscillator provides the pixel clock used by the shift registers. This is divided by two and inverted to obtain the 5 MHz required by the ACRTC as its 2CLK.

A non-inverted form is taken off and used to generate an early form of the MCYC signal called EMCYC. This avoids the internal delays of the ACRTC as a 74ALS74 can be used as the flip-flop, see Fig. 6.1. The state of MCYC is sampled just prior to being changed due to 2CLK and the result inverted to obtain the correct state, (as MCYC toggles of 2CLK).

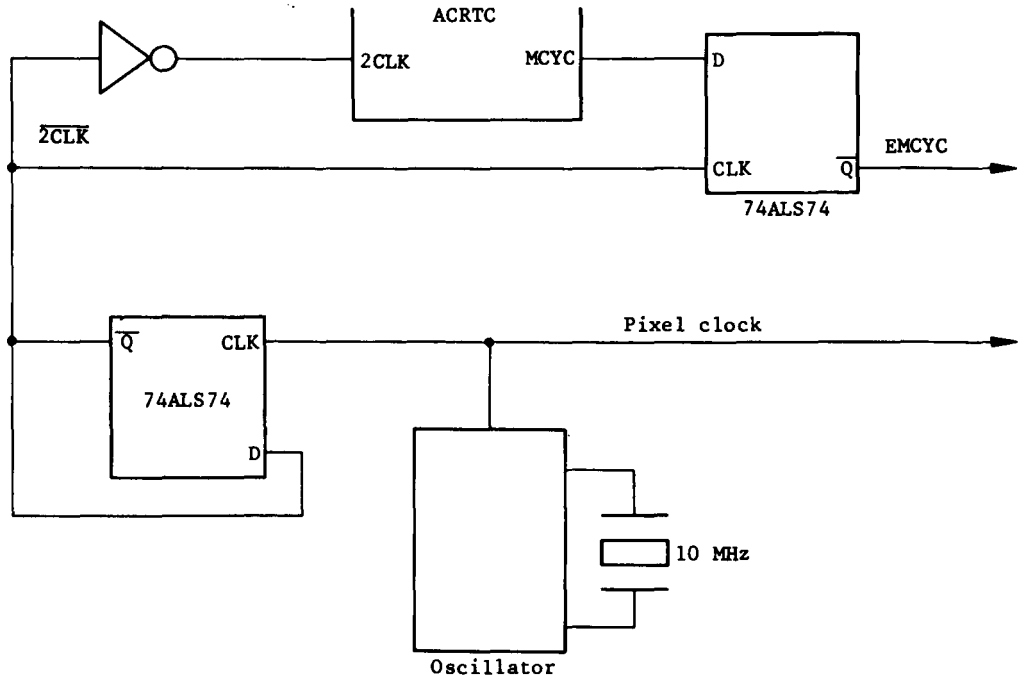


Fig. 6-1 Clock Generator Circuit

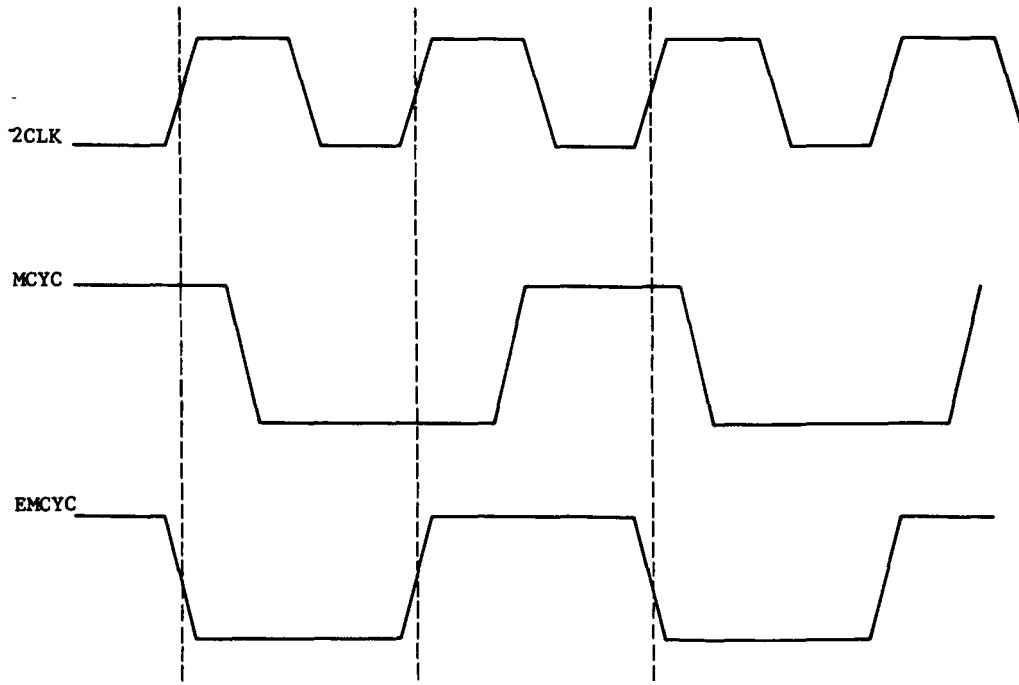


Fig. 6-2 Timing Chart of Clock Generator

6.3 Pixel Clock

6.3.1 Shift Register Load

The shift register load signal occurs at the end of a display cycle, when the data obtained from the DRAM's is stable at the shift register parallel inputs. The Load/$\overline{\text{Shift}}$ input to these registers must be taken high, during which time the pixel clock must have a rising edge.

The loading signal must be inhibited while drawing cycles and dummy cycles take place.

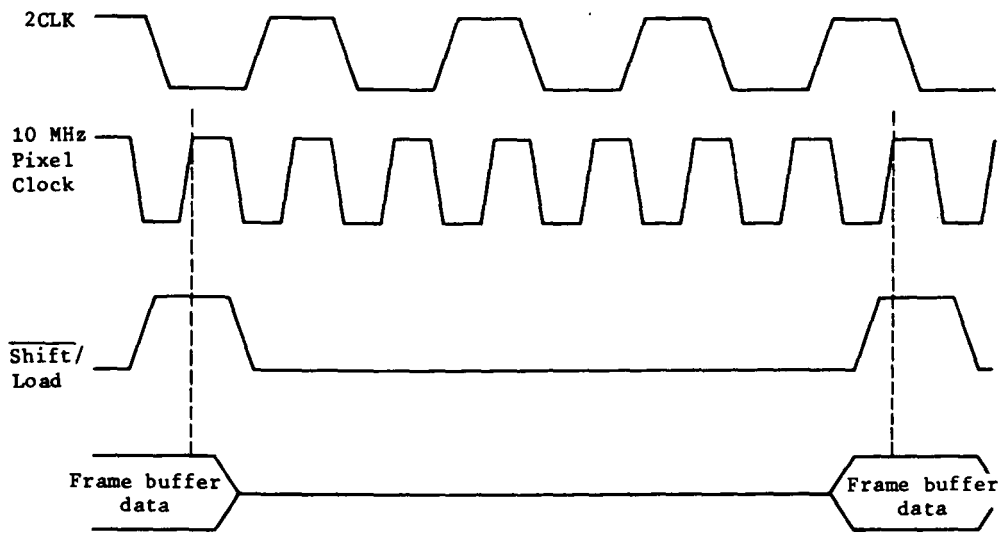


Fig. 6-3 Timing Chart of Shift Register Load

6.3.2 Shift Register Load - Dummy Cycles

When the ACRTC has the opportunity of accessing the frame buffer for drawing purposes, but has no requirement to transfer any pixel data, then it skips the cycle by doing a 'dummy cycle'. These take the form of a display read cycle, as both $\overline{\text{DRAW}}$ and MRD are high. However, there is no $\overline{\text{AS}}$ strobe associated with the address phase and the address output from the ACRTC is 0. During the data phase the ACRTC MAD lines are Hi-Z.

Because the status is similar to a display read cycle, care must be taken to identify these dummy cycles and inhibit loading of the shift registers. One method of spotting these cycles is to preset a flip-flop from $\overline{\text{AS}}$ and clear it at the end of the cycle.

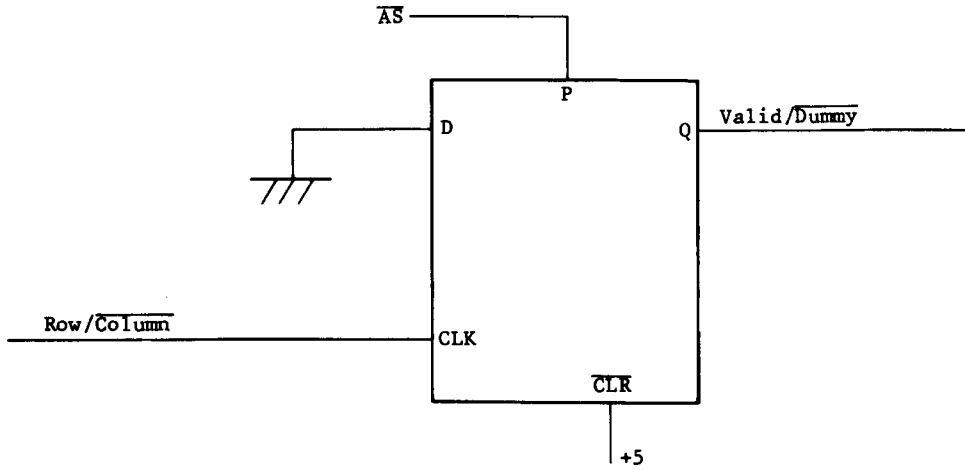


Fig. 6-4 Valid/Dummy Cycle Generator

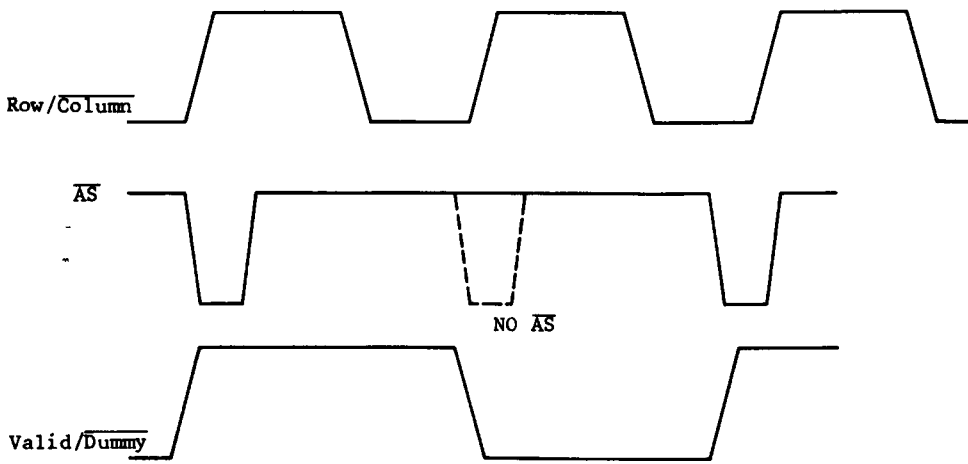


Fig. 6-5 Timing Chart of Valid/Dummy

6.4 Write Enable : WRITE EN

During a write cycle, the addressed bank of DRAM's will be write enabled, prior to the falling edge of $\overline{CAS}$. The ACRTC status signal MRD, can be used for this purpose by using it to enable a 2:4 decoder that takes as its inputr A0 and A1.

5.5 Read Enable : READ EN

When a drawing read cycle is performed, the data multiplexer must be enabled in order to route the data from the appropriate bank to the ACRTC and satisfy the set-up and hold times (Refer to Electrical Specification in the User's Manual, 55 and 56). The timing of this enable can be basically that of the CAS. The logic delays will generally satisfy the data hold time (56). It is not necessary to inhibit this signal during a display read cycle as the ACRTC will ignore any data placed on its Hi-Z MAD lines.

6.6 Attribute Latch Enable : ATT EN

This signal can simply be derived by gating 2CLK with $\overline{HSYNC}$. It can enable transparent latches like 74LS373. The operating speed is not as critical as in other areas of the design.

6.7 Address Latch Enable : ADD EN

The transparent latches type 74ALS373 are enabled by inverting the ACRTC address strobe $\overline{AS}$.

6.8 Row/$\overline{Column}$ Address Select : Row/$\overline{COL}$

After the row address strobe $\overline{RAS}$ has gone low, a short hold time is required before the row address is replaced with the column address. If the RAS signal timing is used to derive the ROW/$\overline{COL}$ signal, then the logic delays must satisfy this address hold time.

An alternative method is to specifically generate a ROW/$\overline{COL}$ signal that occurs some time after $\overline{RAS}$.

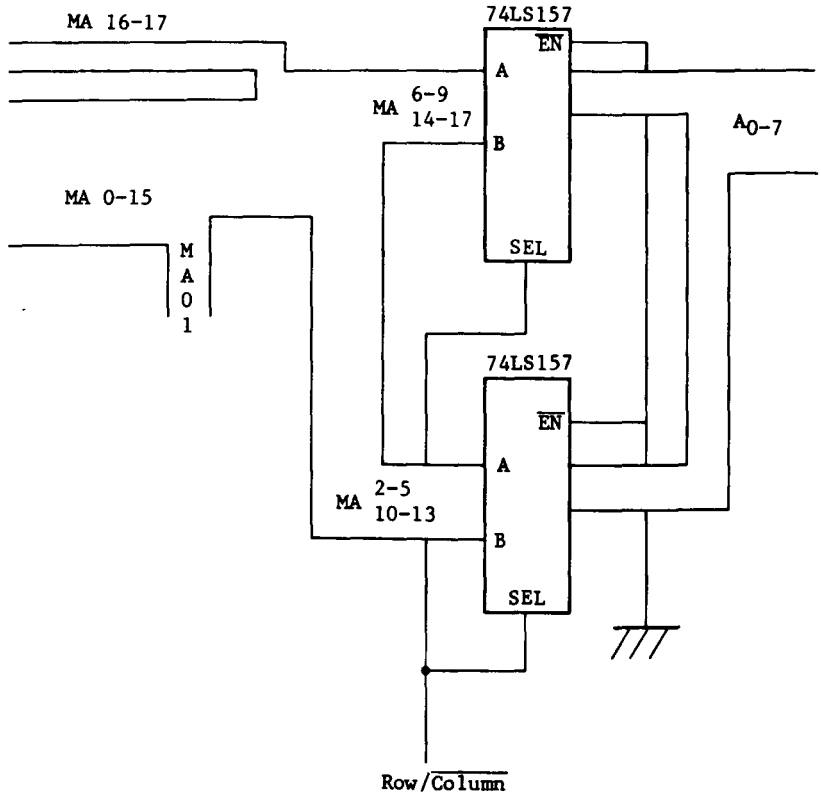


Fig. 6-6 Circuit for Row/$\overline{\text{Column}}$ Selection

6.9 The DRAM Row Address Strobe : $\overline{\text{RAS}}$

$\overline{\text{RAS}}$ must not go low at the DRAM's before the Row address is stable. No set-up time (TASR) is required, but allowance must be made for the address being delayed by the external logic, ie: address latch, multiplexer, buffers and DRAM input capacitance. If a transparent latch is used, e.g.: 74ALS373, then it is not necessary to wait until the tailing edge of $\overline{\text{AS}}$ for a valid address. The address from the ACRTC is stable by (52), i.e.: 70 ns after 2CLK ↓. Allowing 40 ns external delay results in taking $\overline{\text{RAS}}$ low 120 ns after 2CLK ↓.

RAS can remain low for the remaining part of the cycle. The 120 ns high period satisfies the precharge period (T_{rp}, 100 ns).

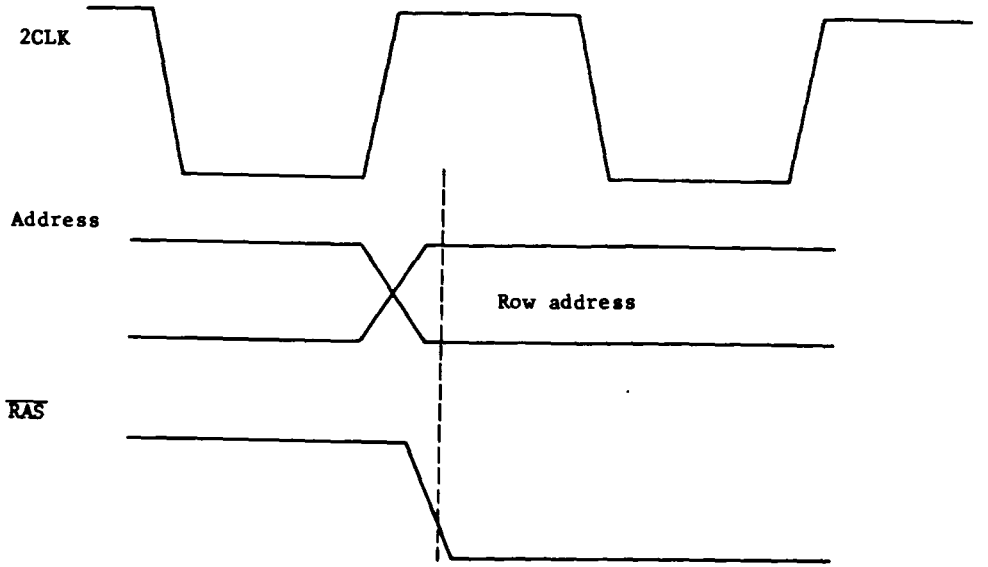


Fig. 6-7 Timing Chart of $\overline{RAS}$ Signal

6.10 The DRAM Column Address Strobe : $\overline{CAS}$

$\overline{CAS}$ has more critical timing requirements than $\overline{RAS}$. It is used by the DRAM's to action four functions.

(a) To capture the column address
(b) To sample the state of the write unit
(c) For a WRITE CYCLE : capture the data being input
(d) For a READ CYCLE : initiate the output of data

The column address is held in the address latch, so when the ROW/$\overline{COL}$ selects it, only the delay due to the multiplexer, buffer and capacitance, need allowing for. The write signal will be stable well before this as it is derived from the ACRTC MRD signal which is available early in the cycle.

It is reasonably simple to satisfy the first two requirements ie: (a) and (b) above. It is the last two ie: (c) and (d), that effectively dictate how soon the $\overline{CAS}$ can be taken low and the result depends on whether it is a read or a write cycle.

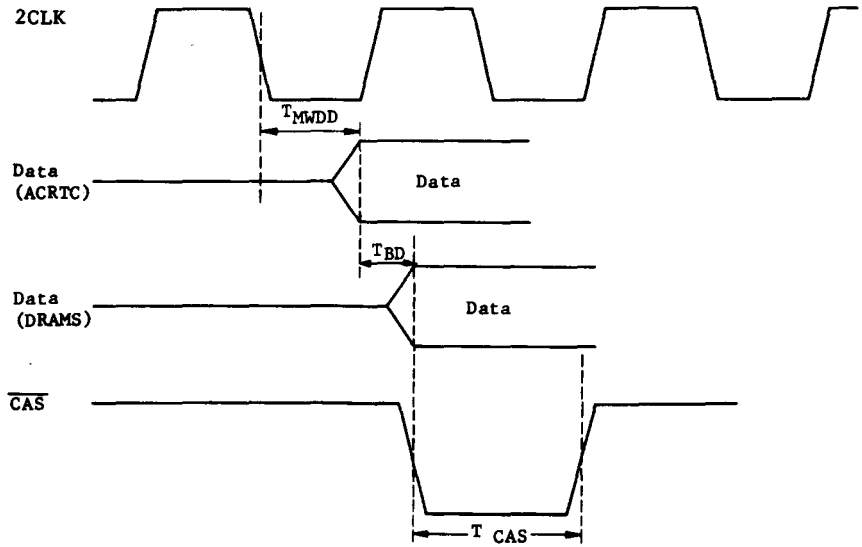


Fig. 6-8 Timing Chart of $\overline{CAS}$ Signal

6.10.1 Write Cycle Timing

The limiting factor is the availability of data from the ACRTC. Data is output from the mid access 2CLK ↓, becoming stable (64); i.e.: 70 ns later. The data buffers will add some further delay so the data will not be settled at the DRAM pins until approx. 20 ns later, a total of 40 ns. At this time $\overline{CAS}$ can be taken low. The DRAM's require no set-up time (T_{ASC}) but a hold time of T_{CAH} i.e.: 25 ns.

$\overline{CAS}$ must remain low for T_{CAS}, i.e.: 75 ns. This can overlap into the next access cycle provided the write signal timing parameters are observed. The ACRTC MRD signal is updated at the start of each cycle.

6.10.2 Read Cycle Timing

Data is available from the DRAM's T_{CAC} i.e.: 75 ns after $\overline{CAS}$ has gone low. This data must be routed through the data multiplexer to the ACRTC and meet the data set-up time (55), i.e.: 40 ns. This data is captured on the 2CLK ↓ at the end of the access cycle. The data multiplexer would typically use 74ALS 253, providing, a delay of 10 ns including tracking etc.

The total delay from $\overline{CAS}$ ↓ to 2CLK ↓ would therefore be:

75 + 10 + 40 = 125 ns

$\overline{CAS}$ can be terminated just after the end of the cycle. The ACRTC requires a data hold time (56) of 10 ns. The data multiplexer outputs can be disabled at this point also, in fact the $\overline{CAS}$ signal timing can be used to enable the multiplexer.

6.10.3 Read and Write Timing Considerations

If the frame buffer timing is not too critical then it is possible to simplify the generation of $\overline{CAS}$ by making the read cycle and write cycle timing common. This technique does not optimize the memory throughout, but is simpler to implement.

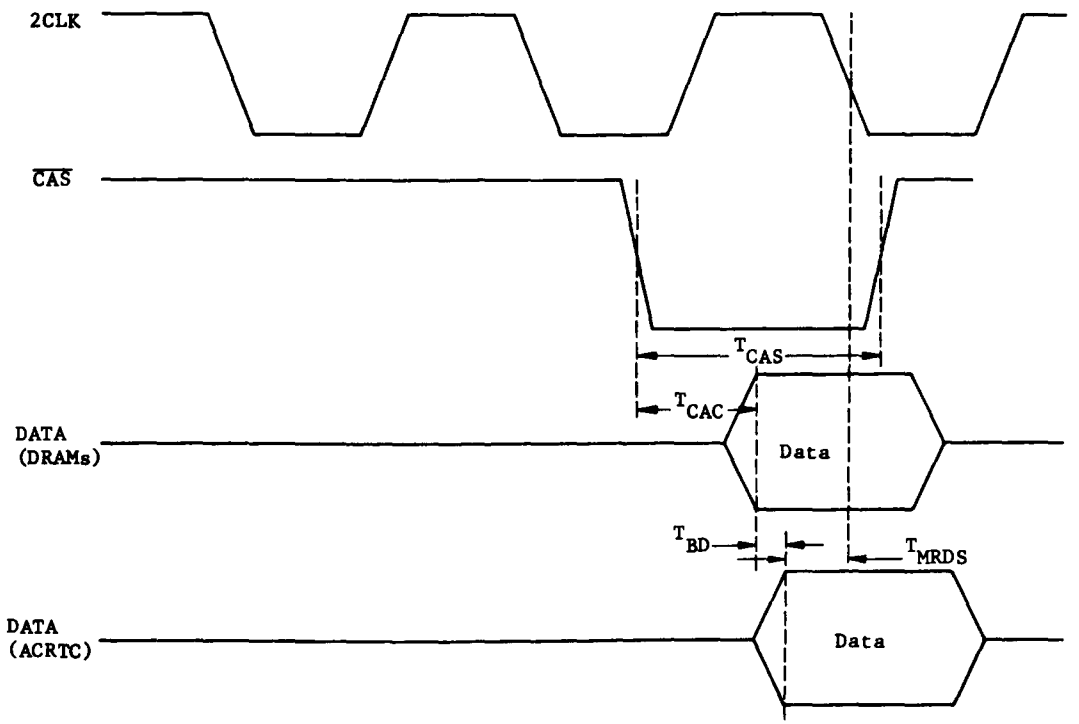


Fig. 6-9 Frame Buffer Read/Write Timing

So the half memory cycle time is 225 ns = clock period of 2CLK. Hence 2CLK = 4.4 MHz.

Hence, if the 6 MHz ACRTC part and 150 ns access time DRAM's are used, the fastest the ACRTC can be clocked using common $\overline{CAS}$ timing for read and write cycles, is about 4.5 MHz.

6.10.4 Refresh Cycles

When refreshing of DRAM's is enabled, they occur during the Horizontal Sync period, ie: when $\overline{HSYNC}$ is low. When refresh is disabled, then drawing cycles can occupy this period, suitable for use with STATIC RAM.

The ACRTC places the refresh address from its 8 bit counter onto the appropriate MAD signal lines, (according to the setting of GAI), during the address phase of the cycle. As no data is transferred during data phase, the $\overline{CAS}$ signal must not be generated. Hence, it is necessary to use the $\overline{HSYNC}$ output from the ACRTC to inhibit $\overline{CAS}$ generation during the refresh cycles of DRAM's.

6.10.5 General Comments on Timing

If the frame buffer timing is not critical, then it is possible to simplify the generation of $\overline{CAS}$ by making the read and write cycles both common timing:

eg: for the memory cycle time

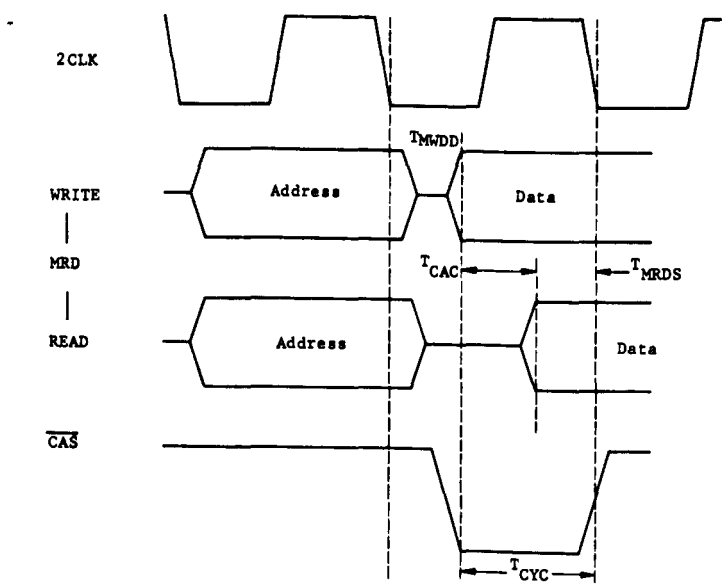


Fig. 6-10 Frame Buffer Read/Write Timing

Half cycle = 225 ns = 2CLK = 4.4 MHz

therefore, rule of thumb:

If using same read and write $\overline{CAS}$ the max 2CLK 4.5 MHz (if 150 ns DRAMs).

If frame buffer timing is tight, then it is necessary to modify the timing of $\overline{CAS}$ dependent on whether a read or a write cycle is taking place. By optimizing in this way, the frame buffer can be worked harder for greater throughput.

6.11 Timing Generator

Many methods can be employed to provide the timing generator. If the frame buffer timing is not critical, then some short cuts can be taken e.g.: T_{CASW} and T_{CASR} could be the same signal, so reducing the associated logic. Also, the timings could be sourced from signals already available.

Specifically deriving the timing signals could be accomplished by any of the four example methods outlined below:

(i) Delay Line

This makes use of a tapped delay line to provide a series of timings, with EOR to pick up the appropriate delays for each signal. The buffered variety of delay line is best suited as it avoids the problems of tolerance due to loading effects. By splitting the delay line into two sections, the second stage can have more tappings to allow for fine tuning.

The advantage is that no high frequency clock source is needed.

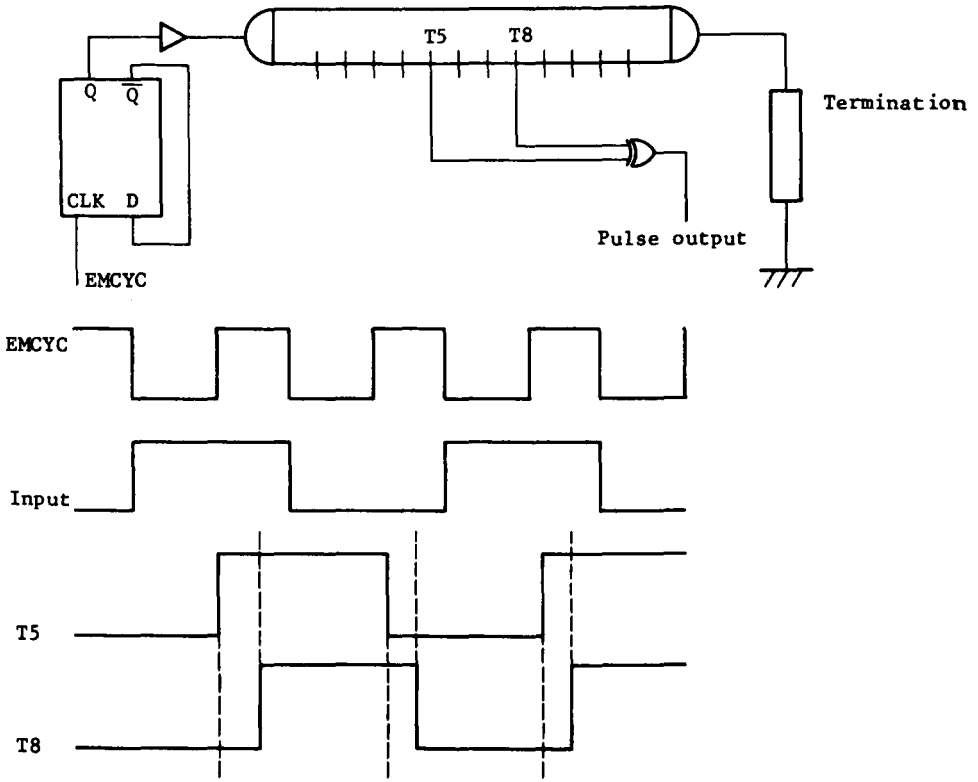


Fig. 6-11 Delay Line

(ii) Shift Register

The analog delay line of method (1) can be replaced by a shift register clocked by a high frequency. This makes the results more able to be reproduced and well defined. However, the resolution of the tappings depends upon the clock frequency; the higher it is, the more stages required for the shift register.

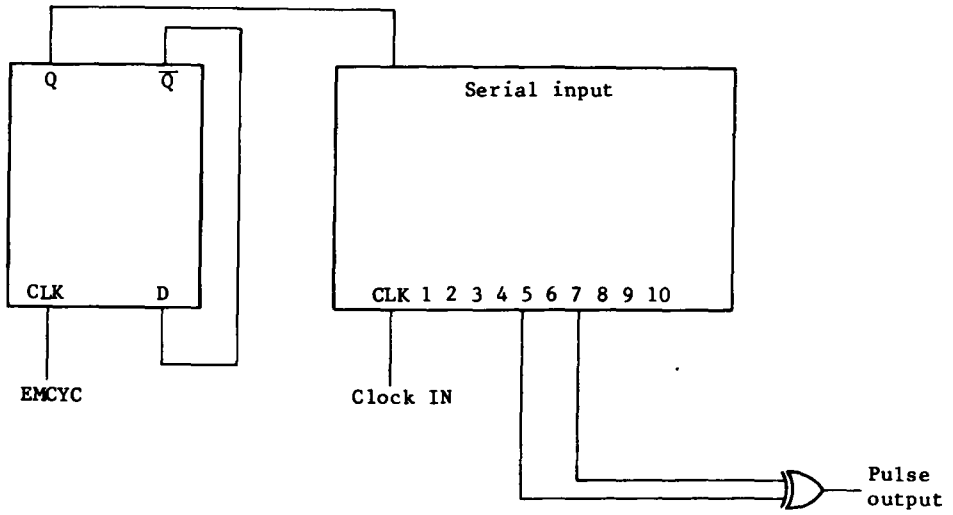


Fig. 6-12 Shift Register

(iii) State Machine

A clock can be used to increment a counter, the output of which is used to address a PROM. The PROM outputs are de-glitched via a latch to provide the timing signals. The counter must be reset at the start of each access cycle.

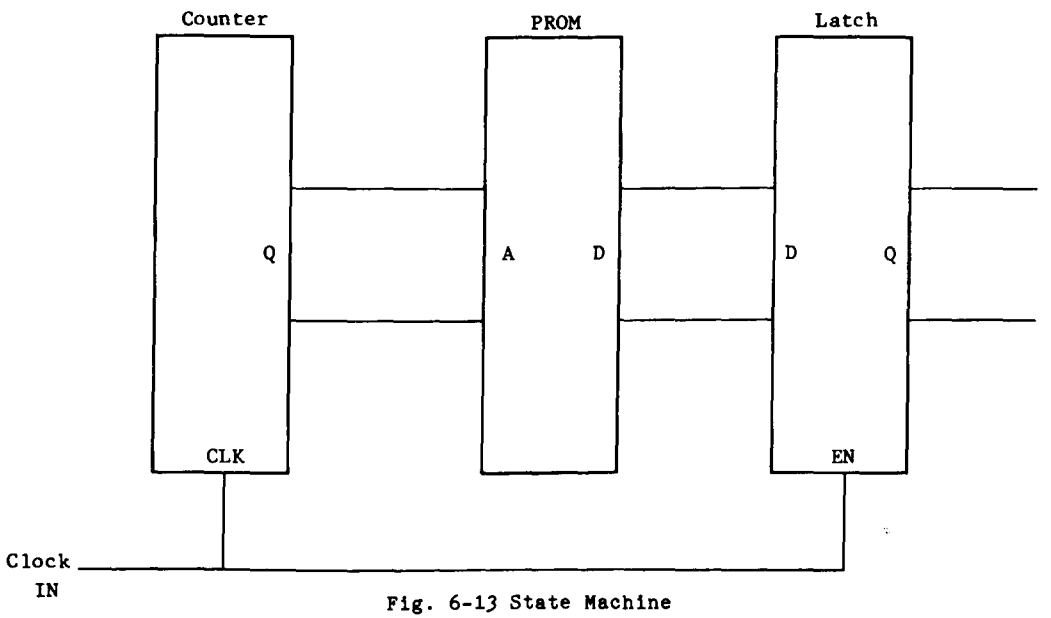


Fig. 6-13 State Machine

(iv) Micro-Sequencer

By providing a feedback path from the latched outputs to the PROM address inputs, the counter can be removed. This can be reduced to one part using a programmable array logic (PAL).

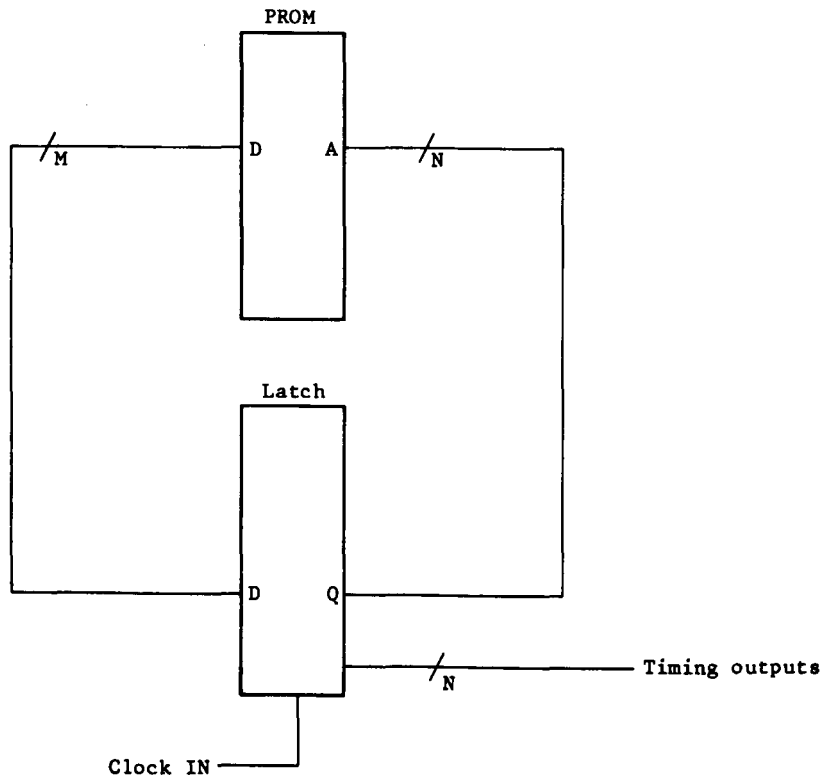


Fig. 6-14 Micro-Sequencer

7. ACRTC PARAMETERS AND RESISTERS

7.1 Reset

A hardware reset is achieved by pulsing the RESET input signal (Pin 6) to a low level for a minimum of 10 2CLK cycles. It forces the ACRTC into the following state, which will persist until the host alters it.

(1) Both the drawing and display operations are terminated.

(2) The DRAM refresh address is placed on the MAD lines in accordance with the Graphic Address Increment (GAI) mode selected by bits 4, 5 and 6 of the Operation Mode Register, (OMR).

(3) The $\overline{\text{HSYNC}}$ output signal, pin 12, assumes a low level. Other signals are affected due to 'start' being cleared.

(4) The ACRTC registers are initialized as follows:

Table 7-1 Initial State of Control Registers

DATA BIT		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRESS REG.	AR									8 BIT ADDRESS							
										NO CHANGE (NC)							
STATUS REG.	SR									CER	ARD	CED	LPD	RFF	RFR	WFR	WFE
										0	0	1	0	0	0	1	1
FIFO ENTRY	FE	16 BIT FIFO DATA ENTRY															
		DATA DELETED, AS POINTERS RESET															
COMMAND CONTROL	CCR	ABT	PSE	DDM	CDM	DRC	GBM			CRE	ARE	CEE	LPE	RFE	RRE	WRE	WEE
		1	0	0	0	0	0			0	0	0	0	0	0	0	0
OPERATION MODE	OMR	M/S	STR	ACP	WSS	CSK		DSK		RAM	GAI			ACM		RSM	
		0	0	NC	NC	NC		NC		NC	NC			NC		NC	
DISPLAY CONTROL	DCR	DSP	SE1	SE0		SE2		SE3		8 BITS OF USER ATTRIBUTES							
		NC	NC	NC		NC		NC		NO CHANGE (NC)							

NC = NO CHANGE

RESET

Initialises both the ABORT Bit (CCR Bit 15) = 1 and the START Bit (OMR Bit 14) = 0

These two bits have an important effect on the device.

ABORT

When set, abandons command execution, clears the FIFO and initialises the status register (SR) to $23.

START

When clear, it halts the display control and drawing operations and the internal time base for the CRT control signals are reset. While it is clear the following signal conditions exist:

Pin 63 $\overline{\text{DISP1}}$)
)
Pin 62 $\overline{\text{DISP2}}$)
) ALL GO TO THE INACTIVE "HIGH" STATE
Pin 1 $\overline{\text{CUD1}}$)
Pin 2 $\overline{\text{CUD2}}$)
Pin 13 $\overline{\text{VSYNC}}$)

Pin 12 $\overline{\text{HSYNC}}$ = DRIVEN TO THE ACTIVE "LOW" STATE

*MAD lines = CARRY THE REFRESH ADDRESS (as per GAI mode)

*Regardless of OMR Bit, RAM mode select.

Only the register values indicated in the table are affected by the application of a hardware RESET. Note that many of the bits within the table are unaffected (no change NC). Their value prior to the reset will be maintained.

The remaining registers and RAM of the ACRTC undergo no change as a result of a RESET. Hence the timing and display control values are left intact as are the drawing parameters and pattern RAM.

Obviously, after power-up neither of the ACRTC registers nor the RAM hold defined values and thus they need to be specified after the first RESET.

A hardware RESET can be simulated in software by:

(1) WRITING $8000 TO CCR - SET ABORT, CLEAR OTHERS
(2) 'AND'ING $3FFF TO OMR - CLEAR MIS AND START, NO CHANGE

The following flowchart, Fig. 7-1 shows a software simulation of a hardware RESET.

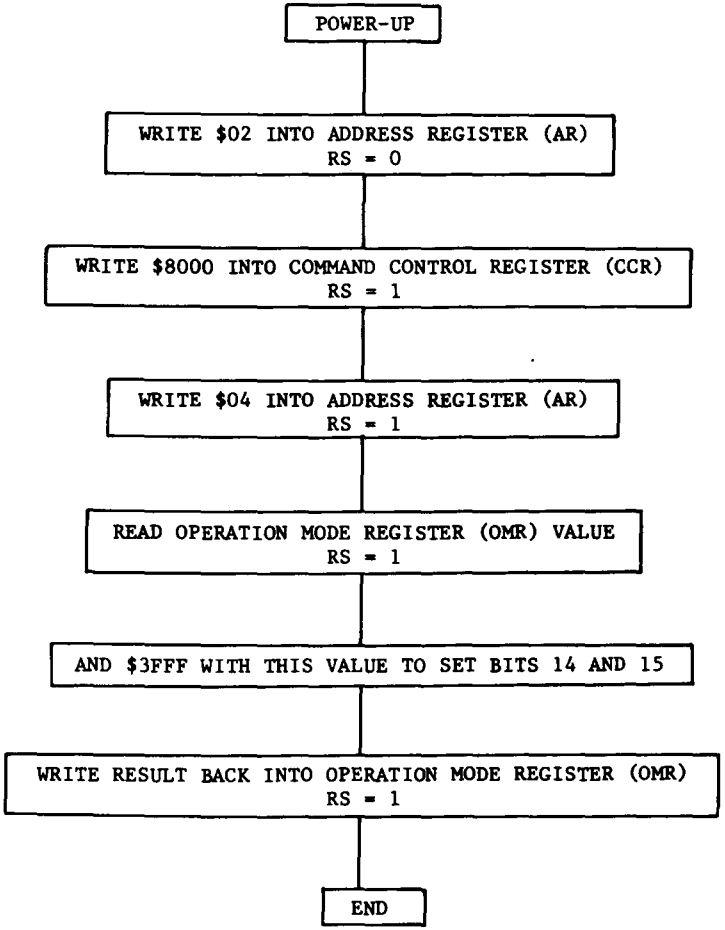


Fig. 7-1 Software Simulation of a Hardware Reset

7.2 Initialization

Following the application of a hardware reset (or the software equivalent), many registers of the ACRTC need initializing. First the timing and display control RAM require loading with values appropriate to the monitor hardware in use and the desired display format. The three control registers CCR, DCR and OMR are then configured, after which drawing commands can be issued.

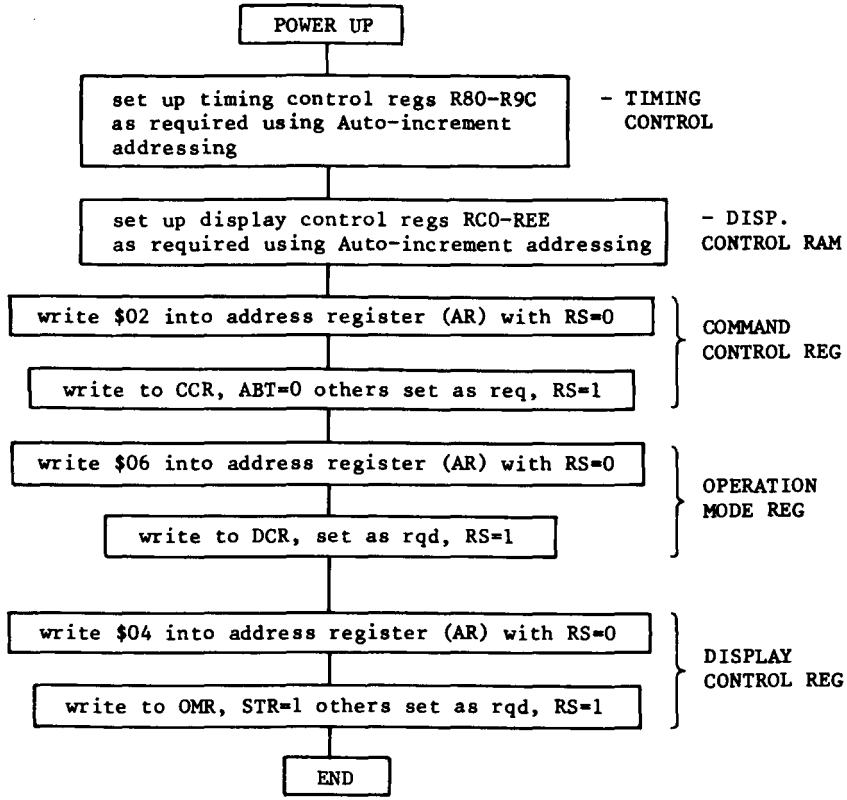


Fig. 7-2 Registers Initialization

7.3 Timing Control RAM

REGISTER ADDRESS R80-R9C

Before starting the configuration of the timing RAM it is necessary to completely specify the requirements of the display monitor hardware and system design.

Two fundamental points are:

(1) All horizontal values used by the ACRTC are in units of memory cycles.

(2) All vertical values are in units of scan lines (rasters).

It is therefore necessary to convert all specifications for the monitor hardware etc. from their time domain values into these units before any registers can be configured.

To aid programming the many Timing Control registers, a chart is provided. In all, 23 fields of data values must be configured. The results of these fields are combined to provide the word values, required by the ACRTC, an example in Table 7-2.

The timing control RAM holds the values that time and configure the display screen. Fig. 7-3 shows how the display screen is specified in terms of the register values.

The ACRTC Users Manual should be consulted for detailed explanation of these registers.

For clarification, there follows a worked example. This is based on the system discussed in Chapter 4 onwards; only the base screen will be implemented.

To recap: SYSTEM SPEC

Scan standard	625 lines interlaced
Scan rate	15.625 KHz
Field rate	50 Hz
Frame rate	25 Hz

Horizontal resolution	512 pixels
Vertical resolution	585 lines
Displayed vertical resolution	512 lines
Frame buffer capacity	= 256 KB (64 × 64K × 1 DRAM)
Frame buffer cycle period	= 400 ns
ACRTC clock frequency	= 5 MHz

Pixel rate = 10 MHz

monitor
(line period = 64 μs
(horizontal sync width = 5.12 μs
(back porch = 5.12 μs
(front porch = 2.56 μs

Vertical sync width = min 200 μs
front porch = 256 μs
back porch = 256 μs

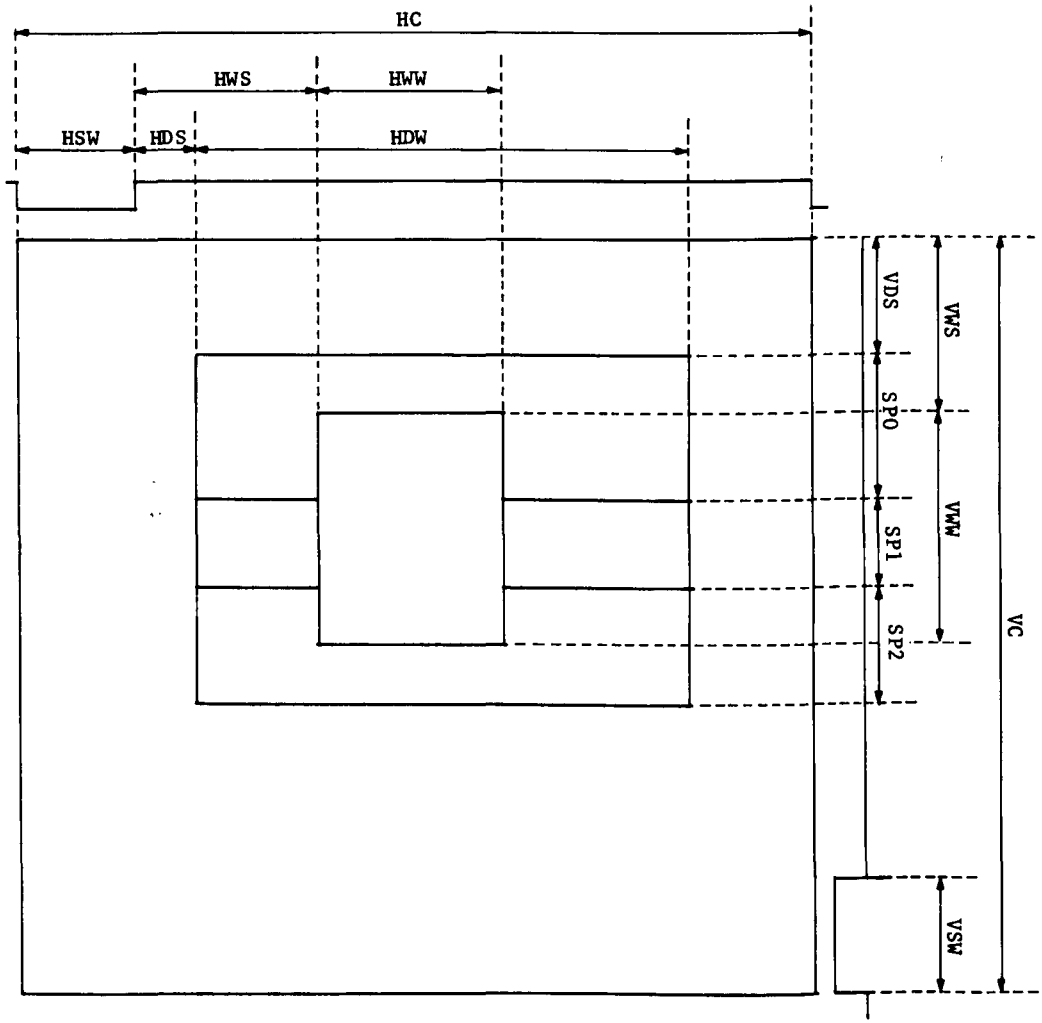


Fig. 7-3 Display Screen Specification

Table 7-2 Timing Control RAM (Set Values)

REG NO.	REGISTER	MNEMONIC	BITS	REG	MNEMONIC	LOAD DEC	VALUE HEX	NOTES
R80	Raster Count	RCR	11~0	Raster Cycle	RC			(1)
R82	Horizontal Sync.	HSR	15~8	Horizontal Cycle	HC	159	9F	(2)
			4~0	Horizontal Sync. Width	HSW	13	0D	
R84	Horizontal Display	HDE	15~8	Horizontal Display Start	HDS	12	0C	(2)
			7~0	Horizontal Display Width	HDW	127	7F	(2)
R86	Vertical Sync.	VSR	11~0	Vertical Cycle	VC	625	271	
R88	Vertical Display	VDR	15~8	Vertical Display Start	VDS	37	25	
			4~0	Vertical Sync. Width	VSW	20	14	
R8A	Split Screen Width	SSW	11~0	Split Screen 1 Width	SP1	512	200	
R8C			11~0	Split Screen 0 Width	SP0	0	0	(3)
R8E			11~0	Split Screen 2 Width	SP2	0	0	(3)
R90	Blink Control	BCR	15~12	Blink On 1	BON1	0	0	(4)
			11~8	Blink Off 1	BOFF1	0	0	(4)
			7~4	Blink On 2	BON2	0	0	(4)
			3~0	Blink Off 2	BOFF2	0	0	(4)
R92	Horizontal Window Display	HWR	15~8	Horizontal Window Start	HWS	0	0	(3)(2)
			7~0	Horizontal Window Width	HWW	0	0	(3)(2)
R94	Vertical Window Display	VWR	11~0	Vertical Window Start	VWS	0	0	(3)
R96			11~0	Vertical Window Width	VWW	0	0	(3)
R98	Graphic Cursor	GCR	15~8	Cursor X End	CXE	0	0	(4)
			7~0	Cursor X Start	CXS	0	0	(4)
R9A			11~0	Cursor Y Start	CYS	0	0	(4)
R9C			11~0	Cursor Y End	CYE	0	0	(4)

Notes:

(1) A Read Only Register
(2) The Load Value is one less than the Required Value
(3) Need only define if particular screen is enabled
(4) Need only define if function is to be used, otherwise = X

7.4 Horizontal Timing

From the specifications the required horizontal timing wave form appears as in Fig. 7-4.

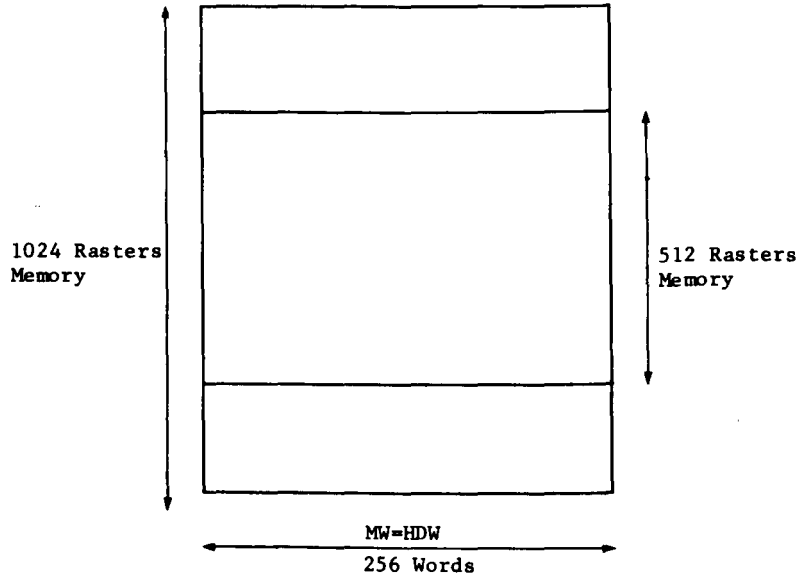


Fig. 7-4 Horizontal Timing

	SET VIA:
LINE PERIOD = 64 µs = 100%	HC
HSYNC = 5.12 µs = 8%	HSW
BACK PORCH = 5.12 µs = 8%	HDS
ACTIVE DISPLAY PERIOD = 51.2 µs = 80%	HDW
FRONT PORCH = 2.56 µs = 4%	REMAINING TIME

The frame buffer cycle time is 400 ns

(i) Line Period:

This is set via HC, the Horizontal Cycle field;

64 µs required: $\frac{64\ \mu s}{400\ ns}$ = 160 cycles

Note:

As this is an even number, it is suitable for interlaced operation.

HC is loaded with one less than this value, thus;

160 - 1 = 159 = $ 9F

HC = $9F

(ii) Horizontal Sync. Period:

This is set via HSW, the Horizontal Sync. width field, 5.12 μs required:

$\frac{5.12\ \mu s}{400\ ns}$ = 12.8 approx. = 13 cycles

13 = $0D

HSW = $0D

Note:

- This value is greater than 3, and will allow RCR to be read.
- Will it satisfy the DRAM refresh needs?

For DRAM's refresh:

HSW > N/(Tr × Fh)

N = No. of refresh cycles = 128
Tr = refresh period = 2 ms
Fh = scan frequency = 15.625 KHz

HSW > $128/(2 \times 10^{-3} \times 15.625 \times 10^{3})$ (see chapter 1, section 13, and chapter 2, section 11)

HSW > 4.096

Which is clearly satisfied by the value of 13.

(iii) Back Porch:

Set vis HDS, the Horizontal display start field.

5.12 μs required: $\frac{5.12\ \mu s}{400\ ns}$ = 12.8 approx. 13 cycles

HDS is loaded with one less than this value, thus;
13 - 1 = 12 = $0C

HDS = $0C

(iv) Active Display Period:

Set via HDW, the Horizontal Display width field.

51.2 μs required: $\frac{51.2\ \mu s}{400\ ns}$ = 128 cycles

HDW is loaded with one less than this value, thus;
128 - 1 = 127=$7F

HDW = $7F

(v) Front Porch

This is not specified directly as it is the remainder from the other values.

Front porch = HC - (HSW + HDS + HDW)

Note:

This equation uses the values calculated, not the ones loaded as sometimes these are one less.

Front Porch = 160 - (13 + 13 + 128)
= 160 - 154
= 6 cycles, at 400 ns

Front Porch = 2.4 µs

This is sufficiently close to the desired value of 2.56 µs.

In summary, the four values are therefore:

1. HC = $9F
2. HSW = $0D
3. HDS = $0C
4. HDW = $7F

These can be entered into Table 7-2.

HC and HSW are combined in R82 (HSR).
HDS and HDW are combined in R84 (HDR).

Table 7-2 shows this information accordingly.

The result is that the two registers have these values:

R82 = $9F0D
R84 = $0C7F

7.5 Vertical Timing

Typically the vertical flyback period occupies about 7% of the vertical period. In the case of the 625 line system example, 40 lines are lost per frame, that is 20 lines per field. Only 585 lines can be used for displaying. As the resolution is to be 512 lines, the front and back porches will be 37 lines and 36 lines. This should place the displayed lines almost centrally on the display monitor tube face; refer to Fig. 7-5.

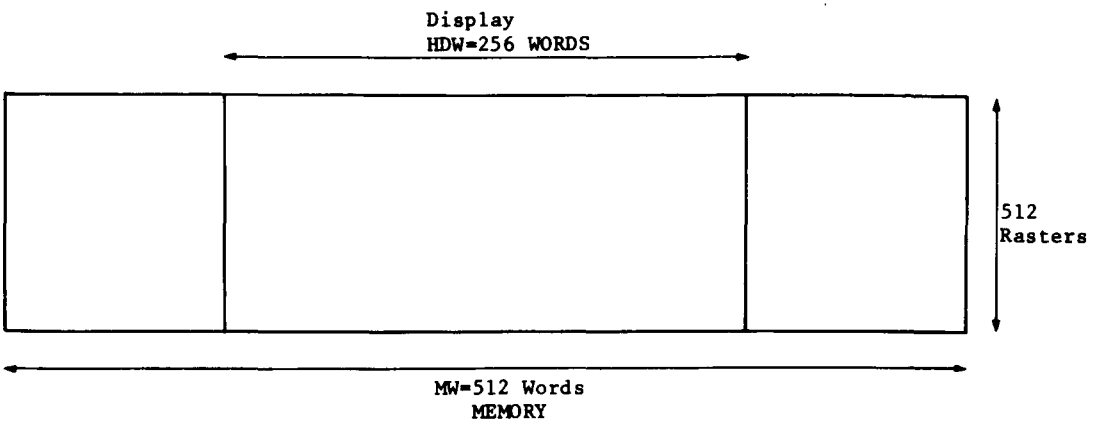


Fig. 7-5 Vertical Timing

(i) Frame Rasters:

Set by VC, the Vertical cycle field;

625 lines required: V = 625 cycles per frame

Note:

V = VC for interlaced sync and video mode.

VC = 625 = $271

(ii) Front Porch:

Set by VDS, the Vertical Display start field; 37 lines required,

VDS = $25 (using interlaced sync and video mode)

(iii) Vertical Sync. Period:

Set by VSW, the Vertical Sync. width field;

20 lines required per one display field (ie: 40 lines per one display frame).

20 × 64 µs = 1.28 ms sync. period, satisfying the monitors minimum requirement of 200 µs.

VSW = 20 = $14

(iv) Display Period:

Set by SP1, the split screen 1 width field (base screen).

512 lines required: SP1 = 512 = $200

Note: Only the base screen is in use.

(v) Back Porch:

This is not specified directly, as it is the remainder from the other values.

Back Porch = VC - (VDS + SP1 + 2 × VSW)

Note:

This equation uses the values calculated, not the ones loaded, as sometimes these are one less. VSW is specified in lines per field, other factors are lines per frame.

Back porch = 625 - (37 + 512 + 40)
Back porch = 36 lines per frame

As no split screens are in use SP0 and SP2 need not be defined. As the address register auto increments, it is simpler to load these registers R8C and R8E with say, $0000 than to specifically skip them. (Normally the value $0 cannot be used).

This also applies to the blink control, window display and graphic cursor control registers. However, if most of these functions are not to be used, then it is efficient to skip a continuous group of registers by reloading the address register.

The unused functions can be left undefined. The registers so far initialized, (R80 - R8A) are the minimum necessary of the timing control RAM, to produce a stable raster timing. The result of the vertical timing calculations are:

VC = $271 VSW = $14
VDS = $25 SP1 = $200

These can be entered into Table 6-2(a). VDS and VSW are combined in R88 (VDR). Table 6-2(b) shows this information accordingly. The result is that the three registers have these values:

R86 = $0271
R88 = $2514
R8A = $0200

The Registers R8C - R90 can be left undefined if the respective functions are not used.

This completes the programming of the Timing Control RAM.

7.6 Display Control RAM

Register Address RC0 - REE

The display format is specified through the use of these registers. In the example that follows, only the base screen will be used for simplicity. The other screens do not need to be defined, if they are not enabled, (the base screen must always be defined even if it is not enabled). Configuration is further simplified as the character mode is not used and neither are the cursors.

The following therefore, represents the minimum amount of initialization of the Display Control Registers:

RC0 - RC6 Upper screen - not defined
RC8 - RCE Base screen - to be defined
RD0 - RD6 Lower screen - not defined
RD8 - RDE Window screen - not defined

RE0 - RE8	Cursor - not defined
REA	Zoom factor - to be defined
REC - REE	Light pen - read only

Base Screen Definition:

RC8 - need not be defined as character mode is not used.

RCA - Memory Width of Base Screen

The hardware design supports 512 KB of frame buffer memory. We only need to consider the base screen.

The display is 512 × 512 pixels, each of 8 bits. As the base screen occupies the whole of this, it represents 256 KB of data, half the frame buffer capacity. With no other screen defined, we have many possibilities for configuring the base screen in relation to frame memory.

The horizontal display width is 512 × 8 bits = 256 words.

Recall that the HDW was set to 128 cycles and using a GAI = +4 and Dual Mode 0.

$$HDW = \frac{128 \times 4}{2} = 256 \text{ words}$$

The base screen memory width can be made greater or equal to this value.

(i) If the memory width is made equal to the display width MW = 256 words.

As the frame buffer capacity = 256K words it will support:

$$\frac{256K}{256} = 1K \text{ or } 1024 \text{ rasters}$$

This will allow vertical scrolling, but not horizontal scrolling.

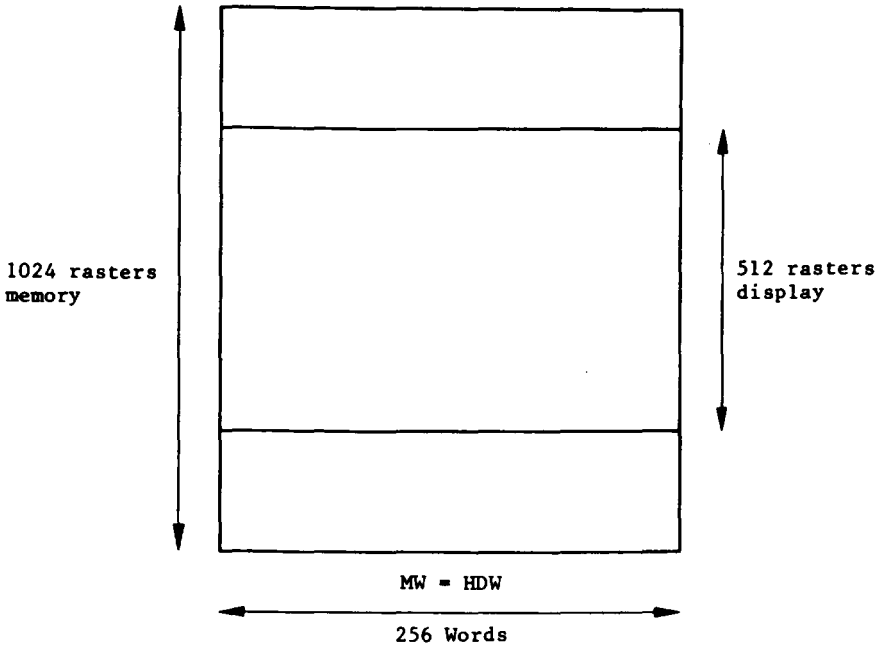


Fig. 7-6(a) Horizontal Timing

(ii) If the memory width is made twice that of the display width,

MW = 2 × 256 = 512 words.

Frame buffer capacity = 256K words, so it will support:

$$\frac{256K}{512} = 512 \text{ rasters}$$

This is the same as the display.

This arrangement will allow scrolling horizontally, but not vertically.

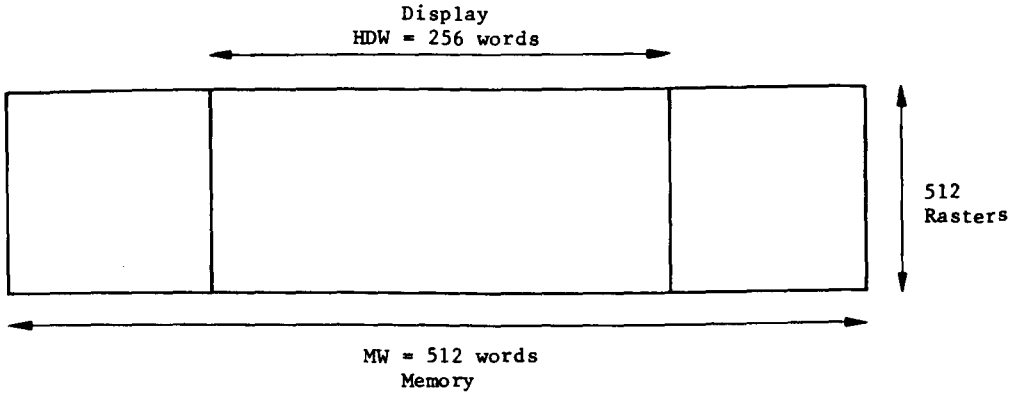


Fig. 7-6(b) Vertical Timing

(iii) If the memory width is made 1.5 times the display width, MW = 1.5 × 256 = 384 words.

Then the frame buffer will support:

$$\frac{256K}{384} = 682.6 \text{ rasters}$$

This will allow the display to be scrolled horizontally and vertically.

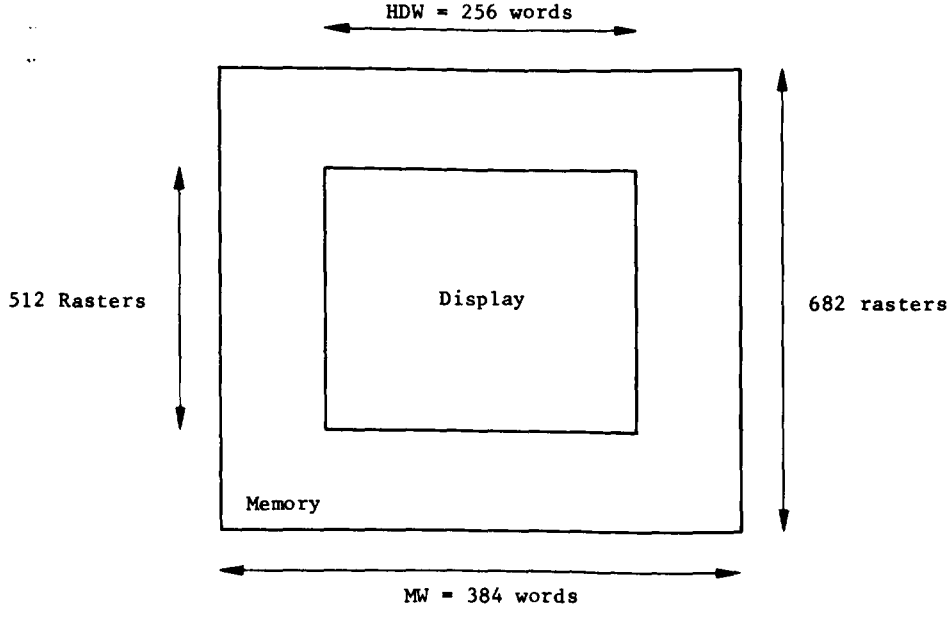


Fig. 7-6(c) Relation between Display Screen and Memory

Choosing the latter result, means that the memory width of the base screen must be set to 384 words:

MW1 = $180

As the base screen is to be defined as a graphics screen, the CHR bit must be 0:

CHR = 0

Combining MW1 & CHR gives:

RCA = $0180

Start Address:

If the display screen is to be positioned centrally in the frame buffer, the screen start address must be offset from that of the frame buffer.

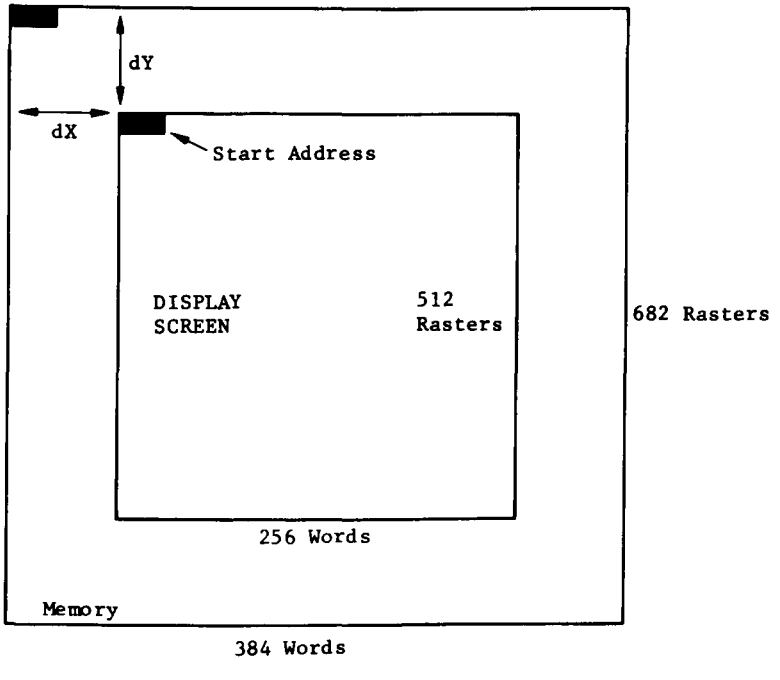


Fig. 7-6(d) Start Address

The offset has both horizontal (dX) and vertical (dY) components:

dX

The memory width is 384 words, the display width is 256 words.

384 - 256 = 128 words total margin

Equally divided between left and right margins gives:

$dX = \frac{128}{2} = 64$ words, the horizontal offset

dY

Likewise, the memory supports 682 rasters, the display uses 512 rasters.

682 - 512 = 170 rasters total margin

Equally divided between top and bottom margins gives:

$dY = \frac{170}{2} = 85$ rasters, the vertical offset

It is now necessary to calculate the word address of the starting point of the screen from these offsets:

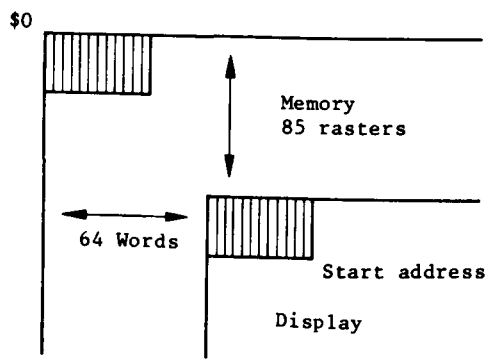


Fig. 7-6(e) Start Address

The start of the 86th raster will be 384 words × 85 = 32640 words from the start of memory. Adding a horizontal offset of 64 words.

32640 + 64 = 32704 words from the start of memory. As the memory starts at address $0:

Name Screen Start Address = $07FC0

This 20 bit value is split between registers RCC and RCE; as no smooth horizontal scrolling is to be applied, the start dot address (SDA) is $0.

Thus

RCC = $0000
RCE = $7FC0

Zoom Vactor - REA

This is the only remaining part of the Display Control RAM that requires initialization for this minimum configuration example. As no zooming is to be applied, both zoom factors are zero.

REA = $0000

This completes the programming of the Display Control RAM.

Table 7-3 shows the values accordingly.

Table 7-3 Display Control RAM (Set Value)

REG NO.	REGISTER		MNEMONIC	BITS	REG	VALUE MNEMONIC	DEC	HEX	NOTES1
RC0	UPPER (Back Ground)	Raster Address 0	RAR0	12∿8 4∿0	Last Raster Address 0 First Raster Address 0	LRA0 FRA0			1
RC2		Memory Width 0	MWR0	15 11∿0	Character/Graphic Memory Width 0	CHR MW0			1
RC4		Start Address 0	SAR0	11∿8 4∿0	Start Dot Address 0 Start Add 0 Hi/Start Raster Add 0	SDA0 SA0H/ SRA0			1
RC6				15∿0	Start Address 0 Low	SA0L			1
RC8	BASE (Back Ground)	Raster Address 1	RAR1	12∿8 4∿0	Last Raster Address 1 First Raster Address 1	LRA1 FRA1			
RCA		Memory Width 1	MWR1	15 11∿0	Character/Graphic Memory Width 1	CHR MW1	0 384	0 180	
RCC		Start Address 1	SAR1	11∿8 4∿0	Start Dot Address 1 Start Add 1 Hi/Start Raster Add 1	SDA1 SA1H/ SRA1	0 0	0 0	
RCE				15∿0	Start Address 1 Low	SA1L	32704	7FC0	1
RD0	LOWER (Back Ground)	Raster Address 2	RAR2	12∿8 4∿0	Last Raster Address 2 First Raster Address 2	LRA2 FRA2			1
RD2		Memory Width 2	MWR2	15 11∿0	Character/Graphic Memory Width 2	CHR MW2			1
RD4		Start Address 2	SAR2	11∿8 4∿0	Start Dot Address 2 Start Add 2 Hi/Start Raster Add 2	SDA2 SA2H/ SRA2			1
RD6				15∿0	Start Address 2 Low	SA2L			1
RD8	WINDOW	Raster Address 3	RAR3	12∿8 4∿0	Last Raster Address Window First Raster Address Window	LRA3 FRA3			1
RDA		Memory Width 3	MWR3	15 11∿0	Character/Graphic Memory Width Window	CHR MW3			1
RDC		Start Address 3	SAR3	11∿8 4∿0	Start Dot Address Window Start Add 3 Hi/Start Raster Add 3	SDA3 SA3H/ SRA3			1
RDE				15∿0	Start Address 3 Low	SA3L			1
RE0	Block Cursor 1		BCUR1	15∿13 12∿8 4∿0	Block Cursor Width 1 Block Cursor Start Raster 1 Block Cursor Start End Raster 1	BCW1 BCSR1 BCER1			1
RE2				15∿0	Block Cursor Address 1	BCA1			1
RE4	Block Cursor 2		BCUR2	15∿13 12∿8 4∿0	Block Cursor Width 2 Block Cursor Start Raster 2 Block Cursor End Raster 2	BCW2 BCSR2 BCER2			1
RE6				15∿0	Block Cursor Address 2	BCA2			1
RE8	Cursor Definition		CDR	15∿14 13∿11 10∿8 5∿3 2∿0	Cursor Mode Cursor On 1 Cursor Off 1 Cursor On 2 Cursor Off 2	CM CON1 COFF1 CON2 COFF2			
REA	Zoom Factor		ZFR	15∿12 11∿8	Horizontal Zoom Factor Vertical Zoom Factor	HZF VZF	0 0	0 0	
REC	Light Pen Address		LPAR	7 4∿0	Character/Graphic Light Pen Address High	CHR LPAH			
REE				15∿0	Light Pen Address Low	LPAL			

Notes:

(1) Need only define if function is to be used, otherwise = X

7.7 Control Registers

The final stage of initialization involves the 3 control registers, CCR, OMR and DCR. The address register does not auto-increment when referencing these control registers, so before each write it is necessary to point to the required control register by suitably loading of the address register.

The preferred order of initialization is:

(1) CCR (R02)
(2) OMR (R04)
(3) DCR (R06)

Together these registers hold 30 fields of control bits and each must carefully be considered in relation to the application. The users manual gives detailed explanations on the function of each field. The following example applies to the example system and represents a simple application for clarity.

(1) Command Control Register (R02)

Command control register (CCR: r02-r03)

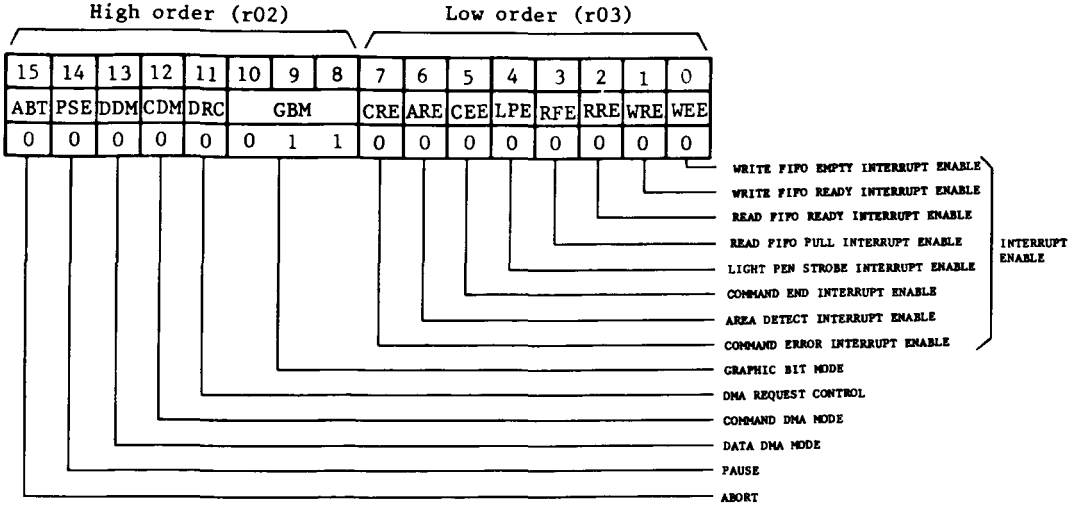


Fig. 7-6(f) Command Control Register

Reset left this register with the value $8000 ie: ABORT set and all others cleared.

Bits 0 - 7:
Enable/Disable the interrupt sources. This example uses polled status to control transfers and so all these can be disabled.

Bits 8 - 10 GBM:
Graphic Bit Mode; this sets the number of bits per pixel. This example uses 8 bits per pixel, and so the mode is "011" i.e.: $3.

Bits 11 - 13:
The DMA control bits; as DMA is not used these are all 0.

Bit 14 PSE Pause:
This bit halts command execution; it must be 0 in order to permit commands to be processed later.

Bit 15 ABT ABORT:
Reset left this bit set; it must now be cleared to enable command execution later.

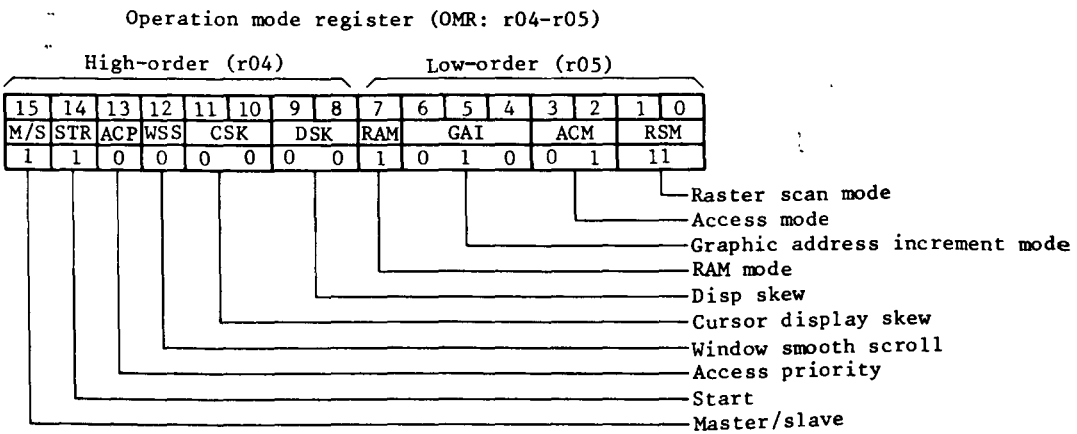
The above values can be written into the CCR in Table 7-4, from which the CCR in Table 7-5 can be completed.

The resulting value is thus:

CCR = $0300

Tables 7-4(a) and 7-5 show the value accordingly.

(2) Operation Mode Register r04



Operation mode register (OMR) values

Fig. 7-6(g) Operation Mode Register

Reset left the two most significant bits cleared, but did not change any of the others.

Bit 0 - 1 Raster Scan Mode:
In order to operate in interface sync and video mode, these must both be set ie: $3.

Bit 2 - 3 ACM Access Mode:
For improved drawing speed the system uses interleaved access mode (DAO), hence these have the value $2.

Bit 4 - 6 GAI Graphic Increment Mode;
The design requires that 64 bits are obtained from the frame buffer per display access, hence the addresses must increment by 4 words, so set GAI = $2.

Bit 7 RAM, RAM Mode:
64k × 1 Dynamic RAMs are used in the frame buffer so refresh must be provided by resetting this bit.

Bit 8 - 11 Skew:
If the external logic delays and access times shift the video signals in relationship to the display signals $\overline{\text{DISP1}}$, $\overline{\text{DISP2}}$, and cursor signals $\overline{\text{CUD1}}$ and $\overline{\text{CUD2}}$, this can be corrected by applying a compensating skew. In that case the horizontal front and back porches are extended and reduced respectively. These horizontal timings may therefore require some adjustment in order to maintain the correct values (HDS, etc.). Assuming no skew is required, all these can be $0. Note that if CSK = $0, cross hair mode cannot be used.

Bit 12 WSS Window Smooth Scroll:
In order to smooth scroll the window data prefetching must be carried out. As no window is to be implemented, this bit can be $0.

Bit 13 ACP Access Priority:
To avoid disruption of the displayed image due to drawing operations, the display process will be given priority over drawing, hence this bit will be $0.

Bit 14 STR Start:
This bit was left cleared by reset, to stop all drawing and displaying. In order to activate these processes, it is necessary to set this bit = $1.

Bit 15 M/S Master/Slave:
As this example system will not be synchronised with an external source, this

bit will be set so that the ACRTC acts in master mode.

Again, Table 7-4(a) and 7-5 can be filled in from these values.

The result is OMR = $C02B.

Tables 7-4(b) and 7-5 show the value accordingly.

(3) Display Control Register R06:

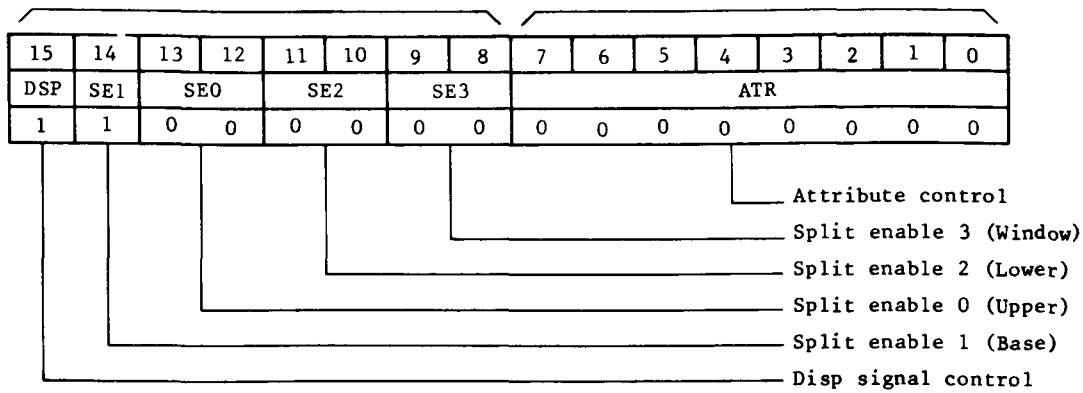


Fig. 7-6(h) Display Control Register

The DCR controls the screen organization and provides for 8 bits of user defined video attributes.

Bits 0 - 7 ATR Attribute Control:
These bits are not used directly by the ACRTC, but are output as the user attributes, together with the other attributes, during horizontal flyback. They are not used in this example and so will be $0, though they can be freely programmed.

Bits 8 - 13 Split Enables:
As these screens are not used and so not defined, they are all cleared to disable these screens.

Bit 14 SE1 Split Enable 1 (base):
This bit enables the base screen. As this screen is in use, it must be set = $1.

Bit 15 DSP, DISP Control:

The $\overline{\text{DISP}}$ signals, together with the $\overline{\text{HSYNC}}$ and $\overline{\text{VSYNC}}$ signals, allow blanking of the video signal and generation of front and back porches. These can be used for driving display monitors.

$\overline{\text{DISP 1}}$ provides a combined horizontal and vertical blanking signal for both background and window screens when this bit is set. In fact, as this example uses only the base screen, the bit could also be cleared, so that $\overline{\text{DISP 1}}$ only applies to the background screen(s) and $\overline{\text{DISP 2}}$ applies to the window.

In order to allow a window screen to be used later, this bit = $1.

Tables 7-4(a) and 7-5 can be completed from the above and $5:

DCR = $C000

Tables 7-4(b) and 7-5 show the value accordingly.

Table 7-4(a) Control Register Table

REG NO.	REGISTER	MNEMONIC	BITS	REG	MNEMONIC
R02	Command Control	CCR	15	Abort	ABT
			14	Pause	PSE
			13	Data DMA Mode	DDM
			12	Command DMA Mode	CDM
			11	DMA Request Control	DRC
			10∿8	Graphic Bit Mode	GBM
			7	Command Error Interrupt Enable	CRE
			6	Area Detect Interrupt Enable	ARE
			5	Command End Interrupt Enable	CEE
			4	Light Pen Strobe Interpt. Enable	LPE
			3	Read FIFO Full Interrupt Enable	RFE
			2	Read FIFO Ready Interrupt Enable	RRE
			1	Write FIFO Ready Interpt. Enable	WRE
			0	Write FIFO Empty Interpt. Enable	WEE
R04	Operation Mode	OMR	15	Master/Slave	M/S
			14	Start	STR
			13	Access Priority	ACP
			12	Window Smooth Scroll	WSS
			11∿10	Cursor Display Skew	CSK
			9∿8	DISP Skew	DSK
			7	RAM Mode	RAM
			6∿4	Graphic Address Increment Mode	GAI
			3∿2	Access Mode	ACM
			1∿0	Raster Scan Mode	RSM
R06	Display Control	DCR	15	DISP Signal Control	DSP
			14	Split Enable 1	SE1
			13∿12	Split Enable 0	SE0
			11∿10	Split Enable 2	SE2
			9∿8	Split Enable 3	SE3
			7∿0	Attribute Control	ATR

Table 7-4(b) Control Register Table (Set Value)

REG NO.	REGISTER	MNEMONIC	BITS	REG	MNEMONIC	LOAD VALUE
R02	Command Control	CCR	15	Abort	ABT	0
			14	Pause	PSE	0
			13	Data DMA Mode	DDM	0
			12	Command DMA Mode	CDM	0
			11	DMA Request Control	DRC	0
			10∿8	Graphic Bit Mode	GBM	3
			7	Command Error Interrupt Enable	CRE	0
			6	Area Detect Interrupt Enable	ARE	0
			5	Command End Interrupt Enable	CEE	0
			4	Light Pen Strobe Interpt. Enable	LPE	0
			3	Read FIFO Full Interrupt Enable	RFE	0
			2	Read FIFO Ready Interrupt Enable	RRE	0
			1	Write FIFO Ready Interpt. Enable	WRE	0
			0	Write FIFO Empty Interpt. Enable	WEE	0
R04	Operation Mode	OMR	15	Master/Slave	M/S	1
			14	Start	STR	1
			13	Access Priority	ACP	0
			12	Window Smooth Scroll	WSS	0
			11∿10	Cursor Display Skew	CSK	0
			9∿8	DISP Skew	DSK	0
			7	RAM Mode	RAM	1
			6∿4	Graphic Address Increment Mode	CAI	2
			3∿2	Access Mode	ACM	1
			1∿0	Raster Scan Mode	RSM	3
R06	Display Control	DCR	15	DISP Signal Control	DSP	1
			14	Split Enable 1	SE1	1
			13∿12	Split Enable 0	SE0	0
			11∿10	Split Enable 2	SE2	0
			9∿8	Split Enable 3	SE3	0
			7∿0	Attribute Control	ATR	0

Table 7-5 Programming Chart Control Register (Set Value)

REGISTER	MNEMONIC	REG NO.	NOTES	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	VALUE HEX
Command Control	CCR	R02		ABT	PSE	DDM	CDM	DRC	GBM			CRE	ARE	CEE	LPE	RFE	RRE	WRE	WEE	
				0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	$0300
Operation Mode	OMR	R04		M/S	STR	ACP	WSS	CSK		DSK		RAM	GAI			ACM		RSH		
				1	1	0	0	0	0	0	0	1	0	1	0	0	1	1	1	$C0A7
Display Control	DCR	R06		DSP	SE1	SE0		SE2		SE3		ATR								
				1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$C000

Volume 2

Hardware Section

Page 123

•

Volume 3

Software Section

Page 193

HITACHI

This page intentionally left blank.

This section is intended to show some examples in designing graphic/character display system equipment and developing software using the ACRTC. A normal CRT capable of displaying of 15 colors, 640 × 400 raster lines in a non-interlace method is employed as the target display device to show the application examples.

The ACRTC is a high-performance, highly-functional graphic display controller which has the following key functions;

1. High-speed graphics drawing.
2. Various display functions such as CRTC control timing, split screen, smooth scroll and zoom.
3. Figure drawing such as circles, ellipses, painting and copying.
4. Various character display control.

The following shows the CRT timing used as the circuit example in this application note. CRT timing which is not used in this application note is omitted.

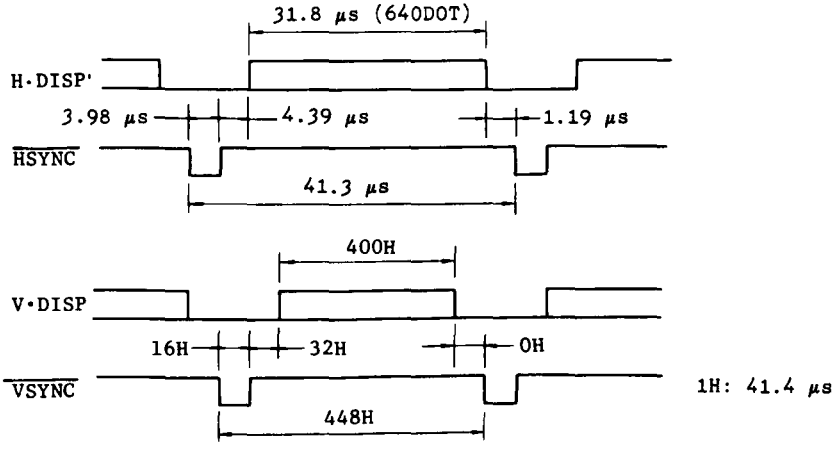


Figure 0-1. CRT Display Timing

This page intentionally left blank.

Volume 2
Hardware

Table of Contents

1. BASIC DESIGN OF THE ACRTC SYSTEM 123
 1.1 Basic Design 123
 1.2 Design Example 130
2. INTERFACE WITH 16-BIT MPU 132
 2.1 Connection to the HD68000/HD8HC000 132
 2.1.1 MPU Read 132
 2.1.2 MPU Write 134
 2.1.3 Interrupt Generation Circuit 135
 2.2 Connection to the HD68450/HD63450 137
3. INTERFACE WITH 8-BIT MPU 146
 3.1 Connection to the HD6809 146
 3.2 Connection to the HD6844 148
4. CRT INTERFACE 150
 4.1 Dot Clock, 2CLK, Load Signal Generation Circuit 150
 4.2 Video Signal Generation Circuit 151
5. FRAME BUFFER 155
 5.1 Memory Organization 155
 5.1.1 Frame Buffer for Graphics Display 155
 5.1.2 Refresh Memory for Character Display 157
 5.2 Memory Access 161
 5.2.1 DRAM Access 161
 5.2.2 SRAM Access 165
6. ATTRIBUTES 167
 6.1 Fetching the Attribute Control Signal 167
 6.2 Smooth Scroll 169
 6.3 Zooming Display 172
 6.4 Cursor 174
 6.4.1 Block Cursor 174
 6.4.2 Cross-hair Cursor 177
 6.4.3 Graphics Cursor 181
 6.5 Blink and Split Control 184
7. CIRCUIT EXAMPLE FOR 16K × 4 DRAM INTERFACE 186
 Application Circuit (a) 186
 Application Circuit (b) 188

This page intentionally left blank.

1. BASIC DESIGN OF THE ACRTC SYSTEM

1.1 Basic Design

Key factors in designing the ACRTC systems are the CRT display timing and the frame buffer memory timing.

When controlling the CRT display with the ACRTC, the main operation performed by the ACRTC is to generate

1 the synchronization signal for controlling the CRT display

2 the frame buffer address to read the display data

based on the 2CLK input to the ACRTC.

The CRT display timing and the memory timing of the frame buffer are defined by the relationship between the following items.

- ACRTC and the display timing
- ACRTC access mode and operation
- ACRTC and the memory access time

1) ACRTC and the display timing

The display data, which are read out from the frame buffer memory based on the display address, are input to the parallel-serial conversion circuit. After synchronizing with the dot clock, the serial data is provided to the CRT display. This process is shown in Fig. 1-1.

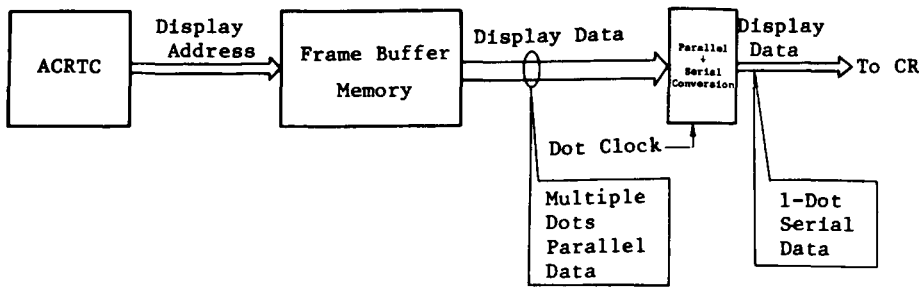


Fig. 1-1 Displaying Memory Data

The dot clock frequency is decided by the number of dots displayed during one horizontal scan period.

For example, when using a CRT of 31.8 μs/one horizontal display period, the dot clock cycle time is [31.8 us/640 pixels = 49.69 ns], and the frequency is 20.126 MHz.

2) ACRTC access mode and the display operation

There are two access modes available for the ACRTC according to how many times the memory is accessed in one display period.

(1) single access mode: the display address is generated once in 1 display period.

(2) dual access modes : one is the superimpose mode which generates the display address twice. The other is interleave mode which generates the display address and drawing address for implementing parallel display and drawing operation.

Each mode is shown in Fig. 1-2. The amount of data read from the memory by one display address is the number of dots displayed during one display period.

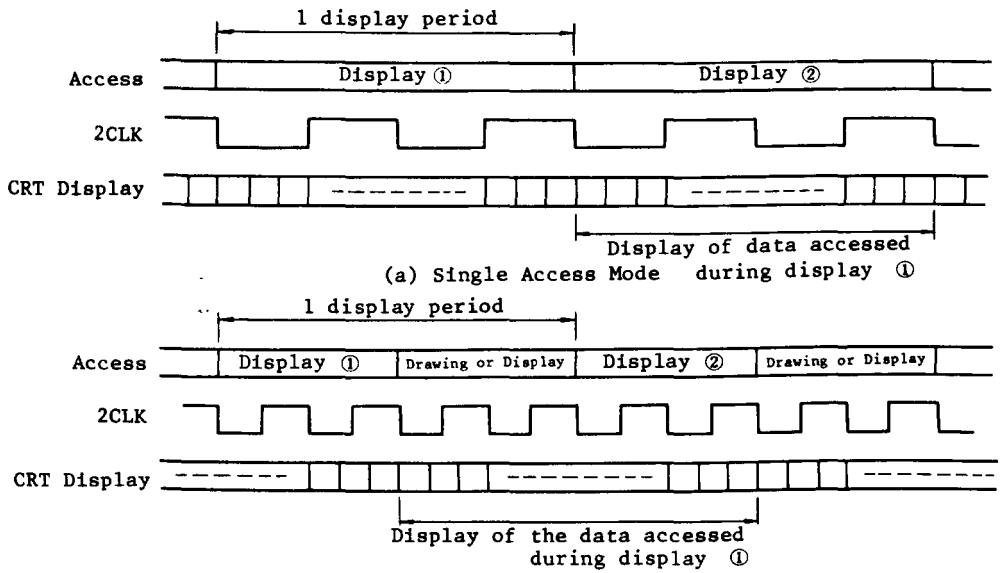


Fig. 1-2 Relationship Between Memory Access Mode and 2CLK

3) ACRTC and memory access time

In the ACRTC, the memory cycle time Ts is expressed by the following equations, where,

T_H: one horizontal display period

Nd: number of dots displayed during one horizontal display period

Ns: number of dots shifted out during one display period.

$$Ts(ns) = \frac{T_H\ (\mu s) \times 1000}{Nd\ (dot)} \times Ns(dot) \quad ; \text{ single access mode}$$

$$Ts(ns) = \frac{T_H\ (\mu s) \times 1000}{Nd\ (dot)} \times Ns(dot) \times \frac{1}{2} \quad ; \text{ dual access mode}$$

* For example, when displaying 16 dots during one display period, Ts is calculated as follows;

$$Ts(ns) = \frac{31.8 \times 1000}{640} \times 16 = 795(ns) \quad ; \text{ single access mode}$$

$$Ts(ns) = \frac{31.8 \times 1000}{640} \times 16 \times \frac{1}{2} = 397.5(ns) \quad ; \text{ dual access mode}$$

In the dual access mode, high speed drawing is performed without flickering, but note that the memory cycle time is descreased to half. If the access time is too short, it should use a high speed memory with shorter access time (cycle time) or increase the shift quantity of the parallel/serial converter.

2CLK, which is the basic clock for the memory access, is generated by dividing the dot clock. For example, when displaying 16 dots in one display period, 2CLK is made by dividing the dot clock by 8 (single access mode) or 4 (dual access mode) to get 397.5 ns or 198.76 ns, respectively.

However, by using the address increment mode of the ACRTC which increases the data quantity read from the frame buffer, the memory access time can be lengthened. This process is shown in Fig. 1-2.

The relationship between the shift quantity during the one display period and division of the clock frequency is indicated in table 1-1. When reading 32 dots, 2CLK remains 397.5 ns even in the dual access mode.

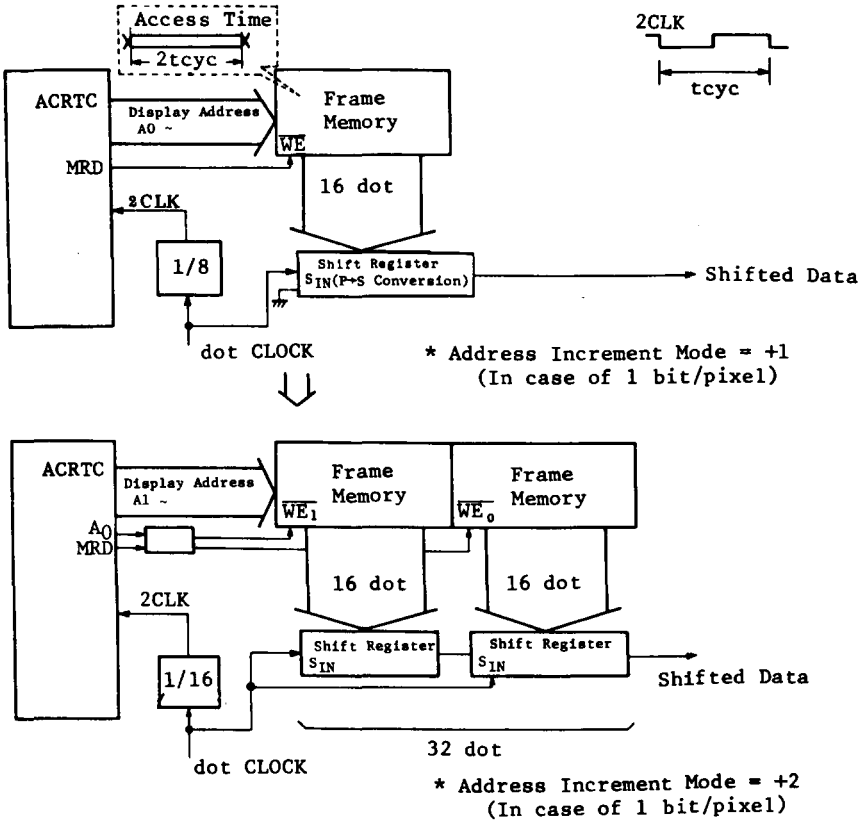


Fig. 1-3 Relation between 2CLK and the Read Data

Table 1-1 Generating 2CLK

Shifted dots. / Access mode	4	8	16	32	64	128
Single access	÷2	÷4	÷8	÷16	÷32	÷64
Dual access	÷1	÷2	÷4	÷8	÷16	÷32

When the cycle time of 2CLK is decided, the ACRTC speed version is decided by the following equation;

$$X\ (\mathrm{MHz}) \geq \frac{2}{Ts\ (\mathrm{ns})}$$ where X is the speed version (X=4, 6, 8)

In the dual access mode,

$X \geq 2/397.5$ ns (=5.03 MHz). Therefore, X is 6 or 8 MHz version.

In the single access mode,

$X \geq 2/795$ ns (=2.52 MHz). Therefore, X is 4, 6 or 8 MHz version.

The number of data bits read during one display period is shown in Table 1-2 according to the bit mode which specifies the data organization for one pixel and the shift quantity of the parallel/serial converter.

Table 1-2 Bit Width of Data Read during 1 Display Period

Shift quantity	4(dots)	8(dots)	16(dots)	32(dots)	64(dots)
1 (bit/pixel)	*	16〔+1/2〕	16〔+2〕	32〔+2〕	64〔+4〕
2 (bit/pixel)	16〔+1/2〕	16〔+1〕	32〔+2〕	64〔+4〕	128〔+8〕
4 (bit/pixel)	16〔+1〕	32〔+2〕	64〔+4〕	128〔+8〕	256〔+16〕
8 (bit/pixel)	32〔+2〕	64〔+4〕	128〔+8〕	256〔+16〕	*
16(bit/pixel)	64〔+4〕	128〔+8〕	256〔+16〕	*	*

Note 1) The unit is 1 bit.

Note 2) 〔 〕shows the address increment mode.

Note 3) When using *, the address lines and the data lines must be connected in a special way.

Pixel and the Display Data

In the case of 4 bits/pixel mode with 16 dots shifted, the memory configuration which enables 64 bit data read from the frame buffer at one time needs to be prepared. In this case, the display address is automatically incremented by 4 according to the setup value of the internal register of the ACRTC (the address increment mode). Fig. 1-4 shows a block diagram of graphic display of 4 bits/pixel with 16 dots shifted.

Note) Note that the frame buffer memory is partitioned because a different quantity of data is accessed for displaying and drawing as shown in Fig. 1-4. In addition, note that the bus buffer is used to separate the bus.

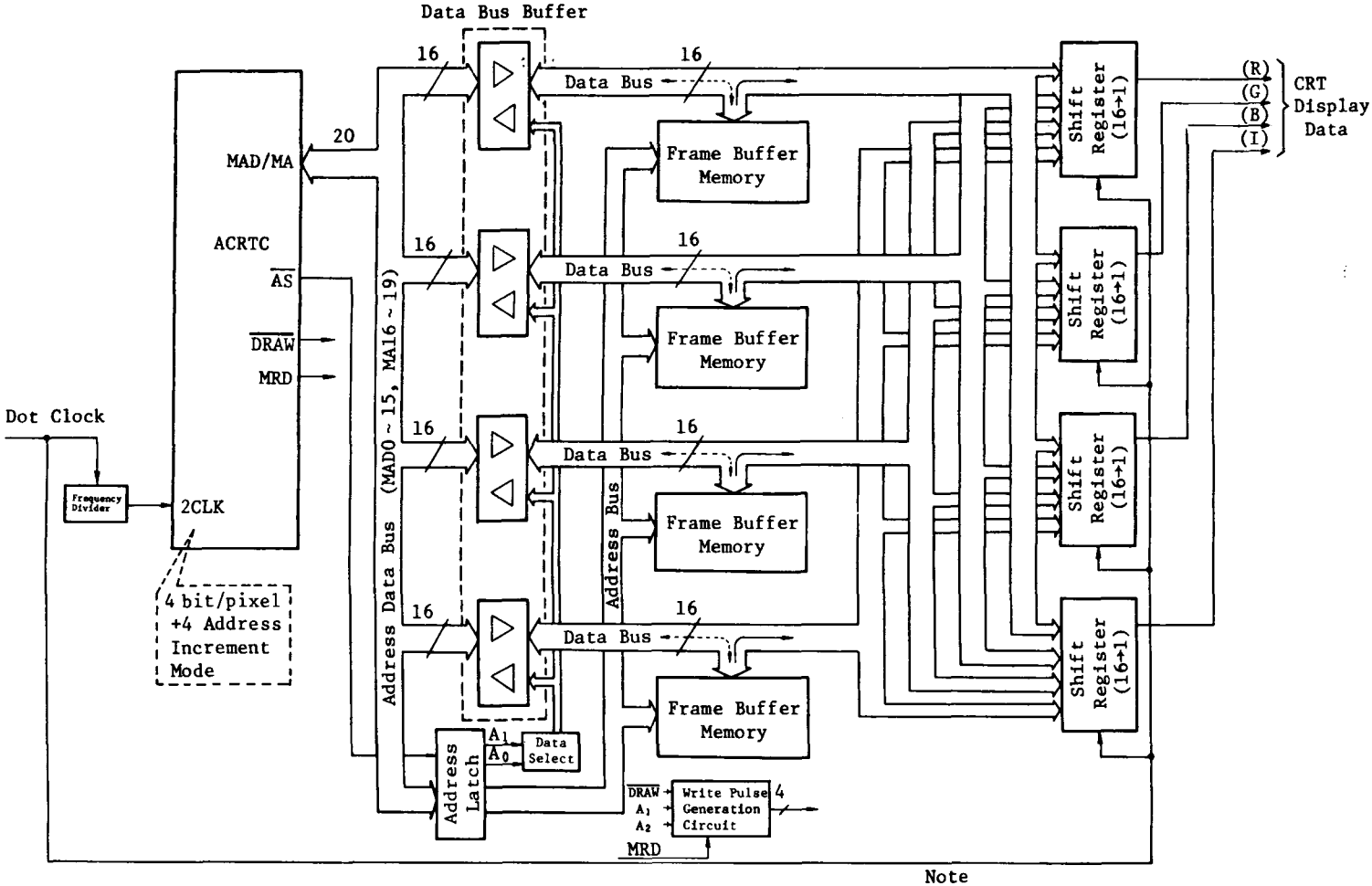


Fig. 1-4 Graphic Display Block Diagram

〔Note〕

ACRTC Frame Buffer

When the ACRTC reads the data from the frame buffer, MAD0 and MAD1 are treated as 'don't care' because the physical address bus of the ACRTC, MAD0 and MAD1, is not connected directly to the address input (A0 ∿ A19: Memory) of the memory. In this case, 16 bits x 4 = 64 bits are read. This procedure is shown in Fig. 1-5. As the data quantity which the ACRTC accesses at a time is 16 bits, the data are selected by MAD0 and MAD1.

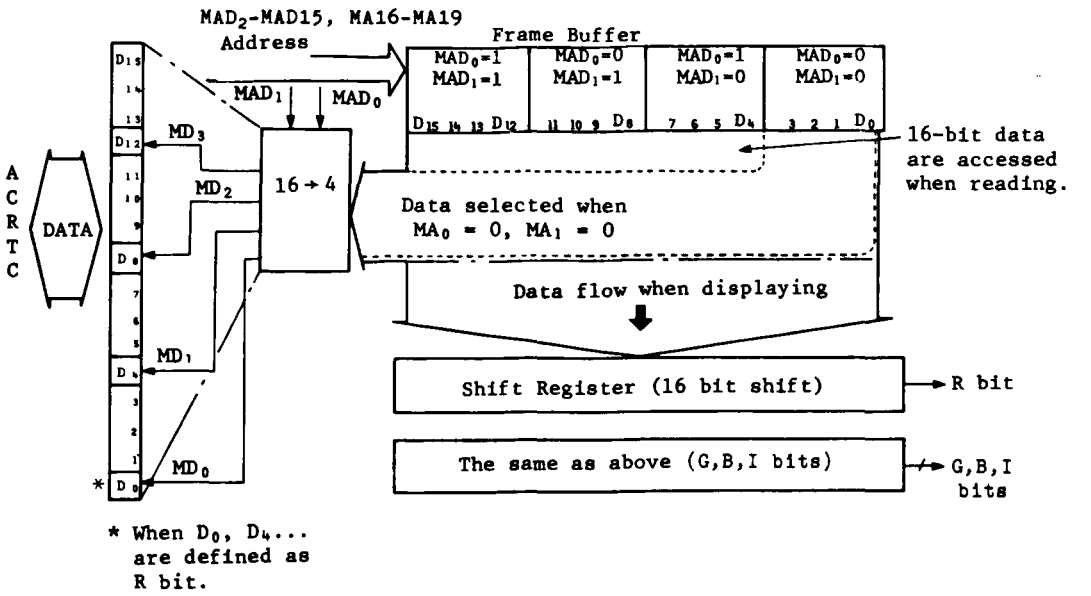


Fig. 1-5 Relation between ACRTC and Pixels (Frame Buffer Data)

Four pixels can be placed in one word, therefore, 4-pixel data are handled at a time. If 1-dot pixel data, D0 bit, is defined as R(Red) bit, 1-word data corresponds with D4, D8 and D12.

ACRTC Frame Buffer (Write)

When the ACRTC writes the data to the memory, write pulse of $\overline{WE}$ is generated by using the MRD signal, the $\overline{DRAW}$ signal, MAD0, and MAD1.

1.2 Design Example

Table 1-3 summarizes the specifications defined in section 1.1.

Table 1-3 Basic Target Specifications of the ACRTC System

No.	Items	Specifications	Remarks
1	CRT display	640 dots x 400 rasters, 15 colors	
2	Dot clock	20.126 MHz	
3	2CLK	5.03 MHz	
4	ACRTC type No.	HD63484-8	
5	Shift quantity	16 dots	
6	Access mode	Dual access mode (Superimpose, Interleave)	
7	Graphic bit mode	4 bits/pixel	
8	Address increment mode	+4 increment	
9	MPU	HD68HC000	
10	Memory type No.	HM50464-12x4 (128KB)	1 CRT screen

The ACRTC provides many display control functions: zooming, horizontal smooth scroll, superimposing two screens. These features can be realized by a small amount of additional external circuits. Users can select which function to use when deciding the system specification. Table 1-4 shows a specification example.

Table 1-4 ACRTC System Specification Example

No.	Item	Specifications	Remarks
1	Character display by character generator	Configuration : 16 dots x 16 rasters (Chinese letters) CRT display : 40 x 25 characters Frame buffer for characters : 2 screens HM6148HP-35 x 8 (4KB)	Characters can be displayed on all the split screens
2	Super-impose	Characters can be superimposed over graphics on the screen	Dual access mode
3	Scroll control	Vertical and horizontal smooth scroll in the base screen or upper screen or lower screen Vertical smooth scroll and horizontal scroll 4 pixel increment in the window	
4	Zoom (enlarge-ment)	1 to 16 times (horizontal and vertical enlargement selectable.)	Base screen only

No.	Item	Specifications	Remarks
5	Cursor display	2 cross hair cursors can be displayed. 1 graphic cursors can be displayed. (size: 8 × 16 ∿ 32 × 4 dots) 2 character cursors can be displayed. Each cursor can blink (Blink speed is selectable). The display position of the cross hair cursor and the graphic cursor can be specified (smooth scroll possible).	
6	DMA transfer	HD63450 (8MHz) DMA mode: Data DMA burst mode Data DMA cycle steal mode Command DMA mode Note)	
7	Interrupt control	Interrupt by the DMAC Interrupt by the ACRTC	
8	Blink	Blink 1 : Whole screen blink for base screen, upper screen, or lower screen (Screen No. is selectable by a jumper line) Blink 2 : All screens blink.	

2. INTERFACE WITH 16-BIT MPU

2.1 Connection to the HD68000/HD68HC000 (16-bit Asynchronous Bus)

The ACRTC provides data transfer acknowledge signal ($\overline{DTACK}$), by which the ACRTC can easily be connected with the HD68000/HD68HC000 using an asynchronous bus (see Fig. 2-1).

In this case, $\overline{CS}$ of the ACRTC is acquired by decoding the HD68000/HD68HC000's FC0 FC1, FC2 $\overline{AS}$, $\overline{LDS}$, $\overline{UDS}$ and address lines. As the ACRTC cannot generate the interrupt vector, external vector generation circuit is required.

2.1.1 MPU Read

Fig. 2-2 shows the MPU read cycle timing of the ACRTC with 16-bits bus.
Fig. 2-3 shows the HD68000/HD68HC000 read cycle timing.

When the ACRTC receives the $\overline{CS}$ signal, it outputs 16-bit data on D0 ∿ D15 in the T2 cycle. The ACRTC asserts $\overline{DTACK}$ in the T3 cycle to inform the MPU of data output. The HD68000/HD68HC000 detects $\overline{DTACK}$ being asserted in the S4 cycle and negates US, $\overline{UDS}$ and $\overline{LDS}$. If $\overline{DTACK}$ is not acknowledged in S4, the HD68000/HD68HC000 goes into the wait cycle. When $\overline{CS}$ is negated, the ACRTC stops data output and negates $\overline{DTACK}$.

As shown above, the ACRTC doesn't need to synchronize with the HD68000/HD68HC000 because the difference in the clock frequency is absorbed by the wait cycle. If the frequency of the CLK of MPU and the 2CLK of ACRTC are the same, three to four wait cycles are usually inserted.

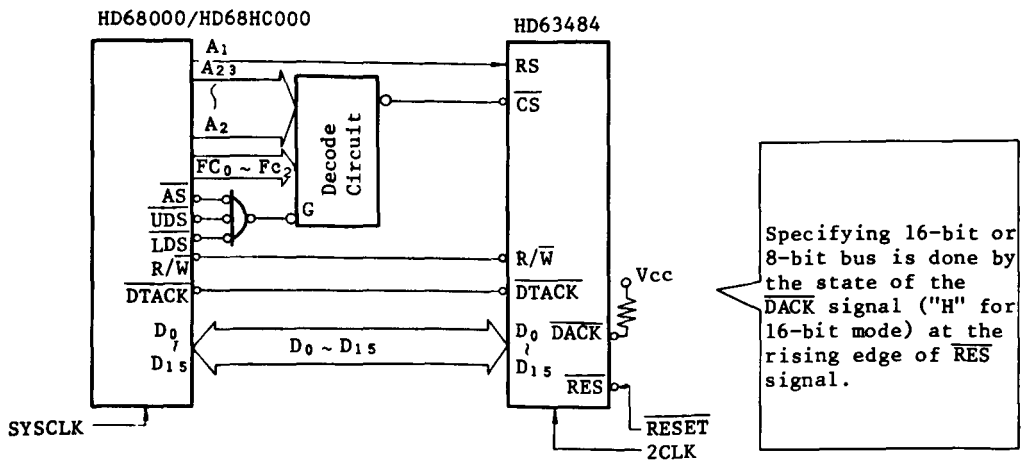


Fig. 2-1 Bus Connection Example with the HD68000 (16-bit Bus)

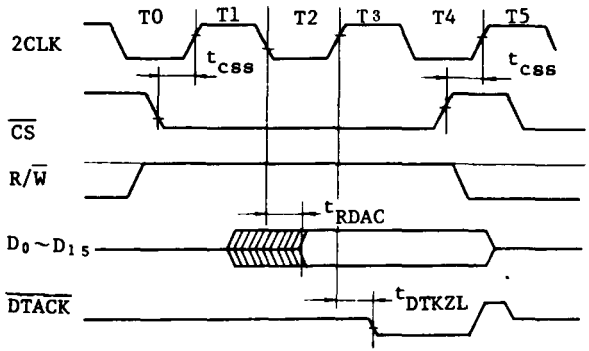


Fig. 2-2 ACRTC Read Cycle Timing: 16-bit Asynchronous Bus (MPU←ACRTC)

Note 1) When deciding the timing of $\overline{BERR}$ signal generation circuit, the phase difference between the HD68000/HD68HC000 CLK and the ACRTC 2CLK needs to be considered.

Note 2) Signals FC0, FC1, and FC2 must be used by the $\overline{CS}$ decoder to prevent $\overline{CS}$ assertion during interrupt acknowledge cycle.

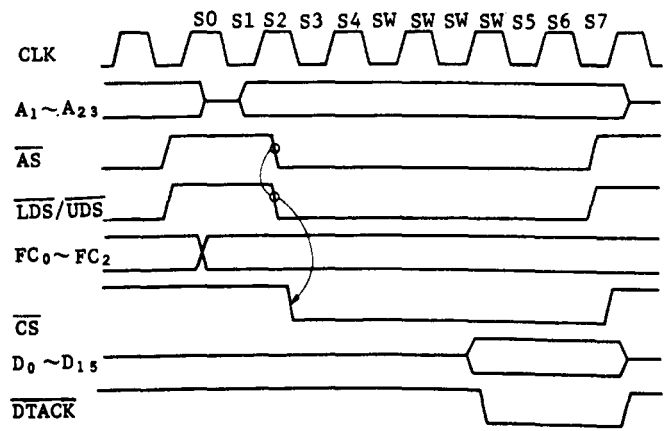


Fig. 2-3 HD68000/HD68HC000 Read Cycle Timing

2.1.2 MPU Write

Fig. 2-4 shows the MPU write cycle timing of the ACRTC when the data bus is 16-bit. Fig. 2-5 shows the HD68000/HD68HC000 write cycle timing. The HD68000/HD68HC000 asserts $\overline{UDS}$ and $\overline{LDS}$ in the S4 cycle. After receiving $\overline{CS}$, ACRTC latches the 16-bit data and asserts $\overline{DTACK}$. The HD68000/HC68HC000 detects $\overline{DTACK}$, terminates data send, and negates the bus control signals.

In the MPU mode, a wait cycle will be inserted if the $\overline{DTACK}$ is not generated at S4.

If a delayed $\overline{AS}$ is used to generate the ACRTC $\overline{CS}$ without using the $\overline{UDS}$ and the $\overline{LDS}$, the number of wait cycles is decreased.

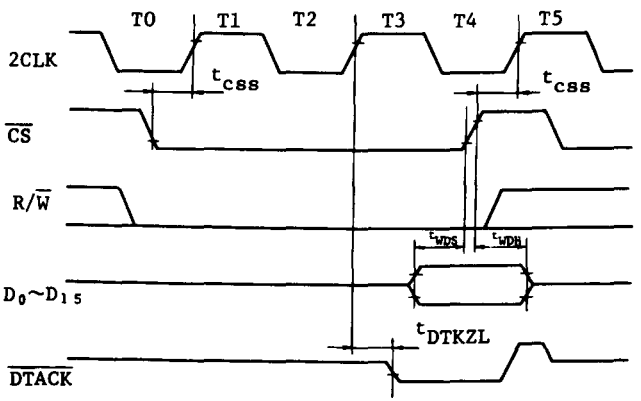


Fig. 2-4 ACRTC Write Cycle Timing: 16-bit Asynchronous Bus (MPU→ACRTC)

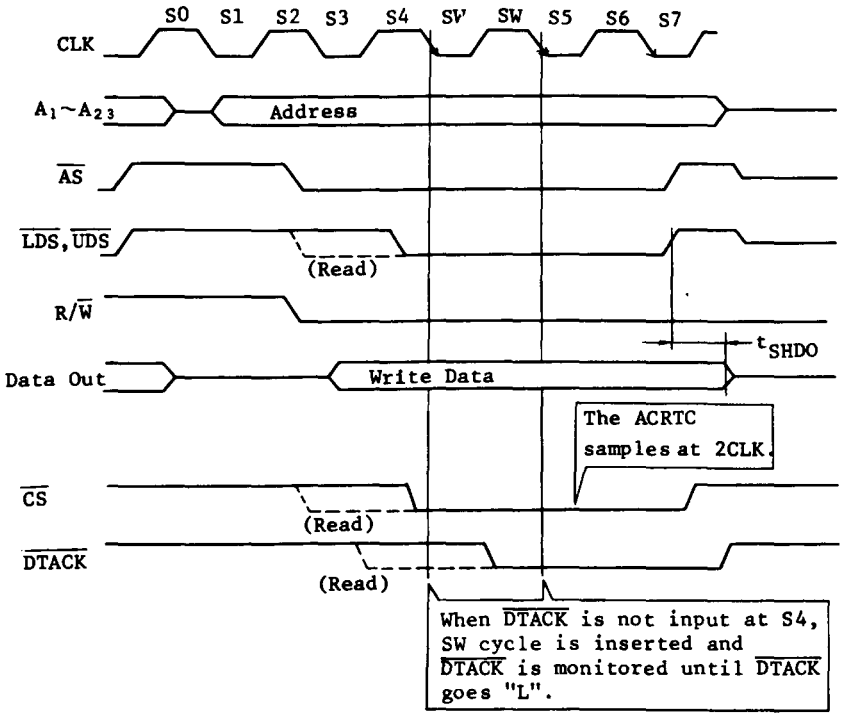


Fig. 2-5 HD68000/HD68HC000 Write Timing

Note) In the HD68000/HD68HC000 interface, the $\overline{CS}$ should not be generated by decoding the address only. Be sure to decode using the control signals such as $\overline{AS}$, $\overline{LDS}$ or $\overline{UDS}$ together with the address signals. In particular, be sure to meet the data hold time for the $\overline{CS}$ rising edge, when the ACRTC fetches data in MPU write cycle.

2.1.3 Interrupt Generation Circuit

The ACRTC can generate an interrupt with the $\overline{IRQ}$ output, however, it can not generate an interrupt vector so, it is necessary to have a vector generation circuit. Fig. 2-6 shows an example of the interrupt generation circuit.

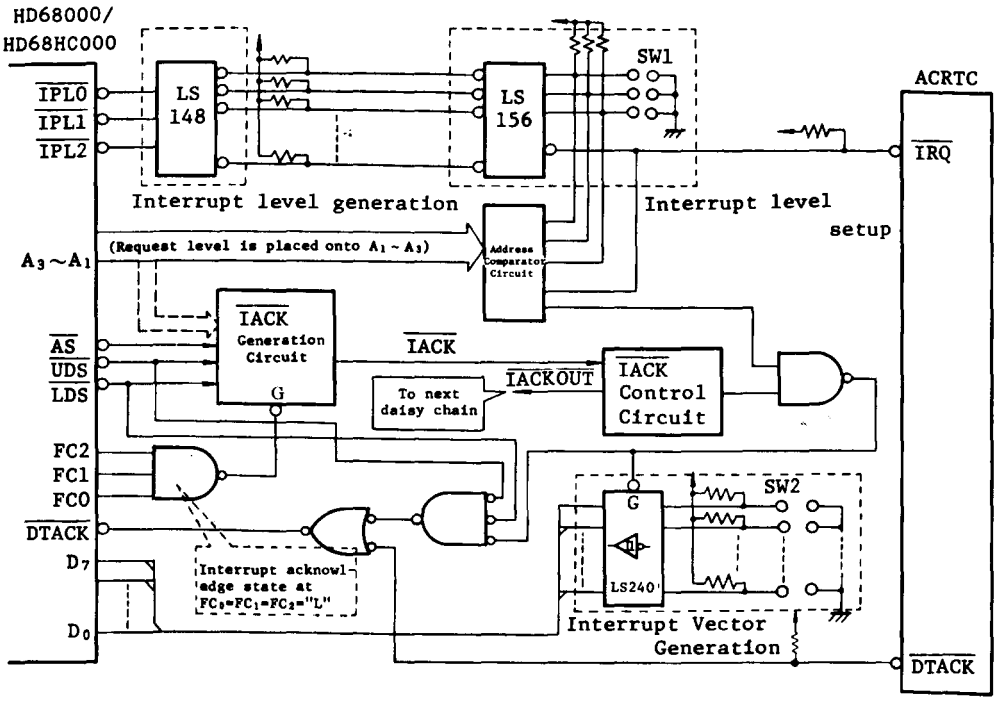


Fig. 2-6 The Example of Interrupt Signal Generation Circuit

In the circuit example of Fig. 2-6, if $\overline{IRQ}$ of the ACRTC is asserted, the interrupt level set by SW1 is input to the HD68000/HD68HC000 by the $\overline{IPL0}$, $\overline{IPL1}$, and $\overline{IPL2}$ lines. After completing the current instruction execution, the HD68000/HD68HC000 performs an interrupt acknowledge sequence. At this time, the interrupt level is output onto A_1 through A_3 by HD68000 and HD68HC000. This interrupt level is compared with the level set by SW1, by the address comparator circuit. If matched, the vector set in SW2 is output to the data bus. As the level signals showing the interrupt acknowledge state are generated on FC0, FC1, and FC2, the interrupt acknowledge timing is obtained, by using these function codes, $\overline{AS}$, $\overline{LDS}$, and $\overline{UDS}$. This timing signal must be used to generate the interrupt vector, and $\overline{DTACK}$ signal.

Interrupt control circuit organized as a daisy chain is required to prevent the contention of interrupts with the same level.

An 8 bit vector is placed on D_0 through D_7 in the interrupt acknowledge cycle, but signals D_8 through D_{15} are not used by the MPU. Therefore, $\overline{UDS}$ is not important in the interrupt acknowledge cycle, but as shown in Fig. 2-7, HD68000/HD68HC000 asserts $\overline{UDS}$ during this cycle.

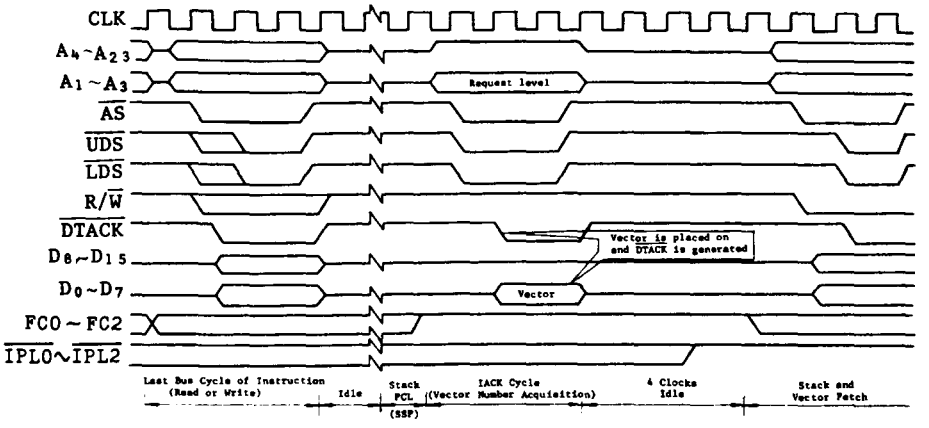


Fig. 2-7 Interrupt Acknowledge Timing

2.2 Connection to HD68450/HD63450: Direct Memory Access Controller (DMAC)

The ACRTC has cycle steal and burst modes for DMA data transfer. High speed data or parameter transfer is possible by using an external DMA controller such as HD68450/HD63450. Both modes are realized with the same hardware, and either can be selected by only setting the registers inside the ACRTC and DMAC.

During DMA transfer, the ACRTC is selected not by $\overline{CS}$ but $\overline{DACK}$. Data transfer between the ACRTC and the main memory is performed by the DMA transfer request ($\overline{DREQ}$) and acknowledge signal ($\overline{DACK}$). The single address mode with $\overline{ACK}$ and $\overline{READY}$ of HD68450/HD63450 DMAC can be used. Fig. 2-8 shows the basic sequence and the data flow when DMA transfer is performed between the ACRTC and the main memory.

As there are two potential bus masters, the MPU and the DMAC, in the system, it is necessary to control the direction of the address bus and the data bus. Fig. 2-9 shows an example block diagram, and Table 2-1 shows the direction control logic for data bus and address bus.

Table 2-1 The direction of the data bus and address bus

Item	Bus master	Access (transfer) direction	DATA bus direction	Address bus direction	Logic
1	MPU	HD68000/HD68HC000 read the ACRTC	MPU ← ACRTC	MPU → ACRTC	$R/\overline{W}\cdot\overline{CS}$ (ACRTC)
2	MPU	HD68000/HD68HC000 writes the ACRTC	MPU → ACRTC	MPU → ACRTC	$\overline{R/\overline{W}\cdot\overline{CS}}$
3	MPU	Interruption acknowledge cycle (response from HD68000/HD68HC000)	MPU ← ACRTC	MPU → ACRTC	$R/\overline{W}\cdot\overline{IACK}$
4	DMAC	DMAC reads from the ACRTC	Memory←ACRTC	Memory←DMAC	$R/\overline{W}\cdot\overline{BGACK}$
5	DMAC	DMAC writes to the ACRTC	Memory→ACRTC	Memory←DMAC	$\overline{R/\overline{W}\cdot\overline{BGACK}}$
6	Other bus master	Access from other bus master			Same as Items 1 and 2

(note) Single addressing mode with $\overline{ACK}$ and $\overline{READY}$ is used.

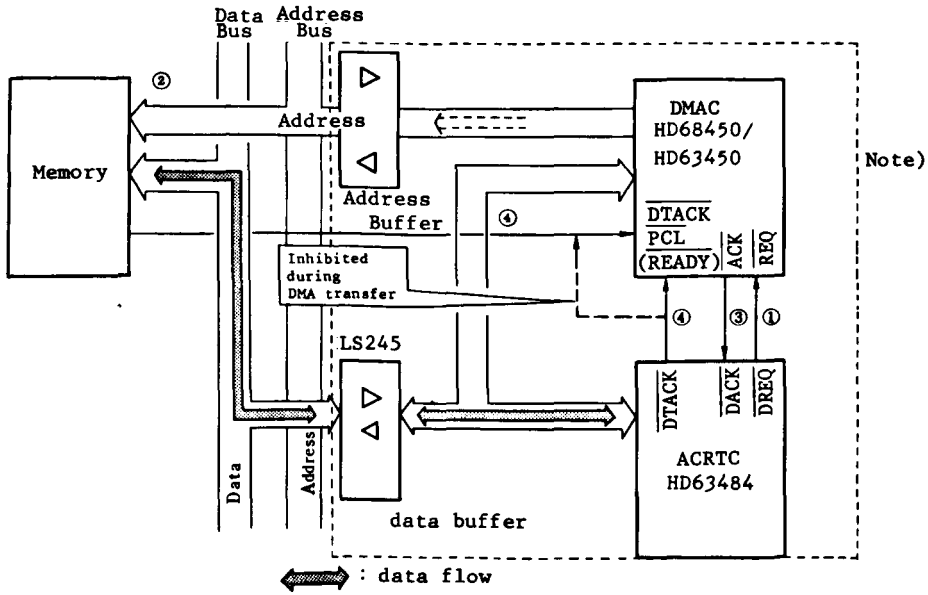


Fig. 2-8 DMA Data Transfer

Basic sequence

1 Transfer request is generated by the ACRTC.

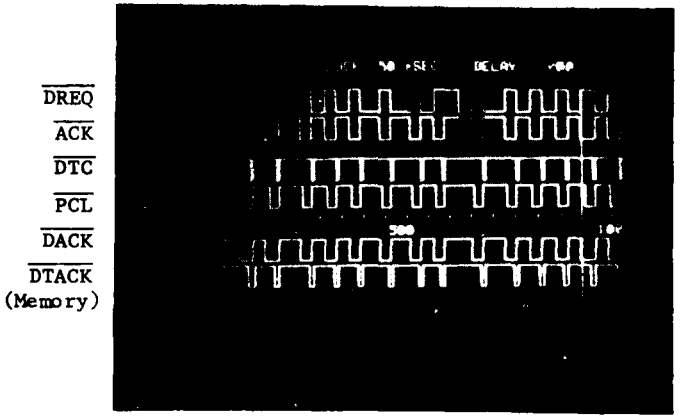
2 DMAC acquires bus mastership and addresses the main memory.

3 $\overline{ACK}$ is output by the DMAC to access the ACRTC.

4 Memory returns $\overline{DTACK}$, and ACRTC applies $\overline{READY}$ ($\overline{DTACK}$) to $\overline{PCL}$ pin of the DMAC to indicate the end of the data transfer bus cycle.

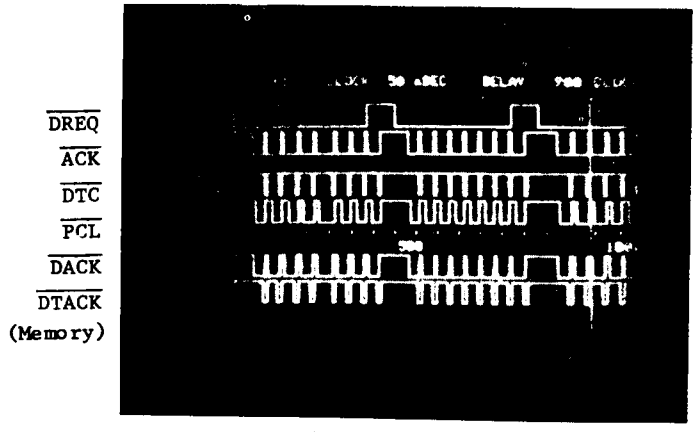
Note 1) In this example, the ACRTC and the DMAC are mapped as I/O for the system bus as shown in Fig. 2-8.

Note 2) DMA transfer of the ACRTC is classified into 2 modes, cycle steal and burst, as shown in Picture 2-1. Both modes can be used by the same hardware. However, DMA transfer cannot be performed when DMAC is in cycle steal mode and the ACRTC is in burst mode. Both ACRTC and DMAC must be set to the same mode.



(a) Cycle Steal Mode

Picture 2-1. DMA Transfer Timing



(b) Burst Mode

Picture 2-1 DMA Transfer Timing

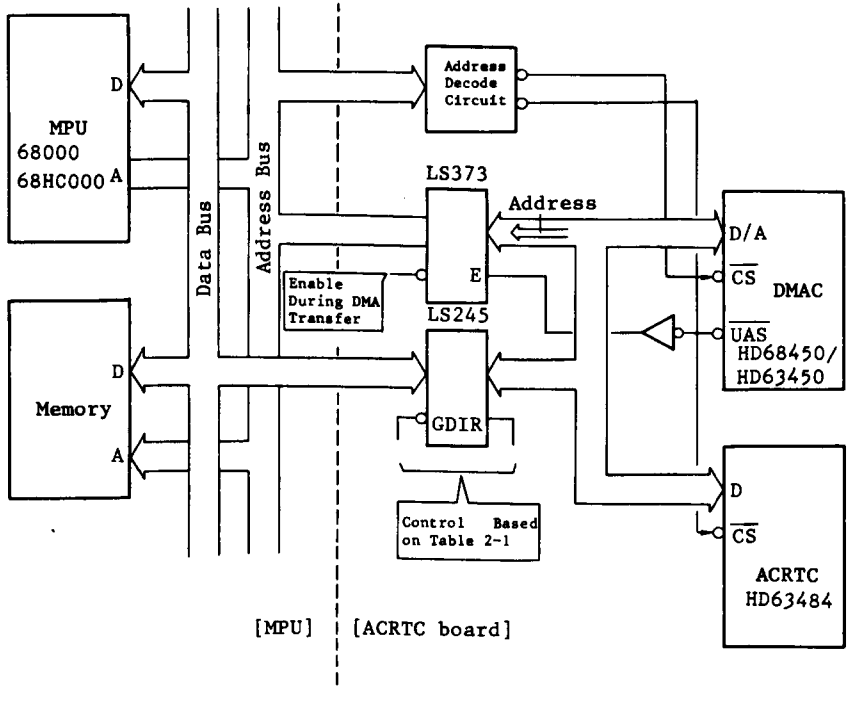


Fig. 2-9 Block Diagram of Bus Control in the DMA

When the MPU is a the bus master, it controls the DIR signal of the data bus buffer with the R/$\overline{W}$ signal. When the DMAC is the bus master, the ACRTC must acknowledge R/$\overline{W}$ in a reversed polarity, compared with the former case. This inverse capability is embodied in the ACRTC and is automatically done internally. Fig. 2-10 shows an example of a bus control circuit and fig. 2-11 shows the address output timing from DMAC. During DMA data transfer, DMAC outputs the address (A_8 through A_{23}: a multiplexed bus) at the cycle of CLK = 1, 2, 3, regardless of the data direction.

According to the block diagram of Fig. 2-9, LS245 is set to high impedance and the address from the DMAC is latched during this cycle.

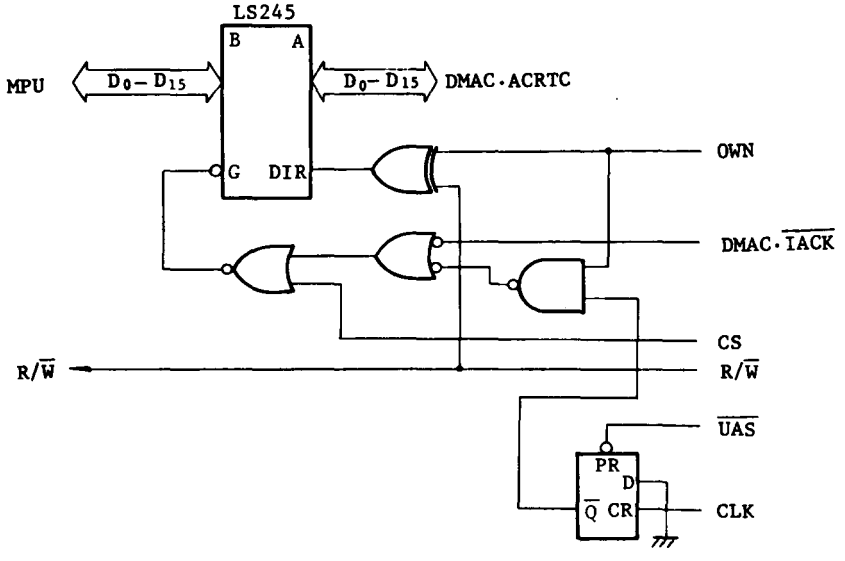


Fig. 2-10 The Example of Bus Control Circuit

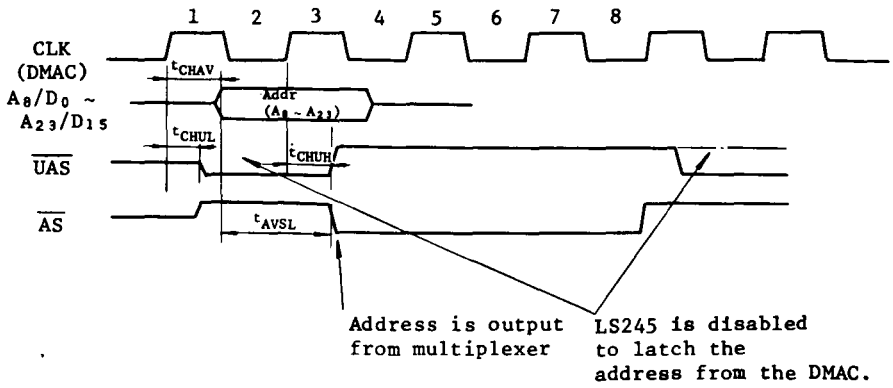


Fig. 2-11 Address Control Timing of the DMAC

During DMA data transfer, the ACRTC is basically controlled by $\overline{ACK}$, $\overline{REQ}$, $\overline{PCL}$ and $\overline{DONE}$. $\overline{REQ}$ is connected to $\overline{DREQ}$ of the ACRTC: the DMAC receives DMA transfer request from the ACRTC. $\overline{ACK}$ is connected to $\overline{DACK}$ of the ACRTC. The DMAC accesses the ACRTC by this signal. $\overline{PCL}$ is used as $\overline{READY}$ input and it is connected to $\overline{DTACK}$ of the ACRTC. $\overline{DONE}$ is I/O signal showing transfer completion, so it is connected to $\overline{DONE}$ of the ACRTC.

As shown in fig. 2-12, $\overline{DTACK}$ of the DMAC becomes an input pin to acknowledge the signal from the main memory, during DMA data transfer, and $\overline{DTACK}$ signal of the ACRTC becomes $\overline{READY}$ signal. Therefore is necessary to switch the source of the $\overline{DTACK}$ to the ACRTC.

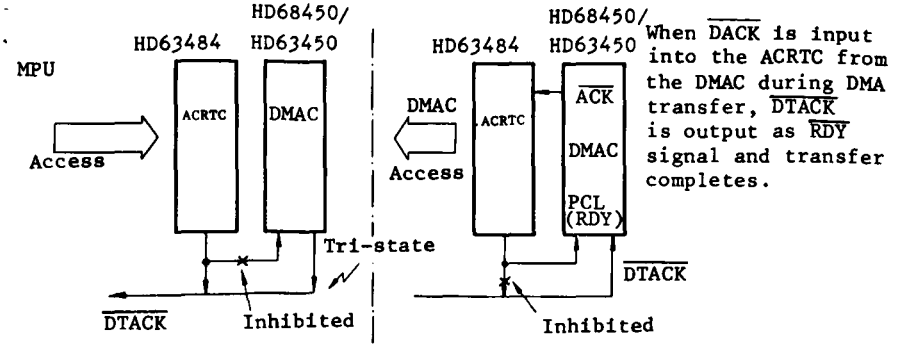


Fig. 2-12 Switching of the $\overline{DACK}$ signal

Since the ACRTC, together with $\overline{CS}$ timing, fetches the data at the rising edge of $\overline{DACK}$, as shown in fig. 2-13, data hold time (t_{DWDH}) is not assured, if $\overline{ACK}$ of DMAC is directly used. So, it is necessary to control $\overline{DACK}$ by $\overline{DTC}$ or $\overline{DS}$ shown in the circuit example of fig. 2-14.

Practically, $\overline{ACK}$ can control $\overline{DACK}$ by using $\overline{AS}$ and $\overline{DTC}$ as shown in Ⓐ Ⓑ timing of Fig. 2-13.

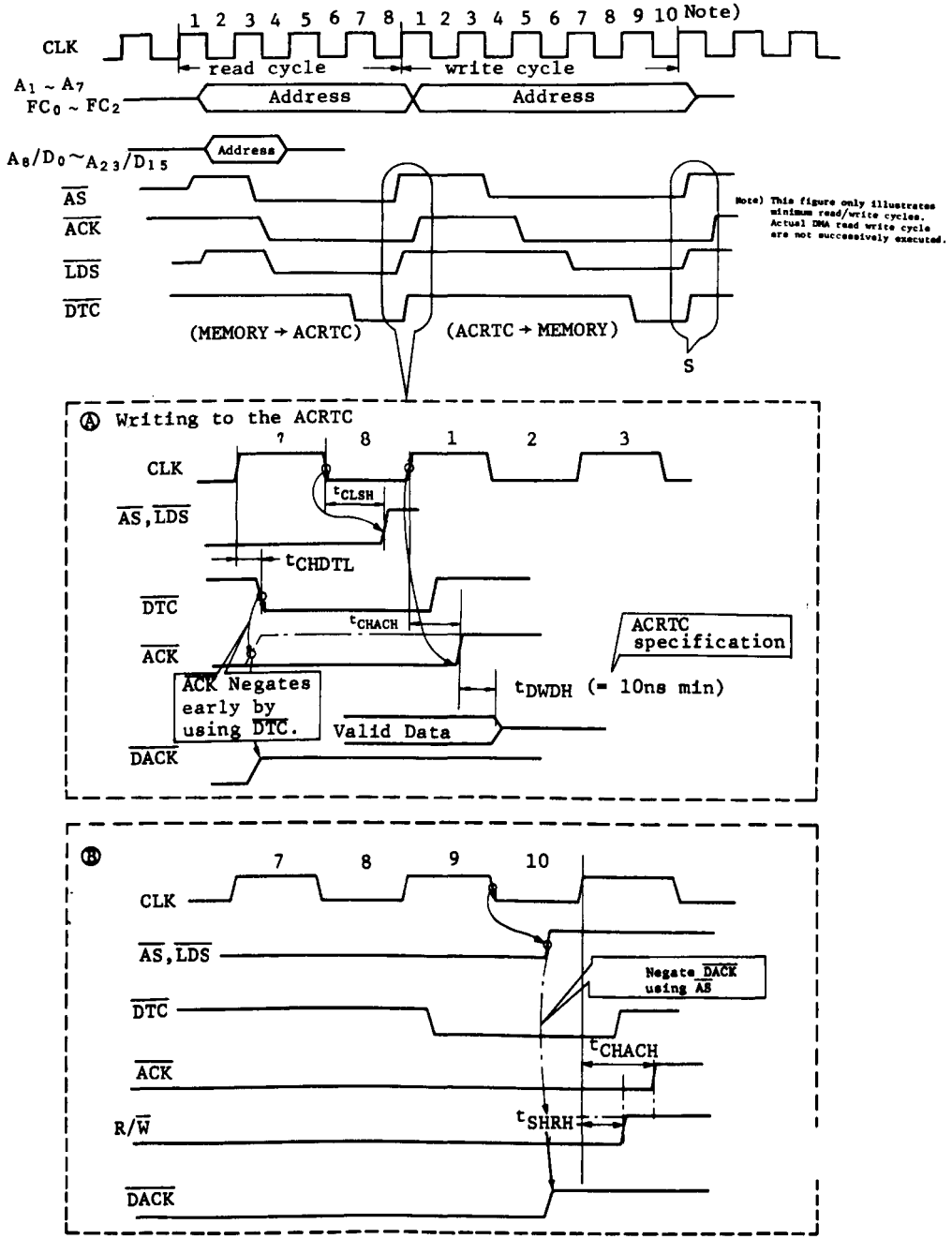


Fig. 2-13 DMAC Timing

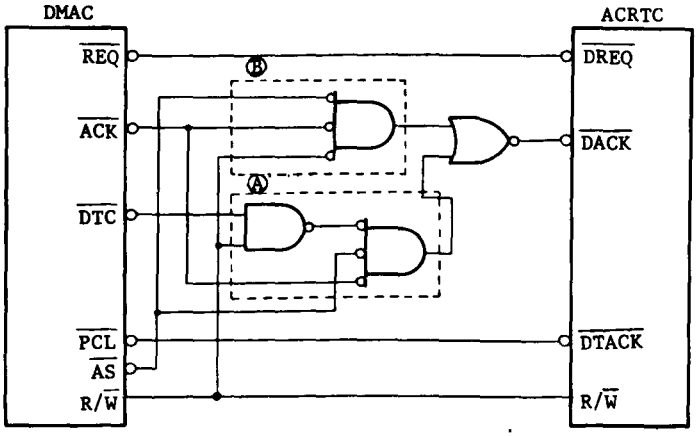


Fig. 2-14 Connection diagram for the DMAC and the ACRTC

3. INTERFACE WITH 8-BIT MPU

3.1 Connection to HD6809 (8 Bit Synchronous Bus)

When the $\overline{DACK}$ signal is "Low" at the rising edge of $\overline{RES}$ signal, the ACRTC is programmed as an 8-bit peripheral. In this case, only data bus signals D_0 through D_7 are used, and high-order bytes D_8 through D_{15} should not be connected. (ACRTC output "High" level on $D_8 \sim D_{15}$.)

Fig. 3-1 shows an example of bus connection to HD6809. Chip select signal input to the ACRTC ($\overline{CS}$) is generated by decoding address A_1 through A_{15} and E, Q clocks, which are HD6809 outputs. Since the HD6809 utilizes a synchronous bus, timing adjustment is required.

· MPU read/write cycle

R/$\overline{W}$ cycle timing of the ACRTC when the data bus is 8-bit is shown in Fig. 3-2. Fig. 3-3 shows R/W cycle timing of HD6809. It should be noted that there exist limitations on the frequency ratio of clock signals of the ACRTC and the HD6809. The ACRTC requires at least 3 2CLK cycles for $\overline{CS}$ assertion time. Since the $\overline{CS}$ signal is obtained by decoding the address signals and clock signals, E and Q of HD6809, $\overline{CS}$ assertion time is 750ns or 375ns when HD6809 is 1000ns, or 500ns, respectively. Therefore, the frequency of 2 CLK signal must be at least 4 MHz, or 8 MHz when HD6809 cycle time is 1000ns, or 500ns, respectively.

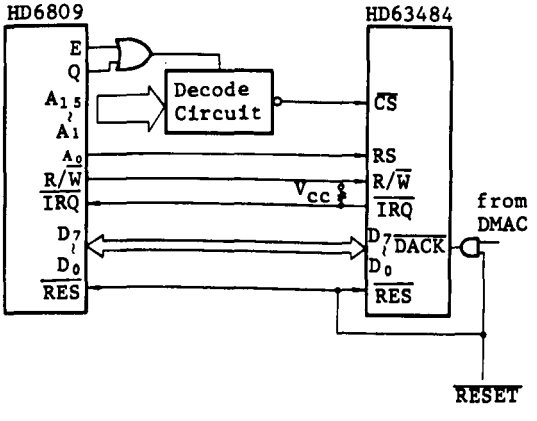


Fig. 3-1 ACRTC Bus Connection to HD6809

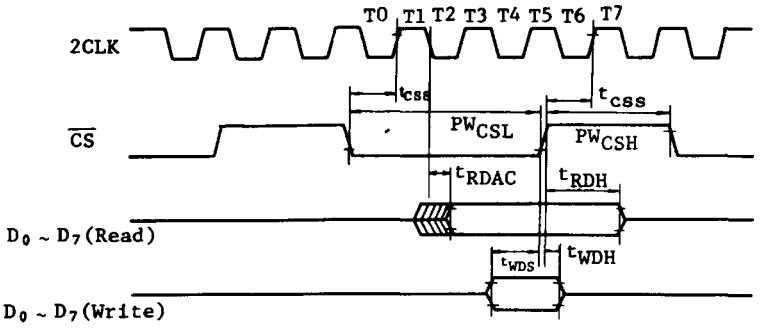


Fig. 3-2 MPU Read/Write Timing : 8-bit Synchronous Bus (MPU↔ACRTC)

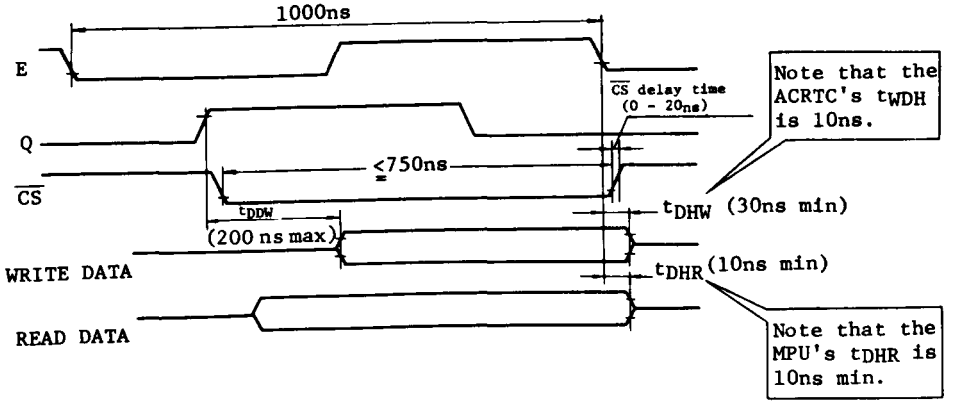


Fig. 3-3 Timing Chart of the HD6809 Read/Write Cycle

On the other hand, 2CLK frequency depends on the dot rate of the CRT so it cannot be changed. Generally, the interface to extend MPU clock using a ready signal is recommended. As shown in Fig. 3-4, a memory ready signal (MRDY) is generated from the external circuit, and is input into the HD6809. "Low" width of $\overline{CS}$ is extended by 4 cycles of 2CLK.

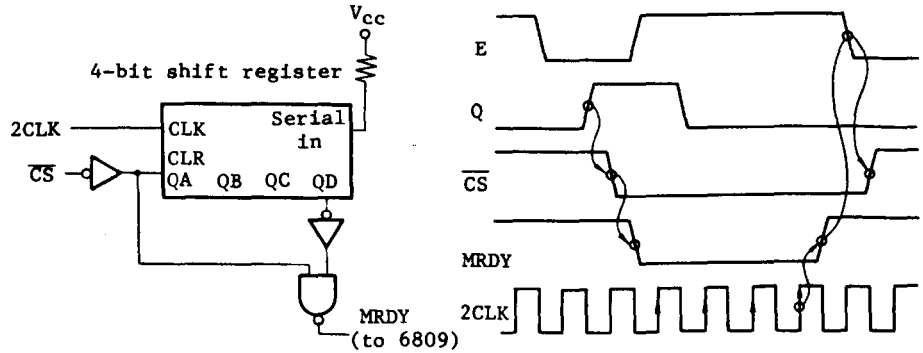


Fig. 3-4 Wait Circuit for the HD6809 and Timing

3.2 Connection to HD6844 (8-bit DMAC)

The ACRTC can perform 8-bit data transfer under the control of an 8-bit DMAC, such as the HD6844. Fig. 3-5 shows an example of a circuit where the HD6809, MPU, and the HD6844 DMAC are used.

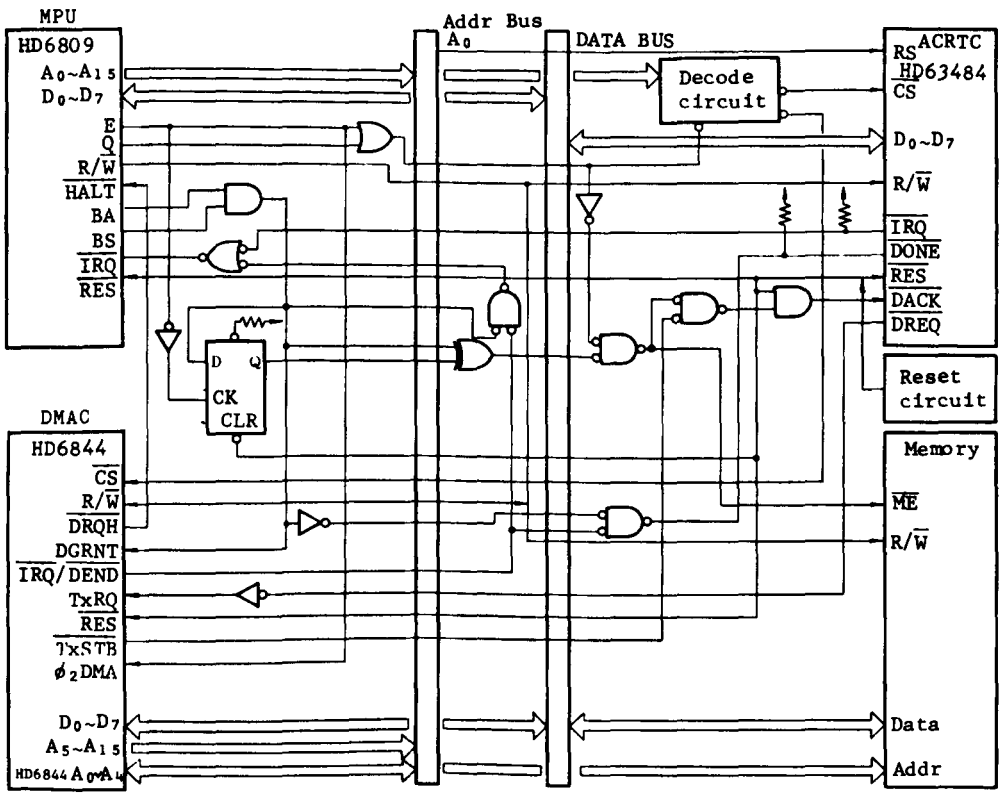


Fig. 3-5 8-Bit DMAC Interface Circuit

Bus arbitration is performed as follows.

1 The ACRTC outputs a transfer request $\overline{\text{DREQ}}$ to DMAC.

2 After acknowledging the $\overline{\text{DREQ}}$ signal, the DMAC requests the MPU to disconnect itself from the bus using the $\overline{\text{HALT}}$ pin of the HD6809.

3 MPU receives $\overline{\text{HALT}}$. After completion of the current cycle, MPU disconnects itself from the bus, sets BA and BS to "H" to signal the bus disconnection.

4 DMAC is informed of bus disconnection by BGRNT signal generated by BA and BS signals, and DMA transfer begins.

The ACRTC performs the DMA data transfer by accepting the $\overline{\text{DACK}}$ signal. For this purpose, the $\overline{\text{TXSTB}}$ signal from the DMAC is applied to the ACRTC $\overline{\text{DACK}}$ pin, as shown in Fig. 3-6. It should be noted that masking of the $\overline{\text{TXSTB}}$ is required to prevent the ACRTC from being improperly accessed during the bus arbitration period. Details are shown in Fig. 3-5.

$\overline{\text{DACK}}$ must be "LOW" at the rising edge of the $\overline{\text{RES}}$ signal so as to program the ACRTC as an 8-bit peripheral. Therefore, the masked $\overline{\text{TXSTB}}$ signal ORed with the $\overline{\text{RES}}$ must be input to the ACRTC $\overline{\text{DACK}}$ pin.

The ACRTC latches the data bus signals at the rising edge of the $\overline{\text{DACK}}$. Therefore, the data hold time and the data setup time for the $\overline{\text{DACK}}$ must be assured.

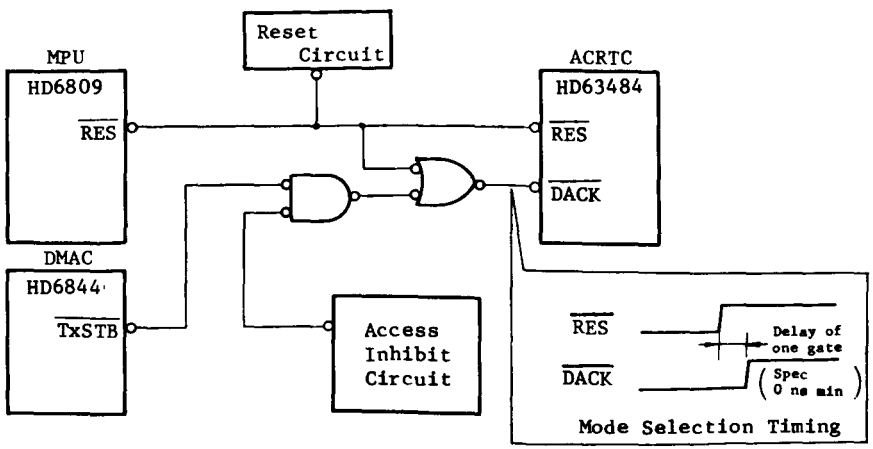


Fig. 3-6 Connection Circuit of $\overline{\text{DACK}}$ and $\overline{\text{RES}}$

4. CRT INTERFACE

4.1 Dot Clock, 2CLK, Load Signal Generation Circuit

The frequency of dot clock depends on the display time during 1 horizontal scanning period and the number of dots displayed. 2CLK is generated by dividing dot clock according to Table 1-1. At the end of the display access, LOAD signal which loads the video memory output to the shift register that converts from parallel to serial is output. And timing of $\overline{RAS}$ and $\overline{CAS}$ for a frame buffer (DRAM) is output. Fig. 4-1 shows the circuit and the timing.

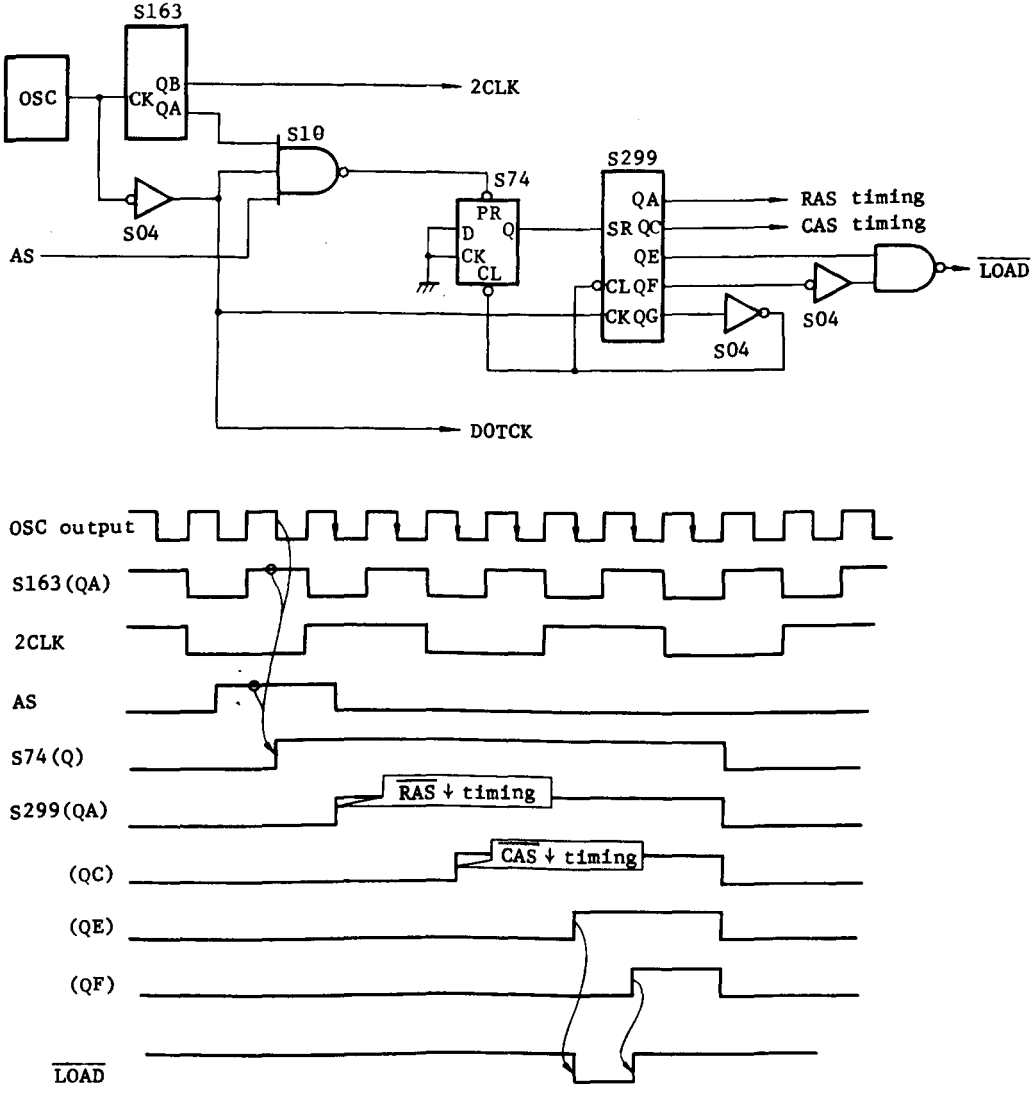


Fig. 4-1 Dot Clock, LOAD Signal Generation Circuit

4.2 Video Signal Generation Circuit

(1) Graphic Display

By directing connecting the frame buffer output to the parallel/serial converter (shift register), graphic display signals can be obtained.

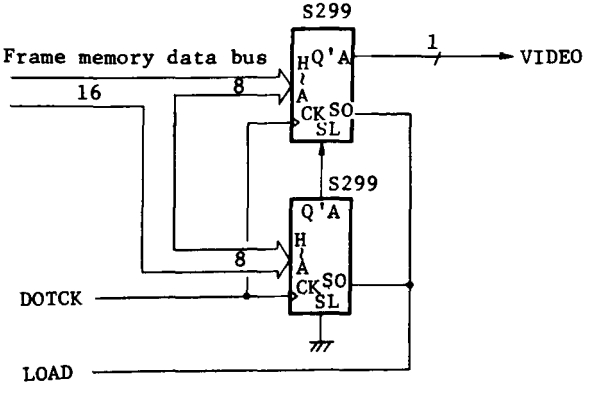


Fig. 4-2 Parallel/Serial Converting Circuit

(2) Character Display

The character display is obtained by the following system. The data video from the refresh memory output (character code) and the raster address signals from the ACRTC are used to access the character generator (CG). The output data from the CG is input to the shift register, in the same way as the graphic display, the shift register converts the data into video signals.

In the case of the character display, the attribute data, such as character color specifications or blink controls, are stored into the refresh memory in parallel with the character codes. The CG output data is modified together according to this data, and variety of character displays can be realized. Fig. 4-3 shows an example of character display circuit having the character pattern of 16 dot × 16 raster through 16 dot × 32 raster and color data attributes. In the case of Fig. 4-3, if the access time of the refresh memory and character generator are sufficiently shorter than the memory access cycle of the ACRTC, the refresh memory data latch and the raster address latch are unnecessary.

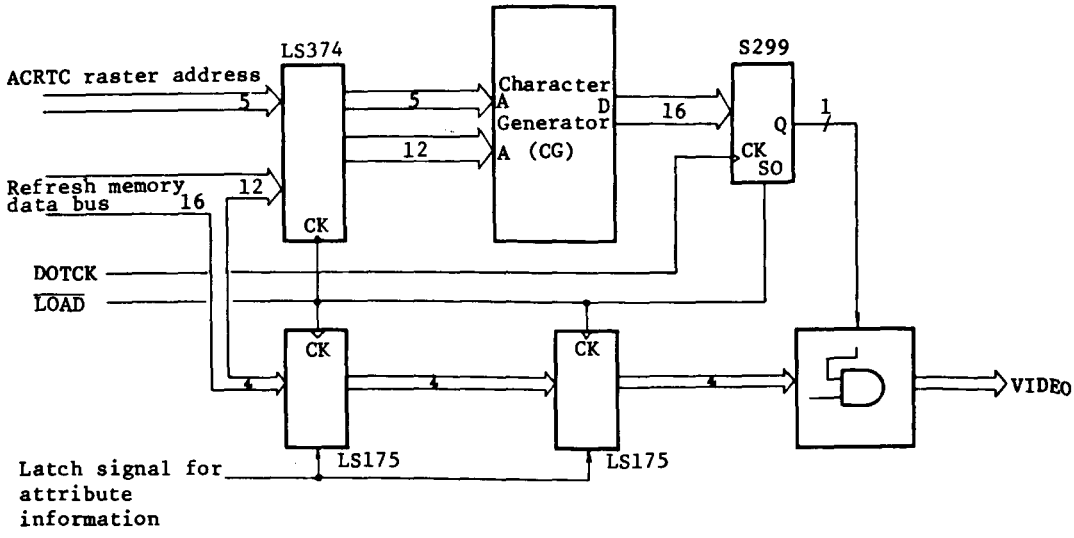


Fig. 4-3 An Example of Character Generating Circuit

(3) Superimposed Display

By setting the frame buffer access mode (ACM:bit 3, 2) in the operation mode register (OMR r04 - 05) to "1 1", the ACRTC can be set to the superimposed mode. Signals are output at the timing shown in Fig. 4-4. Therefore, in the first half of the one display cycle, the addresses which come from the parameters in the background screen register are output. In the latter half, the addresses which come from the parameters in the window screen register are output. By mixing the read-out data during the two memory cycles by using OR or EXOR logic, a superimposition of screens is possible. As for the mixing methods, two methods are employed. One is to execute the logical operation against the output data of the frame buffer directly as shown in Fig. 4-5. The other is to perform logical operation to the serially converted output data. In the both methods, image data from the frame buffer is loaded into each circuit by strobing the load signal into background screen circuit and window screen circuit.

DA1 (DUAL ACCESS 1 MODE)

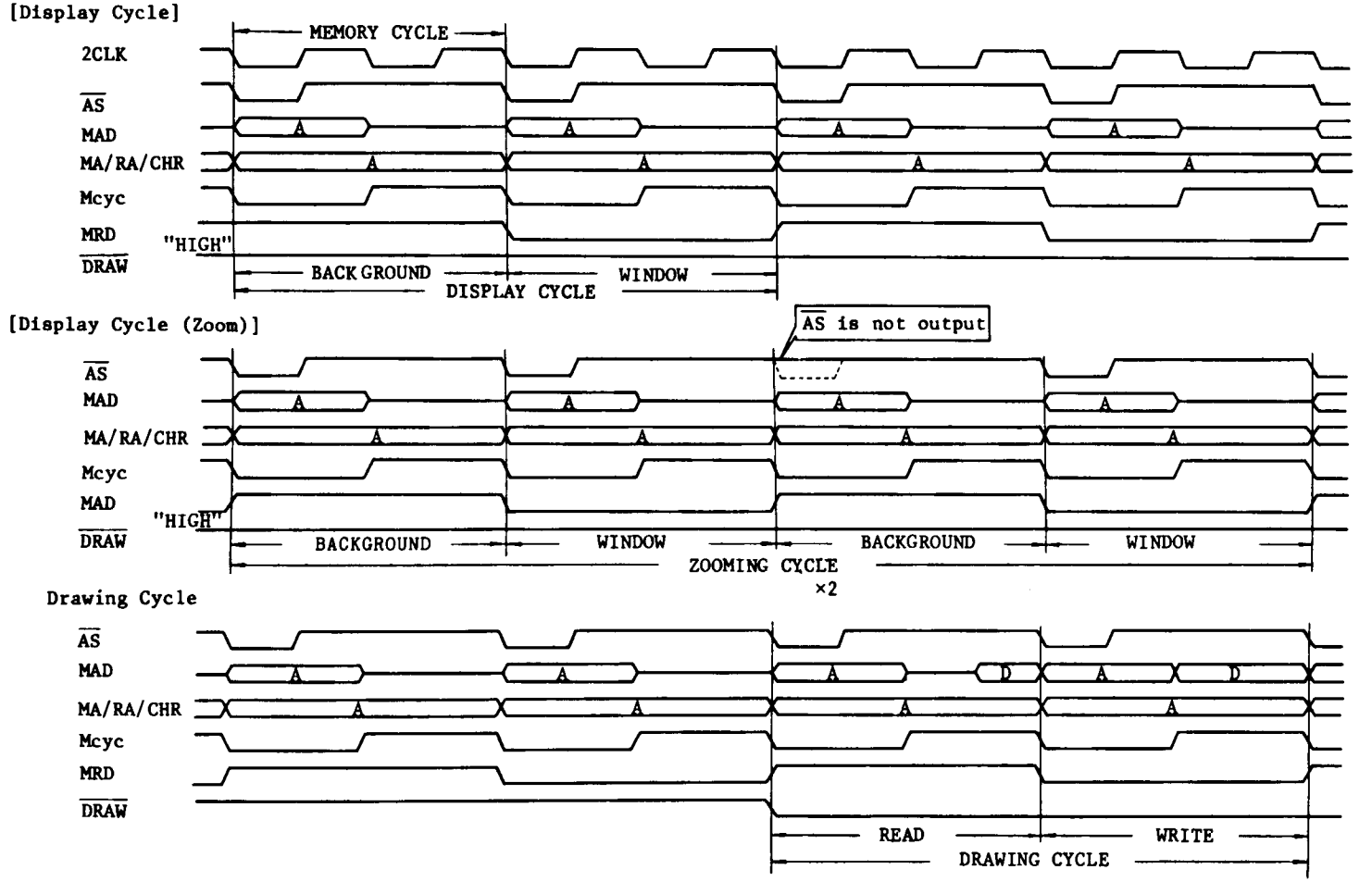


Fig. 4-4 Superimposed Access Mode (Dual Access Mode 1)

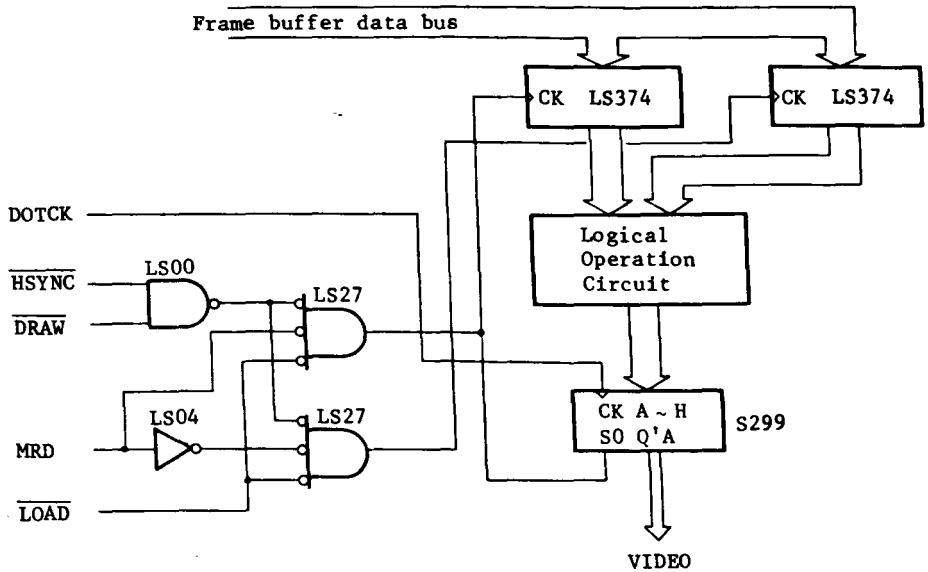


Fig. 4-5 Superimpose Circuit (1)

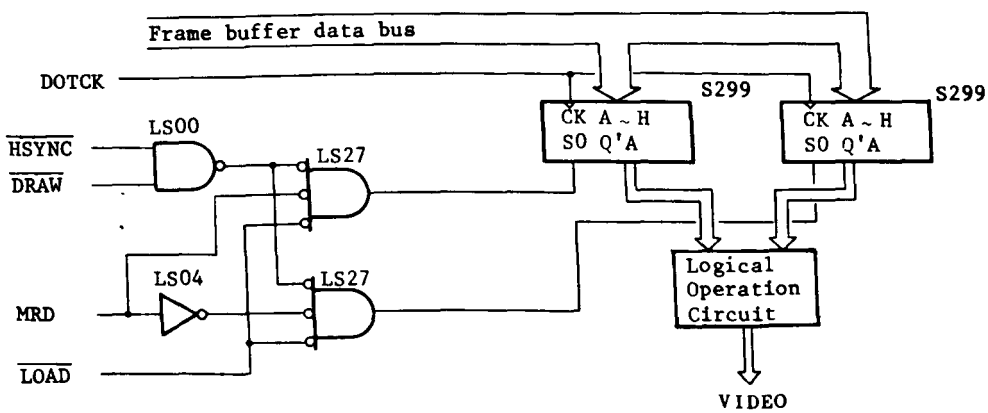


Fig. 4-6 Superimpose Circuit (2)

5. FRAME BUFFER

5.1 Memory Organization

Frame buffer organization is determined by number of display pixel of the CRT and by data number which is read out during one display cycle.

5.1.1 Frame Buffer for Graphic Display

In the case of graphic display, the data which is read out in one display period is determined by (graphic bit mode) × (parallel/serial converter dot shift quantity) as shown in Fig. 1-2 in Chapter 1. For example, in the case that graphic bit mode 4 bits/pixel is used, and parallel to serial conversion of 16 dots are executed, the data number to be read out during one display cycle is 64 bits (4 × 16).

On the other hand, drawing in the frame buffer is done in 1 word (16 bits) units. Therefore, when drawing, the frame buffer is divided into blocks by the lower address, the MRD and the $\overline{\text{DRAW}}$. Fig. 5-1 shows the memory organization of 16 dots parallel/serial conversion in the 4 bits/pixel mode.

As to the memory type, any memory which can be accessed within 1 memory cycle time can be used. However, DRAM's are the most commonly used from the view points of capacity and mounting area. The number of memory chips is determined by the amount of data which is read out in one display period and the number of screens to be stored in the system.

Fig. 5-1 shows the memory type and the minimum necessary number in the case of a 15 color display CRT of 640 dots × 400 rasters, with a memory organization of 4 bits/pixel, 16 dots parallel/serial conversion. However, it is possible to reduce the necessary number by using the page and nibble modes of DRAM's by implementing multiple access to the memory during one display period.

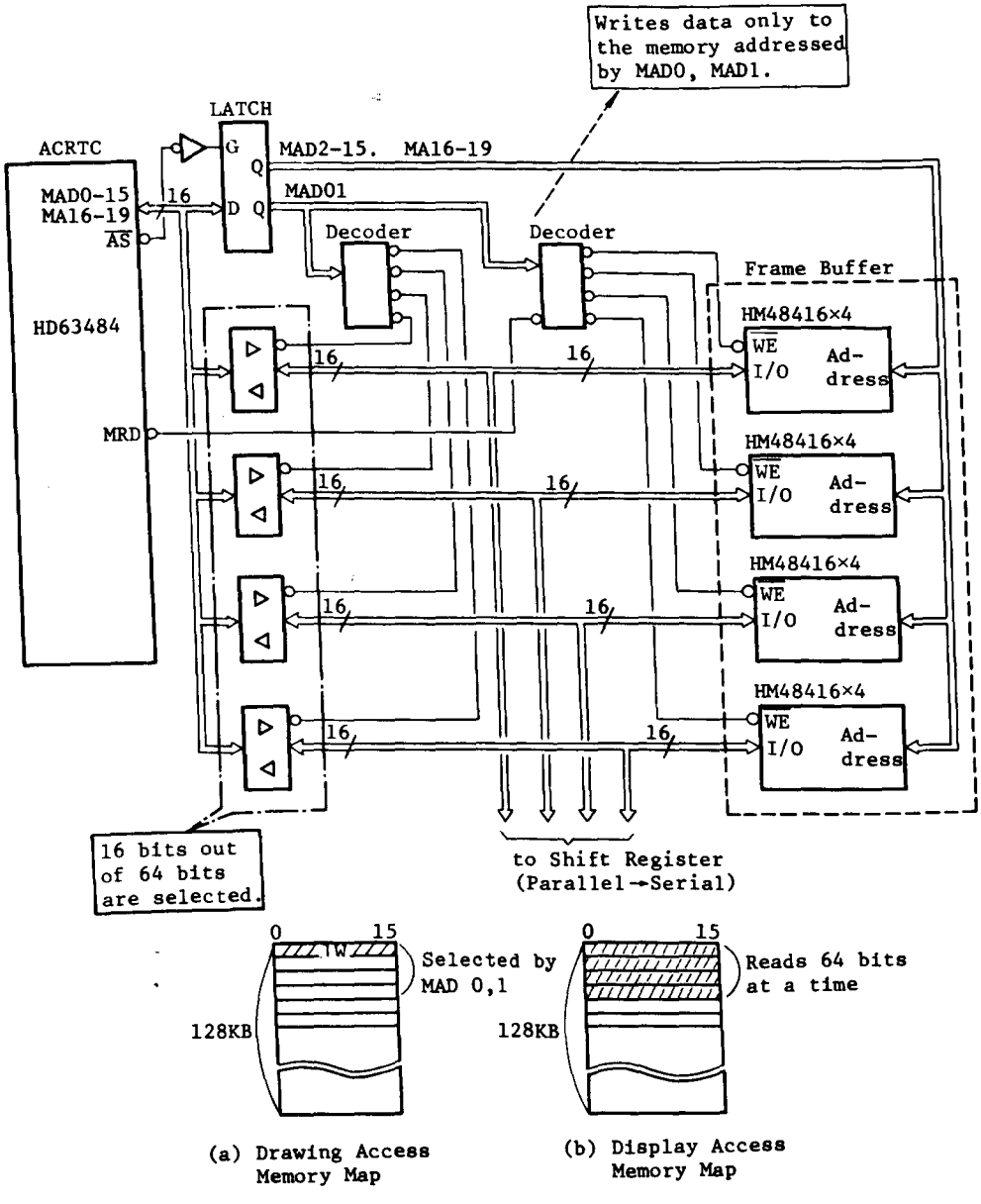


Fig. 5-1 The Example of Memory Organization Using the HM48416

Table 5-1 Memory Configuration for Example System (4 bits/pixel 16 pixel/display cycle)

Using memory type	Memory organization	Minimum necessary chips	Capacity
HM4846	64K × 1bit	64	512 KB CRT 4 screens
HM50256	256K × 1bit	64	2 MB CRT 16 screens
HM48416	16K × 4bit	16	128 KB CRT 1 screen

The memory capacity necessary for 1 CRT screen is as follows.

640 dots/line × 400 lines × 4 bits/pixel

= 1.024,000 bits

= 128K bytes.

HM48416AP-12, HM48416AP-15 HM48416AP-20

16384-word X 4-bit Dynamic Random Access Memory

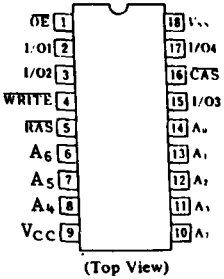
- **FEATURES**
- 16384-word x 4-bit Organization
- Single 5V (±10%)
- Low Power; 303mW Active, 20mW Standby
- High speed: Access Time 120ns/150ns/200ns (max)
- Page mode capability
- Output data controlled by $\overline{CAS}$, $\overline{OE}$
- TTL compatible
- 128 refresh cycles ($A_0 \sim A_6$, 2ms)

HM48416AP-12, HM48416AP-15, HM48416AP-20

(DP-18)

A0~A7	Address Inputs
$\overline{CAS}$	Column Address Strobe
I/O1~I/O4	Data In/Data Out
$\overline{OE}$	Output Enable
$\overline{RAS}$	Row Address Strobe
$\overline{WRITE}$	Read/Write Input
V_{CC}	Power (+5V)
V_{SS}	Ground

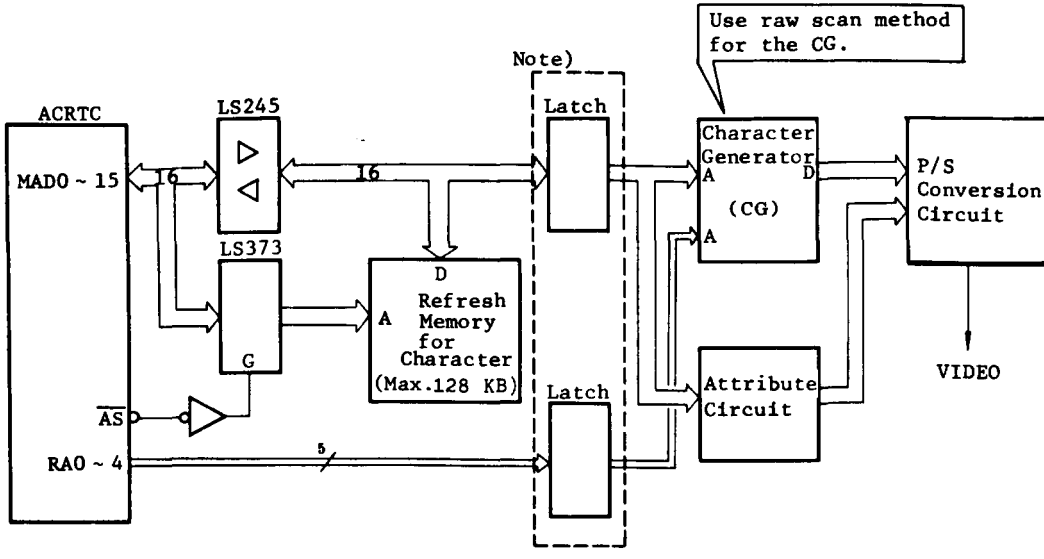
- **PIN ARRANGEMENT**



(Top View)

5.1.2 Refresh Memory for Character Display

In the ACRTC system, in addition to graphic frame buffer, it is possible to equip refresh memory of maximum 128 KB for character display. Fig. 5-2 shows a block diagram (ex.) for the character display; Fig. 5-3 shows the timing.



Note) If total of memory access time and CG access time are sufficiently shorter than one memory cycle, a character display is implemented without using these latches.

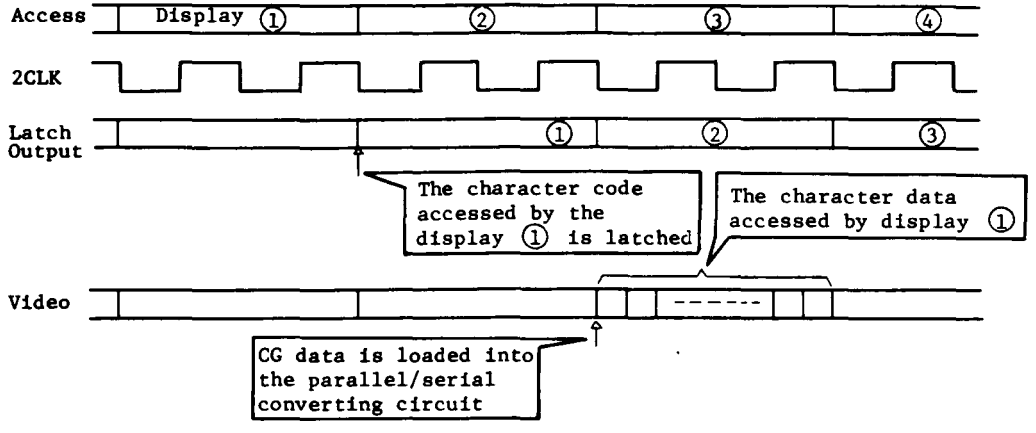
Fig. 5-2 Block Diagram of Character Display

In the ACRTC, each split screen and window screen can be independently set to the character mode, by setting the CHR bit of the memory width register (MWR) to "1". In this case, the refresh memory addresses from MAD 0 - 15, and the character raster addresses from RA0 - 4, are output according to the setup value of the LRA, FRA of the raster address register (RAR).

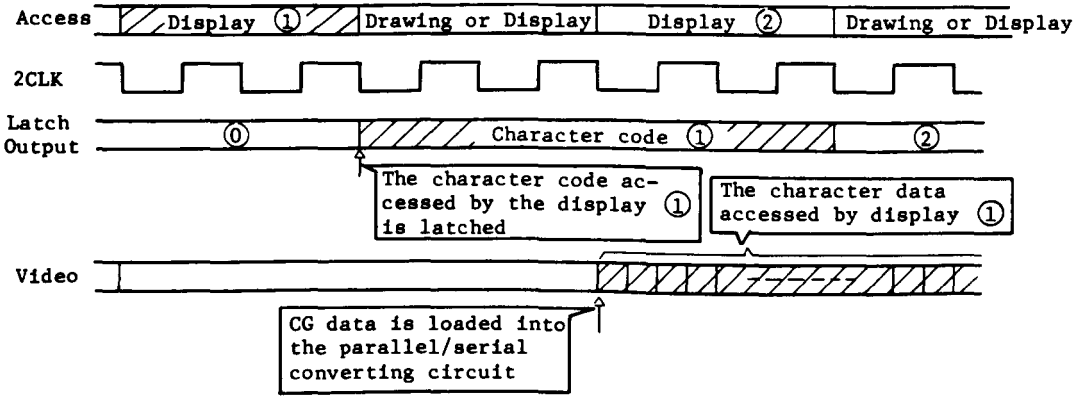
The display addresses which are output from MAD 0 - 15, are successively incremented by plus 1 starting from the one stored in address of the display start address register (SAR). Therefore, character generator (CG) addresses to be displayed (character code) need to be stored in the refresh memory before starting the display. Thus the CG characters which correspond to the character codes are displayed on the CRT. Fig. 5-4 shows the correspondence between the CRT displays and the refresh memory addresses.

The number of the refresh memory data lines must be at least the number of the character generator address lines. If the addresses which are necessary for the CG are 16 bits or less, it is possible to use the remaining data for attribute control such as color data. Any memory which can be

accessed within one ACRTC memory cycle can be used. Fig. 5-2 shows the minimum memory chips and display screens which are necessary to implement the display of 16 dot × 16 raster characters in the 640 dot × 400 raster CRT.



(a) Single Access Mode



(b) Dual Access Mode

Fig. 5-3 Character Display Timing

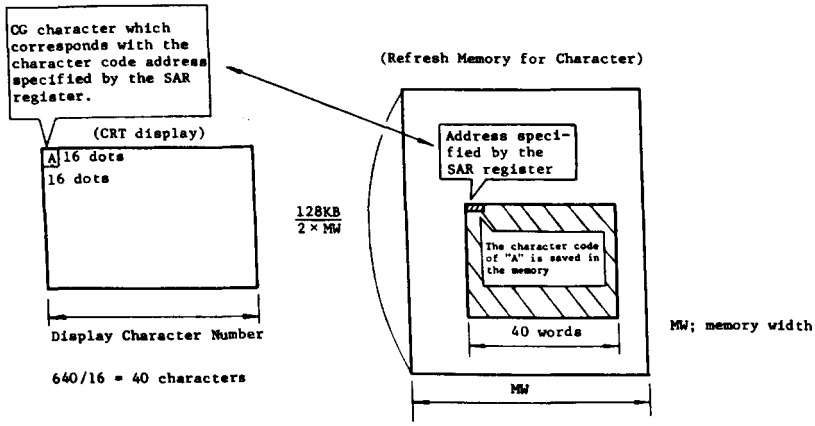


Fig. 5-4 Correspondence between the CRT Display and the Refresh Memory

Table 5-4 Memory Type and the Minimum Necessary Chips

Memory Type	Memory organization	Minimum necessary chips	Capacity
HM6264	8K × 8 bit	2	16 KB (CRT 8 screens)
HM6116	2K × 8 bit	2	4 KB (CRT 2 screens)
HM6148	1K × 4 bit	4	2 KB (CRT 1 screen)
HM48416	16K × 4 bit	4	32 KB (CRT 16 screens)

Note 1) Calculates the CG address as 16 bit.

Note 2) The necessary memory capacity for 1 display screen the CRT is as follows.
(640/16) × (400/16) = 1000 (W) (2 KB)

5.2 Memory Access

Operating synchronized with the 2CLK, the ACRTC accesses the memory within 2 cycles (MCYC: 1 memory cycle) of 2CLK, regardless of single/dual access mode

5.2.1 DRAM Access

There are two ways to access the frame buffer: DRAM early write cycle to write data at the falling edge of $\overline{CAS}$ and delayed write cycle to write data at the falling edge of $\overline{WRITE}$. For details of both access modes, refer to the memory data sheet.

In the case of writing the drawing data into the frame buffer using the early write cycle as shown in Fig. 5-5, the ACRTC outputs the drawing data with MCYC = "H", then the $\overline{CAS}$ needs to be driven "Low" after the drawing data has been output. On the contrary, in the case of reading data out of the frame buffer, as shown in Fig. 5-5, the $\overline{CAS}$ needs to be driven "Low" to satisfy t_{MRDS} of the ACRTC. This is because $\overline{CAS}$ falling with the same timing causes insufficient ACRTC read data setup time (t_{MRDS}) when the 2CLK cycle time is shortened in high speed application.

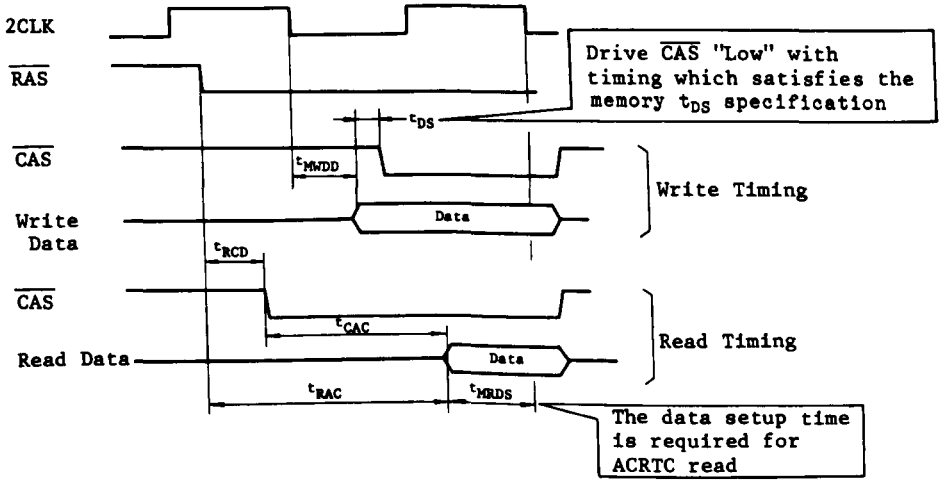


Fig. 5-5 Memory Read/Write Timing

Thus the falling edge of $\overline{CAS}$ must be handled with care when the early write cycle is used. Further, in the case of using a delayed write cycle to execute the $\overline{WRITE}$ operation at the falling edge of the DRAM WRITE, drawing access can be performed without changing the $\overline{CAS}$ timing. A circuit example using the delayed write cycle is given in Fig. 5-6, and the timing is given in Fig. 5-7.

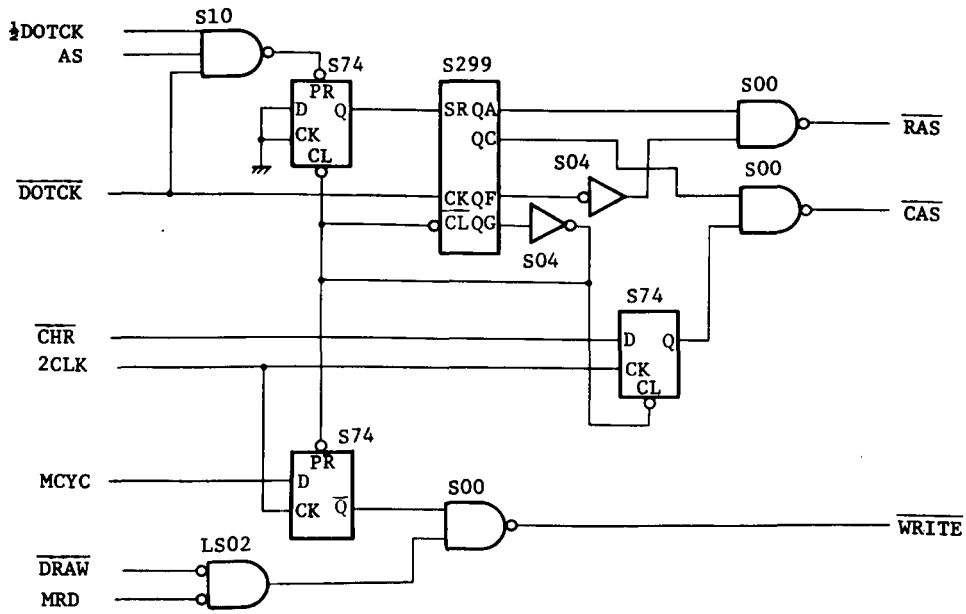


Fig. 5-6 DRAM Access Circuit

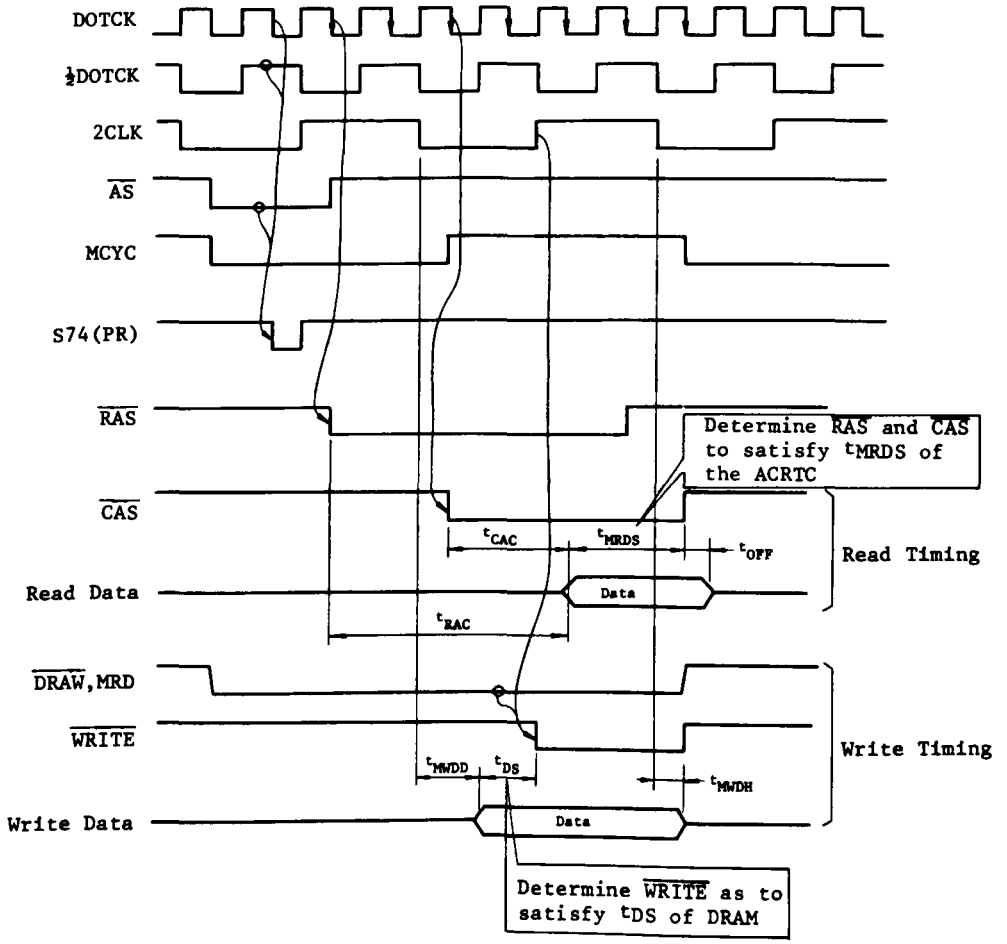
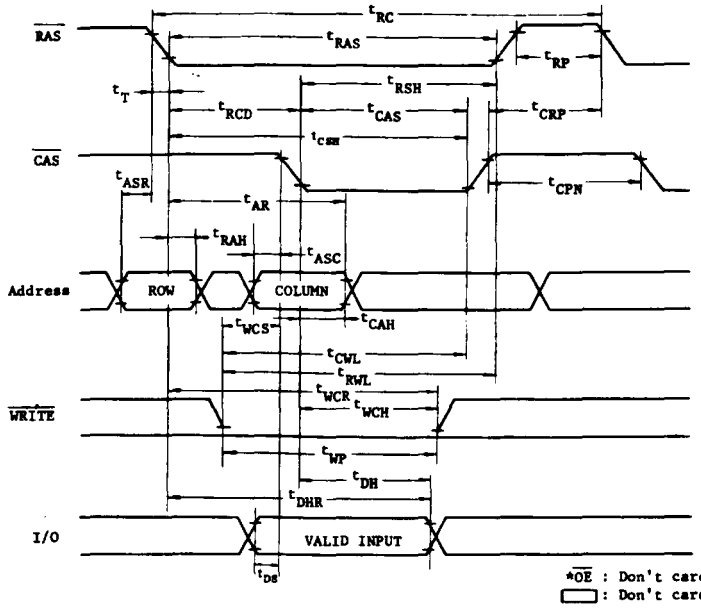


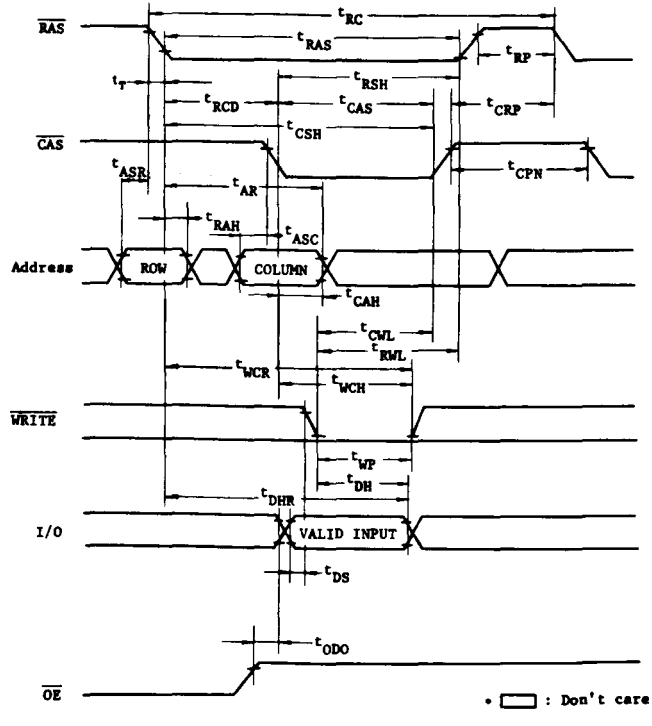
Fig. 5-7 DRAM Access Timing

Difference between Early Write and Delayed Write

- Early Write Cycle



- Delayed Write Cycle



(Notes)

Damping resistors are to be inserted between the DRAM and ACRTC to avoid undershooting of signals of $\overline{RAS}$, $\overline{CAS}$, $\overline{WRITE}$ and address. The data bus driver of the frame buffer is controlled by $\overline{DRAW}$, MRD, and $\overline{CAS}$ signals as shown in Fig. 5-8.

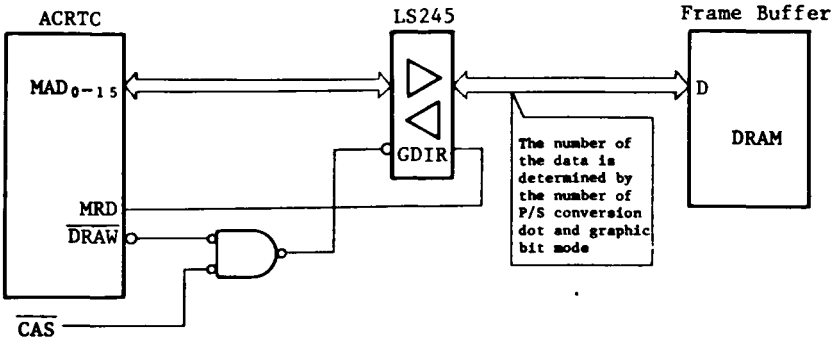


Fig. 5-8 Data Bus Control Circuit

The ACRTC outputs the refresh address during $\overline{HSYNC}$ "Low" period to refresh the DRAM when the DRAM mode is selected by setting the RAM bit in the OMR register to "0". The refresh period can be specified by the value of the horizontal synchronous pulse width (HSW) in the horizontal synchronous register (HSR). Therefore, any DRAM refresh can be done according to the DRAM refresh timing specification.

For example, when using the CRT timing is as shown in Fig. 0-1, and the HM48416 as DRAM, the DRAM must be refreshed 128 times every 2 ms. The refresh frequency of the ACRTC is 2 ms when $0A is set to HSW, (2 ms/41.3 us) x 10 times ≒ 484 times. This satisfies the DRAM refresh requirement

5.2.2 SRAM Access

When SRAM is used for the frame buffer, it can be accessed with a simpler circuit than that of the DRAM. However, SRAM has less memory capacity than that of DRAM, so it is recommended in a system in which large frame buffer capacity is not required or high-speed access is required. In Fig. 5-9, an example of a circuit when the access mode of SRAM is controlled by $\overline{CS}$, and the timing are given in Fig. 5-10. The SRAM which provides time to satisfy the data setup time (t_{MRDS}) of the ACRTC is recommended. The Write operation is executed at the rising edge of $\overline{CS}$ when controlling $\overline{CS}$. Note that RAM data hold

time (t_{MWDH}) must exceed the specified values.

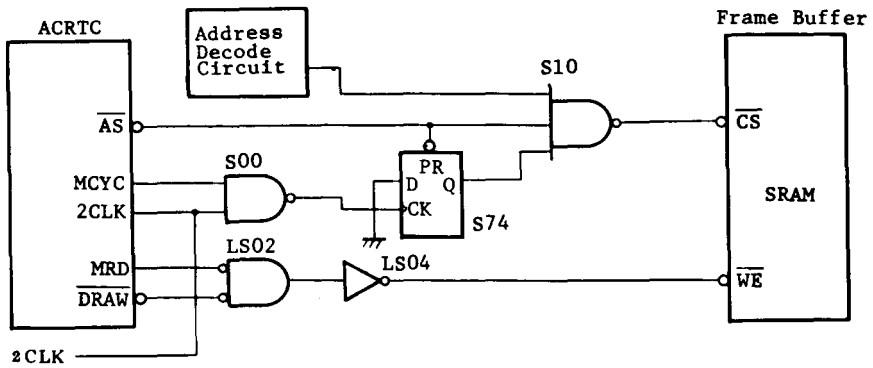


Fig. 5-9 The Example of SRAM Access Circuit

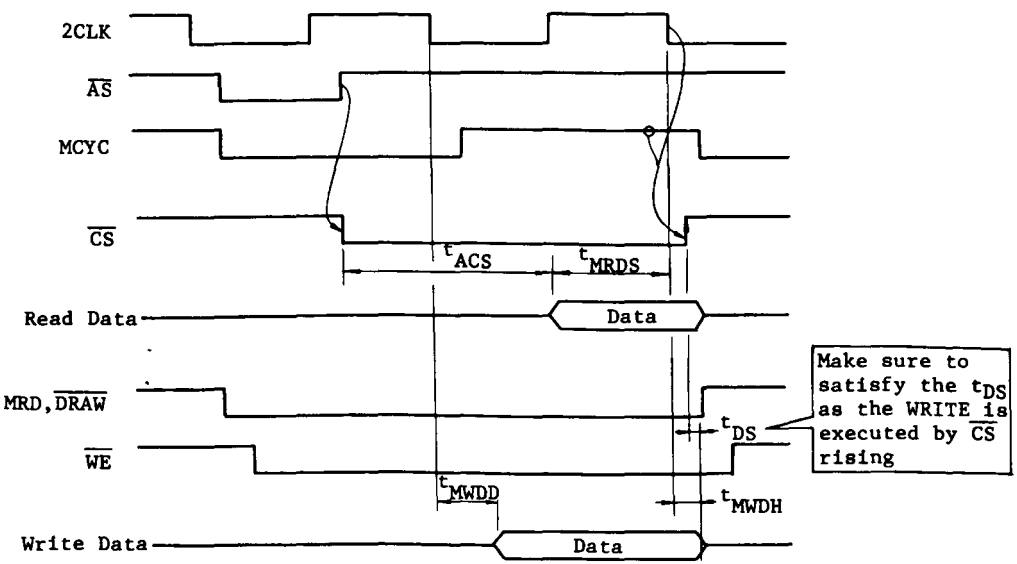


Fig. 5-10 SRAM Access Timing

6. ATTRIBUTE

6.1 Fetching the Attribute Control Signal

The ACRTC attribute control signal as shown in Fig. 6-1 is output at the end of all horizontal retrace period as shown in Fig. 6-2. Therefore, the attribute data is latched at the falling edge of the 2CLK signal when HSYNC is "L". Fig. 6-3 gives the circuit example of latching the attribute data.

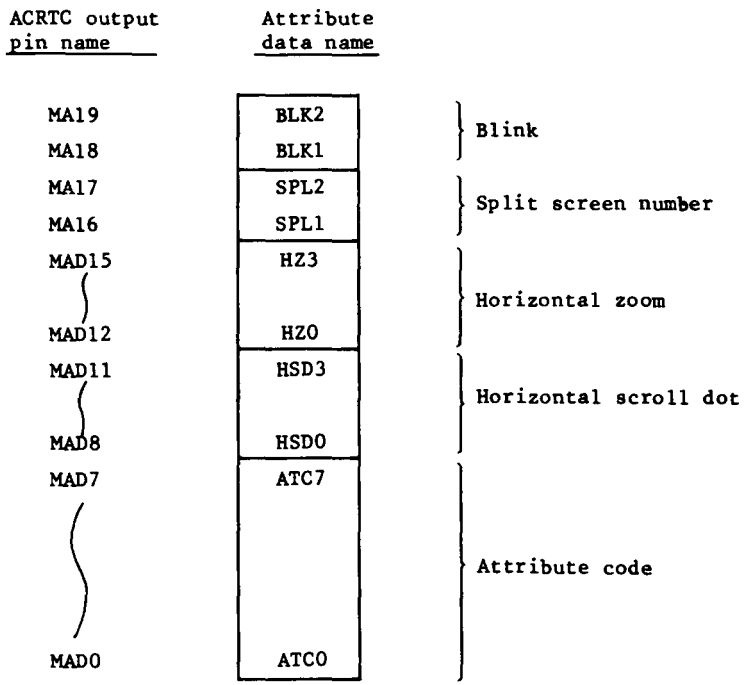


Fig. 6-1 Attribute Control Signal

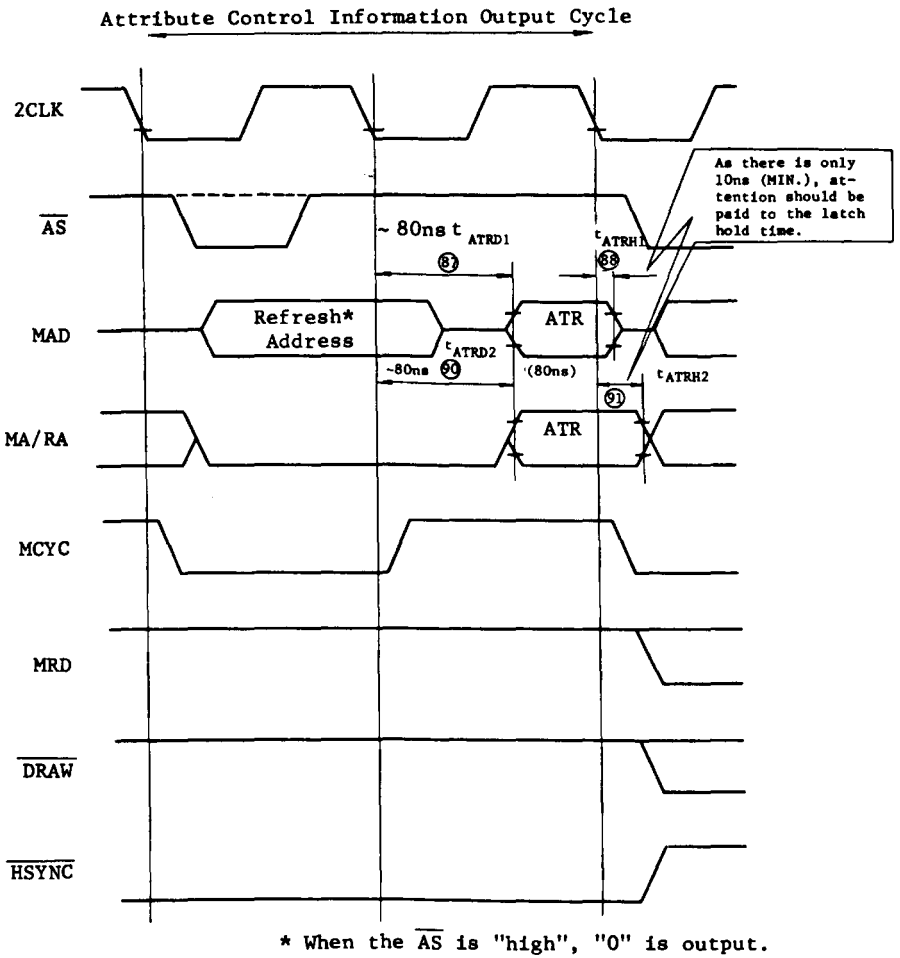


Fig. 6-2 Attribute Control Data Output Timing

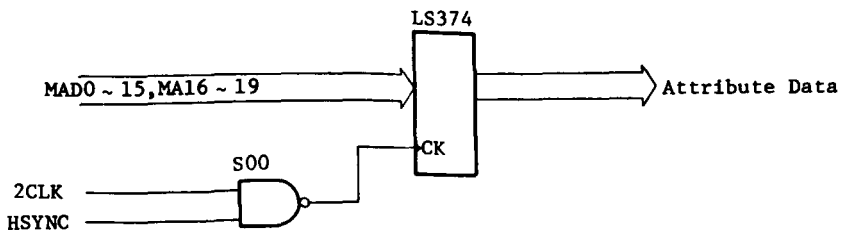


Fig. 6-3 Circuit to Latch Attribute Data

6.2 Smooth Scroll

(1) Vertical Smooth Scroll

The ACRTC controls the display start address for 4 screens independently. As shown in the following equation, a vertical smooth scroll can be executed without an external circuit by offsetting the display start address value by the memory width.

SAR = SAR + N x MWR (N = 0, 1, 2, ...)

SAR: Display start address

MWR: Memory width

In the case of the character screen, scrolling by raster or by line is performed. Then, not only the display start address, but also the start raster address (SRA) needs to be changed.

(2) Horizontal smooth scroll

The horizontal scrolling in units of dot can be implemented by selecting the output from the shift register with a selector with the HSD 0 to 3. Attribute data fetch circuit is shown in Fig. 6-4. In this case, the Shift register is used to prepare 16 signals delayed by units of dots from the parallel/serial converter. Then, $\overline{\text{DISP1}}$ and $\overline{\text{DISP2}}$ signals must be delayed by the external circuit (skewed), or by setting the DSK bit of the command control register (CCR, r02 to r03).

When executing the horizontal smooth scroll of the window screen with the superimpose mode, "1" should be set in the WSS bit of the CCR register (r02 to 03). In this case, the setup value at the window (SDAW value of SARW) is output onto the horizontal smooth scroll quantity (HSD) of the attribute output.

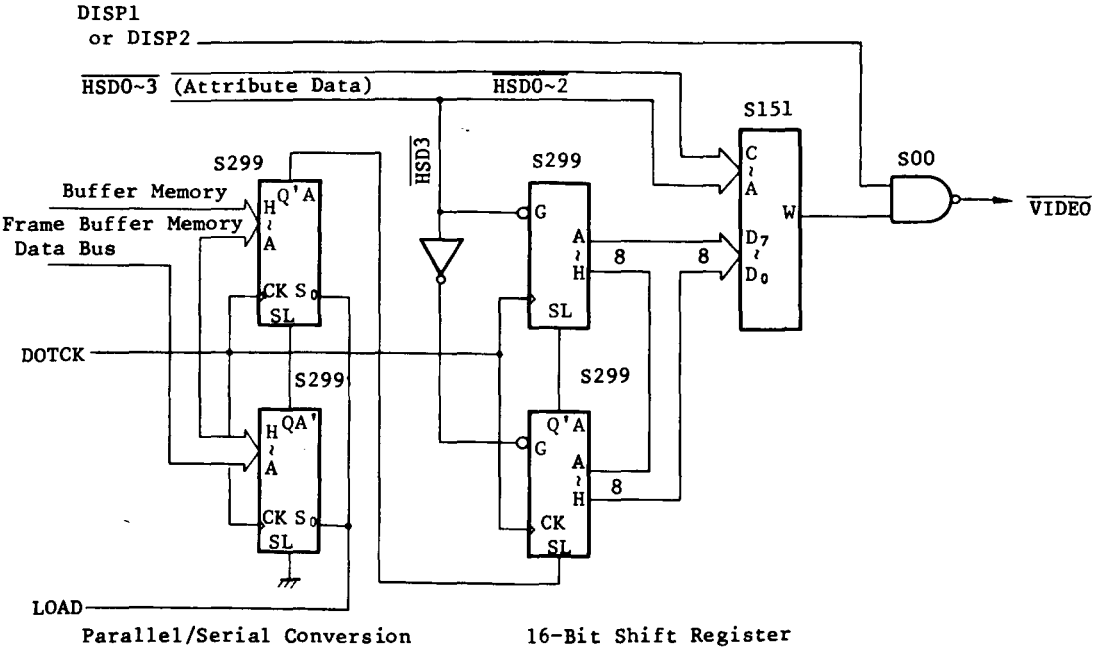


Fig. 6-4 Horizontal Smooth Scroll Circuit

Fig. 6-5 indicates the timing of each signal when horizontal smooth scroll is executed on the base screen and split screen.

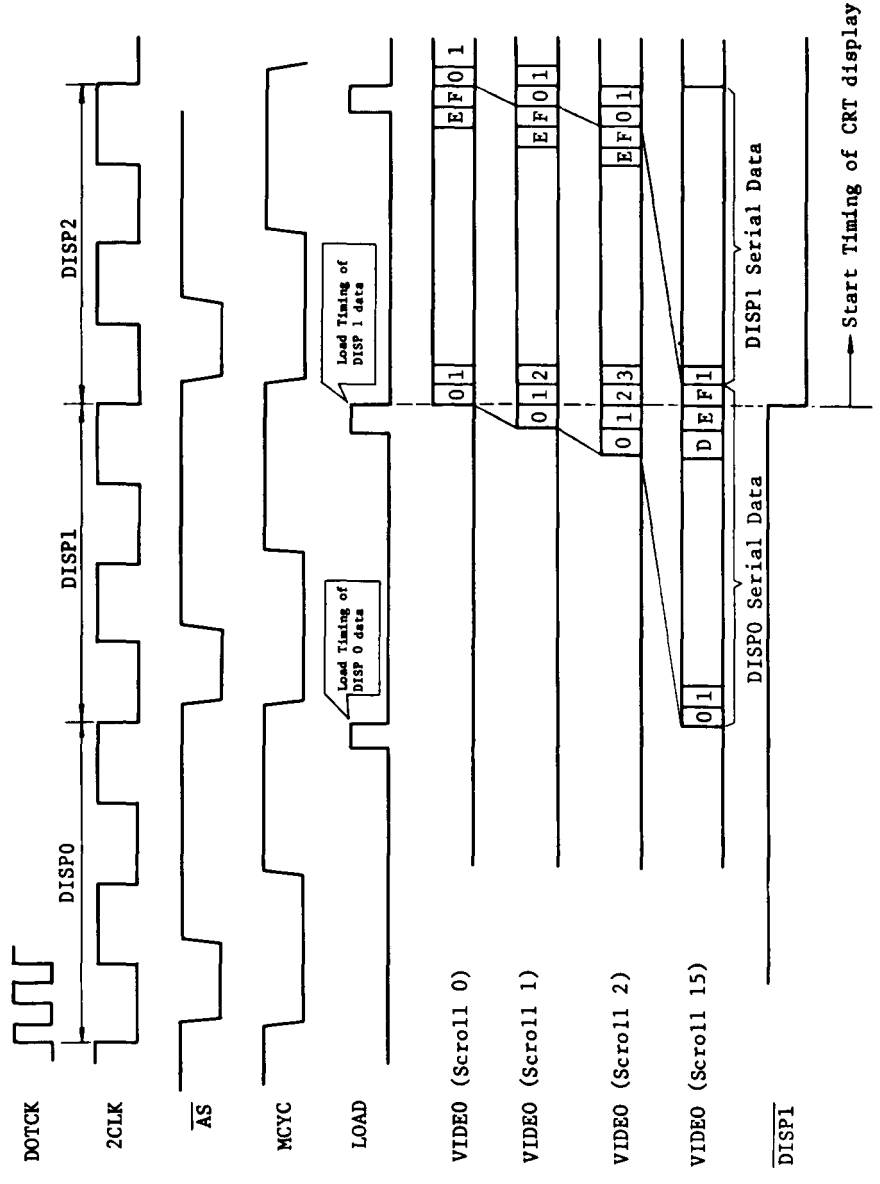
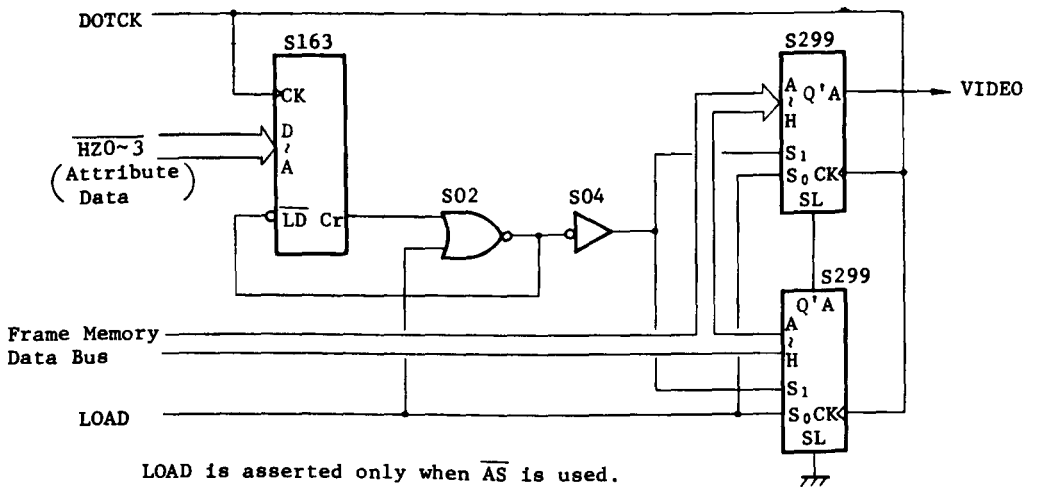


Fig. 6-5 Horizontal Smooth Scroll Timing of Base and Split Screens

6.3 Zooming Display

The ACRTC is capable of magnifying the base screen horizontally and vertically up to 16 times in the base screen by setting the magnification facter in the zoom factor register (ZFR ; rEA). The vertical zooming, can be controlled by the ACRTC internally so an external circuit is unnecessary. As to horizontal zooming, the ACRTC controls only the display address shown in Fig. 6-7. Therefore, it is necessary to modify the shift clock according to the horizontal zoom attribute data (HZ 0-3) or to control the shift operation of the shift register by an external circuit. Fig. 6-6 shows a circuit example of the zooming display.



Note 1) This circuit delays the shift operation according to the horizontal zoom attribute data.

Fig. 6-6 Zoom Display Circuit

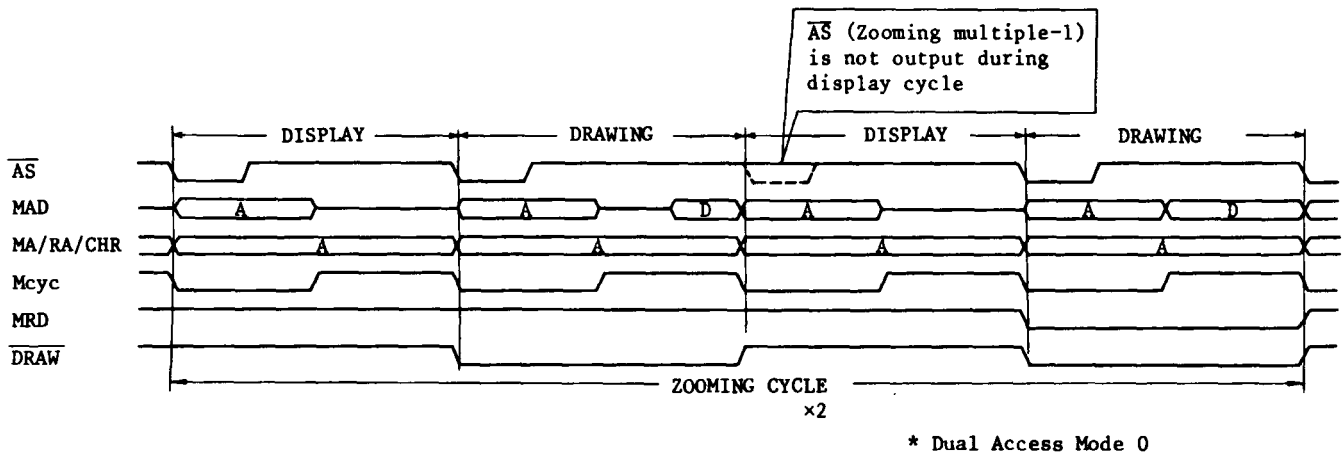


Fig. 6-7 Display Timing of Zoom (×2)

6.4 Cursor

6.4.1 Block Cursor

The ACRTC outputs two block cursor signals, called $\overline{CUD1}$ and $\overline{CUD2}$ according to the internal register setting shown in Table 6-1 with the timing of Fig. 6-8. Therefore, in a system which employs the ACRTC, the block cursor can be displayed with a simple circuit as shown in Fig. 6-9.

Table 6-1 Block Cursor Setup Register

		Cursor 1		Cursor 2		Unit	Set up Value
		Register name	Reg No.	Register name	Reg No.		
Character screen setup		CHR bit (in MW)	rC2,rCA rD2,rDA	CHR bit (in MW)	rC2,rCA rD2,rDA	—	"1"
Mode		CM	rE8	CM	rE8	—	"00" or "01"
Display position		BCA1	rE2	BCA2	rE6	Memory address	Within character display screen
Size	Width	BCW1	rE0	BCW2	rE4	Memory cycle (Note 1)	0 to 7
	Raster	BCSR1 BCER1	rE0	BCSR2 BCER2	rE4	Raster number	0 to 31 (Note 2)
Blink time		CON1 COFF1	rE8	CON2 COFF2	rE9	4-field time	0 to 7
Output signal name		$\overline{CUD1}$		$\overline{CUD2}$		——	————

(Note 1) During the horizontal zooming display,
BCW setting × (zoom factor + 1) × memory cycle.

(Note 2) Setup should be done within the number of rasters per character for each screen.

(Note 3) For details on each register, see the "ACRTC Users Manual".

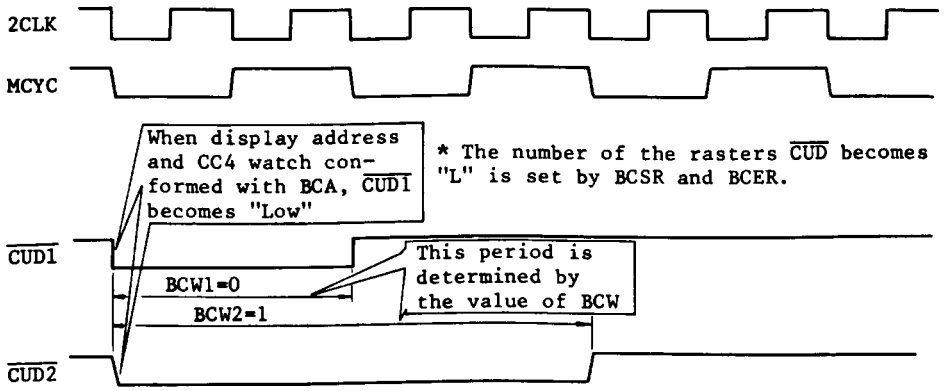
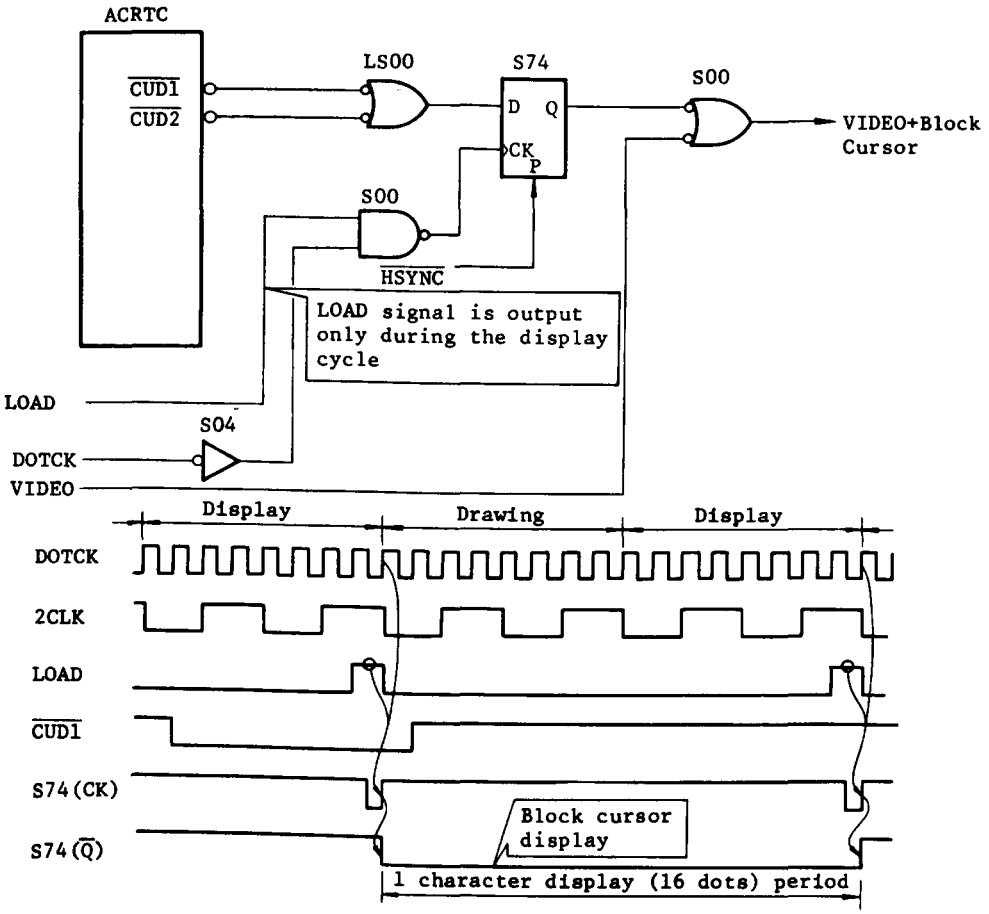


Fig. 6-8 Setting of $\overline{CUD}$ Signal Timing



* The timing when the block cursor 1 is displayed in the dual access mode 0 (extended to 1 display cycle timing for dual access mode).

Fig. 6-9 Character Cursor Display Circuit and Timing

6.4.2 Cross-hair Cursor

As shown in Fig. 6-10, the ACRTC outputs the horizontal component of the cross-hair cursor from the $\overline{CUD1}$, and the vertical component from $\overline{CUD2}$, according to the internal register setup shown in Table 6-2. Therefore, an external circuit that can detect the edges of $\overline{CUD1}$ and $\overline{CUD2}$ is used to display the cross-hair cursor.

Table 6-2 Cross-hair Cursor Register Setup

		Output Signal Name	Register Name	Reg No.	Unit	Setup Value
Mode		——	CM	rE8	——	"11" (Cross-hair cursor mode)
Display position	Vertical cursor	$\overline{CUD1}$	CXS	r99	Memory cycle	0 to 255
			CXE	r98		
	Horizontal cursor	$\overline{CUD2}$	CYS	r9A,r9B	Raster line	0 to 4095
			CYE	r9C,r9D		
Blink time		——	CON1 COFF1	rE8	4-field time	0 to 7

(Note 1) For details on each register, see the "ACRTC Users Manual".

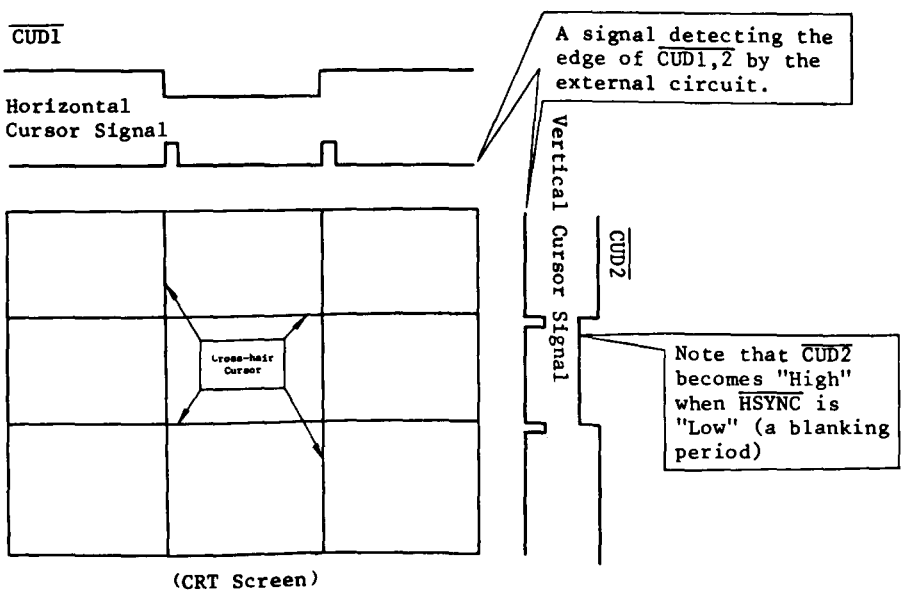


Fig. 6-10 Cross-hair Cursor Display

Fig. 6-11 shows a circuit example for displaying 2 cross hair cursors and Fig. 6-12 shows its timing. In this circuit, the cursor display vertical position is set by rasters and by the memory cycle for horizontally. In order to designate the display horizontal position in units of dots (to allow the cursor's horizontal smooth scroll), an external circuit, for shifting the display position using the attribute code, is needed. Fig. 6-13 shows a circuit example which specifies the vertical cursor display position in units of dots.

(Vertical Cross-Hair Cursor Display Circuit)

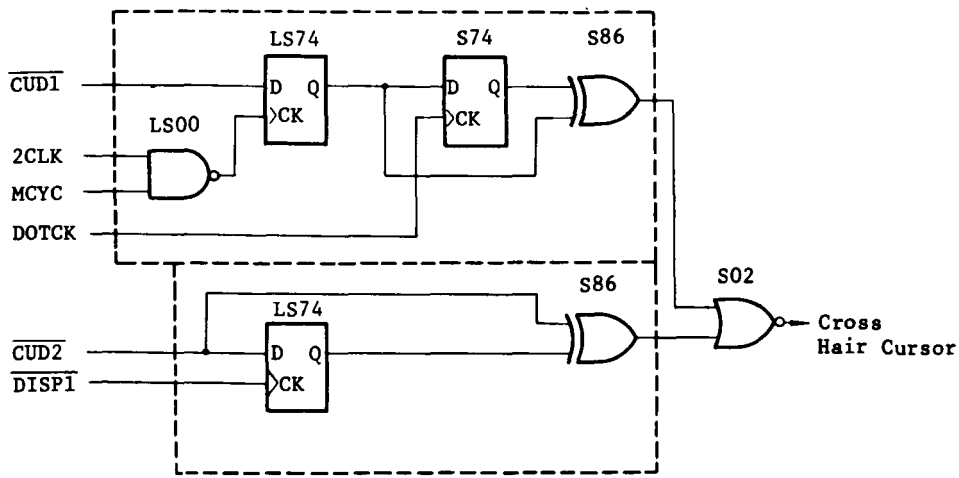
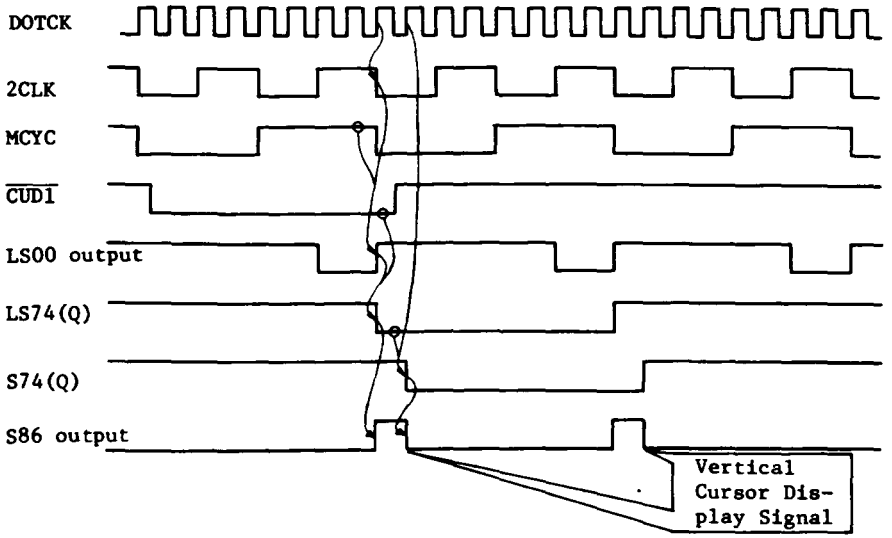
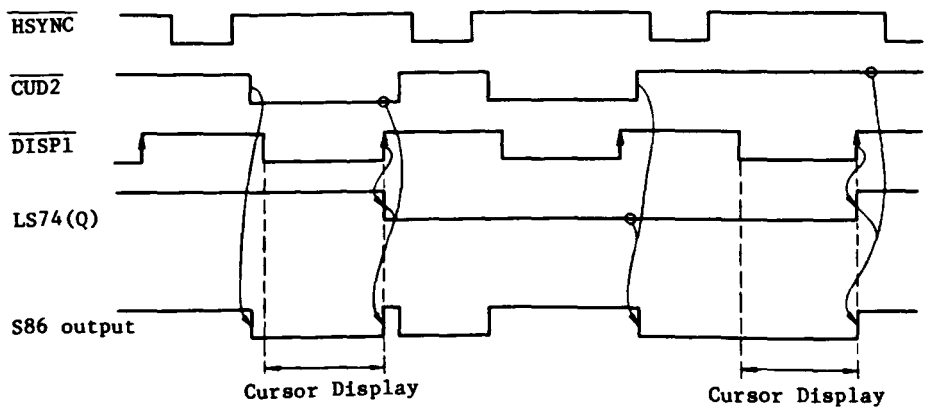


Fig. 6-11 Cross-hair Cursor Display Circuit



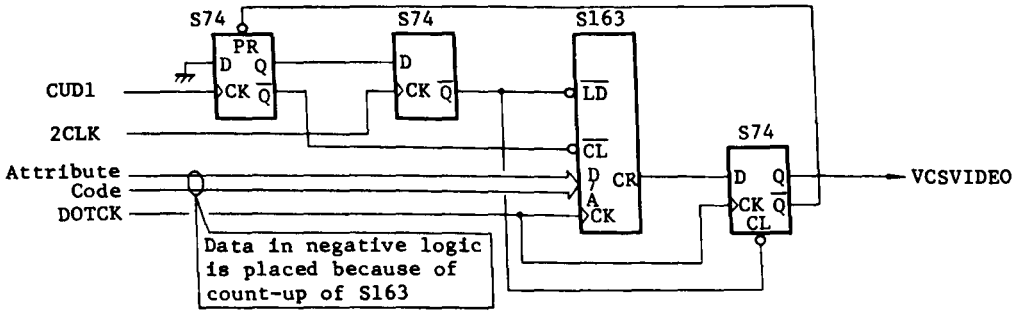
(a) Vertical Cursor Display Timing



(Note 1) When $\overline{\text{DISP1}}$ does not control the display range of CRT screen in the system, be sure to align the phase between the horizontal cursor display and the screen display.

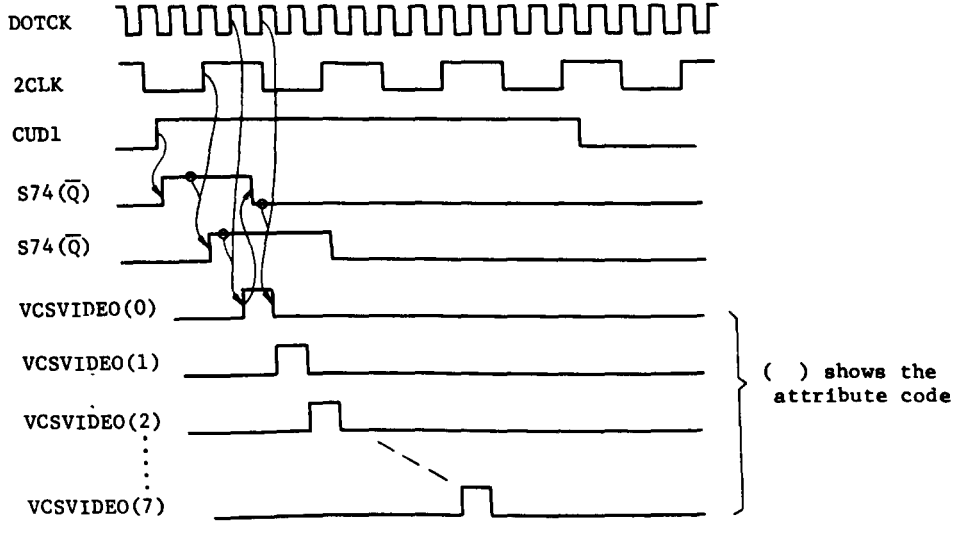
(b) Horizontal Cursor Display Timing

Fig. 6-12 Cursor Display Timing



(Note 1) In this circuit, only one vertical cursor is displayed. To display a second cursor, the duplicate circuit system is required.

(a) Vertical Cursor Display Circuit



(b) Vertical Cursor Display Timing

Fig. 6-13 Vertical Cursor Display (For specifying dot position)

6.4.3 Graphic Cursor

The ACRTC outputs the composite signal of the horizontal and vertical direction component of the graphic cursor from $\overline{CUD1}$ as shown in Fig. 6-14, by the internal register setup shown in Table 6-3.

Table 6-3 Graphic Cursor Setting Register

		Output Signal Name	Register Name	Reg No.	Unit	Set Value
Mode		——	CM	rE8	——	"10" (Graphic cursor mode)
Display start position	Horizontal	$\overline{CUD1}$	CXS	r99	Memory cycle	0 to 255
	Vertical		CYS	r9A,r9B	Raster line	0 to 4095
Size	Horizontal		CXE-CXS	r98,r99	Memory cycle	0 to 255 (Within CRT display range
	Vertical		CYE-CYS	r9A-r9D	Raster line	0 to 4095 (Within CRT display raster
Blink time		——	CON1 COFF1	rE8	4-field time	0 to 7

(Note 1) In the graphic cursor mode, $\overline{CUD2}$ outputs the composite signal of vertical and horizontal components of block cursor 1 and 2. However, the value in CSK (r04) should be set to 1 or more.

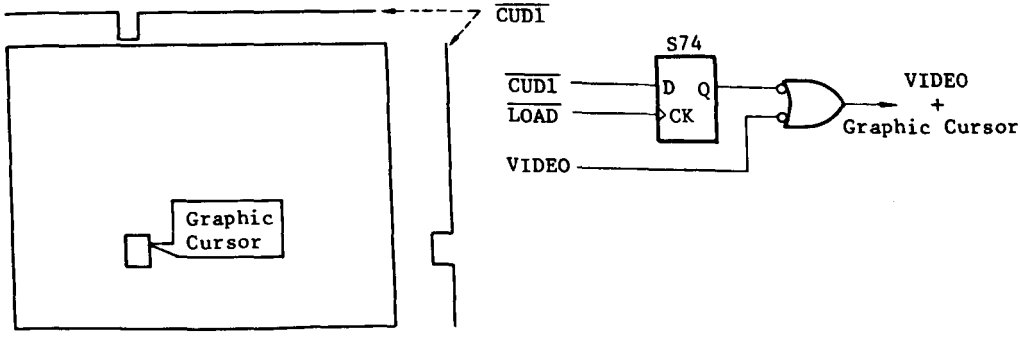


Fig. 6-14 Graphic Cursor Display Circuit

Various shape of graphic cursors can be easily formed providing externally the cursor pattern memory and by controlling its address and display data using the graphic cursor signal ($\overline{CUDI}$).

Fig. 6-15 shows the circuit generating various graphic cursors formed in an 8 × 16 matrix.

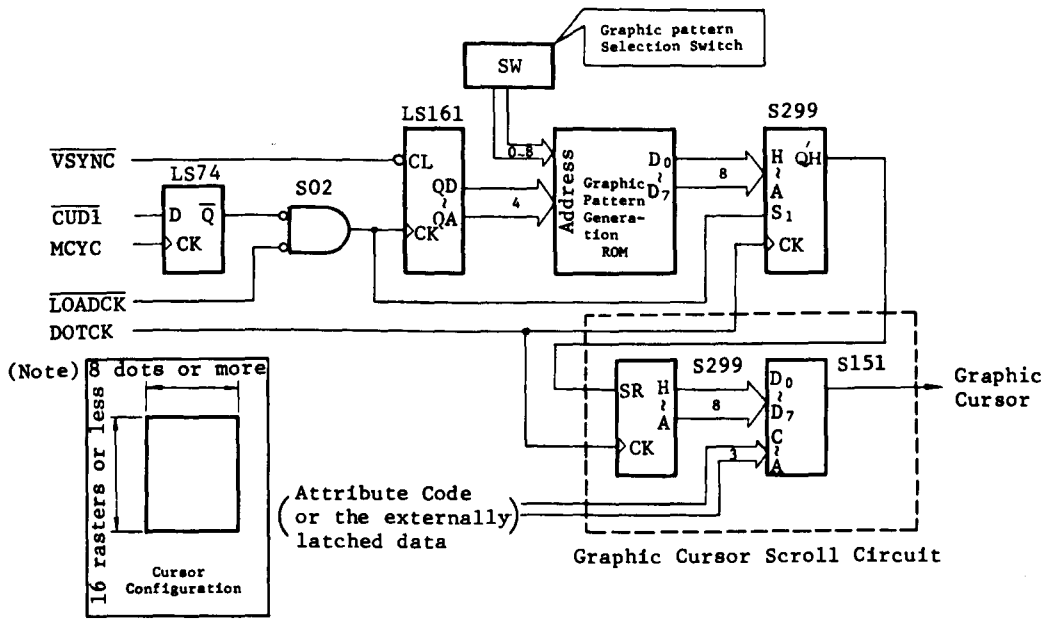
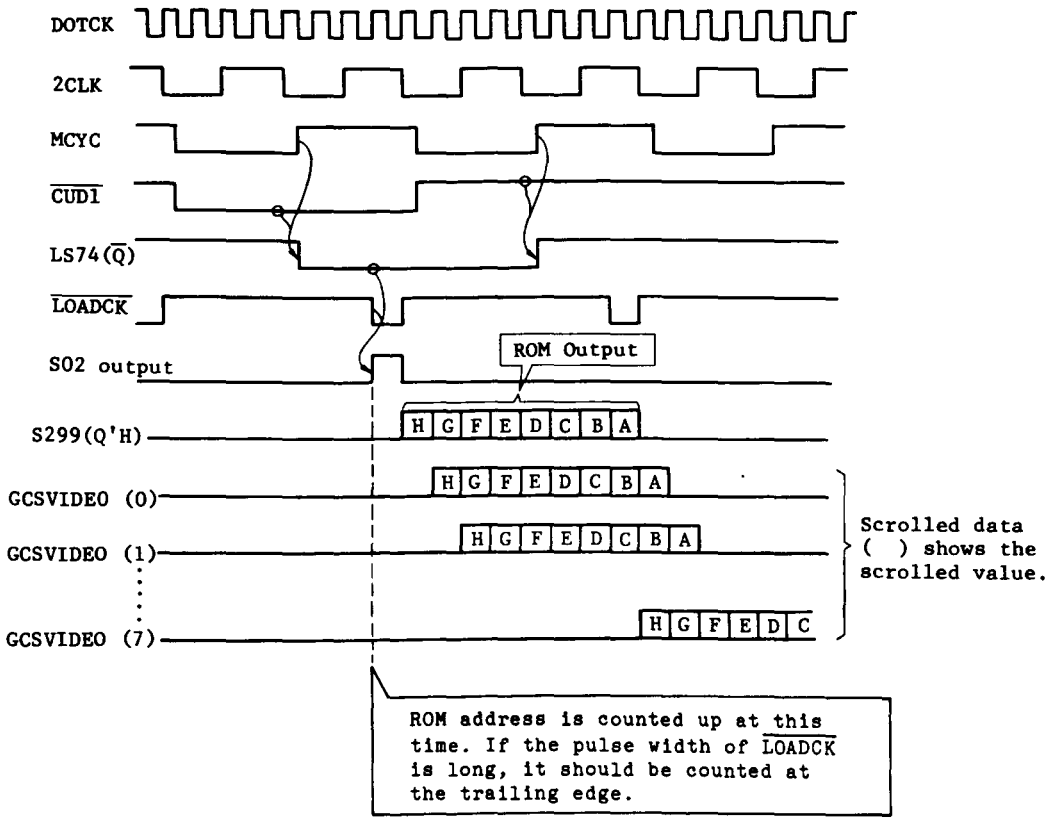


Fig. 6-15 Graphic Cursor Display Circuit



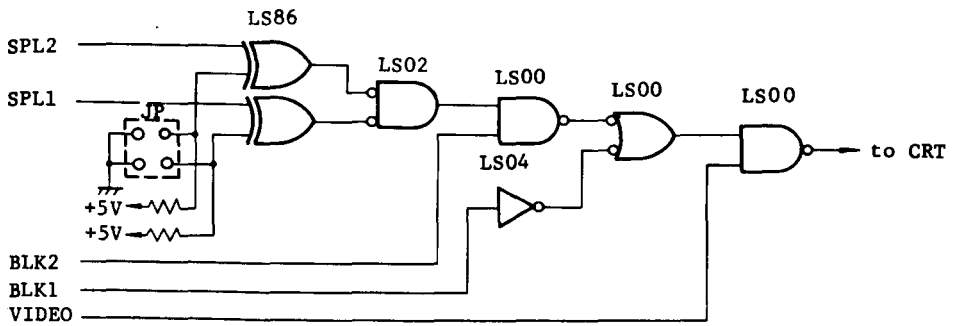
(Note) Total cursor width is (CUDI is "L") 8 dots.

Fig. 6-16 Graphic Cursor Display Timing

6.5 Blink and Split Control

Attribute signals, BLK1 and BLK2 periodically change their output level, "H" "L", every 4-field based on the Blink Control Register (BCR) Setup.

SPL1 and SPL2 output the split screen number which is currently displayed by ACRTC, However, the SPL cannot indicate the window screen information. The BLK1 and BLK2 are used to blink specific characters. Also, by combining them with SPL performs the blink in units of screens. Fig. 6-17 shows the circuit to blink screens.



* BLK2 performs blink for specific screen and BLK1 performs blink for the whole screen.

Fig. 6-17 Screen Blink Circuit

This page intentionally left blank.

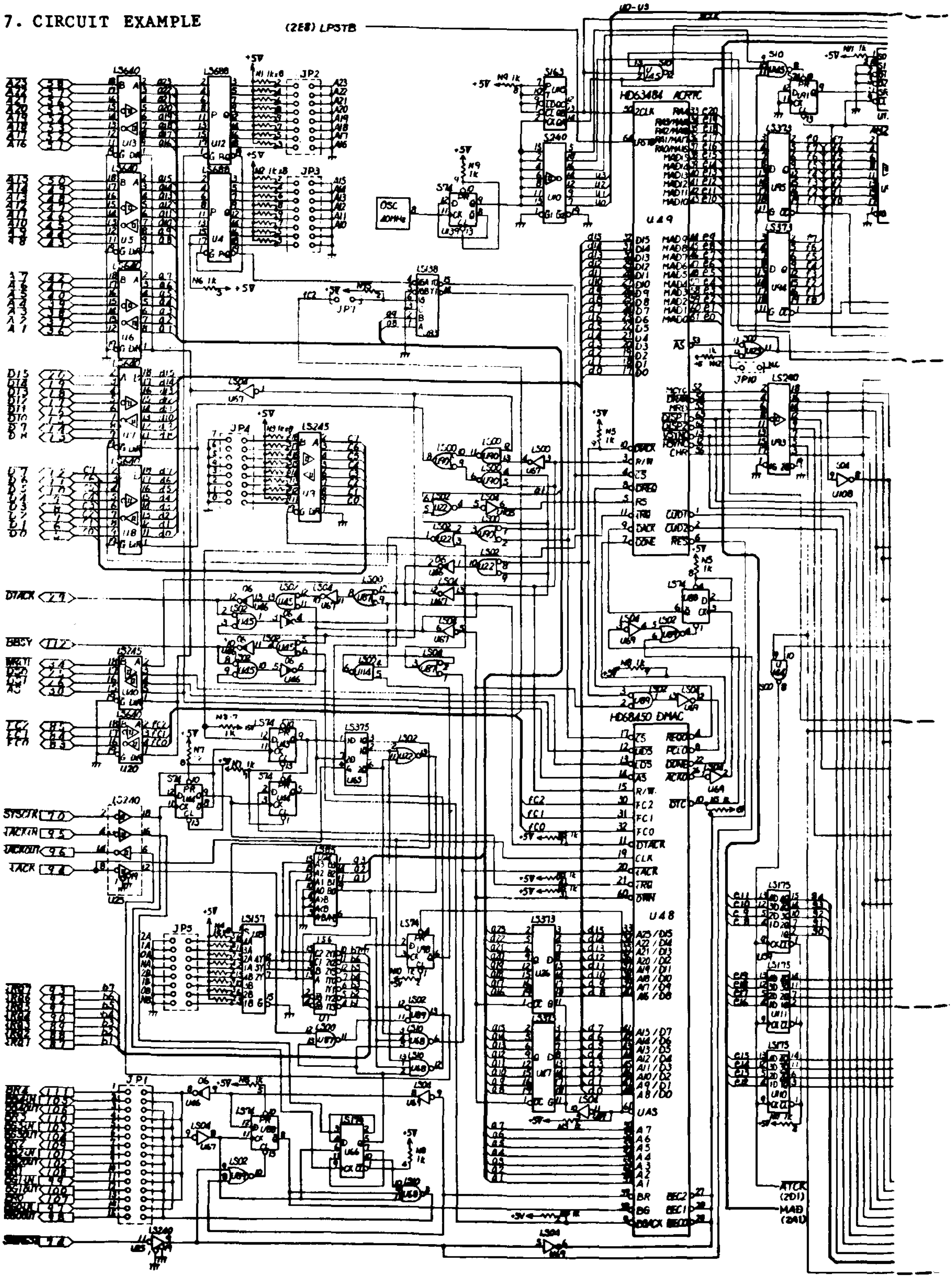
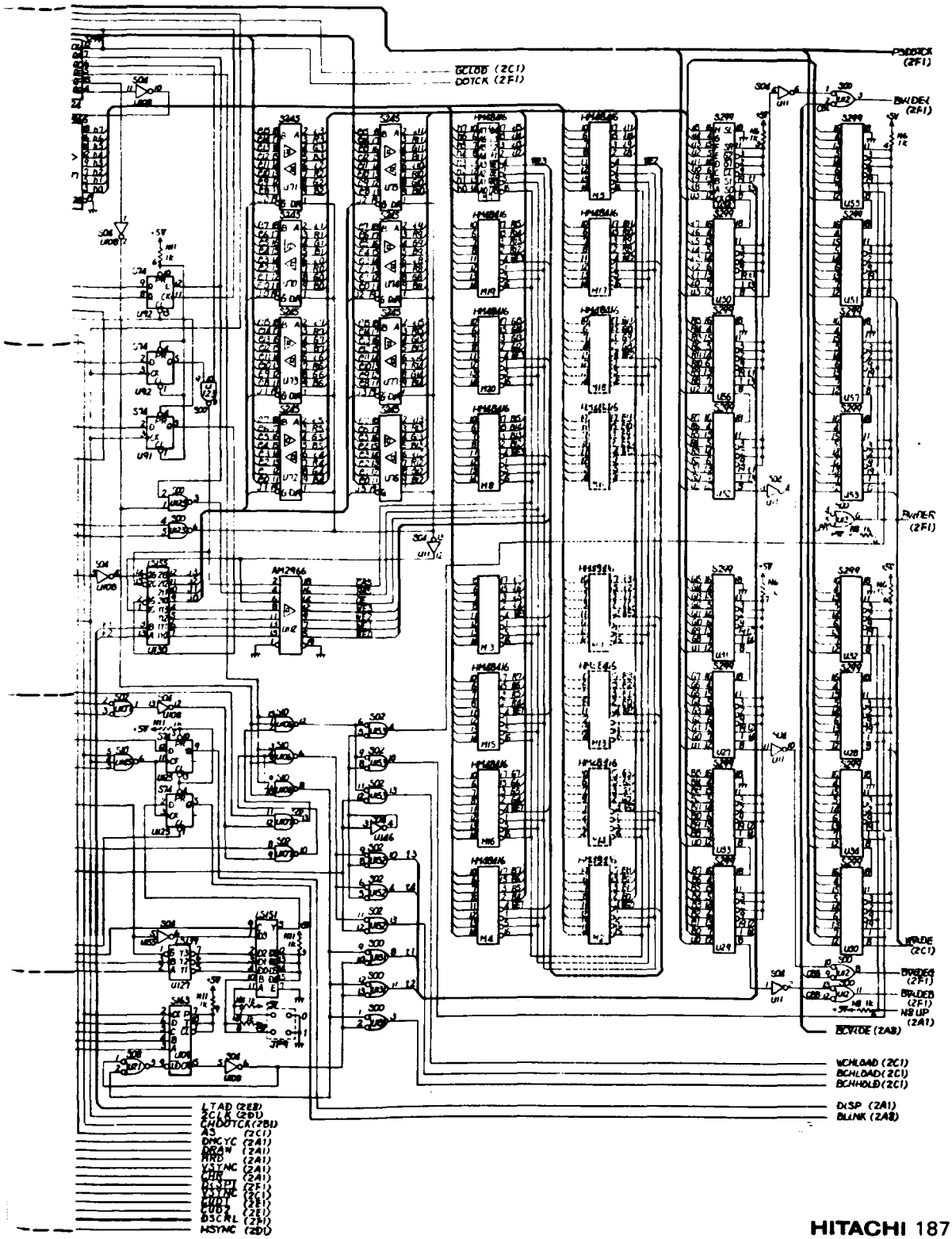


Fig. 7-1 (a) Application Circuit


GCLOB (2C1)
DOTCK (2F1)
BWIDE (2A3)
WCHLOAD (2C1)
BCHLOAD (2C1)
BCHHOLD (2C1)
DISP (2A1)
BLINK (2A3)
LTAD (2E3)
2CLK (2D1)
CHDOTCK (2B1)
AS (2C1)
DMCYC (2A1)
DRAW (2A1)
MRD (2A1)
VSYNC (2A1)
CDR (2A1)
DISPEN (2F1)
VSYNC (3C1)
CURL (2E1)
CUD2 (2E1)
DSCRL (2F1)
HSYNC (2D1)

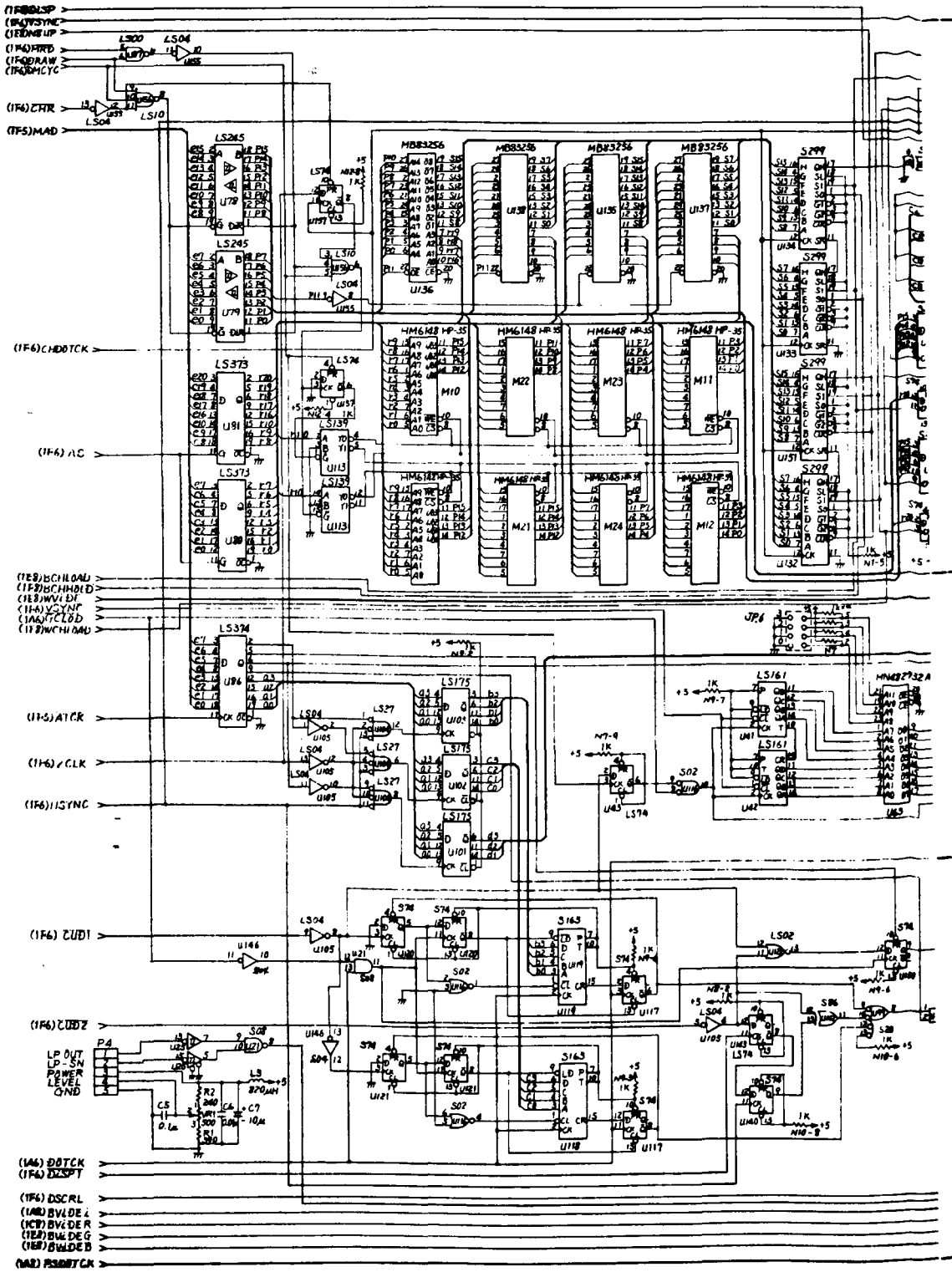
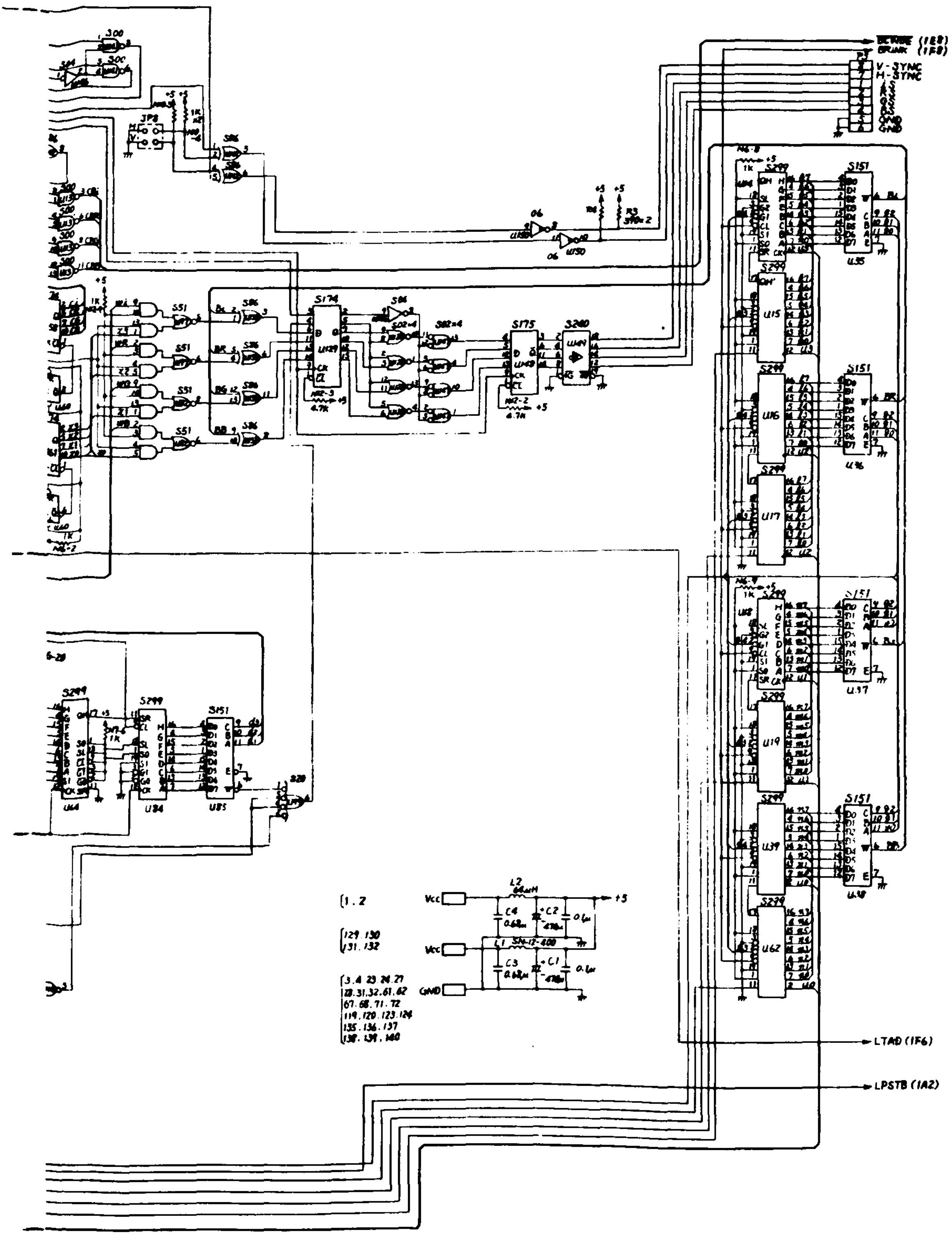


Fig. 7-1 (b) Application Circuit



V-SYNC
H-SYNC
GND
S299
S151
S174
S175
S240
JP8
L2 64μH
C4 0.68μ
C2 470μ
0.1μ
L1 SN-12-400
C3 0.68μ
C1 470μ
Vcc
GND
+5
LTAD (1F6)
LPSTB (1A2)

This page intentionally left blank.

Volume 3
Software

Table of Contents

1. DIRECTIONS FOR USING THE REGISTER SET 193
1.1 Register Configuration and Access Method 193
1.1.1 Register directly accessible by the MPU 193
1.1.2 Registers accessed via FIFO 196
1.2 Screen Configuration 197
1.2.1 Pixel configuration-graphics bit mode (GBM) 197
1.2.2 Memory width (MW) 198
1.3 FIFO 198
1.3.1 Data transfer by program I/O 199
1.4 Pattern RAM 201
1.4.1 Writing to the Pattern RAM 201
1.4.2 Pattern RAM and the drawing command 202
1.5 Drawing Parameter Register 204
1.5.1 Writing to the drawing parameter registers 205
1.5.2 Reading from the drawing parameter registers 205
1.5.3 Color control registers 206
1.5.4 Pattern RAM control registers 207
1.5.5 Area definition registers 208
1.5.6 Pointer control registers 209
1.6 Initialization 211
2. COMMAND TRANSFER 218
2.1 One Word Transfer 218
2.2 Block Transfer 219
3. DIRECTION FOR USING COMMANDS 222
3.1 Coordinate Setup 222
3.2 Screen Clear 224
3.3 Chart Drawing 226
3.3.1 Poly-line chart 226
3.3.2 Bar chart 228
3.3.3 Pie chart 230
3.4 Ellipse Drawing 232
3.5 Character Drawing 234
3.6 Painting Inside a Figure 238
3.7 Software Multi-window 242

4. SCREEN CONTROL 243
4.1 Split Screen 243
4.2 Scroll 246
4.3 Superimposing 249
5. EXAMPLE PROGRAMS 250
5.1 Example of Drawing 250
5.2 Painting the Polygons 251
5.3 Drawing Panda Bears 252

1. DIRECTIONS FOR USING THE REGISTER SET

1.1 Register Configuration and Access Method

Fig. 1-1 shows the programming model of the ACRTC. A group of registers is composed of the control register, FIFO, the drawing parameter register, and the pattern RAM.

There are two types of registers in the ACRTC: one is accessed directly by the MPU, and the other is accessed indirectly by the MPU via FIFO.

1.1.1 Registers Directly Accessible by the MPU

The registers which are directly accessed by the MPU are the address/status register and the control registers.

The address/status register functions as the address register for write, and as the status register for read. One control register is selected by writing the register number of the control register to the address register.

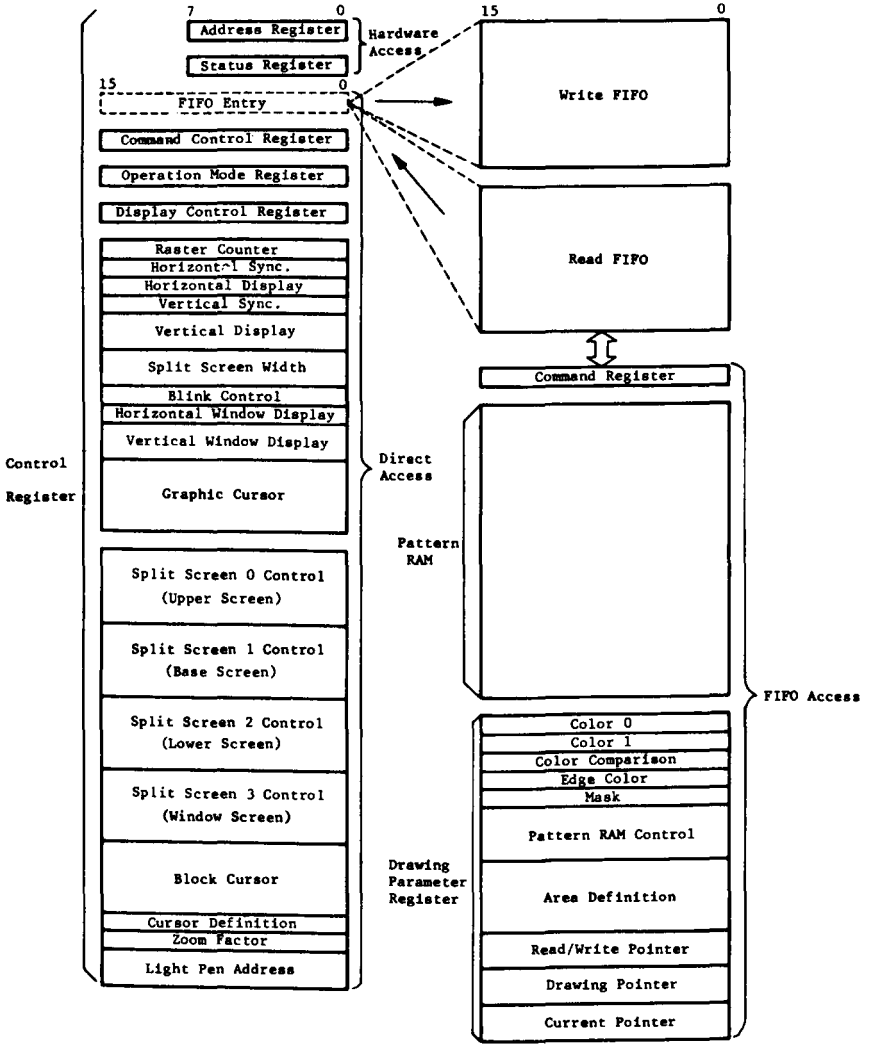


Fig. 1-1 Programming Model

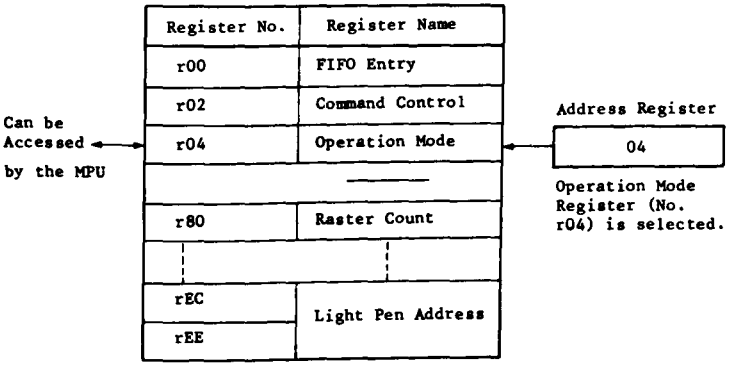


Fig. 1-2 Selection of a Control Register

In the same manner, many of the control registers can be accessed by changing the value of the address register. In the range of r80 to rFF, the contents of the address register are automatically incremented. Therefore, it is unnecessary to rewrite the address register to access the control register consecutively.

In the 16-bit interface mode, the register number (even number) is set in the address register. In the 8-bit interface mode, high byte data or low byte data is accessed by setting an even number respectively or an odd number respectively in the address register.

Example 1) Procedure for Register Access: 8-bit interface

Fig. 1-3 shows the flowchart to setup the graphic cursor register in the 8-bit MPU interface.

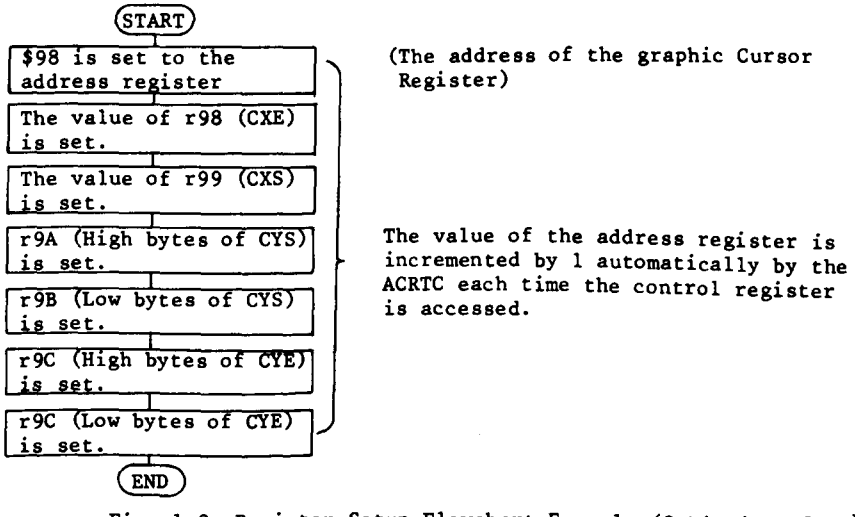


Fig. 1-3 Register Setup Flowchart Example (8-bit interface)

Example 2) Procedure for Register Access: 16-bit interface

Fig. 1-4 shows the flowchart to setup for the graphic cursor register in the 16-bit MPU interface.

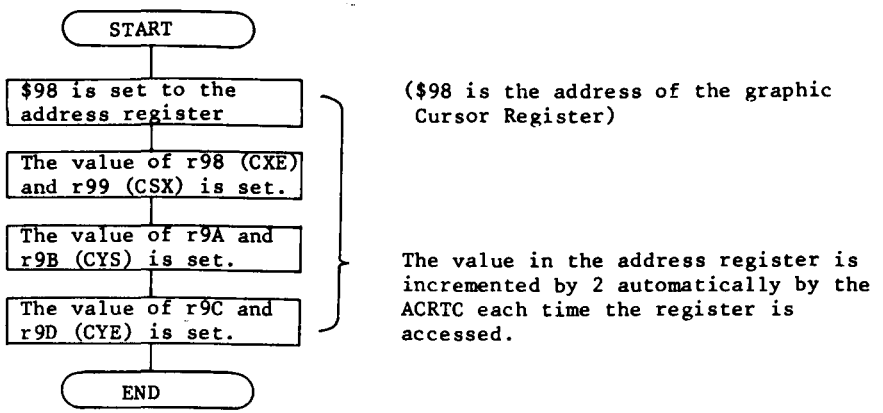


Fig. 1-4 Register Setup Flowchart Example (16-bit interface)

1.1.2 Registers Accessed via FIFO

The command register, the pattern RAM, and the drawing parameter register are accessed by the MPU via the FIFO. Writing to/Reading from the FIFO is performed by specifying the FIFO Entry register.

The command and command parameters written to the write FIFO are transferred to the command register each time the previous command execution is terminated. See Section 2 ''Command Transfer'' for more details.

The pattern RAM is accessed by using the WPTN (Write Pattern RAM) or RPTN (Read Pattern RAM) command. See Section 1.4 "Pattern RAM" for more details.

The drawing parameter registers are accessed by using the WPR (Write Parameter Register) or the RPR (Read Parameter Register) commands. See Section 1.5 ''Drawing Parameter Register'' for more details.

1.2 Screen Configuration

Two-dimensional X-Y coordinate addressing is used to specify the drawing position So, it is necessary to define the relationship between the physical memory address and the two-dimensional logical address space by using the ORG command. After this, the ACRTC can calculate the physical address of the frame buffer from the X-Y coordinate by using the width in the X direction on the basis of the memory width set for each screen.

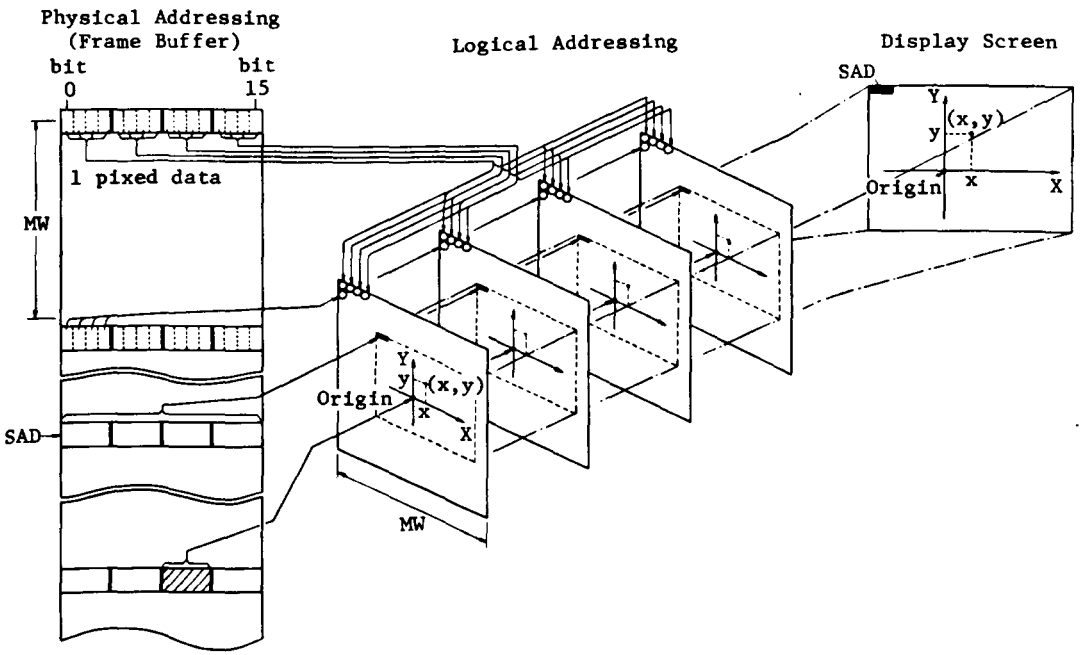


Fig. 1-5 Logical/Physical Addressing

The physical memory of the ACRTC is configured by the memory width (MW) and the data bit/pixel value as shown in Fig. 1-5. In this figure, the data bit/pixel value is 4 bits. The area surrounded by dotted lines in the logical space is specified as the display area, and SAD indicates the display start address location.

1.2.1 Pixel Configuration - Graphic Bit Mode (GBM)

The amount of data assigned to one pixel (bit/pixel) is programmable. There are five choices available as shown in Table 1-1, and the number of colors necessary in user's system can be easily realized.

Table 1-1 Graphic Bit Mode Setting

GBM			Mode	Data Bit per Pixel	Number of colors displayed per pixel	Number of Pixels per Word
10	9	8				
0	0	0	1 bit / pixel	1	2	16
0	0	1	2 bits / pixel	2	4	8
0	1	0	4 bits / pixel	4	16	4
0	1	1	8 bits / pixel	8	256	2
1	0	0	16 bits / pixel	16	65536	1

1.2.2 Memory Width (MW)

The memory width is calculated from the size of the drawing screen in the horizontal direction.

$$MW = \frac{\text{Number of pixels in the horizontal direction}}{\text{Number of pixels / word}}$$

Note: Specify the number of pixels in the horizontal direction to make MW an integer.

The ACRTC supports four screens: the base split screen, the upper split screen, the lower split screen, and the window screen. The memory width can be set for each of these screens individually.

1.3 FIFO

The ACRTC has an internal FIFO to achieve high-speed, effective interface with the MPU. The capacity of the FIFO is 8 words each for the Read FIFO and the Write FIFO (16 words in total).

FIFO Entry (r00) in the control register set is used to read from, or write to, the FIFO.

To read or write data using the FIFO, it is necessary to check the status of the FIFO before the data transfer.

There are two ways to check the status of the FIFO:

a) Checking the FIFO status by reading the status register.

b) Checking the FIFO status by Interrupt.

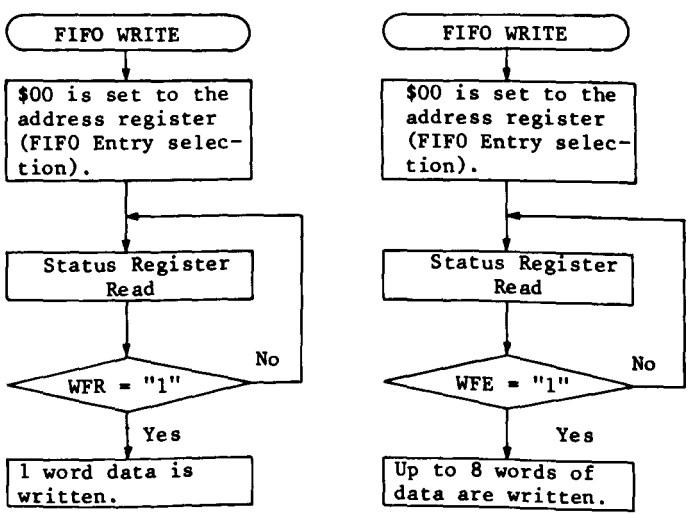
In case of a), the status is shown in bit 0 to bit 3 of the status register.

- Read FIFO Full (RFF: bit 3)
  The Read FIFO is full.
- Read FIFO Ready (RFR: bit 2)
  Data already exists in the Read FIFO.
- Write FIFO Ready (WFR: bwt 1)
  The next word or byte can be written to the Write FIFO.
- Write FIFO Empty (WFE: bit 0)
  The Write FIFO is empty.

In case b), interrupt can be enabled for the above 4 states independently in the command control register.

1.3.1 Data Transfer by Program I/O

Example 1) The Data Transfer to the FIFO - 1

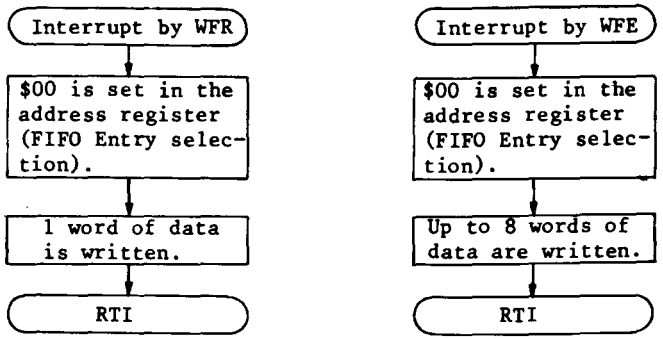


a) Checking Procedure of WFR b) Checking Procedure of WFE

Fig. 1-6 Procedure of the Data Transfer along with Checking the Status Register

In case a), it is confirmed by WFR that at least 1 word is vacant in the FIFO: 1 word of data is written to the FIFO. In case b), it is confirmed by WFE that the FIFO is empty: up to 8 words of data are written to the FIFO. Procedure a) is usually adopted for the command/parameter transfer, while b) is effective when transferring large blocks of data or display list commands.

Example 2) Writing the Data to the FIFO - 2



a) Using Interrupt by WFR b) Using Interrupt by WFE

Fig. 1-7 Data Transfer by Interrupt

In case a), it is confirmed that there is at least 1-word vacancy in the FIFO by the interrupt which occurs when WFR is set. So 1 word of data are written to the FIFO. In case b), it is confirmed that the FIFO is empty by the interrupt which occurs when WFE is set. So up to 8 words of data can be written to the FIFO.

1.4 Pattern RAM

The ACRTC has on-chip 16 word pattern RAM Most graphic drawing commands perform drawing by referring to the pattern RAM data Therefore, it is necessary to write the data to the pattern RAM before issuing a graphic drawing command.

1.4.1 Writing to the Pattern RAM

Write Pattern (WPTN) command is used to write the data to the pattern RAM. The pattern RAM address (PRA), $0 to $F, are allocated to the pattern RAM, and each PRA represents 1 word (16 bits) of pattern RAM. WPTN command can be issued by writing the following to the Write FIFO: PRA at which the writing starts, number of words, and the pattern data.

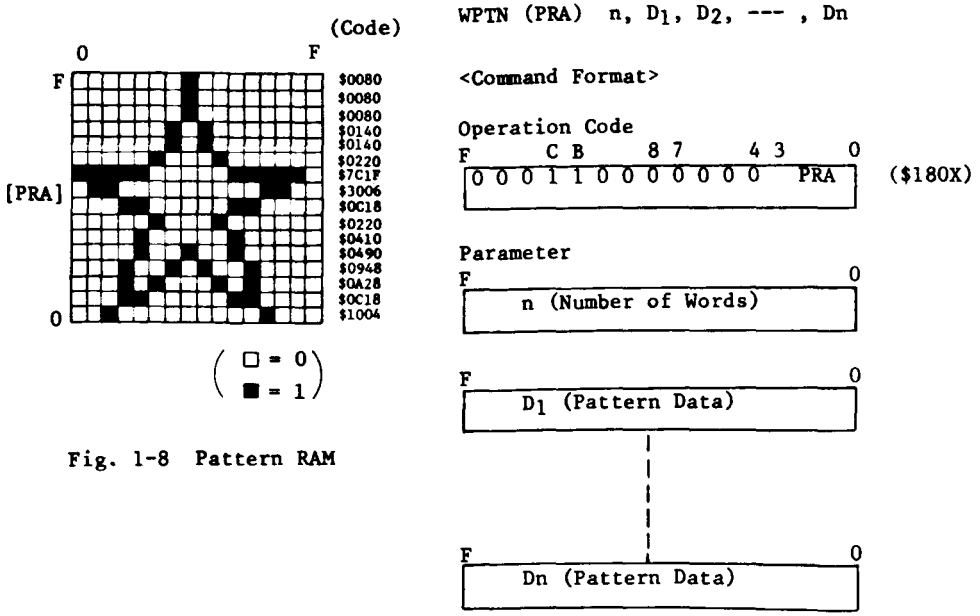


Fig. 1-8 Pattern RAM

Fig. 1-9 WPTN Command

The following gives an example of writing the figure in Fig. 1-8 to the pattern RAM (writing starts from address 0).

```
WPTN    16, $1004, $0C18, $0A28, $0948, $0490, $0410, $0220, $0C18
($1800)'    $3006, $7C1F, $0220, $0140, $0140, $0080, $0080, $0080
(PRA = 0)
```

Example 1-1 Writing to the Pattern RAM

1.4.2 Pattern RAM and the Drawing Command

Graphic drawing commands are classified into two groups: line drawing and plane drawing.

- Line drawing group (LINE, RCT, PLL, PLG, CRCL, ELPS, ARC, EARC, DOT)

When the pattern RAM data is used as binary data, 16 different line information can be stored because each word in the pattern RAM holds 1 line information.

The line information which is used for drawing is selected by the pattern point (PPY) in the drawing parameter register. The line information is the bit information within the range specified by the pattern start position (PSX) and the pattern end position (PEX). Reference to the pattern RAM starts from the bit specified by the pattern pointer (PPX), then the pattern pointer (PPX) is shifted as the drawing proceeds. Arrows in the Fig. 1-10 show the reference direction.

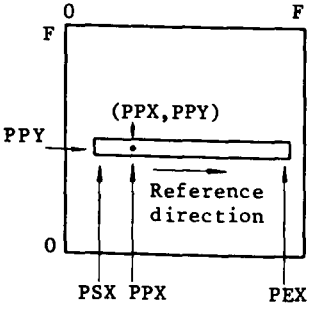


Fig. 1-10 Line Drawing Setting

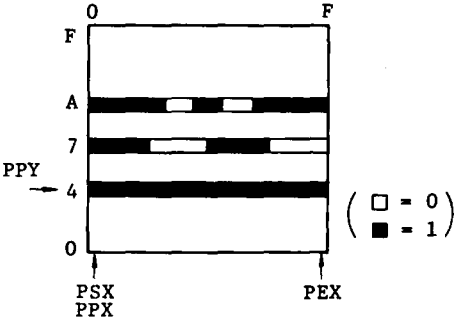


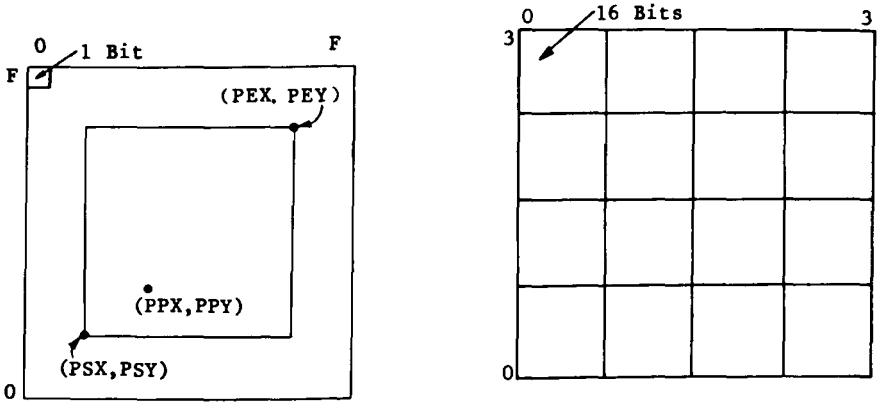
Fig. 1-11 Example of Line Type

If the line information such as Fig. 1-11 is stored in the pattern RAM, the figure is drawn with a solid line when pattern pointer is set to "4". The figure is drawn with dotted lines when the pattern pointer (PPY) is set to "7", and with broken lines when being set to "A".

- Plane Drawing Group

A figure of the arbitrary size of up to 16 dots × 16 dots can be stored when the pattern RAM data is used as binary data for a plane. Plane information, which is in the rectangle area specified by the pattern start position (PSX, PSY) and the pattern end position (PEX, PEY), is used for drawing a plane.

Reference to the pattern RAM starts from the bit specified by the pattern pointer (PPX, PPY), then the pattern pointer is shifted as the drawing proceeds.



a) Pattern RAM "indirect" mode (COL = is, 01, 10)

b) Pattern RAM "direct" mode (COL = 11)

Fig. 1-12 Plane Drawing Setup

The information in the pattern RAM can be also used as frame buffer color data. In this case, the bit information of the pattern RAM is used as the color information (in 16 bits / pixel mode) of up to 4 pixels × 4 pixels. For line drawing, 4 types of line information of up to 4 dots can be stored in the pattern RAM. Also for plane drawing, a figure of up to 4 pixels × 4 pixels can be stored. The value of the pattern start position (PSX, PSY), the pattern end position (PEX, PEY), and the pattern pointer (PPX, PPY) must be 0 to 3.

Whether the pattern RAM data are used as binary data or used as color data is specified by setting the "COL" bit in the graphic drawing command.

Table 1-2 Color, Mode

COL	Color Mode
00	When Pattern RAM data = 0, Color Register 0 is used. When Pattern RAM data = 1, Color Register 1 is used.
01	When Pattern RAM data = 0, drawing is suppressed. When Pattern RAM data = 1, Color Register 1 is used.
10	When Pattern RAM data = 0, Color Register 0 is used. When Pattern RAM data = 1, drawing is suppressed.
11	Pattern RAM contents are directly used as color data.

1.5 Drawing Parameter Register

The ACRTC has an internal drawing parameter register set which is used for the color control, pattern RAM control, area definition, and pointer control. Table 1-3 shows the drawing parameter registers.

Table 1-3 Drawing Parameter Registers

Register No.	Read/ Write *	Name of Register	Abbr.	Data (H) 15	14	13	12	11	10	9	8	Data (L) 7	6	5	4	3	2	1	0
Pr00	R/W	Color 0	CL0	CL0															
Pr01	R/W	Color 1	CL1	CL1															
Pr02	R/W	Color Comparison	CCMP	CCMP															
Pr03	R/W	Edge Color	EDG	EDG															
Pr04	R/W	Mask	MASK	MASK															
Pr05	R/W	Pattern RAM Control	PRC	PPY								PZCY					PPX		PZCX
↓				PSY								■					PSX		■
Pr07				PEY								PZY					PEX		PZX
Pr08	R/W	Area Definition **	ADR	XMIN															
↓				YMIN															
				XMAX															
Pr0B				YMAX															
Pr0C	R/W	Read Write Pointer	RWP	DN	■							RWPH							
Pr0D				RWPL								■							
Pr0E	–	.....	–	–															
Pr0F				–															
Pr10	R	Drawing Pointer	DP	DN	■							DPAH							
Pr11				DPAL														DPD	
Pr12	R	Current Pointer	CP	X															
Pr13				Y															
Pr14	–	.....	–	–															
Pr15				–															

* R .... Register readable by a Read Parameter Register (RPR) command
W .... Register writable by a Write Parameter Register (WPR) command
– .... Access is not allowed
■ .... Always set to "0"
** ..... Set binary complements for negative values of X and Y axis.

1.5.1 Writing to the Drawing Parameter Register

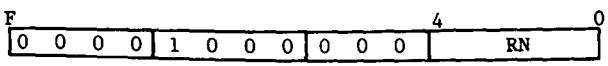
The Write Parameter Register (WPR) command is used to write the data to a specific drawing parameter register.

<Mnemonic>

WPR (RN) D

<Command Format>

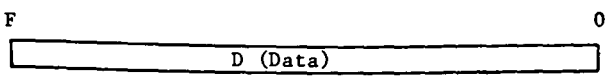
Operation Code



($08XX)

5 bits

Parameter



"RN" bits in the operation code of WPR command specifies the drawing parameter register number shown in Table 1-3. The WPR command is issued by writing the operation code and the data to the Write FIFO.

An example of writing the data $3333 to the color 1 register is shown below.

WPR (RN=1) ($0801) , $3333

Example 1-2 WPR Command

1.5.2 Reading from the Drawing Parameter Registers

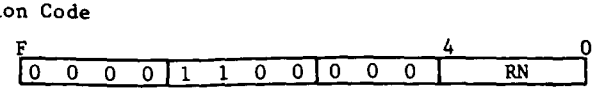
The Read Parameter Register (RPR) command is used to read data from a specific drawing parameter register.

<Mnemonic>

RPR (RN)

<Command Format>

Operation Code



($0C0X)

5 bits

The "RN" bits in the operation code of RPR command specifies the drawing parameter register number shown in Table 1-3. The RPR command is issued by writing the operation code to the Write FIFO. The data read out from the register is set to the Read FIFO after issuing RPR command, therefore, it is necessary to empty the Read FIFO before issuing RPR command.

The example of reading the mask register (MASK) data is shown below.

```
RPR
($0C04)
(RN=4)
```

Example 1-3 RPR Command

1.5.3 Color Control Registers

The drawing parameter register (Pr0-Pr4) is used for the color control.

(1) Color 0, Color 1 Register

Color 0 and Color 1 are the registers which define the drawing color. These registers also define the drawing color which is specified by the binary data in the pattern RAM. On the other hand, by defining Color 0 = Color 1 regardless of the contents of the pattern RAM, solid color lines or planes can be drawn.

(2) Color Comparison Register

The color comparison register defines the comparison color in the color operation mode. The color operation mode is specified by "OPM" bit in the graphic drawing operation code. The color comparison register is used when "OPM = 100" or OPM = 101". Table 1-4 shows the color operation modes.

Table 1-4 Color Operation Mode

OPM		Operation Mode
000	REPLACE	Replaces the frame buffer data with the color data.
001	OR	ORs the frame buffer data with the color data. The result is rewritten to the frame buffer.
010	AND	ANDs the frame buffer data with the color data. The result is rewritten to the frame buffer.
011	EOR	EORs the frame buffer data with the color data. The result is rewritten to the frame buffer.

-to be continued

OPM		Operation Mode
100	CONDITIONAL REPLACE (P = CCMP)	When the frame buffer data at the drawing position (P) is equal to the color comparison register (CCMP), the frame buffer data is replaced with the color data.
101	CONDITIONAL REPLACE (P ≠ CCMP)	When the frame buffer data at the drawing position (P) is not equal to the color comparison register (CCMP), the frame buffer data is replaced with the color data.
110	CONDITIONAL REPLACE (P < CL)	When the frame buffer data at the drawing position (P) is less than the color register data (CL), the frame buffer data is replaced with the color data.
111	CONDITIONAL REPLACE (P > CL)	When the frame buffer data at the drawing position (P) is greater than the color register data (CL), the frame buffer data is replaced with the color data.

(3) Edge Color Register

The edge color register defines the boundary edge color which specifies the area to be painted by the paint command.

(4) Mask Register

The mask register is used to mask bits that should not have drawing or other logical operations be performed by the data transfer command (DMOD, MOD, SCLR, SCPY). The bits which are set to "1" in the mask register are modified, and the bits which are set to "0" are not modified.

As the color control register contains 16 bits, the data for 4 pixels can be stored in the mask register when using 4 bits/pixel mode.

1.5.4 Pattern RAM Control Register

The drawing parameter registers (Pr5 ∿ Pr7) are used for the pattern RAM control.

The pattern RAM control register specifies the size of the patterns used for drawing, the pattern scan position, and the zoom coefficient.

The size of the pattern used for drawing is specified by the start point (PSX, PSY) and the end point (PEX, PEY). The reference point on the pattern is specified by the pattern point (PPX, PPY). The value which specifies the size of the pattern is 0 to 15 when using binary data (when color 0 and color 1 are selected and used), and 0 to 3 in case the pattern RAM data is used as the color data.

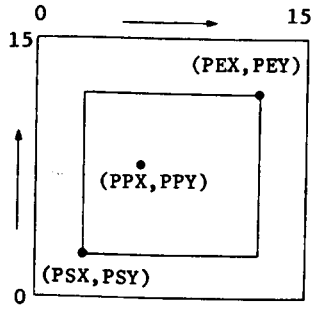


Fig. 1-13 Pattern RAM Setup

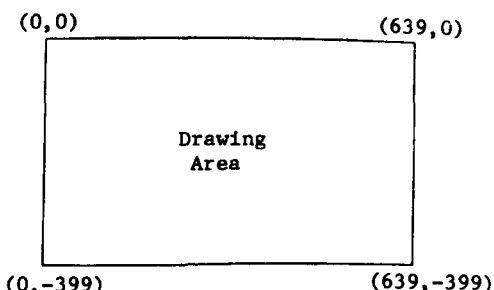
The magnification coefficient of the pattern RAM is set as the pattern zoom (PZX, PZY). The pattern is enlarged up to 16 times by setting between 0 and 15 for the zoom count.

The pattern zoom count (PZCX, PZCY) indicates the status of the ongoing magnification of the drawing. The pattern is enlarged by repeatedly using the data at the reference point. The number of times repeated is the value set in the pattern zoom (PZX, PZY). The pattern zoom count counts the number of time it is repeated.

The pattern zoom count (PZCX, PZCY) needs to be set to zero at the time of intialization. (PZCX, PZCY) is normally used to restart the PAINT command for drawing unpainted areas. (PZX, PZY) (PZCX, PZCY) is pushed onto the stack as "Pattern Pointer" together with the current pointer (CPX, CPY) when the PAINT command detects unpainted areas.

1.5.5 Area Definition Registers

The drawing parameter register (Pr8 ~PrB) is used for area definition. The area of XMIN ≤ X ≤ XMAX, YMIN ≤ Y ≤ YMAX is defined as the drawing area. A negative value is set by using 2's complement. An example of the setting is shown in Fig. 1-14.



XMIN = 0 : $0000
YMIN = -399: $FE71
XMAX = 639 : $027F
YMAX = 0 : $0000

Fig. 1-14 Area Definition Setting

The defined area is referred to by the "AREA" bit in the graphic drawing command. Table 1-5 shows the various drawing area modes.

Table 1-5

AREA	Drawing Area Mode
000	Drawing is executed without Area checking.
001	When attempting to exit the Area, drawing is stopped and the Abort bit (ABT) is set.
010	Drawing suppressed outside the Area - drawing operation continues and the ARD (Area Detect) flag is not set.
011	Drawing suppressed outside the Area - drawing operation continues and the ARD (Area Detect) flag is set.
100	Same as AREA = 000.
101	When attempting to enter the Area, drawing is stopped and the Abort bit (ABT) is set.
110	Drawing suppressed inside the Area - drawing operation continues and the ARD (Area Detect) flag is not set.
111	Drawing suppressed inside the Area - Drawing operation continues and the ARD flag is set.

1.5.6 Pointer Control Registers

The drawing parameter register (PrC ∿ Pr0) and (Pr10 ∿ Pr13) is used to control the pointer.

(1) Read/Write Pointer Registers

Read/write pointer specifies a 20-bit physical frame buffer address for use with the data transfer command (DRD, DWT, DMOD, RD, WT, MOD, CLR, SCLR, CPY, SCPY). The setup is done using physical addresses in the frame buffer memory. One of the four split screens controlled by the ACRTC is selected by DN, and the upper 8 bits and the lower 12 bits of physical address are respectively set as "RWPH" and "RWPL". Read/write pointer must be set before the data transfer command is issued, and the value of the read/write pointer (except "DN") varies after the command is issued.

DN	Selected Screen
00	Upper Screen (Split Screen 0)
01	Base Screen (Split Screen 1)
10	Lower Screen (Split Screen 2)
11	Window Screen

Fig. 1-15 "DN" Setup

(2) Drawing Pointer Register

The drawing pointer contains the physical drawing address calculated during drawing commands. The drawing pointer can be set only by issuing the ORG command. "DN" indicates the drawing screen, and the upper 8 bits and lower 12 bits of the physical drawing address of the drawing screen are indicated by "DPAH" and "DPAL". "DPD" indicates the pixel address of the drawing point in one memory word. The pixel address varies according to the graphic bit mode (GBM) as shown below.

1 bit / pixel:	Specifies the pixel position by the 4 bits of DPD.
2 bits / pixel:	Specifies the pixel position by the upper 3 bits of DPD. The lower one bit is not used.
4 bits / pixel:	Specifies the pixel position by the upper 2 bits of DPD. The lower 2 bits are not used.
8 bits / pixel:	Specifies the pixel position by the upper one bit of DPD. The lower 3 bits are not used.
16 bits / pixel:	DPD is not used.

(3) Current Pointer Register

The current pointer register indicates the X-Y coordinates of the current drawing address. The current pointer moves by the execution of the graphic drawing command. The current pointer is cleared to "0, 0" only by issuing "ORG" command. Negative coordinates are indicated by 2's complement.

1.6 Initialization

To use the ACRTC properly, appropriate values must be set in each control registers according to the hardware configuration and the specification of the CRT.

r82 - r8F and r92 - r97 are set according to the specification of CRT to be connected. Fig. 1-16 shows the register which controls the display screen. One horizontal cycle is the number of memory cycles in one display line, and one vertical cycle is the number of rasters in one frame.

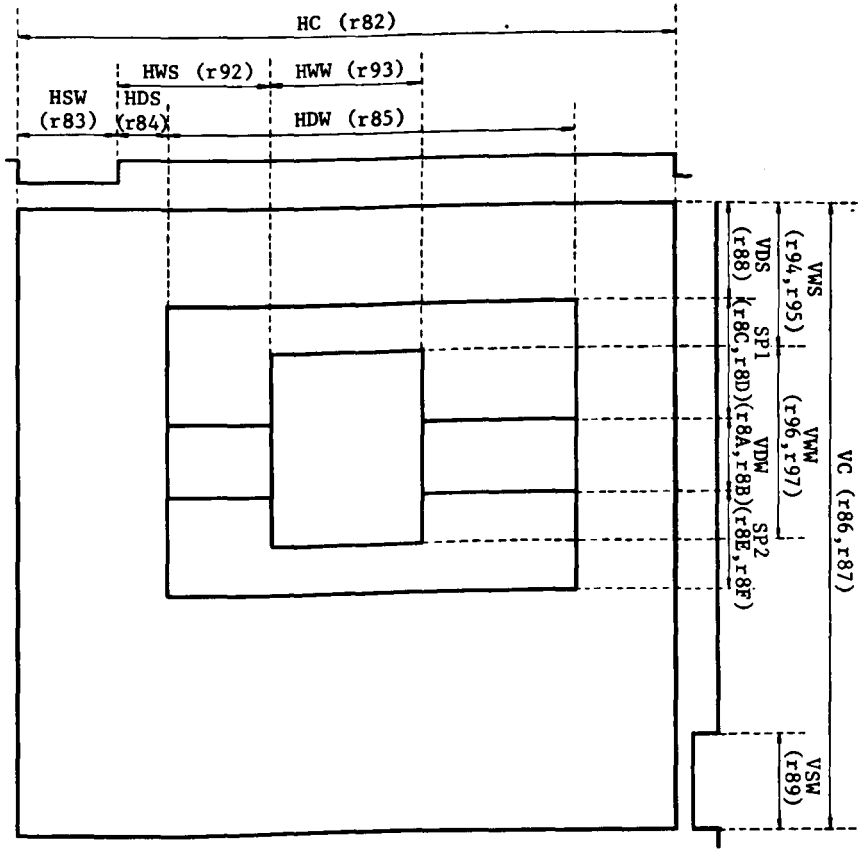


Fig. 1-16 Display Screen Specification

Memory Cycle (Ts) is calculated using the following equations by defining the 1 horizontal display period as "T_H", the number of horizontally displayed dots as "Nd", and the number of displayed dots in 1 display cycle as Ns. This is as follow,

$$Ts\ (ns) = \frac{T_H\ (\mu s) \times 1000}{Nd\ (dot)} \times Ns\ (dot) \quad : \text{Single Access Mode}$$

$$Ts\ (ns) = \frac{T_H\ (\mu s) \times 1000}{Nd\ (dot)} \times Ns\ (dot) \times \frac{1}{2} \quad : \text{Dual Access Mode}$$

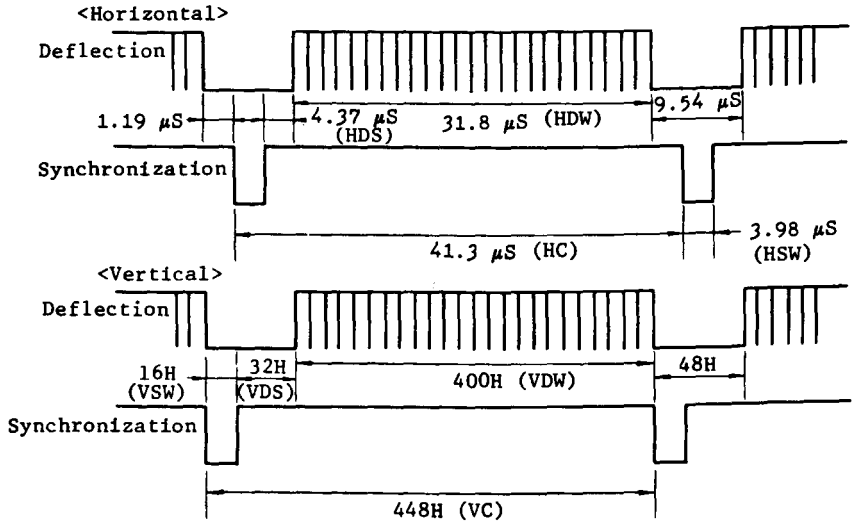


Fig. 1-17 CRT Specification

If the number of dots displayed during 1 display cycle is 16 and 640 × 400 dots are displayed on the CRT with timing shown in Fig. 1-17, the memory cycle is as follows.

$$Ts\ (ns) = \frac{31.8\ (\mu s) \times 1000}{640\ (dot)} \times 16\ (dot) = 795\ (ns) \quad : \text{Single Access Mode}$$

$$Ts\ (ns) = \frac{31.8\ (\mu s) \times 1000}{640\ (dot)} \times 15\ (dot) \times \frac{1}{2} = 397.5\ (ns) \quad : \text{Dual Access Mode}$$

The value set for each register in the dual access mode would be:

$$\text{HC (r82)} = \frac{41.3(\mu s) \times 1000}{397.5\ (ns)} - 1 = 103 \quad (\$67)$$

$$\text{HSW (r83)} = \frac{3.98(\mu s) \times 1000}{397.5\ (ns)} = 10 \quad (\$A)$$

$$\text{HDS (r84)} = \frac{4.37(\mu s) \times 1000}{397.5\ (ns)} - 1 = 10 \quad (\$A)$$

$$\text{HDW (r85)} = \frac{31.8(\mu s) \times 1000}{397.5\ (ns)} - 1 = 79 \quad (\$4F)$$

VC (r86-7) = 448 ($1C0)

VDS (r88) = 32 ($20)

VSW (r83) = 16 ($10)

VDW (r8A-B) = 400 ($190)

The above setup allows the base screen display. When only the base screen is displayed, it is unnecessary to initiate r8C - r8F and r92 - r97.

rC0 - rDF are used to set the screen configuration. The start address and the memory width of the each split screen are set in words. The start address and the memory width can be set without restraint within the frame buffer memory.

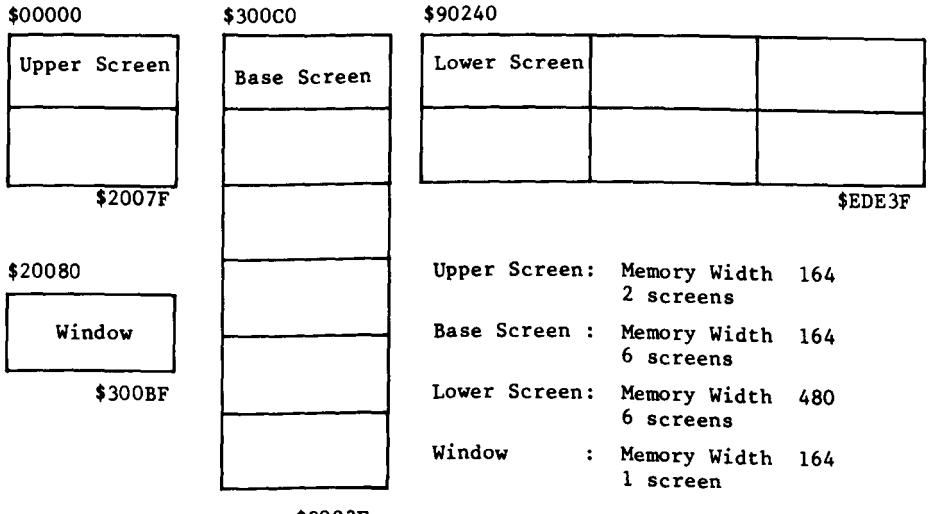


Fig. 1-18 Split Screens Configuration

The following program is used to initialize the ACRTC. This program is written in HD68000 assembly language. The label "ACRTC" within the program is the label set by the program control instruction "ACRTC EQU $XXXXXX", which indicates the address in the system where the ACRTC is mapped.

```
INIT    LEA     INITTBL, A1     :  Sets the start address of the data table.
*
        MOVE    #$82, ACRTC     :  Selects the r82 register.
        MOVE    #13, D2         :  Sets the loop counter.
INITI   MOVE    (A1)+, ACRTC+2: Writes in the setup value to r82 - r9D.
        DBRA    D2, INIT1
*
        MOVE    #$C0, ACRTC     :  Selects the rC0 register.
        MOVE    #21, 02         :  Sets the loop counter.
INIT2   MOVE    (A1)+, ACRTC+2: Writes in the setup value to rC0 - rEB.
        DBRA    D2, INIT2
*
        MOVE    #$02, ACRTC     :  Selects the r02 register.
        MOVE    (A1)+, ACRTC+2: Writes the setup value to r02.
        DBRA    D2, INIT2
*
        MOVE    #$02, ACRTC     :  Selects the r02 register.
        MOVE    (A1)+, ACRTC+2: Writes the setup value to r02.
        MOVE    #$06, ACRTC     :  Selects the r06 register.
        MOVE    (A1)+, ACRTC+2: Writes the setup value to r06.
        MOVE    #$04, ACRTC     :  Selects the r04 register.
        MOVE    (A1)+, ACRTC+2: Writes the setup value to r04.
```

Fig. 1-19 Example of ACRTC Initialization Program

```
*
*****************************
*                           *
*   ACRTC INITIALIZE DATA   *
*                           *
*****************************
*
INITTBL   DC      $680A         R82:HORIZONTAL SYNC.
          DC      $0B50         R84:HORIZONTAL DISPLAY
          DC      $01C0         R86:VERTICAL SYNC.
          DC      $2010         R88:VERTICAL DISPLAY
          DC      $0190         R8A:SPLIT SCREEN WIDTH SP1 (BASE)
          DC      $0000         R8C:SPLIT SCREEN WIDTH SP0 (UPPER)
          DC      $0000         R8E:SPLIT SCREEN WIDTH SP2 (LOWER)
          DC      $0000         R90:BLINK CONTROL
          DC      $0000         R92:H-WINDOW DISPLAY
          DC      $0000         R94:V-WINDOW DISPLAY
          DC      $0000         R96:
          DC      $0000         R98:GRAPHIC CURSOR
          DC      $0000         R9A:
          DC      $0000         R9C:
*
          DC      $0000         RC0:RASTER ADDR. SCREEN 0 (UPPER)
          DC      $00A4         RC2:MEMORY WIDTH
          DC      $0F00         RC4:START ADDR. H
          DC      $0000         RC6:START ADDR. L
*
          DC      $0000         RC8:RASTER ADDR. SCREEN 1 (BASE)
          DC      $00A4         RCA:MEMORY WIDTH
          DC      $0F08         RCC:START ADDR. H
          DC      $00C0         RCE:START ADDR. L
*
          DC      $0000         RD0:RASTER ADDR. SCREEN 2 (LOWER)
          DC      $01E0         RD2:MEMORY WIDTH
          DC      $0F09         RD4:START ADDR. H
          DC      $0240         RD6:START ADDR. L
*
          DC      $0000         RD8:RASTER ADDR. SCREEN 3 (WINDOW)
          DC      $00A4         RDA:MEMORY WIDTH
          DC      $0002         RDC:START ADDR. H
          DC      $0080         RDE:START ADDR. L
*
          DC      $0000         RE0:CHARACTER CURSOR
          DC      $0000         RE2:
          DC      $0000         RE4:
          DC      $0000         RE6:
          DC      $0000         RE8:
          DC      $0000         REA:ZOOM FACTOR
*
          DC      $0200         R02:COMMAND CONTROL
          DC      $C128         R04:SYNC. CONTROL
          DC      $4000         R06:DISPLAY CONTROL
```

Fig. 1-20 Example of ACRTC Initialization Data Table

Table 1-6 (a) Programming Model

CONTROL REGISTER

CS	RS	RW	Reg. No.	Register Name	Abbre.	DATA (H) 15	14	13	12	11	10	9	8	DATA (L) 7	6	5	4	3	2	1	0	Setup Value
1	-	-	-	-	-	.....								.....								—
0	0	0	AR	Address Register	AR	.....								Address								—
0	0	1	SR	Status Register	SR	.....								CER	ARD	CED	LPD	RFF	RFR	WFR	WFE	—
		1/0	r00	FIFO Entry	FE	FE																—
		1/0	r02	Command Control	CCR	ABT	PSE	DDM	CDM	DRC	GBM			CRE	ARE	CEE	LPE	RFE	RRE	WRE	WEE	$0200
		1/0	r04	Operation Mode	OMR	M S	STR	ACP	WSS	CSK	DSK			RAM	GAI		ACM		RSM			$C128
		1/0	r06	Display Control	DCR	DSP	SE1	SE0		SE2	SE3			A T R								$4000
			r08 ↓ r7E	(undefined)	-	.....																—
		1	r80	Raster Count	RCR	.....								R C								—
		1/0	r82	Horizontal Sync.	HSR	H C								.....				H S W				$880A
			r84	Horizontal Display	HDR	H D S								H D W								$0B50
		1/0	r86	Vertical Sync.	VSR	.....								V C								$01C0
		1/0	r88	Vertical Display	VDR	V D S								.....				V S W				$2010
		1/0	r8A	Split Screen Width	SSW	.....								S P 1								$0190
		1/0	r8C			.....								S P 0								$0000
		1/0	r8E			.....								S P 2								$0000
		1/0	r90	Blink Control	BCR	BON1				BOFF1				BON2				BOFF2				$0000
		1/0	r92	Horizontal Window Display	HWR	H W S								H W W								$0000
		1/0	r94	Vertical Window Display	VWR	.....								V W S								$0000
		1/0	r96			.....								V W W								$0000
		1/0	r98	Graphic Cursor	GCR	C X E								C X S								$0000
		1/0	r9A			.....								C Y S								$0000
		1/0	r9C			.....								C Y E								$0000
		-	r9E ↓ rBE	(undefined)	-	.....																—
0	1	1/0	rC0	Upper Screen Base	RAR0	.....				L R A 0				.....				F R A 0				$0000
		1/0	rC2	Raster Addr.0	MWR0	CHR	.....			M W 0												$00A4
		1/0	rC4	Memory Width 0	SAR0	.....				S D A 0				.....				SA0H/SRA0				$0F00
		1/0	rC6	Start Addr.0		S A 0 L																$0000
		1/0	rC8	Raster Addr.1	RAR1	.....				L R A 1				.....				F R A 1				$0000
		1/0	rCA	Memory Width 1	MWR1	CHR	.....			M W 1												$00A4
		1/0	rCC	Start Addr. 1	SAR1	.....				S D A 1				.....				SA1H/SRA1				$0F03
		1/0	rCE			S A 1 L																$00C0
		1/0	rD0	Raster Addr.2	RAR2	.....				L R A 2				.....				F R A 2				$0000
		1/0	rD2	Memory Width 2	MWR2	CHR	.....			M W 2												$01E0
		1/0	rD4	Start Addr.2	SAR2	.....				S D A 2				.....				SA2H/SRA2				$0F09
		1/0	rD6			S A 2 L																$0240
		1/0	rD8	Raster Addr.3	RAR3	.....				L R A 3				.....				F R A 3				$0000
		1/0	rDA	Memory Width 3	MWR3	CHR	.....			M W 3												$00A4
		1/0	rDC	Start Addr.3	SAR3	.....				S D A 3				.....				SA3H/SRA3				$0002
		1/0	rDE			S A 3 L																$0080
		1/0	rE0	Block Cursor 1	BCUR1	B C W 1				B C S R 1				.....				B C E R 1				$0000
		1/0	rE2			B C A 1																$0000
		1/0	rE4	Block Cursor 2	BCUR2	B C W 2				B C S R 2				.....				B C E R 2				$0000
		1/0	rE6			B C A 2																$0000
		1/0	rE8	Cursor Definition	CDR	C M		CON1		COFF1				.....		CON2		COFF2				$0000
		1/0	rEA	Zoom Factor	ZFR	H Z F				V Z F				.....								$0000
		1	rEC	Light Pen Address	LPAR	.....								CHR	.....			L P A H				—
		1	rEE			L P A L																—
		-	rF0 ↓ rFE	(undefined)	-	.....																—

Table 1-6 (b) Programming Model (Initialized Functions) for the Example

Items	Initialized Function
Command Execution	Enabled
Data DMA Transfer	Not Used
Graphic Bit Mode	4 Bit / Pixel
Interruption	Disabled
Operation Mode	Master Mode
Display Operation	Start to Display
Drawing Priority	Display Priority
Window Smooth Scroll	Not Used
Cursor Skew	Not Used
Display Timing Skew	1 Memory Cycle Skewed
RAM Mode	Dynamic
Graphic Address Increment Mode	+4
Access Mode	Dual Access Mode
Raster Scan Mode	Non-Interlace
Display Control	DISP1 = Background, DISP2 = Window
Base Screen	Displayed
Upper Screen	Not Displayed
Lower Screen	Not Displayed
Window Screen	Not Displayed
Attribute Control	Not Used
Cursor	Not Used
Block	Not Used
Zoom up Display	× 1
Light Pen	Not Used

2. COMMAND TRANSFER

There are two ways to transfer commands and parameters to the ACRTC: the one-word transfer which transfers only a word, and the block transfer which continuously transfers plural commands.

2.1 One Word Transfer

Commands and parameters are transferred to the ACRTC by writing one-word data to the Write FIFO.

The program written in the HD68000 assembler is shown below. This program writes the data stored in the HD68000'S D0 register to the Write FIFO. The flowchart is shown in Fig. 2-2.

```
CWRITE   MOVE  ACRTC, D1 :  Read the status register data to D1.
         BTST  #1, D1    :  Check WFR.
         BEQ   CWRITE    :  WFR=0 (Loop if FiFo is full.)
*
         MOVE  #0, ACRTC :  Select the FIFO.
         MOVE  D0,ACRTC+2 :  Write the data to WRITE FIFO.
         RTS
```

Fig. 2-1 One-Word Transfer Program

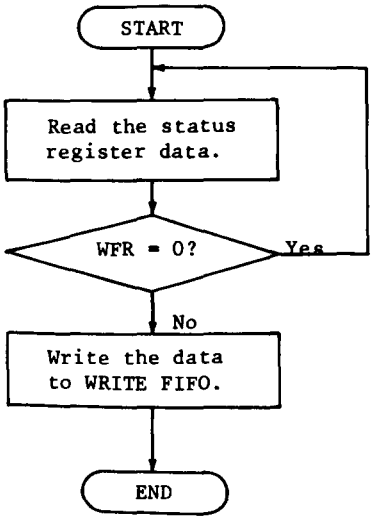


Fig. 2-2 One-Word Transfer Flowchart

2.2 Block Transfer

The program which continuously transfers plural commands and parameters to the ACRTC is shown below. The program transfers a specified number of commands and parameters set in the data table. The amount of data to be transferred is set to the front of the table, and its value is <Number of Data> - 1. The subroutine of Fig. 2-1 is used to transfer the data to the ACRTC.

The start address of the data table must be set in the HD68000'S A1 register before starting the block transfer.

```
CWRITE   MOVE   (A1)+, D2:  Read the number of times the transfer is to be
                            repeated.
CTWR     MOVE   (A1)+, D0:  Read the data to D0.
         BSR    CWRITE   :  Transfer the data to the ACRTC.
         DBRA   D2, CTWR :  Repeat transfer the specified number of times.
         RTS
```

Fig. 2-3 Block Transfer Program

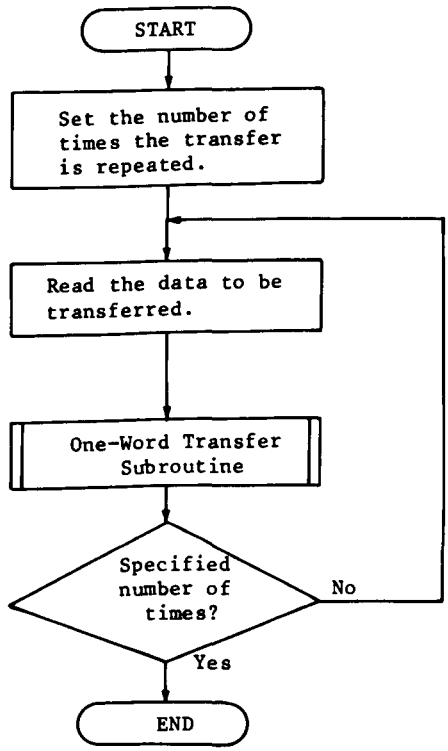


Fig. 2-4 Block Transfer Flowchart

An example of block transfer execution is shown below. This program clears the base screen and defines the drawing screen. The drawing parameter register is set simultaneously. The data in Fig. 2-5 is used by defining the command table of the ACRTC as shown in Fig. 2-6. In Fig. 2-6, bits such as "AREA", "COL" and "OPM" within the command are set to "0".

```
        LEA   DATA, A1 :  Set the start address of the data table to A1.
        BSR   CWRITE   :  The block transfer
        RTS
*
DATA    DC    52                              :  <Number of Data> - 1
        DC    WPR+$C, $4030                   ]  Set Read/Write Pointer
        DC    WPR+$D, $0C00                   ]
        DC    CLR, $0000, 164, -400           :  Clear the screen
        DC    ORG, $4030, $0C00               :  Specify the drawing screen
        DC    WPTN, 16                        ]  Write $FFFF to the pattern
        DC    -1, -1, -1, -1, -1, -1, -1, -1  ]  RAM.
        DC    -1, -1, -1, -1, -1, -1, -1, -1  ]
        DC    WPR, $0000                      :  CL0
        DC    WPR+$1, $FFFF                   :  CL1
        DC    WPR+$2, $0000                   :  CCR
        DC    WPR+$3, $FFFF                   :  EDG
        DC    WPR+$4, $0000                   :  MSK
        DC    WPR+$5, $0000                   :  PP, PZC
        DC    WPR+$6, $0000                   :  PS
        DC    WPR+$7, $F1F1                   :  PE, PZ
        DC    WPR+$8, 0                       ]
        DC    WPR+$9, -399                    ]  AREA
        DC    WPR+$A, 639                     ]  (0, -399) - (639, 0)
        DC    WPR+$B, 0                       ]
```

Fig. 2-5 Example of Block Transfer Program

```
ORG         EQU     %0000010000000000
WPR         EQU     %0000100000000000
RPR         EQU     %0000110000000000
WPTN        EQU     %0001100000000000
RPTN        EQU     %0001110000000000
DRD         EQU     %0010010000000000
DWT         EQU     %0010100000000000
DMOD        EQU     %0010110000000000
*
RD          EQU     %0100010000000000
WT          EQU     %0100100000000000
MOD         EQU     %0100110000000000
OLR         EQU     %0101100000000000
SCLR        EQU     %0101110000000000
CPY         EQU     %0110000000000000
SCPY        EQU     %0111000000000000
*
AMOVE       EQU     %1000000000000000
RMOVE       EQU     %1000010000000000
ALINE       EQU     %1000100000000000
RLINE       EQU     %1000110000000000
ARCT        EQU     %1001000000000000
RRCT        EQU     %1001010000000000
APLL        EQU     %1001100000000000
RPLL        EQU     %1001110000000000
APLG        EQU     %1010000000000000
RPLG        EQU     %1010010000000000
CRCL        EQU     %1010100000000000
ELPS        EQU     %1010110000000000
AARC        EQU     %1011000000000000
RARC        EQU     %1011010000000000
AEARC       EQU     %1011100000000000
REARC       EQU     %1011110000000000
AFRCT       EQU     %1100000000000000
RFRCT       EQU     %1100010000000000
PAINT       EQU     %1100100000000000
DOT         EQU     %1100110000000000
PTN         EQU     %1101000000000000
ABCPY       EQU     %1110000000000000
RGCPY       EQU     %1111000000000000
```

Fig. 2-6 ACRTC Command Table

3. DIRECTIONS FOR USING COMMANDS

3.1 Coordinate Setup

ACRTC Graphic drawing is performed by specifying the dot position with a two-dimensional X-Y coordinate. The origin can be set at any location in the frame buffer memory. Also, a different origin position can be set for each split screen.

"ORG" command is used to define the logical X-Y coordinate origin on the frame buffer. "ORG" command sets the screen number by "DN", and the physical address of the frame buffer memory origin point by "DPAH" and "DPAL". "DPD" also sets the bit position within the word specified by "DPAH" and "DPAL". When using 4 bits / pixel mode, the upper 2 bits of ''DPD'' are valid, thereby setting the pixel address in units of dots is possible.

By issuing the ORG command, two-dimensional coordinates are configured referring to the memory width of the split screen specified by "DN". The parameters "DPH" and "DPL" are written into the drawing pointer in the drawing parameter register, and at the same time, the current pointer in the drawing parameter register is reset to "0".

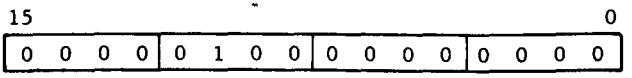
Fig. 3-1 shows an example of the ORG command on the base screen.

<Mnemonic>

ORG DPH, DPL

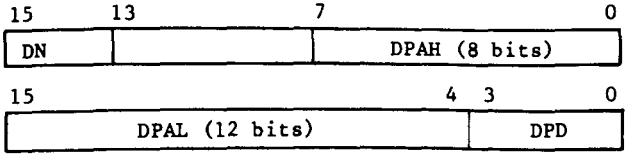
<Command Format>

Operation Code



DN	Screen No.
00	Upper Screen
01	Base Screen
10	Lower Screen
11	Window

Parameters

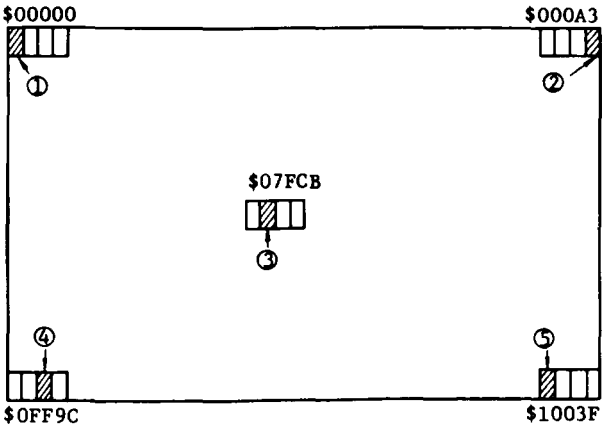


Note)

DPH = DN + DPAH

DPL = DPAL + DPD

Fig. 3-1 ORG Command



(4 bits / pixel
Memory Width:
164 words
Unit: Word)

① :

```
DC      2
DC      ORG, $4000, $0000
```

② :

```
DC      2
DC      ORG, $4000, $0A3C
```

③ :

```
DC      2
DC      ORG, $4007, $FCB4
```

④ :

```
DC      2
DC      ORG, $400F, $F9C8
```

⑤ :

```
DC      2
DC      ORG, $4010, $03F0
```

Fig. 3-2 ORG (Origin Point) Setup Example

3.2 Screen Clear

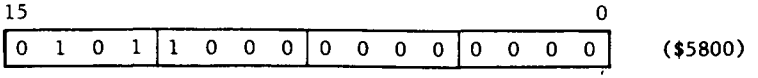
The "CLR" command is used to clear the drawing screen. The "CLR" command clears portions of the frame buffer area by writing a specified color code to the area to be cleared. The area is specified by the command parameters and read/write pointer in the drawing parameter register where the address is specified by using physical address. As processing is performed in unit of words, parameters are also in unit of word. The negative value is set by 2's complement. The color data is specified in units of words, therefore, the color is specified by units of 4 pixels in the 4 bits / pixel mode. Normally, the same color is specified for all 4 dots; however the screen can be cleared with mixed colors other than the solid 16 colors by deliberately setting different color codes for each pixel in the word. The read/write pointer moves to the termination point after the command execution as shown in Fig. 3-3.

<Mnemonic>

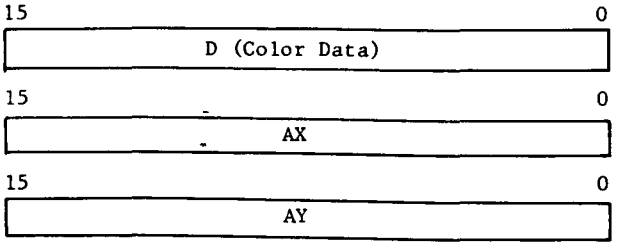
CLR D, AX, AY

<Command Format>

Operation Code



Parameters



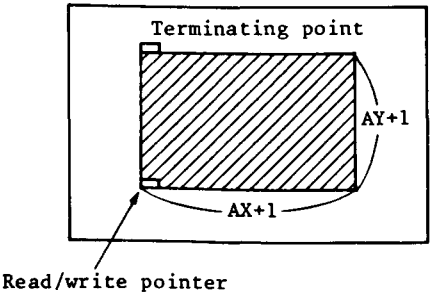
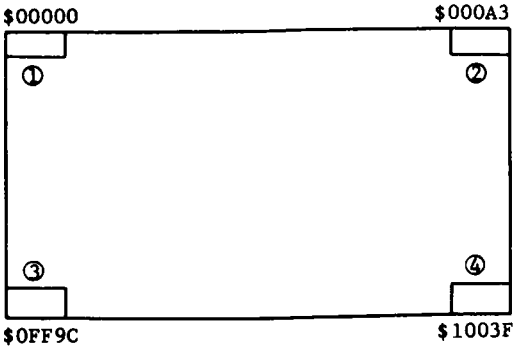


Fig. 3-3 CLR Command



(4 bits / pixel
Memory Width:
164 words
Unit: Word
640 × 400 dots)

Fig. 3-4 Drawing Screen Example

There are four ways to clear the drawing screen shown in Fig. 3-4 with color "0" using the CLR command as indicated below (Note whether the value of AX and AY is positive or negative).

①

```
DC      7
DC      WPR + $C, $4000
DC      WPR + $D, $0000
DC      CLR, $0000, 163, -399
```

②

```
DC      7
DC      WPR + $C, $4000
DC      WPR + $D, $0A30
DC      CLR, $0000, -163, -399
```

③

```
DC      7
DC      WPR + $C, $400F
DC      WPR + $D, $F9C0
DC      CLR, $0000, 163, 399
```

④

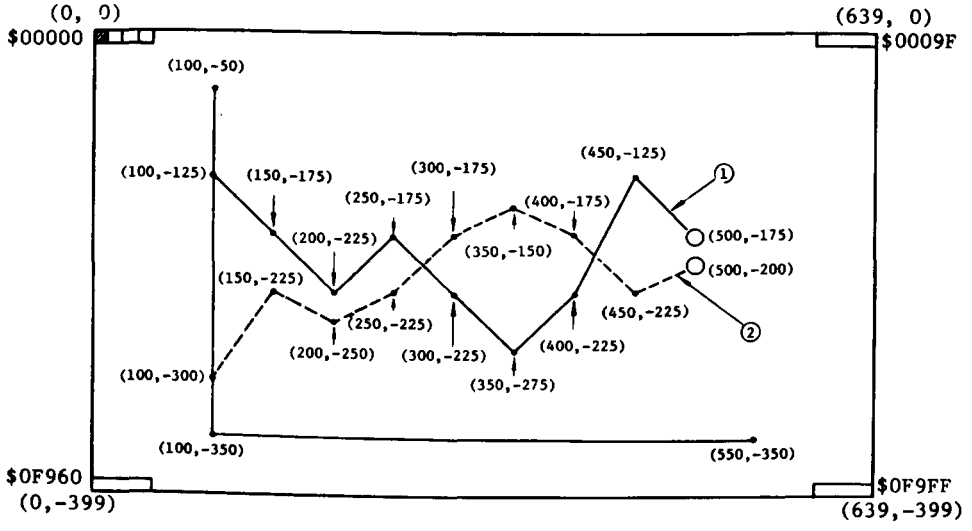
```
DC      7
DC      WPR + $C, $4010
DC      WPR + $D, $03F0
DC      CLR, $0000, -163, 399
```

3.3 Chart Drawing

Charts can be easily drawn by using the graphic drawing commands of the ACRTC.

3.3.1 Poly-line Chart

An example of a program which draws the poly-line chart in Fig. 3-5 is shown in Fig. 3-6.



Base Screen
4 bits / pixel mode
Memory Width: 160 words
Display Start Address: $00000

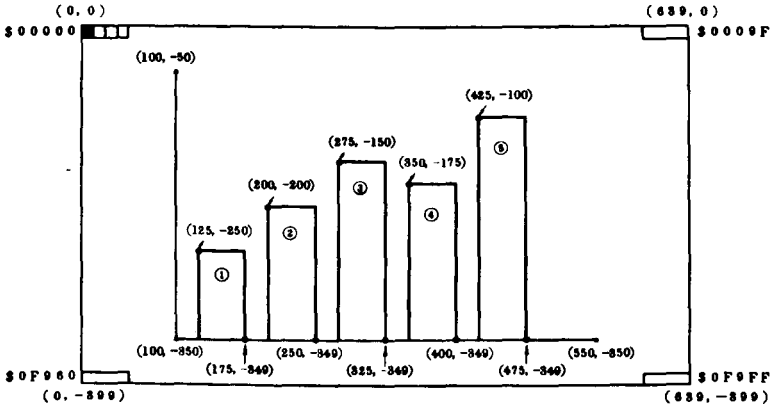
Fig. 3-5 Poly-Line Chart Example

```
GRAPH 1  DC    77
         DC    ORG, $4000, $0000     :  Set the origin point.
         DC    WPR + $C, $4000       ]  Set read/write pointer.
         DC    WPR + $D, $0000       ]
         DC    CLR, $0000, 159, -399:  Clear the screen.
         DC    WPTN, 2, $FFFF, $F0F0:  Set the solid and dotted-line data.
         DC    WPR, $0000            :  Set CL0.
         DC    WPR + 1, $FFFF        :  Set CL1.
         DC    WPR + 5, $0000        :  Select the solid line with PPY = 0.
         DC    WPR + 6, $0000        :  Set PS.
         DC    WPR + 7, $F0F0        :  Set PE and PZ.

         DC    AMOVE, 100, -50  :  Move the current pointer to (100, -50)
         DC    APPL, 2, 100, -350, 550 -350:  Draws X-Y axis.

         DC    AMOVE, 100, -125 :  Moves the current pointer to (100, -125)
         DC    APLL, 8, 150, -175, 200, -225   ]
         DC    250, -175, 300, -225, 350, -275 ]  Draw poly-line 1.
         DC    400, -225, 450, -125, 500, -175 ]

         DC    WPR + 5, $1000   :  Select the dotted line with PPY = 1.
         DC    AMOVE, 100, -300 :  Move the current pointer to (100, -300)
         DC    APLL, 8, 150, -225, 200, -250   ]
         DC    250, -225, 300, -175, 350, -150 ]  Draw poly-line 2.
         DC    400, -175, 450, -225, 500, -200 ]
```

Fig. 3-6 Poly-Line Chart Example Program List

3.3.2 Bar Chart

An example of a program which draws the bar chart should below is shown in Fig. 3-8.



Base Screen
4 bits / pixel mode
Memory Width: 160 words
Display Start Address: $00000

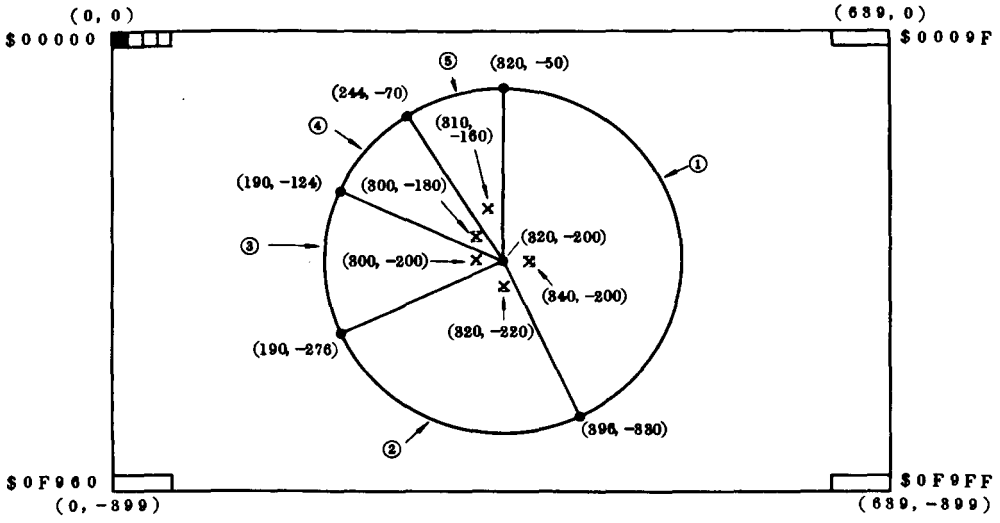
Fig. 3-7 Bar Chart Example

```
GRAPH 2   DC   73
          DC   ORG, $4000, $0000              : Set the origin point.
          DC   WPR + $C, $4000                : Set the read/write pointer.
          DC   WPR + $D, $0000
          DC   CLR, $0000, 159, -399          : Clear the screen.

          DC   WPR, $FFFF                     : Set CL0.
          DC   WPR + 1, $FFFF                 : Set CL1.
          DC   AMOVE, 100, -50  : Move the current pointer to (100, -50)
          DC   APLL, 2, 100, -350, 550, -350: Draw X-Y axis.

          DC   WPR, $9999                     : Set CL0.
          DC   WPR + 1, $9999                 : Set CL1.
          DC   AMOVE, 125, -250 : Move the current pointer to (125, -250)
          DC   AFRCT, 175, -349               : Draw bar 1.

          DC   WPR, $AAAA                     : Set CL0.
          DC   WPR +1, $AAAA                  : Set CL1.
          DC   AMOVE, 200, -200 : Move the current pointer to (200, -200)
          DC   AFRCT, 250, -349               : Draw bar 2.

          DC   WPR, $BBBB                     : Set CL0.
          DC   WPR + 1, $BBBB                 : Set CL1.
          DC   AMOVE, 275, -150 : Move the current pointer to (275, -150)
          DC   AFRCT, 325, -349               : Draw bar 3.

          DC   WPR, $CCCC                     : Set CL0.
          DC   WPR + 1, $CCCC                 : Set CL1.
          DC   AMOVE, 350, -175 : Move the current pointer to (350, -175)
          DC   AFRCT, 400, -349               : Draw bar 4.

          DC   WPR, $DDDD                     : Set CL0.
          DC   WPR + 1, $DDDD                 : Set CL1.
          DC   AMOVE, 425, -100 : Move the current pointer to (425, -100)
          DC   AFRCT, 475, -349               : Draw bar 5.
```

Fig. 3-8 Bar Chart Example Program List

3.3.3 Pie Chart

An example of the program which draws the circle chart in Fig. 3-9 is shown in Fig. 3-10.



Base Screen
4 bits / pixel mode
Memory Width: 160 words
Display Start Address: $00000

Fig. 3-9 Circle Chart Example

```
GRAPH 3  DC   85
         DC   ORG, $4000, $0000     : Set the origin point.
         DC   WPR + $C, $4000       ]
         DC   WPR + $D, $0000       ] Set the read/write pointer.
         DC   CLR, $0000, 159, -399: Clear the screen.

         DC   WPR, $FFFF            : Set CL0.
         DC   WPR + 1, $FFFF        : Set CL1.
         DC   WPR + 3, $FFFF        : Set EDG.
         DC   AMOVE, 320, -200      : Move the current pointer to (320, -200)
         DC   CRCL, 150             : Draw a circle with a radius of 150.

         DC   AMOVE, 320, -200      : Move the current pointer to (320, -50)
         DC   ALINE, 320, -50       : Draw a straight line.
         DC   AMOVE, 396, -330      : Move the current pointer to (396, -330)
         DC   APLL, 2, 320, -200, 190, -276: Draw a straight line.
         DC   AMOVE, 190, -124      : Move the current pointer to ((190, -124)
         DC   APLL, 2, 320, -200, 244, -70: Draw a straight line.

         DC   WPR, $9999            : Set CL0.
         DC   WPR + 1, $9999        : Set CL1.
         DC   AMOVE, 340, -200      : Move the current pointer to (340, -200)
         DC   PAINT                 : Paint in the area 1.

         DC   WPR, $AAAA            : Set CL0.
         DC   WPR + 1, $AAAA        : Set CL1.
         DC   AMOVE, 320, -220      : Move the current pointer to (320, -220)
         DC   PAINT                 : Paint in the area 2.

         DC   WPR, $BBBB            : Set CL0.
         DC   WPR + 1, $BBBB        : Set CL1.
         DC   AMOVE, 300, -200      : Move the current pointer to (300, -200)
         DC   PAINT                 : Paint in the area 3.

         DC   WPR, $CCCC            : Set CL0.
         DC   WPR + 1, $CCCC        : Set CL1.
         DC   AMOVE, 300, -180      : Move the current pointer to (300, -180)
         DC   PAINT                 : Paint in the area 4.

         DC   WPR, $DDDD            : Set CL0.
         DC   WPR + 1, $DDDD        : Set CL1.
         DC   AMOVE, 310, -160      : Move the current pointer to (310, -160)
         DC   PAINT                 : Paint in the area 5.
```

Fig. 3-10 Circle Chart Example Program List

3.4 Ellipse Drawing

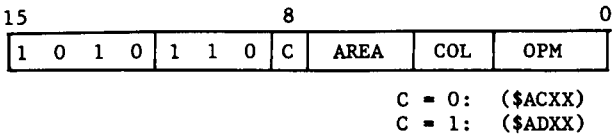
The "ELPS" command is used to draw ellipses. The parameter setup procedure necessary for ellipse drawing is shown below.

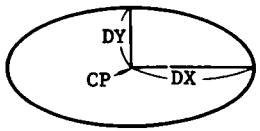
<Mnemonic>

ELPS a,b, DX

<Command Format>

Operation Code





$a : b = DX^2 : DY^2$

Parameters

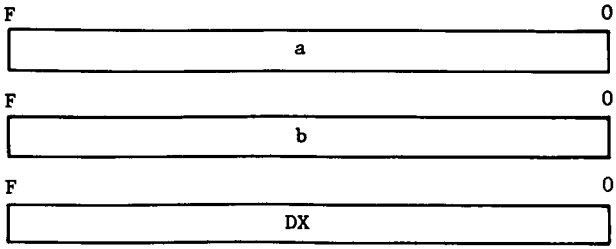


Fig. 3-11 ELPS Command

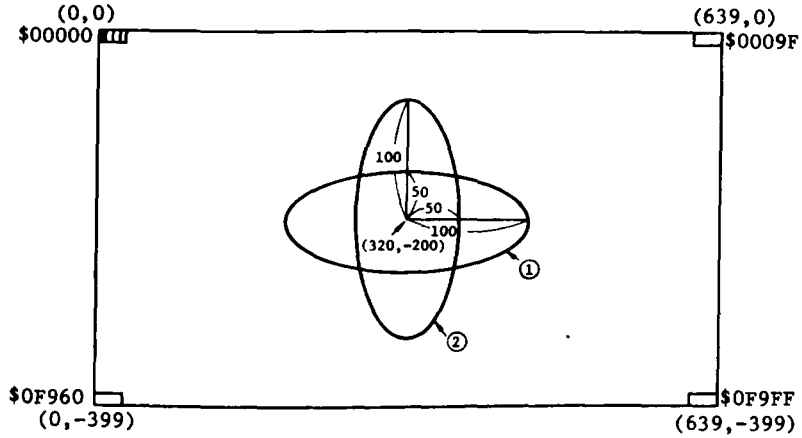
"a" and "b", the ratio of the square of the radius in X axis (DX) radius, and the Y axis (DY) radius, as well as "DX" value are set to the necessary parameters. Supposing DX = 10 and DY = 6, the value of "a" and "b" is calculated as follows:

$$DX = 10 \qquad DY = 6$$
$$(a : b) = (10^2 : 6^2) = (100 : 36) = (25 : 9)$$
$$a = 25, \quad b = 9$$

Bit 8 (c) in the operation code decides whether the ellipse drawing is to be performed clockwise (c=1) or counter-clockwise (c=0).

After the ellipse is drawn the current pointer moves to the center of the ellipse, thereby the current pointer does not move after the ''ELPS'' command.

An example of the program which draws the ellipse in Fig. 3-12 is shown in Fig. 3-13.



Base Screen
4 bits / pixel mode
Memory Width: 160 words
Display Start Address: $00000

Fig. 3-12 Ellipse Drawing

```
ELLIPSE   DC   25
          DC   ORG, $4000, $0000      :  Sets the origin point.
          DC   WPR + $C, $4000        ]  Sets the read/write pointer.
          DC   WPR + $D, $0000        ]
          DC   CLR, $0000, 159, -399: Clears the screen.

          DC   WPR, $FFFF             :  Sets CL0.
          DC   WPR + 1, $FFFF         :  Sets CL1.

          DC   AMOVE, 320, -200       :  Move the current pointer to (320, -200)
        * DC   ELPS, 4, 1, 100        :  Draw ellipse 1.
          DC   ELPS, 1, 4, 50         :  Draw ellipse 2.
```

Fig. 3-13 Ellipse Drawing Example Program List

* $a : b = (100)^2 : (50)^2 = 10{,}000 : 2{,}500 = 4 : 1$

3.5 Character Drawing

To draw the character patterns on the graphic screen, it is very convenient to use the pattern RAM. The pattern RAM can store a figure pattern of up to 16 × 16 dots. The character is drawn by issuing the pattern (PTN) command after storing the character data in the pattern RAM. The Pattern command can magnify the figure stored in the pattern RAM from 1 up to 16 times in the X direction, (in the) Y direction, or in both. Moreover, this command can rotate the pattern by 45 degrees. Such functions of Pattern command allows variety of character drawing. As the character pattern data are used by defining them in the main memory of the MPU, any form of characters and figures can be drawn.

The character drawing program example and its flowchart is shown in Fig. 3-15, and an example of the program written in HD68000 assembly language is shown in Fig. 3-16. This program draws the characters to the area specified by the current pointer using the character code stored in the "D0" register of the HD68000.

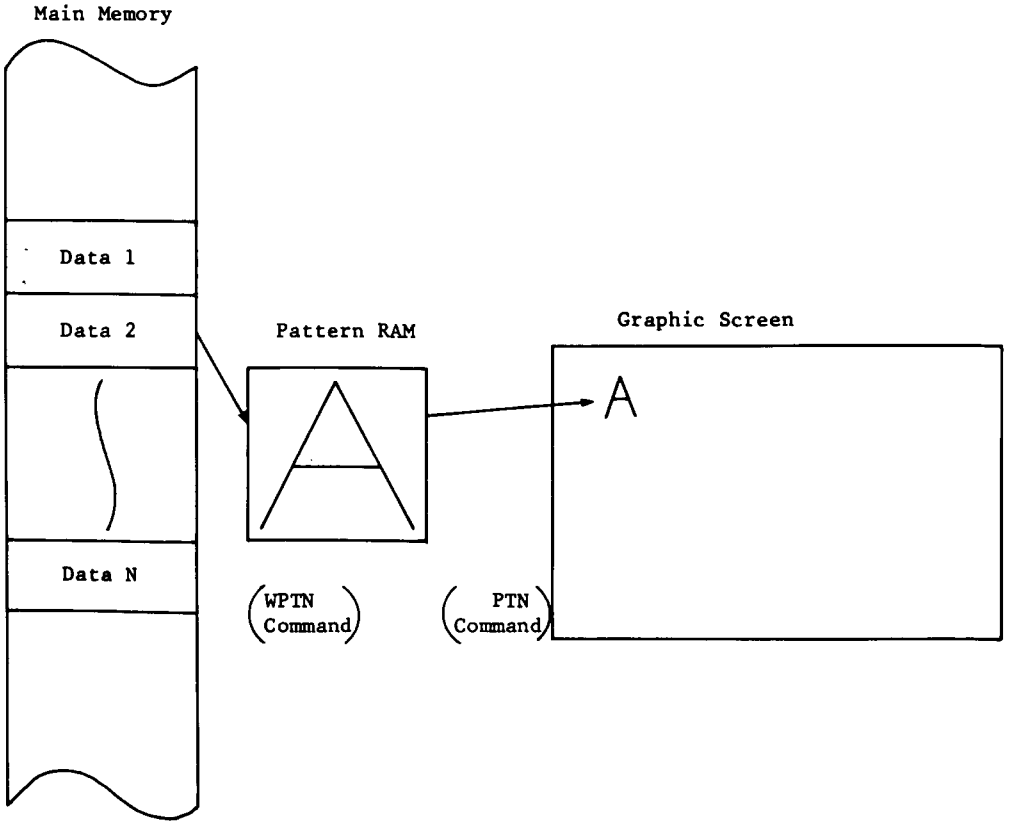


Fig. 3-14 Character Drawing Example

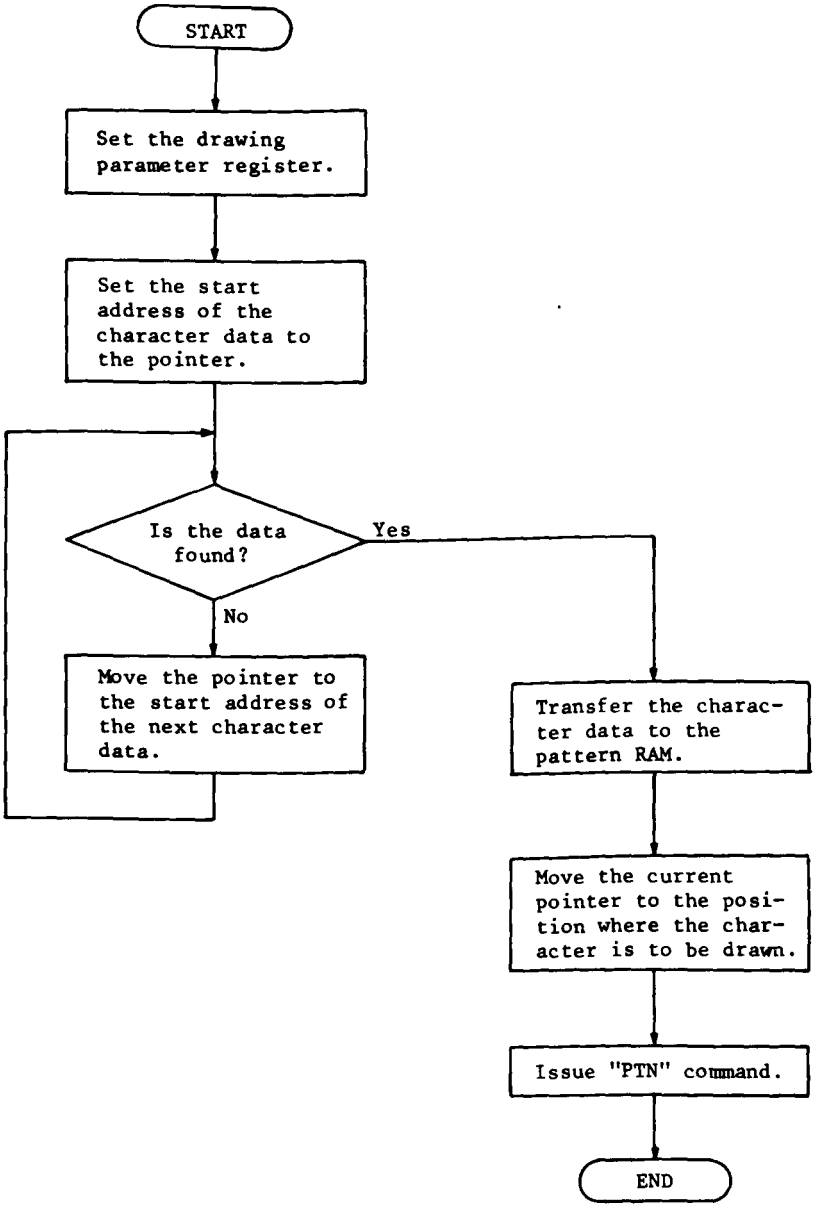


Fig. 3-15 Character Drawing Example Flowchart

```
*
*****************************
*                           *
*       P R I N T           *
*                           *
*****************************
*
PRINT     LEA       PRAM, A1
          BSR       CTWRTE        Initialize the drawing parameter register.
*
          LEA       KANJI, A2     Set the beginning address of the character data
KSERCH    CMP       (A2)+, D0
          BEQ       PAT           Character data No. ?
          ADDA.L    #32, A2
          LEA       KEND, A0
          CMPA.L    A0, A2                  KANJI END ?
          BEQ       PAT
          BRA       KSERCH
*
PAT       MOVE      #WPTN, D0               Issue the WPTN command
          BSR       CWRITE
          MOVE      #16, D0
          BSR       CWRITE
*
          MOVE      #15, D2                 Transfer the character data
PLOOP     MOVE      (A2)+, D0
          BSR       CWRITE
          DBRA      D2, PLOOP
*
          BSR       CTWRTE                  Issue the PTN command
          RTS
*
PRAM      DC        9
          DC        WPR, $0000              CL0
          DC        WPR+1, $FFFF            CL1
          DC        WPR+5, $0000            (PPY, PPX), (PZCY, PZCX)
          DC        WPR+6, $0000            (PSY, PSX)
          DC        WPR+7, $F0F0            (PEY, PEX), (PZY, PZX)
*
          DC        1
          DC        PTN, $0F0
```

Fig. 3-16 Character Drawing Program Example

```
*
****************************
*                          *
*    K A N J I   D A T A   *
*                          *
****************************
*
KANJI     DC        $2121                    Code NO
          DC        $0000, $0000, $0000, $0000, $0000, $0000, $0000, $0000
          DC        $0000, $0000, $0000, $0000, $0000, $0000, $0000, $0000
*
          DC        $2380                    0
          DC        $0000, $03E0, $0410, $0808, $0808, $0808, $0808, $0808
          DC        $0808, $0808, $0808, $0808, $0808, $0410, $03E0, $0000
*
          DC        $2381                    1
          DC        $0000, $03E0, $0080, $0080, $0080, $0080, $0080, $0080
          DC        $0080, $0080, $0080, $0080, $00A0, $00C0, $0080, $0000
*
          DC        $2382                    2
          DC        $0000, $0FF8, $0808, $0008, $0010, $0020, $0040, $0180
          DC        $0200, $0400, $0800, $0808, $0808, $0410, $03E0, $0000
*
          DC        $2383                    3
          DC        $0000, $03E0, $0410, $0808, $0800, $0800, $0800, $0400
          DC        $0380, $0400, $0800, $0800, $0808, $0410, $03E0, $0000
*
          DC        $2384                    4
          DC        $0000, $07C0, $0100, $0100, $0FFC, $0104, $0108, $0108
          DC        $0110, $0120, $0120, $0140, $0140, $0180, $0100, $0000
*
          DC        $2385                    5
          DC        $0000, $03E0, $0410, $0808, $0800, $0800, $0800, $0808
          DC        $0418, $03E8, $0008, $0008, $0008, $0008, $07F8, $0000
*
          DC        $2386                    6
          DC        $0000, $03E0, $0410, $0808, $0808, $0808, $0808, $0808
          DC        $0418, $03E8, $0008, $0008, $0808, $0410, $03E0, $0000
*
          DC        $2387                    7
          DC        $0000, $0100, $0100, $0100, $0100, $0100, $0100, $0100
          DC        $0200, $0200, $0400, $0408, $0808, $0808, $0FF8, $0000
*
          DC        $2388                    8
          DC        $0000, $03E0, $0410, $0808, $0808, $0808, $0808, $0410
          DC        $03E0, $0410, $0808, $0808, $0808, $0410, $03E0, $0000
*
          DC        $2389                    9
          DC        $0000, $03E0, $0410, $0808, $0800, $0800, $0BE0, $0C10
          DC        $0808, $0808, $0808, $0808, $0808, $0410, $03E0, $0000
*

*
KEND      DC        $0000
          DC        $AAAA, $5555, $AAAA, $5555, $AAAA, $5555, $AAAA, $5555
          DC        $AAAA, $5555, $AAAA, $5555, $AAAA, $5555, $AAAA, $5555
```

Fig. 3-17 Character Data Table Example

3.6 Painting inside a Figure

The "PAINT" command is used to paint inside a enclosed figure. "PAINT" command paints inside the enclosed area, surrounded by the edge color, with the pattern stored in the pattern RAM. There are two ways to paint: the solid color painting (non-tiling) and the pattern painting (tiling). Selection is not done by the PAINT command, but by the data stored in the pattern RAM and the drawing parameter register.

A figure is painted with a single color regardless of the pattern RAM data if the color registers (CL0, CL1) in the drawing parameter register are set with the same color, or if all the pattern RAM data are set to all "1" or all "0"

If the color registers (CL0, CL1) are set with different colors, the figure stored in the pattern RAM is drawn (or painted) inside the figure. In this case, the pattern RAM control register in the drawing parameter register must be set to use the pattern RAM.

If the shape of the figure is complicated, there may be some areas left unpainted. Information about unpainted areas (coordinates, the pattern pointer) is passed to the Read FIFO. Therefore, unpainted areas can be painted by having the MPU reissue the PAINT command using the information stored in the Read FIFO.
In this case, a total of 3 words, the current pointer "CPX and CPY", and the pattern pointer register (Pr05) are passed to the Read FIFO as the stack point information in order to paint the unpainted area. The PAINT command needs to be re-issued using this stack information after moving the current pointer (CPX, CPY) with AMOVE and setting the pattern point register (Pr05). The painting operation sequence is terminated when the PAINT command has been issued using all the stack points.
The Read FIFO, therefore, must be emptied before issuing the command. The Read FIFO may become full in case of a complicated figure, thereby the stack information set in the Read FIFO must be stored into the system memory in such a case.

E=0: The data in the "EDG" register is used as the edge color.
E=1: A color other than the data in the "EDG" register is used as the edge color.

<Mnemonic>

PAINT

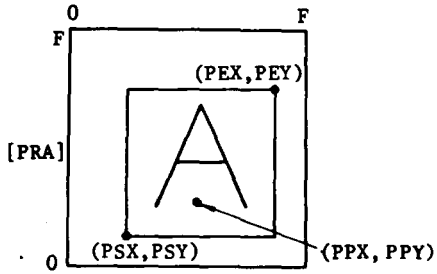
<Command Format>

Operation Code

1	1	0	0	1	0	0	E	AREA	0	0	0	0	0

E=0: ($C8X0)
E=1: ($C9XX)

Pattern RAM



Note) COL and OPM are not programmable for the PAINT command. They are always ''00'', ''000''.

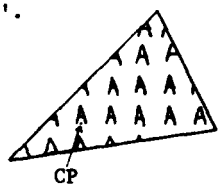


Fig. 3-18 PAINT Command

An example of the program which paints in the figure in Fig. 3-19 with a single color and patterns are shown in Fig. 3-20.

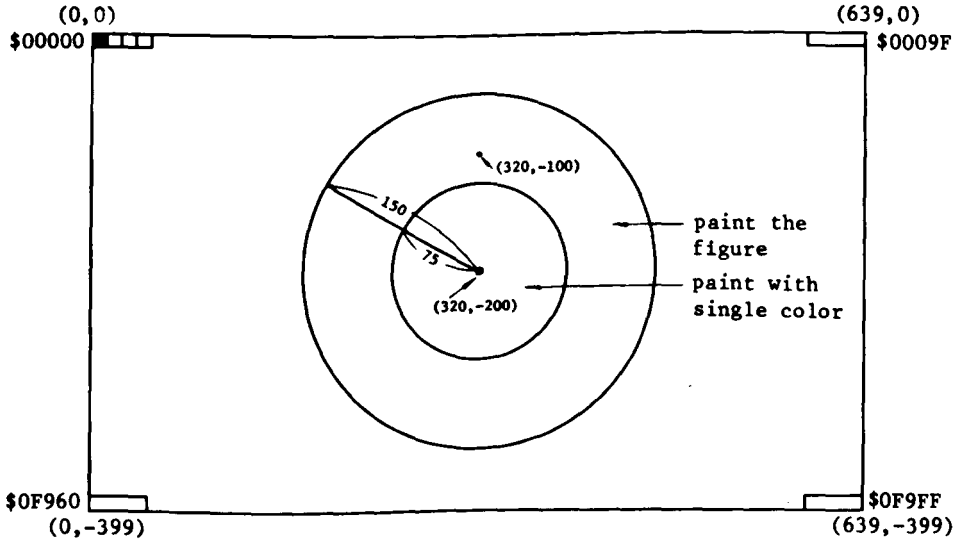


Fig. 3-19 Painting Example

Base Screen
4 bits / pixel mode
Memory Width: 160 words
Display Start Address: $0000

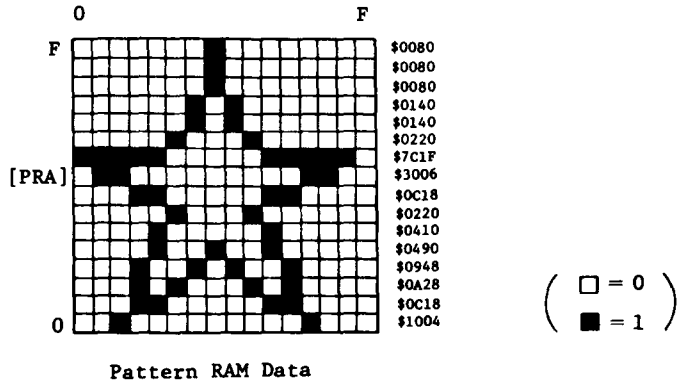


Fig. 3-19 Painting Example (continued)

```
   DC    60
   DC    ORG, $4000, $0000        Set the origin point.
   DC    WPR + $C, $4000   ]
   DC    WPR + $D, $0000   ]      Set the read/write pointer.
   DC    CLR, $0000, 159, -399:   Clear the screen.
*
   DC    WPTN, 16                 :  Set the pattern RAM.
   DC    $1004, $0C18, $0A28, $0948, $0490, $0410, $0220, $0C18
   DC    $3006, $7C1F, $0220, $0140, $0140, $0080, $0080, $0080
*
   DC    WPR, $FFFF               :  Set CL0.
   DC    WPR + 1, $FFFF           :  Set CL1.
   DC    WPR + 3, $FFFF           :  Set EDG.
   DC    AMOVE, 320, -200         :  Move the current pointer to (320, -200).
   DC    CRCL, 150                :  Draw a circle with a radius of 150.
   DC    CRCL, 75                 :  Draw a circle with a radius of 75.
*
   DC    WPR, $9999               :  Set CL0.
   DC    WPR + 1, $0000           :  Set CL1.
   DC    PAINT                    :  Plain Color Painting.
*
   DC    WPR, $AAAA               :  Set CL0.
   DC    WPR + 1, $BBBB           :  Set CL1.
   DC    WPR + 5, $0000           :  Set PP.
   DC    WPR + 6, $0000           :  Set PS.
   DC    WPR + 7, $F0F0           :  Set PE and PZ.
   DC    AMOVE, 320, -100         :  Move the current pointer to (320, -100).
   DC    PAINT                    :  Paint in the figure.
```

Fig. 3-20 Painting Program Example

3.7 Software multi-window

The ACRTC provides a window display function. But only one window is displayed, so multiple ACRTC operation or a multiple window software is required to impliment a multi-window display.

A multi-window display through software is easily provided by copying the graphic data to the display screen area. The ACRTC copies the specified area to another area with the copy commands, like CPY, SCPY, AGCPY and RGCPY. The multi-window display is realized by issuing these copy commands.

Scrolling within the window is performed by moving the source pointer, and the Window position on the CRT screen is shifted by moving the destination pointer.

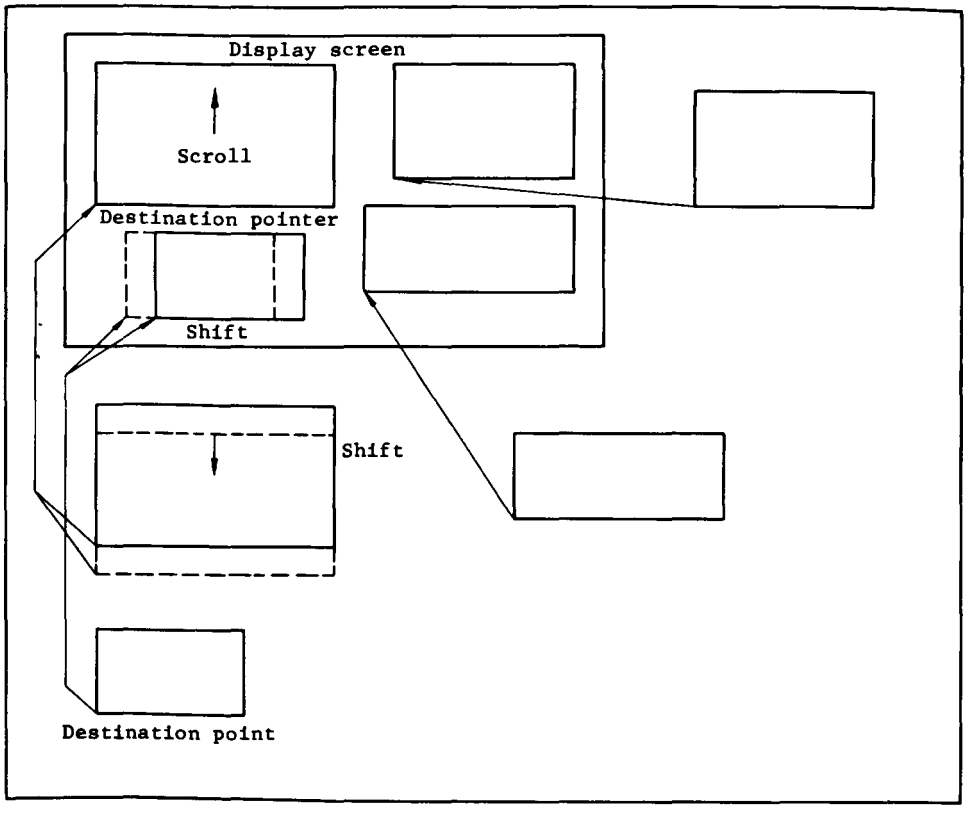


Fig. 3-21 Software Multi-window Support Example

4.1 Split Screen

The ACRTC controls the four screens in the display screen (three horizontally split screens and a window screen).
Various screen configurations are provided by specifying the screen split positions and the window size.
The screen split configuration is shown in Fig. 4-1, and the split screen control registers in Table 4-1.

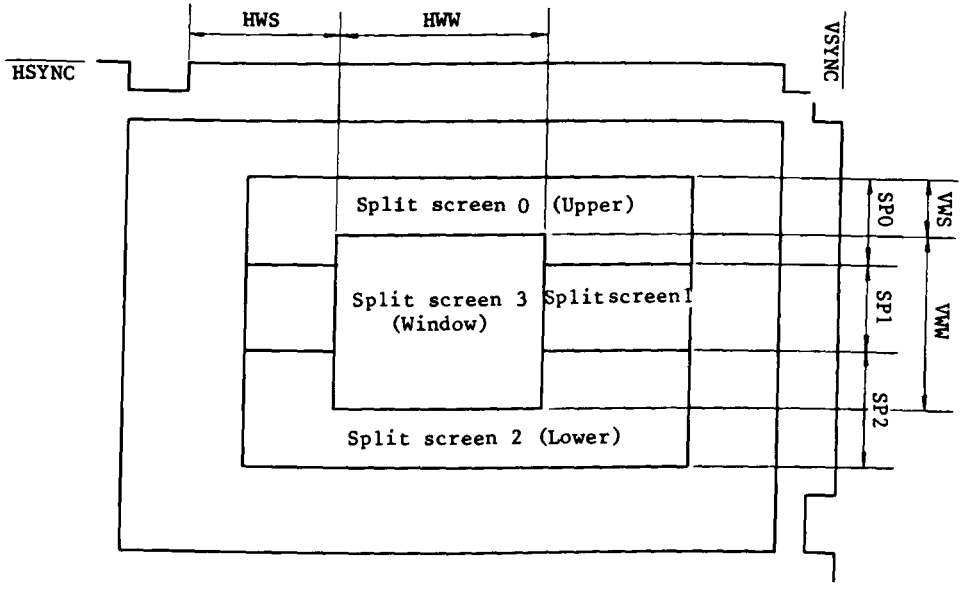


Fig. 4-1 Split Screen Configuration

Reg. No.	Register name	Abbr.	Data (H) F	E	D	C	B	A	9	8	Data (L) 7	6	5	4	3	2	1	0
r06	Display control	DCR	DSP	SE1	SE0	SE2	SE3	ATR										
r8A	Split screen width	SSW	-------					SP1 (Base)										
r8C			-------					SP0 (Upper)										
r8E			-------					SP2 (Lower)										
r92	Horizontal window display	HWR	HWS								HWW							
r94	Vertical window display	VWR	-------					VWS										
r96			-------					VWW										

Table 4-1 Split Screen Control Register

To split the display screen horizontally into three, it is necessary to specify the split screen display width (SP0, SP1 and SP2) as raster counts and set the split screen enable bits (SE0, SE1 and SE2) in the display control register (DCR) to "1" for SE1 and to "11" for (SE0, SE2). The specified value must satisfy the following equation.

SP0 + SP1 + SP2 = Vertical display width

When split screen 0 is being displayed, the base screen is shifted down below it. So display start address (1) in the base screen need to be modified to display start address (2) to avoid lowering the base screen as shown in Fig. 4-2.

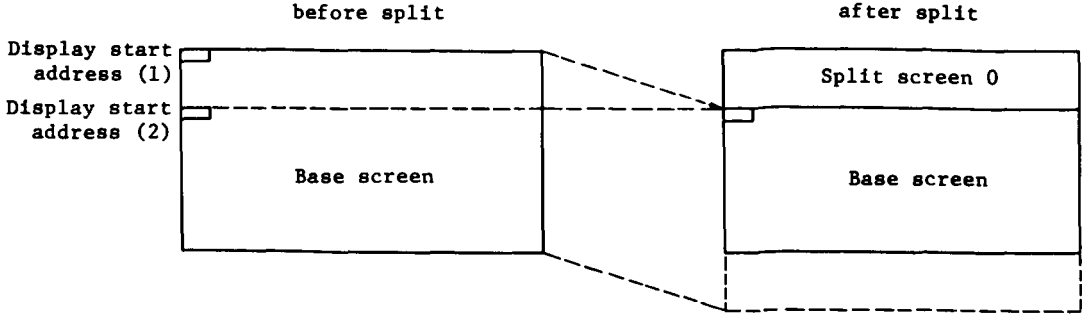


Fig. 4-2 Relation between Base Screen and Split Screen

When displaying split screen 2, the position of the base screen on the CRT is not affected.

To display the window screen, its position and size need to be specified in the horizontal window display register (HWS and HWW) in units of memorys counts and the in vertical window display register (VWS and VWW) in units of rasters, and the window enable bit (SE3) in the display control register (DCR) must be set to "11". The window screen is moved horizontally under "HWS" control, and vertically under "VWS" control.

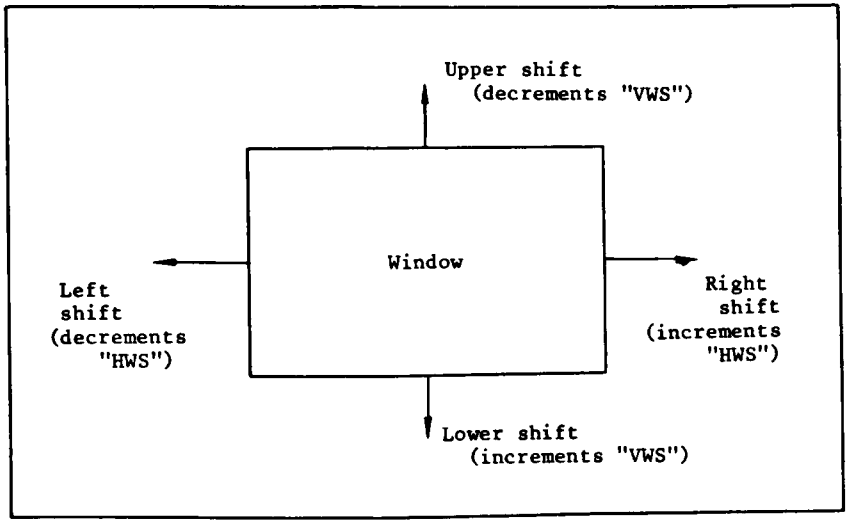


Fig. 4-3 Window Shift

4.2 Scroll

The ACRTC provides a horizontal and vertical smooth scroll function on the graphic screen. The screen is vertically scrolled by increasing or decreasing the start address ("SAH" and "SAL" in the display start address registers (SAR0, SAR1, SAR2 and SAR3)), and horizontally scrolled by controlling "SDA" in the display start address register together with "SAH" and "SAL".

The display start address is set in "SAH" and "SAL" as the physical address. The horizontal shift is set in "SDA" in unit of dots.

The display start address is rewritable any time. Smooth scroll without snow (flickering) is performed by rewriting it while the scanning is in non-displaying period. Each split screen is separately scrolled because each screen has a display start address register.

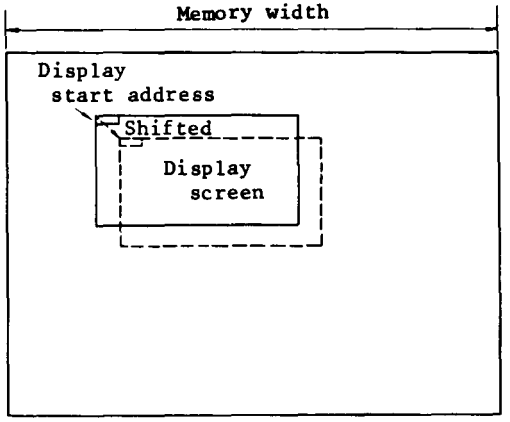


Fig. 4-4 Scroll and the Display Screen

An example of the smooth scroll program written in HD68000 is shown below.

```
ACRTC   EQU     $XXXXXX         : Define the ACRTC.
*
UP      MOVE    #$CC, ACRTC     : Select the rCC register.
        MOVE    ACRTC+2, D0     : Load "SAH" to D0.
        SWAP    D0              : Switch the upper and lower 16 bits of D0.
        MOVE    ACRTC+2, D0     : Load "SAL" to D0.
*
        ADDI.L  # (Memory Width), D0: Add the memory width to the display
                                      start address.
UDRL    MOVE    #$80, ACRTC     : Select the r80 register.
        MOVE    ACRTC+2, D1     : Load the raster to D1.
        CMPI    # (Undisplayed Raster Value), D1 : Non-display raster
        BNE     UDRL
*
        MOVE    #$CC, ACRTC     : Select the rCC register.
        SWAP    D0
        MOVE    D0, ACRTC+2     : Save the data in "SAH".
        SWAP    D0
        MOVE    D0, ACRTC+2     : Save the data in "SAL".
        RTS
```

Fig. 4-5 Upward Smooth Scroll Program

```
DOWN    MOVE    #$CC, ACRTC     : Select the rCC register.
        MOVE    ACRTC+2, D0     : Load "SAH" to D0.
        SWAP    D0
        MOVE    ACRTC+2, D0     : Load "SAL" to D0.
*
        SUBI.L  # (Memory Width), D0 : Subtract the memory width from the
                                       display start address.
*
        BRA     UDRL
```

Fig. 4-6 Downward Smooth Scroll Program Example

```
RIGHT   MOVE    #$CC, ACRTC  :  Select the rCC register.
        MOVE    ACRTC+2, D0  :  Load "SDA" and "SAH" to D0.
        ADDI    #$100, D0    :  Add 1 to "SDA".
        MOVE    D0, D1       :  Save D0 to D1.
        SWAP    D0
        MOVE    ACRTC+2, D0  :  Load "SAL" to D1.
*
        ANDI    #$0F00, D1
        BNE     UDRL         :  "SDA" = 0 ?
*
        SUBI.L  #$4, D0      :  Subtract 4 from the display start address.
        BRA     UDRL                (in case of 4 bits/pixel)
```

Fig. 4-7 Example Program of the Smooth Scroll to the Right

```
KEFT    MOVE    #$CC, ACRTC  :  Select the rCC register.
        MOVE    ACRTC+2, D0  :  Load "SDA" and "SAH" to D0.
        SUBI    #$100, D0    :  Subtract 1 from "SDA".
        MOVE    D0, D1
        SWAP    D0
        MOVE    ACRTC+2, D0  :  Load "SAL" to D0.
*
        ANDI    #$0F00, D1
        CMPI    #$0F00, D1   :  "SDA" = $F?
        BNE     UDREL
*
        ADD.L   #$4, D0      :  Add 4 to the display start address.
        BRA     UDRL                (in case of 4 bits/pixel)
```

Fig. 4-8 Example Program of the Smooth Scroll to the Left

4.3 Superimposing

The ACRTC can superimpose the background screen and the window. The background screen can be horizontally split to 3 parts. The graphic screen and the character screen can also be superimposed. This function allows the replacement and clearing of the figure without re-drawing the background screen. Also cross hair cursors and the graphic cursors can be supported by software using this function.

The background screen and the window are superimposed by setting "11" to the frame buffer access mode bit (ACM) in the operation mode register (OMR) and, thereby, selecting the superimpose mode. To perform smooth scroll or the window, the window smooth scroll bit (WSS) in the operation mode register is to be set to "1". In this case, attribute information for the horizontal smooth scroll of the base screen is not output.

Note) To superimpose the window screen on the base screen, an external circuit for superimposing must be provided in the video signal generation circuit.

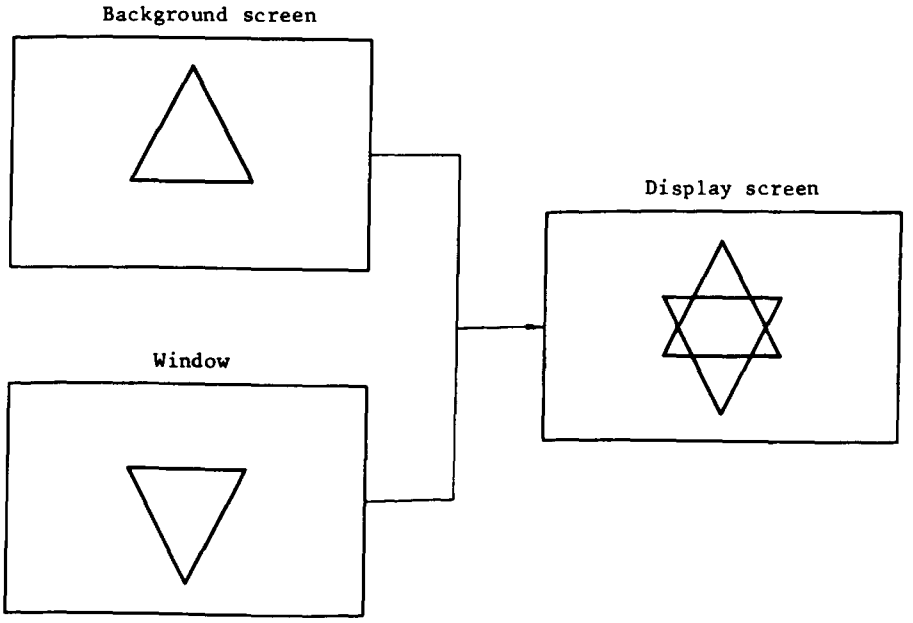


Fig. 4-9 Superimposing

5. EXAMPLE PROGRAMS

Examples of the programs for drawing various figures are shown below. Each program is written by the assembly language of HD68000. See Section "1.6" for the ACRTC modes and the screen configuration. As each program is relocatable, each program operates at any arbitrary memory address.

5.1 Example of Drawing

This program draws Fig. 5-1. Fig. 5-4 shows the source program list.

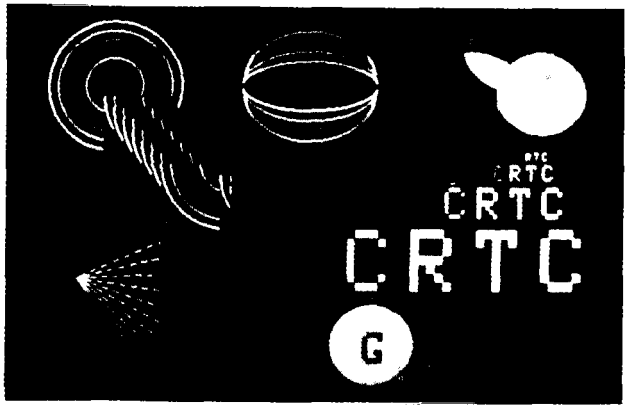


Fig. 5-1 Drawing Example (ACRTC)

5.2 Painting the Polygons

This program paints (tiling) the inside of the polygon shown in the Fig. 5-2. The source program list is shown in Fig. 5-5.

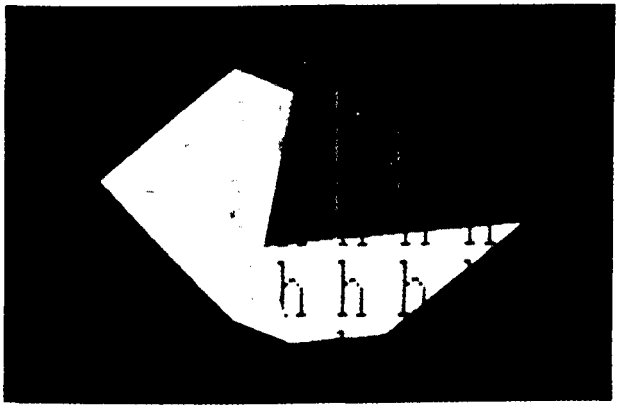


Fig. 5-2 Drawing Example (Painting the Polygons)

5.3 Drawing Panda Bears

This program draws a panda bear as shown in the Fig. 5-3. The source program list is shown in Fig. 5-6.

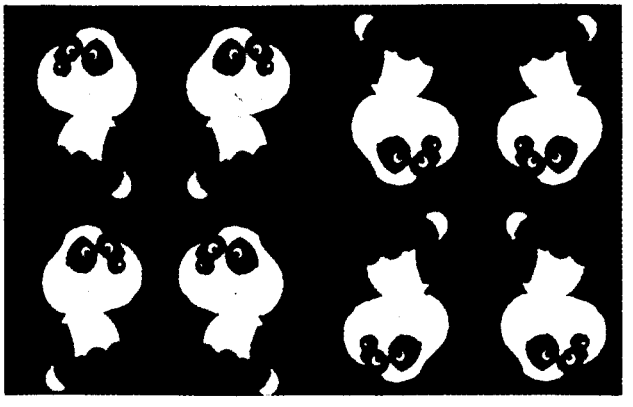


Fig. 5-3 Drawing Example (Panda Bear)

```
                                    OPT       CEX
00A00000            ACRTC   EQU       $A00000
                    *
                    ****************************
                    *                          *
                    *   ACRTC COMMAND TABLE    *
                    *                          *
                    ****************************
                    *
00000400            ORG     EQU       %0000010000000000
00000800            WPR     EQU       %0000100000000000
00000C00            RPR     EQU       %0000110000000000
00001800            WPTN    EQU       %0001100000000000
00001C00            RPTN    EQU       %0001110000000000
00002400            DRD     EQU       %0010010000000000
00002800            DWT     EQU       %0010100000000000
00002C00            DMOD    EQU       %0010110000000000
                    *
00004400            RD      EQU       %0100010000000000
00004800            WT      EQU       %0100100000000000
00004C00            MOD     EQU       %0100110000000000
00005800            CLR     EQU       %0101100000000000
00005C00            SCLR    EQU       %0101110000000000
00006000            CPY     EQU       %0110000000000000
00007000            SCPY    EQU       %0111000000000000
                    *
00008000            AMOVE   EQU       %1000000000000000
00008400            RMOVE   EQU       %1000010000000000
00008800            ALINE   EQU       %1000100000000000
00008C00            RLINE   EQU       %1000110000000000
00009000            ARCT    EQU       %1001000000000000
00009400            RRCT    EQU       %1001010000000000
00009800            APLL    EQU       %1001100000000000
00009C00            RPLL    EQU       %1001110000000000
0000A000            APLG    EQU       %1010000000000000
0000A400            RPLG    EQU       %1010010000000000
0000A800            CRCL    EQU       %1010100000000000
0000AC00            ELPS    EQU       %1010110000000000
0000B000            AARC    EQU       %1011000000000000
0000B400            RARC    EQU       %1011010000000000
0000B800            AEARC   EQU       %1011100000000000
0000BC00            REARC   EQU       %1011110000000000
0000C000            AFRCT   EQU       %1100000000000000
0000C400            RFRCT   EQU       %1100010000000000
0000C800            PAINT   EQU       %1100100000000000
0000CC00            DOT     EQU       %1100110000000000
0000D000            PTN     EQU       %1101000000000000
0000E000            AGCPY   EQU       %1110000000000000
0000F000            RGCPY   EQU       %1111000000000000
                    *
0 00000000 60000092                 BRA       INIT
                    *
                    ****************************
                    *                          *
                    *   ACRTC INITIALIZE DATA  *
                    *                          *
                    ****************************
                    *
```

Fig. 5-4 (1) (ACRTC)

```
 59 0 00000004 880A          INITTBL  DC      $880A                R02:HORIZONTAL SYNC.
 60 0 00000006 0B50                   DC      $0B50                R04:HORIZONTAL DISPLAY
 61 0 00000008 01C0                   DC      448                  R06:VERTICAL SYNC.
 62 0 0000000A 2010                   DC      $2010                R08:VERTICAL DISPLAY
 63 0 0000000C 0190                   DC      400                  R0A:SPLIT SCREEN WIDTH  SP1 (BASE)
 64 0 0000000E 0000                   DC      0                    R0C:SPLIT SCREEN WIDTH  SP0 (UPPER)
 65 0 00000010 0000                   DC      0                    R0E:SPLIT SCREEN WIDTH  SP2 (LOWER)
 66 0 00000012 0000                   DC      0                    R90:BLINK CONTROL
 67 0 00000014 0000                   DC      0                    R92:H-WINDOW DISPLAY
 68 0 00000016 0000                   DC      0                    R94:V-WINDOW DISPLAY
 69 0 00000018 0000                   DC      0                    R96:
 70 0 0000001A 0000                   DC      0                    R98:GRAPHIC CURSOR
 71 0 0000001C 0000                   DC      0                    R9A:
 72 0 0000001E 0000                   DC      0                    R9C:
 73                          *
 74 0 00000020 0000                   DC      0                    RC0:RASTER ADDR.  SCREEN 0
 75 0 00000022 00A4                   DC      164                  RC2:MEMORY WIDTH
 76 0 00000024 0F00                   DC      $0F00                RC4:START ADDR. H
 77 0 00000026 0000                   DC      $0000                RC6:            L
 78                          *
 79 0 00000028 0000                   DC      0                    RC8:RASTER ADDR.  SCREEN 1
 80 0 0000002A 00A4                   DC      164                  RCA:MEMORY WIDTH
 81 0 0000002C 0F03                   DC      $0F03                RCC:START ADDR. H
 82 0 0000002E 00C0                   DC      $00C0                RCE:            L
 83                          *
 84 0 00000030 0000                   DC      0                    RD0:RASTER ADDR.  SCREEN 2
 85 0 00000032 01E0                   DC      480                  RD2:MEMORY WIDTH
 86 0 00000034 0F09                   DC      $0F09                RD4:START ADDR. H
 87 0 00000036 0240                   DC      $0240                RD6:            L
 88                          *
 89 0 00000038 0000                   DC      0                    RD8:RASTER ADDR.  SCREEN 3
 90 0 0000003A 00A4                   DC      164                  RDA:MEMORY WIDTH
 91 0 0000003C 0002                   DC      $0002                RDC:START ADDR. H
 92 0 0000003E 0080                   DC      $0080                RDE:            L
 93                          *
 94 0 00000040 0000                   DC      0                    RE0:BLOCK CURSOR
 95 0 00000042 0000                   DC      0                    RE2:
 96 0 00000044 0000                   DC      0                    RE4:
 97 0 00000046 0000                   DC      0                    RE6:
 98                          *
 99 0 00000048 0000                   DC      0                    RE8:
100 0 0000004A 0000                   DC      0                    REA:ZOOM FACTER
101                         *
102 0 0000004C 8200                   DC      %0000001000000000    R02:COMMAND CONTROL
103 0 0000004E C128                   DC      %1100000100101000    R04:SYNC. CONTROL
104 0 00000050 4000                   DC      %0100000000000000    R06:DISPLAY CONTROL
105                         *
106                         ****************************
107                         *                          *
108                         *   COMMAND TABLE WRITE    *
109                         *                          *
110                         *       (A1)+ -> ACRTC     *
111                         *                          *
112                         ****************************
113                         *
114 0 00000052 48A72000     CTWRTE   MOVEM   D2,-(A7)
115 0 00000056 3419                  MOVE    (A1)+,D2             LOOP COUNTER LOAD
116                         *
```

Fig. 5-4 (2) (ACRTC)

```
117 0 00000058 3019          CTWR    MOVE    (A1)+,D0
118 0 0000005A 6100000C              BSR     CWRITE
119 0 0000005E 51CAFFF8              DBRA    D2,CTWR
120 0 00000062 4C9F0004              MOVEM   (A7)+,D2
121 0 00000066 4E75                  RTS
122
123                          *
124                          ************************
125                          *                      *
126                          *     COMMAND WRITE    *
127                          *                      *
128                          *       D0 -> ACRTC    *
129                          *                      *
130                          ************************
131 0 00000068 40E7          *
132 0 0000006A 46FC2700      CWRITE  MOVE    SR,-(A7)
133 0 0000006E 48E74000              MOVE    #$2700,SR
134 0 00000072 323900A00000  CWR     MOVEM.L D1,-(A7)
135 0 00000078 08010001              MOVE    ACRTC,D1
136 0 0000007C 67F4                  BTST    #1,D1
137                                  BEQ     CWR
138 0 0000007E 33FC000000A0  *
                 0000                MOVE    #0,ACRTC                R00 SELECT
139 0 00000086 33C000A00002          MOVE    D0,ACRTC+2              DATA -> ACRTC
140 0 0000008C 4CDF0002              MOVEM.L (A7)+,D1
141 0 00000090 46DF                  MOVE    (A7)+,SR
142 0 00000092 4E75                  RTS
143
144                          *
145                          ************************
146                          *                      *
147                          *   ACRTC INITIALIZE   *
148                          *                      *
149                          ************************
150 0 00000094 43FAFF8E      *
151                          INIT    LEA     INITTBL(PC),A1
152 0 00000098 33FC008200A0  *
                 0000                MOVE    #$82,ACRTC              R82 SELECT
153 0 000000A0 343C000D              MOVE    #13,D2                  LOOP COUNTER -> D2
154 0 000000A4 33D900A00002  INIT1   MOVE    (A1)+,ACRTC+2           R82-R9D WRITE
155 0 000000AA 51CAFFF8              DBRA    D2,INIT1
156
157 0 000000AE 33FC00C000A0  *
                 0000                MOVE    #$C0,ACRTC              RC0 SELECT
158 0 000000B6 343C0015              MOVE    #21,D2                  LOOP COUNTER -> D2
159 0 000000BA 33D900A00002  INIT2   MOVE    (A1)+,ACRTC+2           RC0-REB WRITE
160 0 000000C0 51CAFFF8              DBRA    D2,INIT2
161
162 0 000000C4 33FC000200A0  *
                 0000                MOVE    #$02,ACRTC              R02 SELECT
163 0 000000CC 33D900A00002          MOVE    (A1)+,ACRTC+2           R02 WRITE
164
165 0 000000D2 33FC000400A0  *
                 0000                MOVE    #$04,ACRTC              R04 SELECT
166 0 000000DA 33D900A00002          MOVE    (A1)+,ACRTC+2           R04 WRITE
167
168 0 000000E0 33FC000600A0  *
                 0000                MOVE    #$06,ACRTC              R06 SELECT
```

Fig. 5-4 (3) (ACRTC)

```
189 0 000000E8 33D900A00002          MOVE      (A1)+,ACRTC+2          R06 WRITE
190                                  *
191                                  ****************************
192                                  *                          *
193                                  *       D  E  M  O   1      *
194                                  *                          *
195                                  ****************************
196                                  *
197 0 000000EE 43FA001A              DEMO1     LEA       DATA1(PC),A1
198                                  *
199 0 000000F2 6100FF5E                        BSR       CTWRTE
200 0 000000F6 6100FF5A                        BSR       CTWRTE
201                                  *
202 0 000000FA 203C0007FFFF          DELAY     MOVE.L    #$7FFFF,D0
203 0 00000100 0480000000001         DD1       SUBI.L    #1,D0
204 0 00000106 66F8                            BNE       DD1
205                                  *
206 0 00000108 60E4                            BRA       DEMO1
207                                  *
208                                  ******************
209                                  *                *
210                                  *   D A T A  1   *
211                                  *                *
212                                  ******************
213                                  *
214             00000140             X0        SET       320
215             FFFFFF88             Y0        SET       -120
216                                  *
217 0 0000010A 000A                  DATA1     DC        10
218 0 0000010C 040040300C00                    DC        ORG,$4030,$0C00
219 0 00000112 080C4020                        DC        WPR+$C,$4020           R/W POINTER-$40200800
220 0 00000116 080D0800                        DC        WPR+$D,$0800
221 0 0000011A S800000000A3                    DC        CLR,$0000,163,-800
                FCE0
222                                  *
223 0 00000122 0308                            DC        774
224 0 00000124 18000010                        DC        WPTN,16
225 0 00000128 040F04110A11                    DC        $040F,$0411,$0A11,$0A0F,$1111,$1111,$110F,$0000
                0A0F11111111
                110F0000
226 0 00000138 1E1111081905                    DC        $1E11,$1109,$1905,$010F,$0111,$1111,$1E0F,$0000
                010F01111111
                1E0F0000
227                                  *
228 0 00000148 0800CCCC                        DC        WPR,$CCCC              CL0 - $CCCC
229 0 0000014C 0801CCCC                        DC        WPR+1,$CCCC            CL1 - $CCCC
230 0 00000150 80000208FFD8                    DC        AMOVE,200+X0,80+Y0
231 0 00000156 AC00000A0007                    DC        ELPS,10,7,44
                002C
232 0 0000015E 0803CCCC                        DC        WPR+3,$CCCC            EDG - $CCCC
233 0 00000162 C800                            DC        PAINT
234 0 00000164 0800AAAA                        DC        WPR,$AAAA              CL0 - $AAAA
235 0 00000168 0801AAAA                        DC        WPR+1,$AAAA            CL1 - $AAAA
236 0 0000016C 80000208FF52                    DC        AMOVE,200+X0,-54+Y0
237 0 00000172 AC00000A0007                    DC        ELPS,10,7,44
                002C
238 0 0000017A 0803AAAA                        DC        WPR+3,$AAAA            EDG - $AAAA
239 0 0000017E C800                            DC        PAINT
```

Fig. 5-4 (4) (ACRTC)

```
220 0 00000180 08008999              DC      WPR,$9999              CL0 - $9999
221 0 00000184 08019999              DC      WPR+1,$9999            CL1 - $9999
222 0 00000188 80000208FECC          DC      AMOVE,200+X0,-108+Y0
223 0 0000018E AC00000A0007          DC      ELPS,10,7,44
                   002C
224 0 00000196 08039999              DC      WPR+3,$9999            EDG - $9999
225 0 0000019A C800                  DC      PAINT
226 0 0000019C 8000014CFFA2          DC      AMOVE,12+X0,28+Y0
227 0 000001A2 E00001D2FFA2          DC      AGCPY,146+X0,28+Y0,108,108
                   006C006C
228 0 000001AC 8000014CFF1C          DC      AMOVE,12+X0,-108+Y0
229 0 000001B2 E00001D2FF1C          DC      AGCPY,146+X0,-108+Y0,108,108
                   006C006C
230 0 000001BC 8000014CFE96          DC      AMOVE,12+X0,-242+Y0
231 0 000001C2 E00001D2FE96          DC      AGCPY,146+X0,-242+Y0,108,108
                   006C006C
232 0 000001CC 800000C8FFA2          DC      AMOVE,-122+X0,28+Y0
233 0 000001D2 E00001D2FFA2          DC      AGCPY,146+X0,28+Y0,108,108
                   006C006C
234 0 000001DC 800000C8FF1C          DC      AMOVE,-122+X0,-108+Y0
235 0 000001E2 E00001D2FF1C          DC      AGCPY,146+X0,-108+Y0,108,108
                   006C006C
236 0 000001EC 800000C8FE96          DC      AMOVE,-122+X0,-242+Y0
237 0 000001F2 E00001D2FE96          DC      AGCPY,146+X0,-242+Y0,108,108
                   006C006C
238                            *
239 0 000001FC 8000014CFF1C          DC      AMOVE,12+X0,-108+Y0
240 0 00000202 E00101D2FFA2          DC      AGCPY+1,146+X0,28+Y0,108,108
                   006C006C
241 0 0000020C 8000014CFF1C          DC      AMOVE,12+X0,-108+Y0
242 0 00000212 E00101D2FE96          DC      AGCPY+1,146+X0,-242+Y0,108,108
                   006C006C
243 0 0000021C 8000014CFFA2          DC      AMOVE,12+X0,28+Y0
244 0 00000222 E00101D2FF1C          DC      AGCPY+1,146+X0,-108+Y0,108,108
                   006C006C
245 0 0000022C 8000014CFE96          DC      AMOVE,12+X0,-242+Y0
246 0 00000232 E00101D2FF1C          DC      AGCPY+1,146+X0,-108+Y0,108,108
                   006C006C
247                            *
248 0 0000023C 800000C8FE96          DC      AMOVE,-122+X0,-242+Y0
249 0 00000242 E00001D2FFA2          DC      AGCPY,146+X0,28+Y0,108,108
                   006C006C
250 0 0000024C 800000C8FFA2          DC      AMOVE,-122+X0,28+Y0
251 0 00000252 E00001D2FE96          DC      AGCPY,146+X0,-242+Y0,108,108
                   006C006C
252 0 0000025C 800000C8FF1C          DC      AMOVE,-122+X0,-108+Y0
253 0 00000262 E00000C8FE96          DC      AGCPY,-122+X0,-242+Y0,108,108
                   006C006C
254                            *
255 0 0000026C 800000C8FF1C          DC      AMOVE,-122+X0,-108+Y0
256 0 00000272 E00100C8FFA2          DC      AGCPY+1,-122+X0,28+Y0,108,108
                   006C006C
257                            *
258 0 0000027C 8000014CFF1C          DC      AMOVE,12+X0,-108+Y0
259 0 00000282 E0030I4CFF1C          DC      AGCPY+3,12+X0,-108+Y0,108,108
                   006C006C
260                            *
261 0 0000028C 800000C8FFA2          DC      AMOVE,-122+X0,28+Y0
```

Fig. 5-4 (5) (ACRTC)

```
262 0 00000292 E000014CFF1C          DC      AGCPY,12+X0,-108+Y0,108,108
                    006C006C
263 0 0000029C 8000014CFFA2          DC      AMOVE,12+X0,26+Y0
264 0 000002A2 E000014CFF1C          DC      AGCPY,12+X0,-108-Y0,108,108
                    006C006C
265 0 000002AC 800000C6FF1C          DC      AMOVE,-122+X0,-108+Y0
266 0 000002B2 E000014CFF1C          DC      AGCPY,12+X0,-108+Y0,108,108
                    006C006C
267 0 000002BC 8000014CFE98          DC      AMOVE,12+X0,-242+Y0
268 0 000002C2 E00001D2FF1C          DC      AGCPY,146+X0,-108+Y0,108,108
                    006C006C
269 0 000002CC 800001D2FF1C          DC      AMOVE,146+X0,-108+Y0
270 0 000002D2 E000014CFF1C          DC      AGCPY,12+X0,-108+Y0,108,108
                    006C006C
271                             *
272 0 000002DC 800001F2FF79          DC      AMOVE,178+X0,-15+Y0
273 0 000002E2 E001014CFE98          DC      AGCPY+1,12+X0,-242+Y0,108,108
                    006C006C
274 0 000002EC 800001B2FF79          DC      AMOVE,114+X0,-15+Y0
275 0 000002F2 E00101D2FE98          DC      AGCPY+1,146+X0,-242+Y0,108,108
                    006C006C
276 0 000002FC 08058000              DC      WPR+5,$8000             PP
277 0 00000300 08068000              DC      WPR+6,$8000             PS
278 0 00000304 0807F353              DC      WPR+7,$F353             PE,PZ
279 0 00000308 0800CCCC              DC      WPR,$CCCC               CL0 - $CCCC
280 0 0000030C 08010000              DC      WPR+1,$0000             CL1 - $0000
281                             *
282 0 00000310 800000F4FEBC          DC      AMOVE,-76+X0,-204+Y0
283 0 00000316 D0081F17              DC      PTM+8,31*256+23
284 0 0000031A 08058080              DC      WPR+5,$8080
285 0 0000031E 08068080              DC      WPR+6,$8080
286 0 00000322 0807F3D3              DC      WPR+7,$F3D3
287 0 00000326 0800AAAA              DC      WPR,$AAAA               CL0 - $AAAA
288 0 0000032A 8000017AFEBC          DC      AMOVE,58+X0,-204+Y0
289 0 00000330 D0081F17              DC      PTM+8,31*256+23
290 0 00000334 08050000              DC      WPR+5,$0000             PP
291 0 00000338 08060000              DC      WPR+6,$0000             PS
292 0 0000033C 08077353              DC      WPR+7,$7353             PE,PZ
293 0 00000340 08009999              DC      WPR,$9999               CL0 - $9999
294 0 00000344 80000200FEBC          DC      AMOVE,192+X0,-204+Y0
295 0 0000034A D0081F17              DC      PTM+8,31*256+23
296 0 0000034E 80000078FFB5          DC      AMOVE,-200+X0,45+Y0
297                             *
298 0 00000354 08009999              DC      WPR,$9999               CL0 - $9999
299 0 00000358 08019999              DC      WPR+1,$9999             CL1 - $9999
300 0 0000035C AC00000A0007          DC      ELPS,10,7,10
                    000A
301 0 00000364 0800AAAA              DC      WPR,$AAAA               CL0 - $AAAA
302 0 00000368 0801AAAA              DC      WPR+1,$AAAA             CL1 - $AAAA
303 0 0000036C AC00000A0007          DC      ELPS,10,7,20
                    0014
304 0 00000374 0800BBBB              DC      WPR,$BBBB               CL0 - $BBBB
305 0 00000378 0801BBBB              DC      WPR+1,$BBBB             CL1 - $BBBB
306 0 0000037C AC00000A0007          DC      ELPS,10,7,30
                    001E
307 0 00000384 0800CCCC              DC      WPR,$CCCC               CL0 - $CCCC
308 0 00000388 0801CCCC              DC      WPR+1,$CCCC             CL1 - $CCCC
309 0 0000038C AC00000A0007          DC      ELPS,10,7,40
```

Fig. 5-4 (6) (ACRTC)

```
                    0028
310 0 00000394 0800DDDD             DC      WPR,$DDDD               CL0 - $DDDD
311 0 00000398 0801DDDD             DC      WPR+1,$DDDD             CL1 - $DDDD
312 0 0000039C AC00000A0007         DC      ELPS,10,7,50
                    0032
313 0 000003A4 0800EEEE             DC      WPR,$EEEE               CL0 - $EEEE
314 0 000003A8 0801EEEE             DC      WPR+1,$EEEE             CL1 - $EEEE
315 0 000003AC AC00000A0007         DC      ELPS,10,7,60
                    003C
316 0 000003B4 0800FFFF             DC      WPR,$FFFF               CL0 - $FFFF
317 0 000003B8 0801FFFF             DC      WPR+1,$FFFF             CL1 - $FFFF
318 0 000003BC AC00000A0007         DC      ELPS,10,7,70
                    0046
319 0 000003C4 800001407FB5         DC      AMOVE,0+X0,45+Y0
320 0 000003CA 0800FFFF             DC      WPR,$FFFF               CL0 - $FFFF
321 0 000003CE 0801FFFF             DC      WPR+1,$FFFF             CL1 - $FFFF
322 0 000003D2 AC00000A0001         DC      ELPS,10,1,70
                    0046
323 0 000003DA 0800EEEE             DC      WPR,$EEEE               CL0 - $EEEE
324 0 000003DE 0801EEEE             DC      WPR+1,$EEEE             CL1 - $EEEE
325 0 000003E2 AC00000A0002         DC      ELPS,10,2,70
                    0046
326 0 000003EA 0800DDDD             DC      WPR,$DDDD               CL0 - $DDDD
327 0 000003EE 0801DDDD             DC      WPR+1,$DDDD             CL1 - $DDDD
328 0 000003F2 AC00000A0003         DC      ELPS,10,3,70
                    0046
329 0 000003FA 0800CCCC             DC      WPR,$CCCC               CL0 - $CCCC
330 0 000003FE 0801CCCC             DC      WPR+1,$CCCC             CL1 - $CCCC
331 0 00000402 AC00000A0004         DC      ELPS,10,4,70
                    0046
332 0 0000040A 0800BBBB             DC      WPR,$BBBB               CL0 - $BBBB
333 0 0000040E 0801BBBB             DC      WPR+1,$BBBB             CL1 - $BBBB
334 0 00000412 AC00000A0005         DC      ELPS,10,5,70
                    0046
335 0 0000041A 0800AAAA             DC      WPR,$AAAA               CL0 - $AAAA
336 0 0000041E 0801AAAA             DC      WPR+1,$AAAA             CL1 - $AAAA
337 0 00000422 AC00000A0006         DC      ELPS,10,6,70
                    0046
338 0 0000042A 08009999             DC      WPR,$9999               CL0 - $9999
339 0 0000042E 08019999             DC      WPR+1,$9999             CL1 - $9999
340 0 00000432 AC00000A0007         DC      ELPS,10,7,70
                    0046
341 0 0000043A 18000010             DC      WPTN,16
342 0 0000043E 041104090405         DC      $0411,$0409,$0405,$040F,$0411,$1511,$1F0F,$0000
                    040F04111511
                    1F0F0000
343 0 0000044E 0E1111110111F        DC      $0E11,$1111,$011F,$0111,$0111,$110A,$0E04,$0000
                    011101111110A
                    0E040000
344 0 0000045E 800002087F74         DC      AMOVE,200+X0,-20+Y0
345 0 00000464 08058000             DC      WPR+5,$8000
346 0 00000468 08068000             DC      WPR+6,$8000
347 0 0000046C 0807F050             DC      WPR+7,$F050
348 0 00000470 08000000             DC      WPR+0,$0000             CL0 - $0000
349 0 00000474 08019999             DC      WPR+1,$9999             CL1 - $9999
350 0 00000478 D0000705             DC      PTN,7*256+5
351 0 0000047C 800001E87F82         DC      AMOVE,168+X0,-38+Y0
352 0 00000482 0807F151             DC      WPR+7,$F151             PE,PZ
```

Fig. 5-4 (7) (ACRTC)

```
353 0 00000486 D0000F0B          DC      PTN,15*256+11
354 0 0000048A 800001A8FF40      DC      AMOVE,104+X0,-72+Y0
355 0 00000490 0807F353          DC      WPR+7,#F353          PE,PZ
356 0 00000494 D0001F17          DC      PTN,31*256+23
357 0 00000498 80000128FEFE      DC      AMOVE,-24+X0,-138+Y0
358 0 0000049E 0807F757          DC      WPR+7,#F757          PE,PZ
359 0 000004A2 D0003F2F          DC      PTN,63*256+47
360 0 000004A6 80000210FF74      DC      AMOVE,208+X0,-20+Y0
361 0 000004AC 08058080          DC      WPR+5,#8080          PP
362 0 000004B0 08068080          DC      WPR+6,#8080          PS
363 0 000004B4 0807F0D0          DC      WPR+7,#F0D0          PE,PZ
364 0 000004B8 0801AAAA          DC      WPR+1,#AAAA          CL1 - #AAAA
365 0 000004BC D0000705          DC      PTN,7*256+5
366 0 000004C0 80000228FF74      DC      AMOVE,232+X0,-20+Y0
367 0 000004C6 0801FFFF          DC      WPR+1,#FFFF          CL1 - #FFFF
368 0 000004CA D0000705          DC      PTN,7*256+5
369 0 000004CE 80000228FF62      DC      AMOVE,232+X0,-38+Y0
370 0 000004D4 0807F1D1          DC      WPR+7,#F1D1          PE,PZ
371 0 000004D8 D0000F0B          DC      PTN,15*256+11
372 0 000004DC 800001F8FF62      DC      AMOVE,184+X0,-38+Y0
373 0 000004E2 0801AAAA          DC      WPR+1,#AAAA          CL1 - #AAAA
374 0 000004E6 D0000F0B          DC      PTN,15*256+11
375 0 000004EA 80000228FF40      DC      AMOVE,232+X0,-72+Y0
376 0 000004F0 0807F3D3          DC      WPR+7,#F3D3          PE,PZ
377 0 000004F4 0801FFFF          DC      WPR+1,#FFFF          CL1 - #FFFF
378 0 000004F8 D0001F17          DC      PTN,31*256+23
379 0 000004FC 800001C8FF40      DC      AMOVE,136+X0,-72+Y0
380 0 00000502 0801AAAA          DC      WPR+1,#AAAA          CL1 - #AAAA
381 0 00000506 D0001F17          DC      PTN,31*256+23
382 0 0000050A 80000228FEFE      DC      AMOVE,232+X0,-138+Y0
383 0 00000510 0807F7D7          DC      WPR+7,#F7D7          PE,PZ
384 0 00000514 0801FFFF          DC      WPR+1,#FFFF          CL1 - #FFFF
385 0 00000518 D0003F2F          DC      PTN,63*256+47
386 0 0000051C 80000168FEFE      DC      AMOVE,40+X0,-138+Y0
387 0 00000522 0801AAAA          DC      WPR+1,#AAAA          CL1 - #AAAA
388 0 00000526 D0003F2F          DC      PTN,63*256+47
389 0 0000052A 80000218FF74      DC      AMOVE,216+X0,-20+Y0
390 0 00000530 08050000          DC      WPR+5,#0000          PP
391 0 00000534 08060000          DC      WPR+6,#0000          PS
392 0 00000538 08077050          DC      WPR+7,#7050          PE,PZ
393 0 0000053C 0801BBBB          DC      WPR+1,#BBBB          CL1 - #BBBB
394 0 00000540 D0000705          DC      PTN,7*256+5
395 0 00000544 80000208FF62      DC      AMOVE,200+X0,-38+Y0
396 0 0000054A 08077151          DC      WPR+7,#7151          PE,PZ
397 0 0000054E D0000F0B          DC      PTN,15*256+11
398 0 00000552 800001E8FF40      DC      AMOVE,168+X0,-72+Y0
399 0 00000558 08077353          DC      WPR+7,#7353          PE,PZ
400 0 0000055C D0001F17          DC      PTN,31*256+23
401 0 00000560 800001A8FEFE      DC      AMOVE,104+X0,-138+Y0
402 0 00000566 08077757          DC      WPR+7,#7757          PE,PZ
403 0 0000056A D0003F2F          DC      PTN,63*256+47
404 0 0000056E 80000220FF74      DC      AMOVE,224+X0,-20+Y0
405 0 00000574 08050080          DC      WPR+5,#0080          PP
406 0 00000578 08060080          DC      WPR+6,#0080          PS
407 0 0000057C 080770D0          DC      WPR+7,#70D0          PE,PZ
408 0 00000580 0801EEEE          DC      WPR+1,#EEEE          CL1 - #EEEE
409 0 00000584 D0000705          DC      PTN,7*256+5
410 0 00000588 80000218FF62      DC      AMOVE,216+X0,-38+Y0
```

Fig. 5-4 (8) (ACRTC)

```
68000 MACRO ASSEMBLER 1.0          ACRTC   .SA 12/13/84 10:09:38

411 0 00000588 080771B1                DC      WPR+7,$71B1                PE,PZ
412 0 0000058C D0000F0B                DC      PTN,15*256+11
413 0 00000590 80000208FF40            DC      AMOVE,200+X0,-72+Y0
414 0 00000598 080773D3                DC      WPR+7,$73D3                PE,PZ
415 0 0000059C D0001F17                DC      PTN,31*256+23
416 0 000005A0 800001E8FEFE            DC      AMOVE,488+X0,-138+Y0
417 0 000005A4 080777D7                DC      WPR+7,$77D7                PE,PZ
418 0 000005A8 D0003F2F                DC      PTN,63*256+47
419 0 000005AC 8000006EFF65            DC      AMOVE,-210+X0,-35+Y0
420 0 000005B0 E0000032FF6F            DC      AGCPY,-270+X0,-25+Y0,70,70
                00460046
421 0 000005BC 80000078FF5B            DC      AMOVE,-200+X0,-45+Y0
422 0 000005C0 E0000032FF6F            DC      AGCPY,-270+X0,-25+Y0,70,70
                00460046
423 0 000005CC 80000082FF51            DC      AMOVE,-190+X0,-55+Y0
424 0 000005D0 E0000032FF6F            DC      AGCPY,-270+X0,-25+Y0,70,70
                00460046
425 0 000005DC 8000008CFF47            DC      AMOVE,-180+X0,-65+Y0
426 0 000005E0 E0000032FF6F            DC      AGCPY,-270+X0,-25+Y0,70,70
                00460046
427 0 000005EC 80000096FF3D            DC      AMOVE,-170+X0,-75+Y0
428 0 000005F0 E0000032FF6F            DC      AGCPY,-270+X0,-25+Y0,70,70
                00460046
429 0 000005FC 800000A0FF33            DC      AMOVE,-160+X0,-85+Y0
430 0 00000600 E0000032FF6F            DC      AGCPY,-270+X0,-25+Y0,70,70
                00460046
431 0 0000060C 800000AAFF29            DC      AMOVE,-150+X0,-95+Y0
432 0 00000610 E0000032FF6F            DC      AGCPY,-270+X0,-25+Y0,70,70
                00460046
433                         *
434 0 00000622 0800AAAA                DC      WPR,$AAAA                  CL0 - $AAAA
435 0 00000626 08019999                DC      WPR+1,$9999                CL1 - $9999
436 0 0000062A 08052000                DC      WPR+5,$2000                PP
437 0 0000062E 08062000                DC      WPR+6,$2000                PS
438 0 00000632 08072333                DC      WPR+7,$2333                PE,PZ
439 0 00000636 80000050FEA2            DC      AMOVE,-240+X0,-230+Y0
440 0 0000063C 800000AAFEA2            DC      ALINE,-150+X0,-230+Y0
441 0 00000642 80000050FEA2            DC      AMOVE,-240+X0,-230+Y0
442 0 00000648 800000AAFEB1            DC      ALINE,-150+X0,-215+Y0
443 0 0000064E 80000050FEA2            DC      AMOVE,-240+X0,-230+Y0
444 0 00000654 800000AAFEC0            DC      ALINE,-150+X0,-200+Y0
445 0 0000065A 80000050FEA2            DC      AMOVE,-240+X0,-230+Y0
446 0 00000660 800000AAFECF            DC      ALINE,-150+X0,-185+Y0
447 0 00000666 80000050FEA2            DC      AMOVE,-240+X0,-230+Y0
448 0 0000066C 800000AAFEDE            DC      ALINE,-150+X0,-170+Y0
449 0 00000672 80000050FEA2            DC      AMOVE,-240+X0,-230+Y0
450 0 00000678 800000AAFEED            DC      ALINE,-150+X0,-155+Y0
451 0 0000067E 80000050FEA2            DC      AMOVE,-240+X0,-230+Y0
452 0 00000684 800000AAFEFC            DC      ALINE,-150+X0,-140+Y0
453 0 0000068A 80000050FEA2            DC      AMOVE,-240+X0,-230+Y0
454 0 00000690 800000AAFF0B            DC      ALINE,-150+X0,-125+Y0
455 0 00000696 80000050FEA2            DC      AMOVE,-240+X0,-230+Y0
456 0 0000069C 800000AAFF1A            DC      ALINE,-150+X0,-110+Y0
457 0 000006A2 80000050FEA2            DC      AMOVE,-240+X0,-230+Y0
458 0 000006A8 800000AAFF29            DC      ALINE,-150+X0,-95+Y0
459 0 000006AE 0800DDDD                DC      WPR,$DDDD                  CL0 - $DDDD
460 0 000006B2 0801EEEE                DC      WPR+1,$EEEE                CL1 - $EEEE
461 0 000006B6 08072535                DC      WPR+7,$2535                PE,PZ
```

Fig. 5-4 (9) (ACRTC)

```
68000 MACRO ASSEMBLER 1.0          ACRTC   .SA 12/13/84 10:09:38

 462 0 000006BA 800000S0FF06            DC      AMOVE,-240+X0,-130+Y0
 463 0 000006C0 880000AAFF29            DC      ALINE,-150+X0,-95+Y0
 464 0 000006C6 800000S0FF06            DC      AMOVE,-240+X0,-130+Y0
 465 0 000006CC 880000AAFF1A            DC      ALINE,-150+X0,-110+Y0
 466 0 000006D2 800000S0FF06            DC      AMOVE,-240+X0,-130+Y0
 467 0 000006D8 880000AAFF0B            DC      ALINE,-150+X0,-125+Y0
 468 0 000006DE 800000S0FF06            DC      AMOVE,-240+X0,-130+Y0
 469 0 000006E4 880000AAFEFC            DC      ALINE,-150+X0,-140+Y0
 470 0 000006EA 800000S0FF06            DC      AMOVE,-240+X0,-130+Y0
 471 0 000006F0 880000AAFEED            DC      ALINE,-150+X0,-155+Y0
 472 0 000006F6 800000S0FF06            DC      AMOVE,-240+X0,-130+Y0
 473 0 000006FC 880000AAFEDE            DC      ALINE,-150+X0,-170+Y0
 474 0 00000702 800000S0FF06            DC      AMOVE,-240+X0,-130+Y0
 475 0 00000708 880000AAFECF            DC      ALINE,-150+X0,-185+Y0
 476 0 0000070E 800000S0FF06            DC      AMOVE,-240+X0,-130+Y0
 477 0 00000714 880000AAFEC0            DC      ALINE,-150+X0,-200+Y0
 478 0 0000071A 800000S0FF06            DC      AMOVE,-240+X0,-130+Y0
 479 0 00000720 880000AAFEB1            DC      ALINE,-150+X0,-215+Y0
 480 0 00000726 800000S0FF06            DC      AMOVE,-240+X0,-130+Y0
 481 0 0000072C 880000AAFEA2            DC      ALINE,-150+X0,-230+Y0
 482                            *
 483                                    END

****** TOTAL ERRORS    0--   0
```

```
68000 MACRO ASSEMBLER 1.0          ACRTC   .SA 12/13/84 10:09:38

SYMBOL TABLE LISTING

SYMBOL NAME      SECT   VALUE       SYMBOL NAME      SECT   VALUE

AARC                    0000B000    INIT1              0   000000A4
ACRTC                   00A00000    INIT2              0   000000BA
AEARC                   0000B800    INITTBL            0   00000004
AFRCT                   0000C000    MOD                    00004C00
AGCPY                   0000E000    ORG                    00000400
ALINE                   00008800    PAINT                  0000C800
AMOVE                   00008000    PTN                    0000D000
APLG                    0000A000    RARC                   0000B400
APLL                    00009800    RD                     00004400
ARCT                    00009000    REARC                  0000BC00
CLR                     00005800    RFRCT                  0000C400
CPY                     00006000    RGCPY                  0000F000
CRCL                    0000A800    RLINE                  00008C00
CTWR              0     00000058    RMOVE                  00008400
CTWRTE            0     00000052    RPLG                   0000A400
CWR               0     00000072    RPLL                   00009C00
CWRITE            0     00000068    RPR                    00006C00
DATA1             0     0000010A    RPTN                   00001C00
DD1               0     00000100    RRCT                   00009400
DELAY             0     000000FA    SCLR                   00005C00
DEMO1             0     000000EE    SCPY                   00007000
DMOD                    00002C00    WPR                    00000800
DOT                     0000CC00    WPTN                   00001800
DRD                     00002400    WT                     00004800
DWT                     00002800    X0                     00000140
ELPS                    0000AC00    Y0                     FFFFFF88
INIT              0     00000094
```

Fig. 5-4 (10) (ACRTC)

```
                                    OPT       CEX
                00A00000    ACRTC   EQU       $A00000
                            *
                            ****************************
                            *                          *
                            *   ACRTC COMMAND TABLE    *
                            *                          *
                            ****************************
                            *
                00000400    ORG     EQU       %0000010000000000
                00000800    WPR     EQU       %0000100000000000
                00000C00    RPR     EQU       %0000110000000000
                00001800    WPTN    EQU       %0001100000000000
                00001C00    RPTN    EQU       %0001110000000000
                00002400    DRD     EQU       %0010010000000000
                00002800    DWT     EQU       %0010100000000000
                00002C00    DMOD    EQU       %0010110000000000
                            *
                00004400    RD      EQU       %0100010000000000
                00004800    WT      EQU       %0100100000000000
                00004C00    MOD     EQU       %0100110000000000
                00005800    CLR     EQU       %0101100000000000
                00005C00    SCLR    EQU       %0101110000000000
                00006000    CPY     EQU       %0110000000000000
                00007000    SCPY    EQU       %0111000000000000
                            *
                00008000    AMOVE   EQU       %1000000000000000
                00008400    RMOVE   EQU       %1000010000000000
                00008800    ALINE   EQU       %1000100000000000
                00008C00    RLINE   EQU       %1000110000000000
                00009000    ARCT    EQU       %1001000000000000
                00009400    RRCT    EQU       %1001010000000000
                00009800    APLL    EQU       %1001100000000000
                00009C00    RPLL    EQU       %1001110000000000
                0000A000    APLG    EQU       %1010000000000000
                0000A400    RPLG    EQU       %1010010000000000
                0000A800    CRCL    EQU       %1010100000000000
                0000AC00    ELPS    EQU       %1010110000000000
                0000B000    AARC    EQU       %1011000000000000
                0000B400    RARC    EQU       %1011010000000000
                0000B800    AEARC   EQU       %1011100000000000
                0000BC00    REARC   EQU       %1011110000000000
                0000C000    AFRCT   EQU       %1100000000000000
                0000C400    RFRCT   EQU       %1100010000000000
                0000C800    PAINT   EQU       %1100100000000000
                0000CC00    DOT     EQU       %1100110000000000
                0000D000    PTN     EQU       %1101000000000000
                0000E000    AGCPY   EQU       %1110000000000000
                0000F000    RGCPY   EQU       %1111000000000000
                            *
0 00000000 60000052         BRA     INIT
                            *
                            ****************************
                            *                          *
                            *   ACRTC INITIALIZE DATA  *
                            *                          *
                            ****************************
                            *
```

Fig. 5-5 (1) (Painting the Polygons)

```
68000 MACRO ASSEMBLER 1.0          PAINT   .SA 12/13/84 10:00:28

  59 0 00000004 680A          INITTBL  DC       $680A                 R02:HORIZONTAL SYNC.
  60 0 00000006 0B50                   DC       $0B50                 R04:HORIZONTAL DISPLAY
  61 0 00000008 01C0                   DC       448                   R06:VERTICAL SYNC.
  62 0 0000000A 2010                   DC       $2010                 R08:VERTICAL DISPLAY
  63 0 0000000C 0190                   DC       400                   R0A:
  64 0 0000000E 0000                   DC       0                     R0C:SPLIT SCREEN WIDTH  SP1
  65 0 00000010 0000                   DC       0                     R0E:                    SP2
  66 0 00000012 0000                   DC       0                     R90:BLINK CONTROL
  67 0 00000014 0000                   DC       0                     R92:H-WINDOW DISPLAY
  68 0 00000016 0000                   DC       0                     R94:V-WINDOW DISPLAY
  69 0 00000018 0000                   DC       0                     R96:
  70 0 0000001A 0000                   DC       0                     R98:GRAPHIC CURSOR
  71 0 0000001C 0000                   DC       0                     R9A:
  72 0 0000001E 0000                   DC       0                     R9C:
  73                          *
  74 0 00000020 0000                   DC       0                     RC0:RASTER ADDR.  SCREEN 0
  75 0 00000022 00A4                   DC       164                   RC2:MEMORY WIDTH
  76 0 00000024 0F00                   DC       $0F00                 RC4:START ADDR. H
  77 0 00000026 0000                   DC       $0000                 RC6:            L
  78                          *
  79 0 00000028 0000                   DC       0                     RC8:RASTER ADDR.  SCREEN 1
  80 0 0000002A 00A4                   DC       164                   RCA:MEMORY WIDTH
  81 0 0000002C 0F03                   DC       $0F03                 RCC:START ADDR. H
  82 0 0000002E 00C0                   DC       $00C0                 RCE:            L
  83                          *
  84 0 00000030 0000                   DC       0                     RD0:RASTER ADDR.  SCREEN 2
  85 0 00000032 01E0                   DC       480                   RD2:MEMORY WIDTH
  86 0 00000034 0F09                   DC       $0F09                 RD4:START ADDR. H
  87 0 00000036 0240                   DC       $0240                 RD6:            L
  88                          *
  89 0 00000038 0000                   DC       0                     RD8:RASTER ADDR.  SCREEN 3
  90 0 0000003A 00A4                   DC       164                   RDA:MEMORY WIDTH
  91 0 0000003C 0002                   DC       $0002                 RDC:START ADDR. H
  92 0 0000003E 0080                   DC       $0080                 RDE:            L
  93                          *
  94 0 00000040 0000                   DC       0                     RE0:CHARACTER CURSOR
  95 0 00000042 0000                   DC       0                     RE2:
  96 0 00000044 0000                   DC       0                     RE4:
  97 0 00000046 0000                   DC       0                     RE6:
  98                          *
  99 0 00000048 0000                   DC       0                     RE8:
 100 0 0000004A 0000                   DC       0                     REA:ZOOM FACTER
 101                          *
 102 0 0000004C 0200                   DC       %0000001000000000     R02:COMMAND CONTROL
 103 0 0000004E C128                   DC       %1100000100101000     R04:SYNC. CONTROL
 104 0 00000050 4000                   DC       %0100000000000000     R06:DISPLAY CONTROL
 105                          *
 106                          ****************************
 107                          *                          *
 108                          *   COMMAND TABLE WRITE    *
 109                          *                          *
 110                          *        (A1)+ -> ACRTC    *
 111                          *                          *
 112                          ****************************
 113                          *
 114 0 00000052 48A72000      CTWRTE   MOVEM    D2,-(A7)
 115 0 00000056 3419                   MOVE     (A1)+,D2              LOOP COUNTER LOAD
 116                          *
```

Fig. 5-5 (2) (Painting the Polygons)

```
117 0 00000058 3019          CTWR    MOVE    (A1)+,D0
118 0 0000005A 8100000C              BSR     CWRITE
119 0 0000005E 51CAFFF8              DBRA    D2,CTWR
120 0 00000062 4C9F0004              MOVEM   (A7)+,D2
121 0 00000066 4E75                  RTS
122                          *
123                          ************************
124                          *                      *
125                          *     COMMAND WRITE    *
126                          *                      *
127                          *       D0 -> ACRTC    *
128                          *                      *
129                          ************************
130                          *
131 0 00000068 40E7          CWRITE  MOVE    SR,-(A7)
132 0 0000006A 46FC2700              MOVE    #$2700,SR
133 0 0000006E 48E74000              MOVEM.L D1,-(A7)
134 0 00000072 323900A00000  CWR     MOVE    ACRTC,D1
135 0 00000078 08010001              BTST    #1,D1
136 0 0000007C 67F4                  BEQ     CWR
137                          *
138 0 0000007E 33FC000000A0          MOVE    #0,ACRTC                R00 SELECT
                 0000
139 0 00000086 33C000A00002          MOVE    D0,ACRTC+2              DATA -> ACRTC
140 0 0000008C 4CDF0002              MOVEM.L (A7)+,D1
141 0 00000090 46DF                  MOVE    (A7)+,SR
142 0 00000092 4E75                  RTS
143                          *
144                          ************************
145                          *                      *
146                          *   ACRTC INITIALIZE   *
147                          *                      *
148                          ************************
149                          *
150 0 00000094 43FAFF6E      INIT    LEA     INITTBL(PC),A1
151                          *
152 0 00000098 33FC008200A0          MOVE    #$82,ACRTC              R82 SELECT
                 0000
153 0 000000A0 343C000D              MOVE    #13,D2                  LOOP COUNTER -> D2
154 0 000000A4 33D900A00002  INIT1   MOVE    (A1)+,ACRTC+2           R82-R8D WRITE
155 0 000000AA 51CAFFF8              DBRA    D2,INIT1
156                          *
157 0 000000AE 33FC00C000A0          MOVE    #$C0,ACRTC              RC0 SELECT
                 0000
158 0 000000B6 343C0015              MOVE    #21,D2                  LOOP COUNTER -> D2
159 0 000000BA 33D900A00002  INIT2   MOVE    (A1)+,ACRTC+2           RC0-REB WRITE
160 0 000000C0 51CAFFF8              DBRA    D2,INIT2
161                          *
162 0 000000C4 33FC000200A0          MOVE    #$02,ACRTC              R02 SELECT
                 0000
163 0 000000CC 33D900A00002          MOVE    (A1)+,ACRTC+2           R02 WRITE
164                          *
165 0 000000D2 33FC000400A0          MOVE    #$04,ACRTC              R04 SELECT
                 0000
166 0 000000DA 33D900A00002          MOVE    (A1)+,ACRTC+2           R04 WRITE
167                          *
168 0 000000E0 33FC000600A0          MOVE    #$06,ACRTC              R06 SELECT
                 0000
```

Fig. 5-5 (3) (Painting the Polygons)

```
68000 MACRO ASSEMBLER 1.0          PAINT   .SA 12/13/84 10:00:28

169 0 000000E8 33D900A00002          MOVE      (A1)+,ACRTC+2          R06 WRITE
170                                  *
171                                  ****************************
172                                  *                          *
173                                  *      D  E  M  O   2       *
174                                  *                          *
175                                  ****************************
176                                  *
177 0 000000EE 43FA0034      DEMO2   LEA       DATA2(PC),A1
178                                  *
179 0 000000F2 6100FF5E              BSR       CTWRTE
180 0 000000F6 6100001C              BSR       DELAY
181                                  *
182 0 000000FA 6100FF56              BSR       CTWRTE                 PAINT1
183 0 000000FE 6100FF52              BSR       CTWRTE                 PAINT2
184 0 00000102 6100FF4E              BSR       CTWRTE                 PAINT3
185 0 00000106 6100FF4A              BSR       CTWRTE                 PAINT4
186 0 0000010A 6100FF46              BSR       CTWRTE                 PAINT5
187                                  *
188 0 0000010E 61000004              BSR       DELAY
189 0 00000112 60DA                  BRA       DEMO2
190                                  *
191 0 00000114 203C0007FFFF  DELAY   MOVE.L    #$7FFFF,D0
192 0 0000011A 048000000001  DD1     SUBI.L    #1,D0
193 0 00000120 66F8                  BNE       DD1
194 0 00000122 4E75                  RTS
195                                  *
196                                  ******************
197                                  *                *
198                                  *   D A T A  2   *
199                                  *                *
200                                  ******************
201                                  *
202 0 00000124 003B          DATA2   DC        59
203 0 00000128 040040300C00          DC        ORG,$4030,$0C00
204 0 0000012C 080C4030              DC        WPR+$C,$4030           R/W POINTER-$40300C00
205 0 00000130 080D0C00              DC        WPR+$D,$0C00
206 0 00000134 5800000000A3          DC        CLR,$0000,163,-399
                FE71
207 0 0000013C 0800FFFF              DC        WPR,$FFFF              CL0 - $FFFF
208 0 00000140 0801FFFF              DC        WPR+1,$FFFF            CL1 - $FFFF
209 0 00000144 08060000              DC        WPR+6,$0000            PS
210                                  *
211 0 00000148 800000F0FFC4          DC        AMOVE,240,-60
212 0 0000014E A00000030158          DC        APLG,3,342,-52,402,-75,301,-82
                FFCC0192FFB5
                012DFFAE
213 0 0000015E 9800000300B8          DC        APLL,3,104,-160,272,-220,301,-82
                FF600110FF24
                012DFFAE
214 0 0000018E 80000192FFB5          DC        AMOVE,402,-75
215 0 00000174 980000050218          DC        APLL,5,536,-200,272,-220,303,-308,402,-300,536,-200
                FF380110FF24
                012FFECC0192
                FED40218FF38
216 0 0000018C 80000068FF60          DC        AMOVE,104,-160
217 0 00000192 9800000200F1          DC        APLL,2,241,-288,303,-308
                FEE2012FFECC
```

Fig. 5-5 (4) (Painting the Polygons)

```
68000 MACRO ASSEMBLER 1.0          PAINT   .SA 12/13/84 10:00:28

218                                     *
219 0 0000019E 002C                     DC      44
220 0 000001A0 08009999                 DC      WPR,$9999               CL0 - $9999
221 0 000001A4 08019999                 DC      WPR+1,$9999             CL1 - $9999
222 0 000001A8 08039999                 DC      WPR+3,$9999             EDG - $9999
223 0 000001AC 800000F0FFC4             DC      AMOVE,240,-60
224 0 000001B2 A00000030156             DC      APLG,3,342,-52,402,-75,301,-82
                FFCC0192FFB5
                012DFFAE
225 0 000001C2 80000156FFCA             DC      AMOVE,342,-54
226 0 000001C8 0800BBBB                 DC      WPR,$BBBB               CL0 - $BBBB
227 0 000001CC 08050000                 DC      WPR+5,$0000             PP,PZC
228 0 000001D0 0807F1F1                 DC      WPR+7,$F1F1             PE,PZ
229 0 000001D4 18000010                 DC      WPTN,16
230 0 000001D8 10040C180A28             DC      $1004,$0C18,$0A28,$0948,$0490,$0410,$0220,$0C18
                0948049004100
                02200C18
231 0 000001E8 30087C1F0220             DC      $3008,$7C1F,$0220,$0140,$0140,$0080,$0080,$0080
                0140014000800
                00800080
232 0 000001F8 C800                     DC      PAINT
233                                     *
234 0 000001FA 002C                     DC      44
235 0 000001FC 0800AAAA                 DC      WPR,$AAAA               CL0 - $AAAA
236 0 00000200 0801AAAA                 DC      WPR+1,$AAAA             CL1 - $AAAA
237 0 00000204 0803AAAA                 DC      WPR+3,$AAAA             EDG - $AAAA
238 0 00000208 800000F0FFC4             DC      AMOVE,240,-60
239 0 0000020E A00000030120             DC      APLG,3,301,-82,272,-220,104,-160
                FFAE0110FF24
                0068FF60
240 0 0000021E 800000F1FFC2             DC      AMOVE,241,-62
241 0 00000224 0800CCCC                 DC      WPR,$CCCC               CL0 - $CCCC
242 0 00000228 08050000                 DC      WPR+5,$0000             PP,PZC
243 0 0000022C 0807F2F2                 DC      WPR+7,$F2F2             PE,PZ
244 0 00000230 18000010                 DC      WPTN,16
245 0 00000234 000008180BA4             DC      $0000,$0618,$09A4,$1084,$1058,$1040,$0040,$0020
                108410581040
                00400020
246 0 00000244 07FC00200020             DC      $07FC,$0020,$0020,$0210,$0410,$0220,$01C0,$0000
                0210041002200
                01C00000
247 0 00000254 C800                     DC      PAINT
248                                     *
249 0 00000256 002C                     DC      44
250 0 00000258 0800BBBB                 DC      WPR,$BBBB               CL0 - $BBBB
251 0 0000025C 0801BBBB                 DC      WPR+1,$BBBB             CL1 - $BBBB
252 0 00000260 0803BBBB                 DC      WPR+3,$BBBB             EDG - $BBBB
253 0 00000264 80000120FFAE             DC      AMOVE,301,-82
254 0 0000026A A00000030192             DC      APLG,3,402,-75,536,-200,272,-220
                FFB50218FF38
                0110FF24
255 0 0000027A 80000192FFB3             DC      AMOVE,402,-77
256 0 00000280 0800DDDD                 DC      WPR,$DDDD               CL0 - $DDDD
257 0 00000284 08050000                 DC      WPR+5,$0000             PP,PZC
258 0 00000288 0807F1F2                 DC      WPR+7,$F1F2             PE,PZ
259 0 0000028C 18000010                 DC      WPTN,16
260 0 00000290 7E0402043FE4             DC      $7E04,$0204,$3FE4,$2224,$3FE5,$2225,$3FF5,$0215
                22243FE52225
```

Fig. 5-5 (5) (Painting the Polygons)

```
68000 MACRO ASSEMBLER 1.0          PAINT   .SA 12/13/84 10:00:28

                    3FF50215
 261 0 000002A0 1FCE12461FC4            DC      $1FCE,$1246,$1FC4,$125F,$1FE4,$0444,$0F8F,$0130
                    125F1FE40444
                    0F8F0130
 262 0 000002B0 C800                    DC      PAINT
 263                        *
 264 0 000002B2 002C                    DC      44
 265 0 000002B4 0800CCCC                DC      WPR,$CCCC               CL0 - $CCCC
 266 0 000002B8 0801CCCC                DC      WPR+1,$CCCC             CL1 - $CCCC
 267 0 000002BC 0803CCCC                DC      WPR+3,$CCCC             EDG - $CCCC
 268 0 000002C0 80000068FF60            DC      AMOVE,104,-160
 269 0 000002C8 A0000030110             DC      APLG,3,272,-220,303,-308,241,-286
                    FF24012FFECC
                    00F1FEE2
 270 0 000002D8 8000006EFF5C            DC      AMOVE,110,-164
 271 0 000002DC 0800EEEE                DC      WPR,$EEEE               CL0 - $EEEE
 272 0 000002E0 08050000                DC      WPR+5,$0000             PP,PZC
 273 0 000002E4 0807F2F3                DC      WPR+7,$F2F3             PE,PZ
 274 0 000002E8 18000010                DC      WPTN,16
 275 0 000002EC 000000001C7C            DC      $0000,$0000,$1C7C,$0410,$0210,$0210,$0110,$0090
                    041002100210
                    01100090
 276 0 000002FC 03F004100810            DC      $03F0,$0410,$0810,$0810,$0810,$0410,$03FC,$0000
                    081008100410
                    03FC0000
 277 0 0000030C C800                    DC      PAINT
 278                        *
 279 0 0000030E 002C                    DC      44
 280 0 00000310 0800DDDD                DC      WPR,$DDDD               CL0 - $DDDD
 281 0 00000314 0801DDDD                DC      WPR+1,$DDDD             CL1 - $DDDD
 282 0 00000318 0803DDDD                DC      WPR+3,$DDDD             EDG - $DDDD
 283 0 0000031C 80000110FF24            DC      AMOVE,272,-220
 284 0 00000322 A00000030218            DC      APLG,3,536,-200,402,-300,303,-308
                    FF380192FED4
                    012FFECC
 285 0 00000332 80000212FF36            DC      AMOVE,530,-202
 286 0 00000338 0800FFFF                DC      WPR,$FFFF               CL0 - $FFFF
 287 0 0000033C 08050000                DC      WPR+5,$0000             PP,PZC
 288 0 00000340 0807F2F1                DC      WPR+7,$F2F1             PE,PZ
 289 0 00000344 18000010                DC      WPTN,16
 290 0 00000348 000000802146            DC      $0000,$0080,$2146,$122C,$0C10,$00E0,$07A0,$08E8
                    122C0C1000E0
                    07A008E8
 291 0 00000358 0808081007E0            DC      $0808,$0810,$07E0,$0040,$0080,$0130,$01C0,$0000
                    004000800130
                    01C00000
 292 0 00000368 C800                    DC      PAINT
 293                        *
 294                                    END

****** TOTAL ERRORS     0--    0
```

Fig. 5-5 (6) (Painting the Polygons)

```
68000 MACRO ASSEMBLER 1.0          PAINT   .SA 12/13/84 10:00:28

SYMBOL TABLE LISTING

SYMBOL NAME      SECT   VALUE       SYMBOL NAME      SECT   VALUE

AARC                    0000B000    INIT              0     00000094
ACRTC                   00A00000    INIT1             0     000000A4
AEARC                   0000B800    INIT2             0     000000BA
AFRCT                   0000C000    INITTBL           0     00000004
AGCPY                   0000E000    MOD                     00004C00
ALINE                   00008800    ORG                     00000400
AMOVE                   00008000    PAINT                   0000C800
APLG                    0000A000    PTN                     0000D000
APLL                    00009800    RARC                    0000B400
ARCT                    00009000    RD                      00004400
CLR                     00005800    REARC                   0000BC00
CPY                     00008000    RFRCT                   0000C400
CRCL                    0000A800    RGCPY                   0000F000
CTWR              0     00000058    RLINE                   00008C00
CTWRTE            0     00000052    RMOVE                   00008400
CWR               0     00000072    RPLG                    0000A400
CWRITE            0     00000068    RPLL                    00009C00
DATA2             0     00000124    RPR                     00000C00
DD1               0     0000011A    RPTN                    00001C00
DELAY             0     00000114    RRCT                    00009400
DEMO2             0     000000EE    SCLR                    00005C00
DMOD                    00002C00    SCPY                    00007000
DOT                     0000CC00    WPR                     00000800
DRD                     00002400    WPTN                    00001800
DWT                     00002800    WT                      00004800
ELPS                    0000AC00
```

Fig. 5-5 (7) (Painting the Polygons)

```
68000 MACRO ASSEMBLER 1.0        PANDA   .SA 12/13/84 10:23:59

                                         OPT       CEX
            00A00000     ACRTC       EQU       $A00000
                         *
                         ****************************
                         *                          *
                         *   ACRTC COMMAND TABLE    *
                         *                          *
                         ****************************
                         *
            00000400     ORG         EQU       %0000010000000000
            00000800     WPR         EQU       %0000100000000000
            00000C00     RPR         EQU       %0000110000000000
            00001800     WPTN        EQU       %0001100000000000
            00001C00     RPTN        EQU       %0001110000000000
            00002400     DRD         EQU       %0010010000000000
            00002800     DWT         EQU       %0010100000000000
            00002C00     DMOD        EQU       %0010110000000000
                         *
            00004400     RD          EQU       %0100010000000000
            00004800     WT          EQU       %0100100000000000
            00004C00     MOD         EQU       %0100110000000000
            00005800     CLR         EQU       %0101100000000000
            00005C00     SCLR        EQU       %0101110000000000
            00006000     CPY         EQU       %0110000000000000
            00007000     SCPY        EQU       %0111000000000000
                         *
            00008000     AMOVE       EQU       %1000000000000000
            00008400     RMOVE       EQU       %1000010000000000
            00008800     ALINE       EQU       %1000100000000000
            00008C00     RLINE       EQU       %1000110000000000
            00009000     ARCT        EQU       %1001000000000000
            00009400     RRCT        EQU       %1001010000000000
            00009800     APLL        EQU       %1001100000000000
            00009C00     RPLL        EQU       %1001110000000000
            0000A000     APLG        EQU       %1010000000000000
            0000A400     RPLG        EQU       %1010010000000000
            0000A800     CRCL        EQU       %1010100000000000
            0000AC00     ELPS        EQU       %1010110000000000
            0000B000     AARC        EQU       %1011000000000000
            0000B400     RARC        EQU       %1011010000000000
            0000B800     AEARC       EQU       %1011100000000000
            0000BC00     REARC       EQU       %1011110000000000
            0000C000     AFRCT       EQU       %1100000000000000
            0000C400     RFRCT       EQU       %1100010000000000
            0000C800     PAINT       EQU       %1100100000000000
            0000CC00     DOT         EQU       %1100110000000000
            0000D000     PTN         EQU       %1101000000000000
            0000E000     AGCPY       EQU       %1110000000000000
            0000F000     RGCPY       EQU       %1111000000000000
                         *
0 00000000 60000092              BRA       INIT
                         *
                         ****************************
                         *                          *
                         *   ACRTC INITIALIZE DATA  *
                         *                          *
                         ****************************
                         *
```

Fig. 5-6 (1) (Panda Bear)

```
 59 0 00000004 680A          INITTBL  DC       $680A                 R82:HORIZONTAL SYNC.
 60 0 00000006 0B50                   DC       $0B50                 R84:HORIZONTAL DISPLAY
 61 0 00000008 01C0                   DC       448                   R86:VERTICAL SYNC.
 62 0 0000000A 2010                   DC       $2010                 R88:VERTICAL DISPLAY
 63 0 0000000C 0190                   DC       400                   R8A:
 64 0 0000000E 0000                   DC       0                     R8C:SPLIT SCREEN WIDTH  SP1
 65 0 00000010 0000                   DC       0                     R8E:                    SP2
 66 0 00000012 0000                   DC       0                     R90:BLINK CONTROL
 67 0 00000014 0000                   DC       0                     R92:H-WINDOW DISPLAY
 68 0 00000016 0000                   DC       0                     R94:V-WINDOW DISPLAY
 69 0 00000018 0000                   DC       0                     R96:
 70 0 0000001A 0000                   DC       0                     R98:GRAPHIC CURSOR
 71 0 0000001C 0000                   DC       0                     R9A:
 72 0 0000001E 0000                   DC       0                     R9C:
 73                          *
 74 0 00000020 0000                   DC       0                     RC0:RASTER ADDR.  SCREEN 0
 75 0 00000022 00A4                   DC       164                   RC2:MEMORY WIDTH
 76 0 00000024 0F00                   DC       $0F00                 RC4:START ADDR. H
 77 0 00000026 0000                   DC       $0000                 RC6:            L
 78                          *
 79 0 00000028 0000                   DC       0                     RC8:RASTER ADDR.  SCREEN 1
 80 0 0000002A 00A4                   DC       164                   RCA:MEMORY WIDTH
 81 0 0000002C 0F03                   DC       $0F03                 RCC:START ADDR. H
 82 0 0000002E 00C0                   DC       $00C0                 RCE:            L
 83                          *
 84 0 00000030 0000                   DC       0                     RD0:RASTER ADDR.  SCREEN 2
 85 0 00000032 01E0                   DC       480                   RD2:MEMORY WIDTH
 86 0 00000034 0F09                   DC       $0F09                 RD4:START ADDR. H
 87 0 00000036 0240                   DC       $0240                 RD6:            L
 88                          *
 89 0 00000038 0000                   DC       0                     RD8:RASTER ADDR.  SCREEN 3
 90 0 0000003A 00A4                   DC       164                   RDA:MEMORY WIDTH
 91 0 0000003C 0002                   DC       $0002                 RDC:START ADDR. H
 92 0 0000003E 0080                   DC       $0080                 RDE:            L
 93                          *
 94 0 00000040 0000                   DC       0                     RE0:CHARACTER CURSOR
 95 0 00000042 0000                   DC       0                     RE2:
 96 0 00000044 0000                   DC       0                     RE4:
 97 0 00000046 0000                   DC       0                     RE6:
 98                          *
 99 0 00000048 0000                   DC       0                     RE8:
100 0 0000004A 0000                   DC       0                     REA:ZOOM FACTER
101                         *
102 0 0000004C 0200                   DC       %0000001000000000     R02:COMMAND CONTROL
103 0 0000004E C128                   DC       %1100000100101000     R04:SYNC. CONTROL
104 0 00000050 4000                   DC       %0100000000000000     R06:DISPLAY CONTROL
105                         *
106                         ****************************
107                         *                          *
108                         *   COMMAND TABLE WRITE    *
109                         *                          *
110                         *       (A1)+ -> ACRTC     *
111                         *                          *
112                         ****************************
113                         *
114 0 00000052 48A72000     CTWRTE   MOVEM    D2,-(A7)
115 0 00000056 3419                  MOVE     (A1)+,D2              LOOP COUNTER LOAD
116                         *
```

Fig. 5-6 (2) (Panda Bear)

```
117 0 00000058 3019          CTWR    MOVE    (A1)+,D0
118 0 0000005A 6100000C              BSR     CWRITE
119 0 0000005E 51CAFFF8              DBRA    D2,CTWR
120 0 00000062 4C9F0004              MOVEM   (A7)+,D2
121 0 00000066 4E75                  RTS
122                          *
123                          ************************
124                          *                      *
125                          *     COMMAND WRITE    *
126                          *                      *
127                          *       D0 -> ACRTC    *
128                          *                      *
129                          ************************
130                          *
131 0 00000068 40E7          CWRITE  MOVE    SR,-(A7)
132 0 0000006A 46FC2700              MOVE    #$2700,SR
133 0 0000006E 48E74000              MOVEM.L D1,-(A7)
134 0 00000072 323900A00000  CWR     MOVE    ACRTC,D1
135 0 00000078 08010001              BTST    #1,D1
136 0 0000007C 67F4                  BEQ     CWR
137                          *
138 0 0000007E 33FC000000A0          MOVE    #0,ACRTC            R00 SELECT
                 0000
139 0 00000086 33C000A00002          MOVE    D0,ACRTC+2          DATA -> ACRTC
140 0 0000008C 4CDF0002              MOVEM.L (A7)+,D1
141 0 00000090 46DF                  MOVE    (A7)+,SR
142 0 00000092 4E75                  RTS
143                          *
144                          ************************
145                          *                      *
146                          *   ACRTC INITIARIZE   *
147                          *                      *
148                          ************************
149                          *
150 0 00000094 43FAFF6E      INIT    LEA     INITTBL(PC),A1
151                          *
152 0 00000098 33FC008200A0          MOVE    #$82,ACRTC          R82 SELECT
                 0000
153 0 000000A0 343C000D              MOVE    #13,D2              LOOP COUNTER -> D2
154 0 000000A4 33D900A00002  INIT1   MOVE    (A1)+,ACRTC+2       R82-R9D WRITE
155 0 000000AA 51CAFFF8              DBRA    D2,INIT1
156                          *
157 0 000000AE 33FC00C000A0          MOVE    #$C0,ACRTC          RC0 SELECT
                 0000
158 0 000000B6 343C0015              MOVE    #21,D2              LOOP COUNTER -> D2
159 0 000000BA 33D900A00002  INIT2   MOVE    (A1)+,ACRTC+2       RC0-REB WRITE
160 0 000000C0 51CAFFF8              DBRA    D2,INIT2
161                          *
162 0 000000C4 33FC000200A0          MOVE    #$02,ACRTC          R02 SELECT
                 0000
163 0 000000CC 33D900A00002          MOVE    (A1)+,ACRTC+2       R02 WRITE
164                          *
165 0 000000D2 33FC000400A0          MOVE    #$04,ACRTC          R04 SELECT
                 0000
166 0 000000DA 33D900A00002          MOVE    (A1)+,ACRTC+2       R04 WRITE
167                          *
168 0 000000E0 33FC000800A0          MOVE    #$08,ACRTC          R08 SELECT
                 0000
```

Fig. 5-6 (3) (Panda Bear)

```
68000 MACRO ASSEMBLER 1.0          PANDA   .SA 12/13/84 10:23:59

169 0 000000E8 33D900A00002          MOVE      (A1)+,ACRTC+2          R06 WRITE
170                                  *
171                                  ****************************
172                                  *                          *
173                                  *     D  E  M  O   3        *
174                                  *                          *
175                                  ****************************
176                                  *
177 0 000000EE 43FA023C      DEMO3   LEA       DATA3(PC),A1
178                                  *
179 0 000000F2 33FC000600A0          MOVE      #$06,ACRTC             R06 SELECT
                0000
180 0 000000FA 33FC000000A0          MOVE      #$0000,ACRTC+2
                0002
181                                  *
182 0 00000102 33FC00EA00A0          MOVE      #$EA,ACRTC             REA SELECT  ZOOM
                0000
183 0 0000010A 33FC100000A0          MOVE      #$1000,ACRTC+2
                0002
184                                  *
185 0 00000112 33FC000600A0          MOVE      #$06,ACRTC             R06 SELECT
                0000
186 0 0000011A 33FC400000A0          MOVE      #$4000,ACRTC+2
                0002
187                                  *
188 0 00000122 33FC00CC00A0          MOVE      #$CC,ACRTC             RCC SELECT
                0000
189 0 0000012A 33FC000300A0          MOVE      #$0003,ACRTC+2
                0002
190                                  *
191 0 00000132 6100FF1E              BSR       CTWRTE
192 0 00000136 610001C8              BSR       PDELAY
193                                  *
194 0 0000013A 43FA01F0              LEA       DATA3(PC),A1
195 0 0000013E 61000198              BSR       DTWRTE
196 0 00000142 610001BC              BSR       PDELAY
197                                  *
198 0 00000146 343C0077              MOVE      #119,D2
199 0 0000014A 610000D0      PA1     BSR       SCROOL2
200 0 0000014E 51CAFFFA              DBRA      D2,PA1
201                                  *
202 0 00000152 343C0059              MOVE      #89,D2
203 0 00000156 61000104      PA2     BSR       DOWN
204 0 0000015A 51CAFFFA              DBRA      D2,PA2
205                                  *
206 0 0000015E 33FC00EA00A0          MOVE      #$EA,ACRTC             REA SELECT
                0000
207 0 00000166 33FC000000A0          MOVE      #0,ACRTC+2
                0002
208                                  *
209 0 0000016E 33FC00CC00A0          MOVE      #$CC,ACRTC             RCC SELECT
                0000
210 0 00000176 33FC0F0300A0          MOVE      #$0F03,ACRTC+2
                0002
211                                  *
212 0 0000017E 61000180              BSR       PDELAY
213 0 00000182 6100FECE              BSR       CTWRTE
214 0 00000186 6100FECA              BSR       CTWRTE
```

Fig. 5-6 (4) (Panda Bear)

```
68000 MACRO ASSEMBLER 1.0          PANDA   .SA 12/13/84 10:23:59

215                                *
216  0 0000018A 33FC00DC00A0      MOVE     #$DC,ACRTC              RDC SELECT
                0000
217  0 00000192 33FC000300A0      MOVE     #$0003,ACRTC+2
                0002
218  0 0000019A 33FC00C000A0      MOVE     #$00C0,ACRTC+2
                0002
219                                *
220  0 000001A2 33FC009200A0      MOVE     #$92,ACRTC              R92 SELECT
                0000
221  0 000001AA 33FC191300A0      MOVE     #$1913,ACRTC+2
                0002
222  0 000001B2 33FC002000A0      MOVE     #32,ACRTC+2
                0002
223  0 000001BA 33FC019000A0      MOVE     #400,ACRTC+2
                0002
224                                *
225  0 000001C2 33FC000600A0      MOVE     #$06,ACRTC              R06 SELECT
                0000
226  0 000001CA 33FC430000A0      MOVE     #$4300,ACRTC+2
                0002
227                                *
228  0 000001D2 343C00C7          MOVE     #199,D2
229  0 000001D6 610000A8    PA5   BSR      UW
230  0 000001DA 51CAFFFA          DBRA     D2,PA5
231                                *
232  0 000001DE 343C00C7          MOVE     #199,D2
233  0 000001E2 610000D6    PA6   BSR      DW
234  0 000001E6 51CAFFFA          DBRA     D2,PA6
235                                *
236  0 000001EA 33FC00DE00A0      MOVE     #$DE,ACRTC              RDE SELECT
                0000
237  0 000001F2 33FC00C000A0      MOVE     #$00C0,ACRTC+2
                0002
238                                *
239  0 000001FA 33FC000600A0      MOVE     #$06,ACRTC              R06 SELECT
                0000
240  0 00000202 33FC400000A0      MOVE     #$4000,ACRTC+2
                0002
241                                *
242  0 0000020A 203C0007FFFF      MOVE.L   #$7FFFF,D0
243  0 00000210 048000000001 DD1  SUBI.L   #1,D0
244  0 00000216 66F8              BNE      DD1
245  0 00000218 6000FED4          BRA      DEMO3
246                               *
247                               *
248                               ********************
249                               *                  *
250                               *   UP  SCROOL     *
251                               *                  *
252                               ********************
253                               *
254  0 0000021C 40E7         SCROOL2  MOVE     SR,-(A7)
255  0 0000021E 46FC2700           MOVE     #$2700,SR
256  0 00000222 33FC00CC00A0      MOVE     #$CC,ACRTC              RCC SELECT
                0000
257  0 0000022A 303900A00002      MOVE     ACRTC+2,D0
258  0 00000230 4840              SWAP     D0
```

Fig. 5-6 (5) (Panda Bear)

```
259 0 00000232 303900A00002          MOVE    ACRTC+2,D0
260                          *
261 0 00000238 06400148              ADDI    #164*2,D0
262                          *
263 0 0000023C 610000D0      SCDLAY  BSR     RDELAY
264 0 00000240 33FC00CC00A0          MOVE    #$CC,ACRTC          RCC SELECT
                 0000
265 0 00000248 4840                  SWAP    D0
266 0 0000024A 33C000A00002          MOVE    D0,ACRTC+2
267 0 00000250 4840                  SWAP    D0
268 0 00000252 33C000A00002          MOVE    D0,ACRTC+2
269 0 00000258 46DF                  MOVE    (A7)+,SR
270 0 0000025A 4E75                  RTS
271                          *
272                          ********************
273                          *                  *
274                          *   DOWN SCROOL    *
275                          *                  *
276                          ********************
277                          *
278 0 0000025C 40E7          DOWN    MOVE    SR,-(A7)
279 0 0000025E 46FC2700              MOVE    #$2700,SR
280 0 00000262 33FC00CC00A0          MOVE    #$CC,ACRTC          RCC SELECT
                 0000
281 0 0000026A 303900A00002          MOVE    ACRTC+2,D0
282 0 00000270 4840                  SWAP    D0
283 0 00000272 303900A00002          MOVE    ACRTC+2,D0
284                          *
285 0 00000278 048000000148          SUBI.L  #164*2,D0
286 0 0000027E 60BC                  BRA     SCDLAY
287                          *
288                          ********************
289                          *                  *
290                          * WINDOW UP SCROOL *
291                          *                  *
292                          ********************
293                          *
294 0 00000280 33FC00DC00A0  UW      MOVE    #$DC,ACRTC          RDC SELECT
                 0000
295 0 00000288 303900A00002          MOVE    ACRTC+2,D0
296 0 0000028E 4840                  SWAP    D0
297 0 00000290 303900A00002          MOVE    ACRTC+2,D0
298                          *
299 0 00000296 0680000000A4          ADDI.L  #164,D0
300                          *
301 0 0000029C 61000070      UD      BSR     RDELAY
302 0 000002A0 33FC00DC00A0          MOVE    #$DC,ACRTC          RDC SELECT
                 0000
303 0 000002A8 4840                  SWAP    D0
304 0 000002AA 33C000A00002          MOVE    D0,ACRTC+2
305 0 000002B0 4840                  SWAP    D0
306 0 000002B2 33C000A00002          MOVE    D0,ACRTC+2
307 0 000002B8 4E75                  RTS
308                          *
309                          **********************
310                          *                    *
311                          * WINDOW DOWN SCROOL *
312                          *                    *
```

Fig. 5-6 (6) (Panda Bear)

```
68000 MACRO ASSEMBLER 1.0          PANDA   .SA 12/13/84 10:23:59

                                   **********************
                                   *
000002BA 33FC00DC00A0 DW         MOVE      #$DC,ACRTC              RDC SELECT
         0000
000002C2 303900A00002            MOVE      ACRTC+2,D0
000002C8 4840                    SWAP      D0
000002CA 303900A00002            MOVE      ACRTC+2,D0
                                   *
000002D0 0480000000A4            SUBI.L    #164,D0
000002D6 60C4                    BRA       UD
                                   *
                                   **********************
                                   *                    *
                                   *  DELAY - CTWRTE    *
                                   *                    *
                                   **********************
                                   *
000002D8 48A72000      DTWRTE    MOVEM     D2,-(A7)
000002DC 3419                    MOVE      (A1)+,D2
000002DE 3019          DTWR      MOVE      (A1)+,D0
000002E0 6100FD86                BSR       CWRITE
000002E4 6100000C                BSR       DDELAY
000002E8 51CAFFF4      DTEE      DBRA      D2,DTWR
000002EC 4C9F0004                MOVEM     (A7)+,D2
000002F0 4E75                    RTS
                                   *
000002F2 323C0001      DDELAY    MOVE      #1,D1
000002F6 61000016      DDEY      BSR       RDELAY
000002FA 51C9FFFA                DBRA      D1,DDEY
000002FE 4E75                    RTS
                                   *
00000300 323C0077      PDELAY    MOVE      #119,D1
00000304 61000008      PPD       BSR       RDELAY
00000308 51C9FFFA                DBRA      D1,PPD
0000030C 4E75                    RTS
                                   *
                                   **********************
                                   *                    *
                                   *   RASTER  DELAY    *
                                   *                    *
                                   **********************
                                   *
0000030E 48E78000      RDELAY    MOVEM.L   D0,-(A7)
00000312 33FC008000A0 RDEY       MOVE      #$80,ACRTC              R80 SELECT
         0000
0000031A 303900A00002            MOVE      ACRTC+2,D0
00000320 0C400191                CMPI      #401,D0
00000324 66EC                    BNE       RDEY
00000328 4CDF0001                MOVEM.L   (A7)+,D0
0000032A 4E75                    RTS
                                   *
                                   ******************
                                   *                *
                                   *   D A T A 3    *
                                   *                *
                                   ******************
                                   *
0000032C 01AD          DATA3     DC        429
```

Fig. 5-6 (7) (Panda Bear)

```
68000 MACRO ASSEMBLER 1.0          PANDA   .SA 12/13/84 10:23:58

369 0 0000032E 080C4030                    DC      WPR+$C,$4030            R/W POINTER-$40300C00
370 0 00000332 080D0C00                    DC      WPR+$D,$0C00
371 0 00000336 5800999900A3                DC      CLR,$9999,163,-800
                FCE0
372 0 0000033E 040040320C80                DC      ORG,$4032,$0C80
373 0 00000344 08050000                    DC      WPR+5,0
374 0 00000348 08060000                    DC      WPR+6,0                 PP,PZC
375 0 0000034C 08070000                    DC      WPR+7,0                 PS
376 0 00000350 0800FFFF                    DC      WPR,$FFFF               PE,PZ
377 0 00000354 0801FFFF                    DC      WPR+1,$FFFF             CL0 - $FFFF
378 0 00000358 0803FFFF                    DC      WPR+3,$FFFF             CL1 - $FFFF
379                                                                        EDG - $FFFF
380 0 0000035C 80000086FF8D        *       DC      AMOVE,134,-115
381 0 00000362 B000007CFFA9                DC      AARC,124,-87,143,-64
                008FFFC0
382 0 0000036C B000004EFFA1                DC      AARC,78,-95,130,-45
                0082FFD3
383 0 00000376 B000006AFFBA                DC      AARC,106,-70,91,-39
                005BFFD9
384 0 00000380 B0000074FF93                DC      AARC,116,-109,70,-52
                0046FFCC
385 0 0000038A B000004FFFB0                DC      AARC,79,-80,74,-109
                004AFF93
386 0 00000394 B0000730000                 DC      AARC,115,0,134,-115
                0086FF8D
387 0 0000039E 8000005AFFB0                DC      AMOVE,90,-80
388 0 000003A4 C800                        DC      PAINT
389 0 000003A8 08000000                    DC      WPR,$0000               CL0 - $0000
390 0 000003AA 08010000                    DC      WPR+1,$0000             CL1 - $0000
391 0 000003AE 08030000                    DC      WPR+3,$0000             EDG - $0000
392 0 000003B2 8000055FFD5                 DC      AMOVE,85,-43
393 0 000003B8 B0000051FFE2                DC      AARC,81,-30,75,-19
                004BFFED
394 0 000003C2 B0000062FFCC                DC      AARC,98,-52,80,-40
                003CFFD8
395 0 000003CC B0000046FFD8                DC      AARC,70,-40,75,-48
                004BFFD0
396 0 000003D8 88000045FFCA                DC      ALINE,69,-54
397 0 000003DC B0000048FFC5                DC      AARC,72,-59,69,-65
                0045FFBF
398 0 000003E6 B0000049FFB9                DC      AARC,73,-71,73,-78
                0049FFB2
399 0 000003F0 B000004DFFB9                DC      AARC,77,-71,80,-86
                0050FFBE
400 0 000003FA B0000051FFCD                DC      AARC,81,-51,85,-56
                005FFFC6
401 0 00000404 B0000058FFCD                DC      AARC,88,-51,85,-43
                0055FFD5
402 0 0000040E 80000054FFC8                DC      AMOVE,84,-55
403 0 00000414 A8000005                    DC      CRCL,5
404 0 00000418 80000055FFC8                DC      AMOVE,85,-56
405 0 0000041E A8000003                    DC      CRCL,3
406 0 00000422 C800                        DC      PAINT
407 0 00000424 80000048FFBA                DC      AMOVE,72,-70
408 0 0000042A A8000002                    DC      CRCL,2
409 0 0000042E 80000050FFE7                DC      AMOVE,80,-25
410 0 00000434 C800                        DC      PAINT
411 0 00000438 8000006EFFD3                DC      AMOVE,110,-45
```

Fig. 5-6 (8) (Panda Bear)

```
68000 MACRO ASSEMBLER 1.0          PANDA   .SA 12/13/84 10:23:59

412 0 0000043C B0000071FFC2          DC      AARC,113,-62,110,-78
                 006EFFB2
413 0 00000446 B0000071FFC2          DC      AARC,113,-62,129,-72
                 0081FFB8
414 0 00000450 B0000067FFBB          DC      AARC,103,-69,110,-45
                 006EFFD3
415 0 0000045A 8000006CFFC5          DC      AMOVE,108,-59
416 0 00000460 A8000005              DC      CRCL,5
417 0 00000464 8000006BFFC4          DC      AMOVE,107,-60
418 0 0000046A A8000003              DC      CRCL,3
419 0 0000046E C800                  DC      PAINT
420 0 00000470 8000006EFFCE          DC      AMOVE,110,-50
421 0 00000478 C800                  DC      PAINT
422 0 00000478 80000082FFD3          DC      AMOVE,130,-45
423 0 0000047E B00000AAFFE2          DC      AARC,170,-30,144,-64
                 0090FFC0
424 0 00000488 B000009BFFCA          DC      AARC,155,-54,189,-51
                 00A9FFCD
425 0 00000492 B0000075FFBF          DC      AARC,117,-65,161,-32
                 00A1FFE0
426 0 0000049C B0000093FFD7          DC      AARC,147,-41,130,-45
                 0082FFD3
427 0 000004A6 80000096FFD8          DC      AMOVE,150,-40
428 0 000004AC C800                  DC      PAINT
429 0 000004AE 8000003AFFA5          DC      AMOVE,58,-91
430 0 000004B4 8800003CFFA7          DC      ALINE,60,-89
431 0 000004BA B0000044FFA6          DC      AARC,68,-90,74,-95
                 004AFFA1
432 0 000004C4 B000005AFFAB          DC      AARC,90,-85,105,-95
                 0069FFA1
433 0 000004CE 8800006CFFA0          DC      ALINE,108,-96
434 0 000004D4 80000086FF81          DC      AMOVE,134,-127
435 0 000004DA B0000083FF7D          DC      AARC,131,-131,126,-130
                 007EFF7E
436 0 000004E4 B00000BCFF81          DC      AARC,188,-159,120,-160
                 0078FF80
437 0 000004EE B0000071FF8D          DC      AARC,113,-163,109,-157
                 006DFF83
438 0 000004F8 B0000080FF8E          DC      AARC,96,-162,86,-152
                 0056FF88
439 0 00000502 B0000048FF83          DC      AARC,72,-157,57,-152
                 0039FF88
440 0 0000050C B0000060FF82          DC      AARC,96,-174,52,-172
                 0034FF54
441 0 00000516 B0000048FF82          DC      AARC,72,-174,85,-190
                 0055FF42
442 0 00000520 B0000062FF4B          DC      AARC,98,-181,105,-194
                 0069FF3E
443 0 0000052A B0000071FF51          DC      AARC,113,-175,130,-185
                 0082FF47
444 0 00000534 B000008BFF4F          DC      AARC,139,-177,141,-168
                 008DFF5A
445 0 0000053E B0000084FF62          DC      AARC,132,-158,143,-158
                 008FFF64
446 0 00000548 B000004BFF5E          DC      AARC,75,-162,127,-126
                 007FFF82
447 0 00000552 CC00                  DC      DOT
448 0 00000554 80000082FF74          DC      AMOVE,130,-140
```

Fig. 5-6 (9) (Panda Bear)

```
68000 MACRO ASSEMBLER 1.0          PANDA   .SA 12/13/84 10:23:59

448 0 0000055A C800                    DC      PAINT
450 0 0000055C 80000041FF8F            DC      AMOVE,65,-113
451 0 00000562 B0000052FFA1            DC      AARC,82,-95,82,-119
                0052FF89
452 0 0000056C B000006BFF8F            DC      AARC,107,-145,71,-137
                0047FF77
453 0 00000576 80000041FF8F            DC      AMOVE,65,-113
454 0 0000057C B000003AFF91            DC      AARC,58,-111,59,-103
                003BFF99
455 0 00000586 B0000035FF99            DC      AARC,53,-103,47,-105
                002FFF97
456 0 00000590 B000002DFF92            DC      AARC,45,-110,48,-115
                0030FF8D
457 0 0000059A B0000031FF87            DC      AARC,49,-121,54,-123
                0036FF85
458 0 000005A4 88000049FF77            DC      ALINE,73,-137
459 0 000005AA 80000035FF9C            DC      AMOVE,53,-100
460 0 000005B0 C800                    DC      PAINT
461 0 000005B2 0800FFFF                DC      WPR+0,$FFFF             CL0 - $FFFF
462 0 000005B6 0801FFFF                DC      WPR+1,$FFFF             CL1 - $FFFF
463 0 000005BA 08039999                DC      WPR+3,$9999             EDG - $9999
464 0 000005BE 8000005BFF8D            DC      AMOVE,91,-115
465 0 000005C4 B000006CFF6E            DC      AARC,108,-146,72,-143
                0048FF71
466 0 000005CE 80000086FF82            DC      AMOVE,134,-126
467 0 000005D4 B000004BFF5F            DC      AARC,75,-161,125,-114
                007DFF8E
468 0 000005DE 8000006EFF88            DC      AMOVE,110,-120
469 0 000005E4 C900                    DC      PAINT+$100
470                             *
471 0 000005E8 80000083FF45            DC      AMOVE,131,-187
472 0 000005EC B0000078FF5B            DC      AARC,120,-165,140,-167
                008CFF59
473 0 000005F6 80000083FF45            DC      AMOVE,131,-187
474 0 000005FC B000008BFF4D            DC      AARC,139,-179,143,-168
                008FFF5A
475 0 00000606 0803FFFF                DC      WPR+3,$FFFF             EDG - $FFFF
476 0 0000060A 80000091FF51            DC      AMOVE,145,-175
477 0 00000610 C800                    DC      PAINT
478 0 00000612 0800CCCC                DC      WPR+0,$CCCC             CL0 - $CCCC
479 0 00000618 0801CCCC                DC      WPR+1,$CCCC             CL1 - $CCCC
480 0 0000061A 8000007AFF5B            DC      AMOVE,122,-165
481 0 00000620 B000008BFF62            DC      AARC,139,-158,142,-165
                008EFF5B
482 0 0000062A 8000008AFF58            DC      AMOVE,138,-168
483 0 00000630 8000087FF5C             DC      ALINE,135,-164
484 0 00000638 80000082FF58            DC      AMOVE,130,-170
485 0 0000063C 8800082FF5C             DC      ALINE,130,-164
486 0 00000642 8000005AFF8A            DC      AMOVE,90,-150
487 0 00000648 8800005CFF80            DC      ALINE,92,-160
488 0 0000064E 80000085FF8B            DC      AMOVE,101,-149
489 0 00000654 88000068FF84            DC      ALINE,102,-156
490 0 0000065A 8000004BFF72            DC      AMOVE,75,-142
491 0 00000660 8800004AFF68            DC      ALINE,74,-152
492 0 00000668 8000003CFF8D            DC      AMOVE,60,-147
493 0 0000066C 8800003EFF65            DC      ALINE,62,-155
494 0 00000672 08000000                DC      WPR+0,$0000             CL0 - $0000
495 0 00000676 08010000                DC      WPR+1,$0000             CL1 - $0000
```

Fig. 5-6 (10) (Panda Bear)

```
68000 MACRO ASSEMBLER 1.0          PANDA   .SA 12/13/84 10:23:58

 496 0 0000067A 800000SAFF8E            DC      AMOVE,80,-114
 497 0 00000680 B000007300000           DC      AARC,115,0,109,-116
                006DFF8C
 498                            *
 499 0 0000068A 0017                    DC      23
 500 0 0000068C 800000B4FF3B            DC      AMOVE,180,-197
 501 0 00000692 E00000B4FF3B            DC      AGCPY,180,-197,-150,181
                FF8A00B5
 502 0 0000069C 8000001EFE85            DC      AMOVE,30,-379
 503 0 000006A2 E00000B4FF3B            DC      AGCPY,180,-197,150,181
                009800B5
 504 0 000006AC 800000B4FE85            DC      AMOVE,180,-379
 505 0 000006B2 E000001EFF3B            DC      AGCPY,30,-197,150,181
                009800B5
 506                            *
 507 0 000006BC 0007                    DC      7
 508 0 000006BE 80000154FFF0            DC      AMOVE,340,-16
 509 0 000006C4 E1000154FE85            DC      AGCPY+$100,340,-379,-320,362
                FEC0016A
 510                            *
 511                                    END

****** TOTAL ERRORS     0--    0
```

```
68000 MACRO ASSEMBLER 1.0          PANDA   .SA 12/13/84 10:23:59

SYMBOL TABLE LISTING

SYMBOL NAME      SECT   VALUE       SYMBOL NAME      SECT   VALUE

AARC                    0000B000    INITTBL           0     00000004
ACRTC                   00A00000    MOD                     00004C00
AEARC                   0000B800    ORG                     00000400
AFRCT                   0000C000    PA1               0     0000014A
AGCPY                   0000E000    PA2               0     00000158
ALINE                   00008800    PA5               0     000001D6
AMOVE                   00008000    PA6               0     000001E2
APLG                    0000A000    PAINT                   0000C800
APLL                    00009800    PDELAY            0     00000300
ARCT                    00009000    PPD               0     00000304
CLR                     00005800    PTN                     0000D000
CPY                     00006000    RARC                    0000B400
CRCL                    0000A800    RD                      00004400
CTWR              0     00000058    RDELAY            0     0000030E
CTWRTE            0     00000052    RDEY              0     00000312
CWR               0     00000072    REARC                   0000BC00
CWRITE            0     00000068    RFRCT                   0000C400
DATA3             0     0000032C    RGCPY                   0000F000
DD1               0     00000210    RLINE                   00008C00
DDELAY            0     000002F2    RMOVE                   00008400
DDEY              0     000002F6    RPLG                    0000A400
DEMO3             0     000000EE    RPLL                    00009C00
DMOD                    00002C00    RPR                     00000C00
DOT                     0000CC00    RPTN                    00001C00
DOWN              0     0000025C    RRCT                    00009400
DRD                     00002400    SCDLAY            0     0000023C
DTEE              0     000002E8    SCLR                    0000SC00
DTWR              0     000002DE    SCPY                    00007000
DTWRTE            0     000002D8    SCROOL2           0     0000021C
DW                0     000002BA    UD                0     0000029C
DWT                     00002800    UW                0     00000280
ELPS                    0000AC00    WPR                     00000800
INIT              0     00000094    WPTN                    00001800
INIT1             0     000000A4    WT                      00004800
INIT2             0     000000BA
```

Fig. 5-6 (11) (Panda Bear)

HITACHI AMERICA, LTD.
SEMICONDUCTOR AND IC DIVISION

HEADQUARTERS

Hitachi, Ltd.
New Marunouchi Bldg., 5-1,
Marunouchi 1-chome
Chiyoda-ku, Tokyo 100, Japan
Tel: Tokyo (03) 212-1111
Telex: J22395, J22432, J24491,
J26375 (HITACHY)
Cable: HITACHY TOKYO

U.S. SALES OFFICE

Hitachi America, Ltd.
Semiconductor and IC Division
2210 O'Toole Avenue
San Jose, CA 95131
Tel: 408-435-8300
Telex: 17-1581
Twx: 910-338-2103
Fax: 408-435-2748
Fax: 408-435-2749
Fax: 408-435-2782

REGIONAL OFFICES

NORTHEAST REGION

Hitachi America, Ltd.
5 Burlington Woods Drive
Burlington, MA 01803
617/229-2150

SOUTH CENTRAL REGION

Hitachi America, Ltd.
Two Lincoln Centre, Suite 865
5420 LBJ Freeway
Dallas, TX 75240
214/991-4510

NORTH CENTRAL REGION

Hitachi America, Ltd.
500 Park Blvd., Suite 415
Itasca, IL 60143
312/773-4864

NORTHWEST REGION

Hitachi America, Ltd.
2210 O'Toole Avenue
San Jose, CA 95131
408/435-2200

SOUTHWEST REGION

Hitachi America, Ltd.
21600 Oxnard St., Suite 600
Woodland Hills, CA 91367
818/704-6500

SOUTHEAST REGION

Hitachi America, Ltd.
4901 N.W. 17th Way, Suite 302
Fort Lauderdale, FL 33309
305/491-6154

DISTRICT OFFICES

- Hitachi America, Ltd.
  1700 Galloping Hill Rd.
  Kenilworth, NJ 07033
  201/245-6400
- Hitachi America, Ltd.
  3500 W. 80th Street, Suite 660
  Bloomington, MN 55431
  612/831-0408
- Hitachi America, Ltd.
  80 Washington St., Suite 302
  Poughkeepsie, NY 12601
  914/485-3400
- Hitachi America, Ltd.
  6 Parklane Blvd., #558
  Dearborn, MI 48126
  313/271-4410
- Hitachi America, Ltd.
  6161 Savoy Dr., Suite 850
  Houston, TX 77036
  713/974-0534
- Hitachi America, Ltd.
  5775 Peachtree-Dunwoody Rd.
  Suite 270C
  Atlanta, GA 30342
  404/843-3445
- Hitachi America, Ltd.
  18300 Von Karman Avenue, Suite 730
  Irvine, CA 92715
  714/553-8500
- Hitachi (Canadian) Ltd.
  2625 Queensview Dr.
  Ottawa, Ontario, Canada K2A 3Y4
  613/596-2777

HITACHI

We make things possible

Hitachi America, Ltd.
Semiconductor and IC Division
2210 O'Toole Avenue, San Jose, CA 95131
1-408-435-8300

10M

Printed in U.S.A.