HITACHI MICROCOMPUTER SYSTEM

HD63484 ACRTC

Advanced CRT Controller

APPLICATION NOTE II

—CIRCUIT & SOFTWARE—

HITACHI

680-3-07

July 1985 680-3-07

Examples of circuit and examples of characteristics described in this manual are designed to explain typical application examples of the Hitachi Semiconductors.

Pay attention to the following points in using this manual.

1. The contents of this manual may be changed without prior notice.

2. No part or all of this manual may be reproduced or republished in any form without prior permission of the company.

3. The company is not responsible for any damage that may be caused by an accident on the users' side.

4. This manual does not provide any guarantee of implementation of industrial ownership on other right or approval of implementation right.

5. Hitachi corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Hitachi product. No other circuit patent licenses are implied.

Foreword

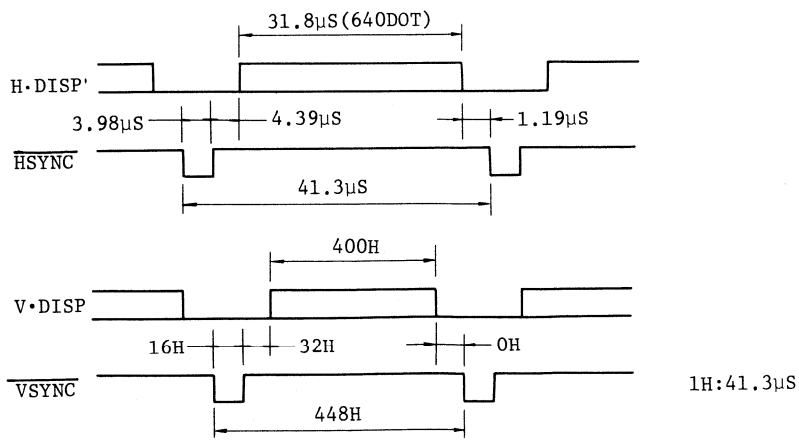
This manual is intended to show some examples in designing graphic/character display system equipment and developing software using the ACRTC. A normal CRT copable of displaying of 15 colors, 640 × 400 raster lines in a non-interlace method is employed as the target display device to show the application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

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

1. High-speed graphic drawing and X-Y Coordinate.
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.

* ACRTC : Advanced CRT Controller



CRT Display Timing

TABLE OF CONTENTS

VOLUME I

HARDWARE

VOLUME II

SOFTWARE

VOLUME I
HARDWARE

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 relation ship 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.

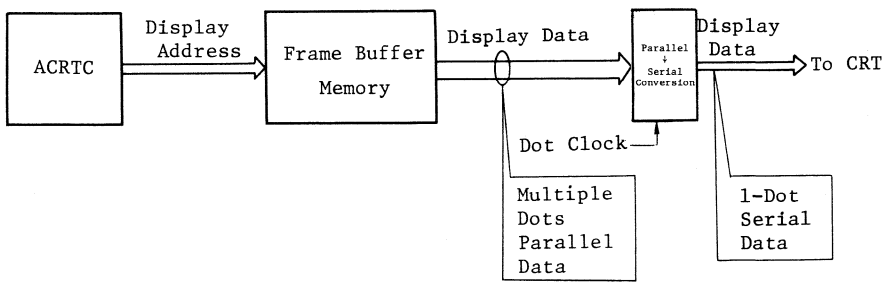


Fig. 1-1 Displaying Memory Data

The dot clock frequency is decided by the number of dot 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 (31.8μs/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 mode : 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.

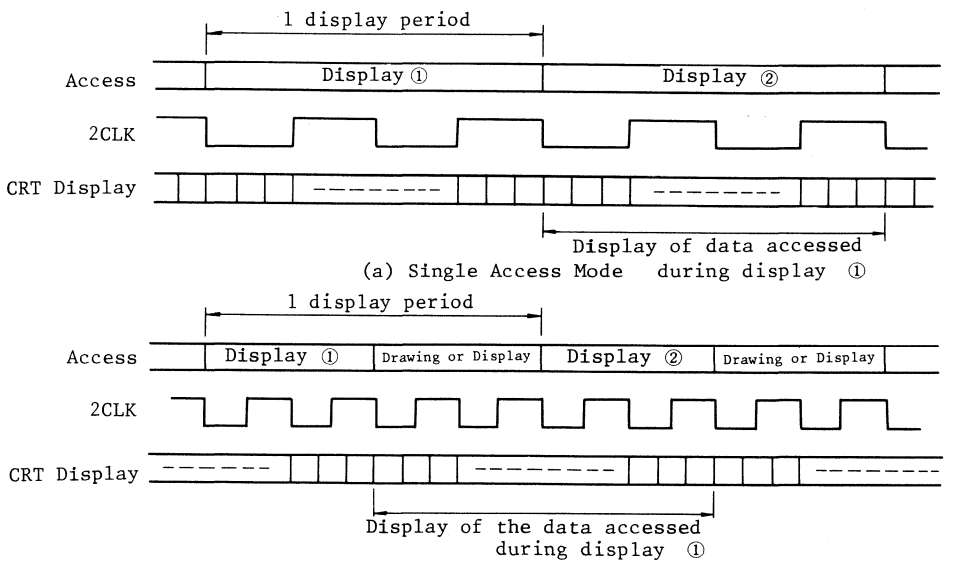


Fig. 1-2 Relationship Between Memory Access Mode and 2CLK

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 shifted quantity of the parallel/serial converter.

2CLK, which is the basic clock for the memory access, is generated by diving 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 397.5ns or 198.76ns, 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 dividing the clock frequency is indicated in table 1-1. When reading 32 dots, 2CLK remains 397.5 ns even in the dual access mode.

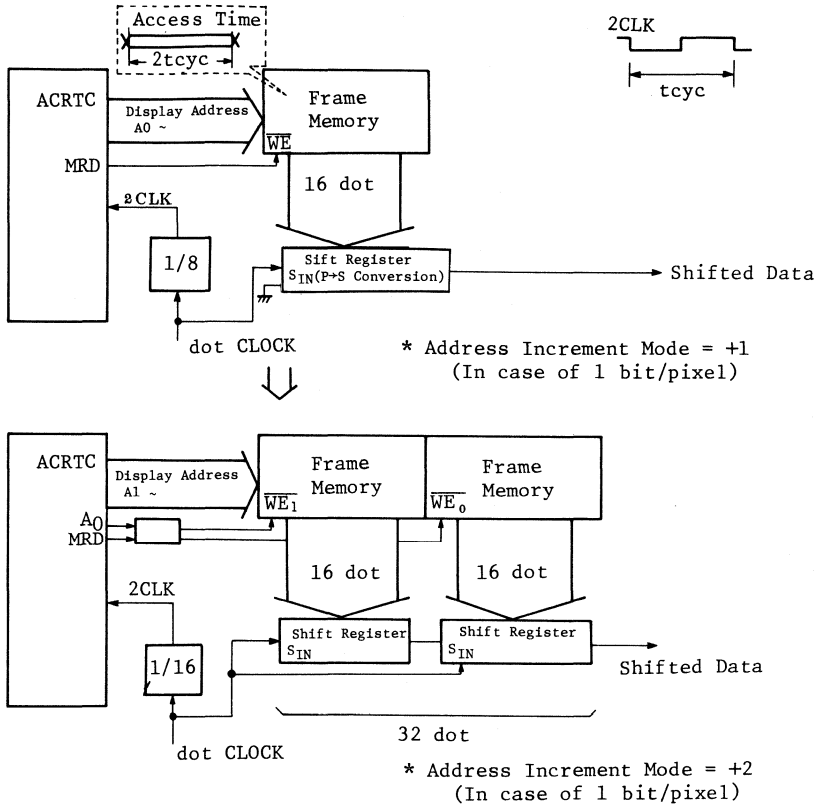


Fig. 1-3 Relation between 2CLK and the Read Data

Table 1-1 Generating 2CLK

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

Note) It can't apply to the ACRTC R Mask version.

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

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

In the dual access mode,

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

In the single access mode,

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

The number of data 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 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.

** cannot be applied for HD63484 R mask version.

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.

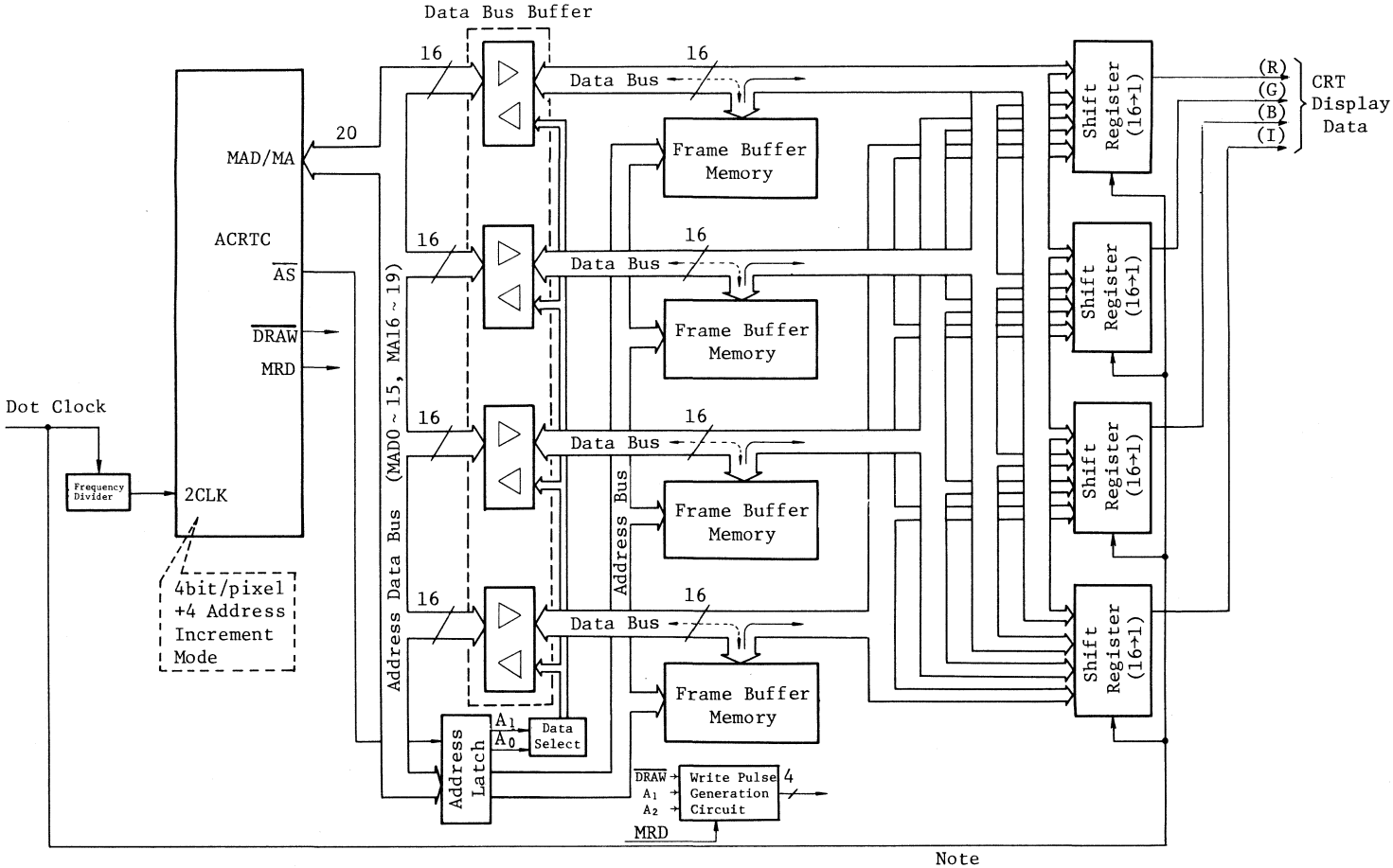


Fig. 1-4 Graphic Display Block Diagram

〔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 ∿: 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.

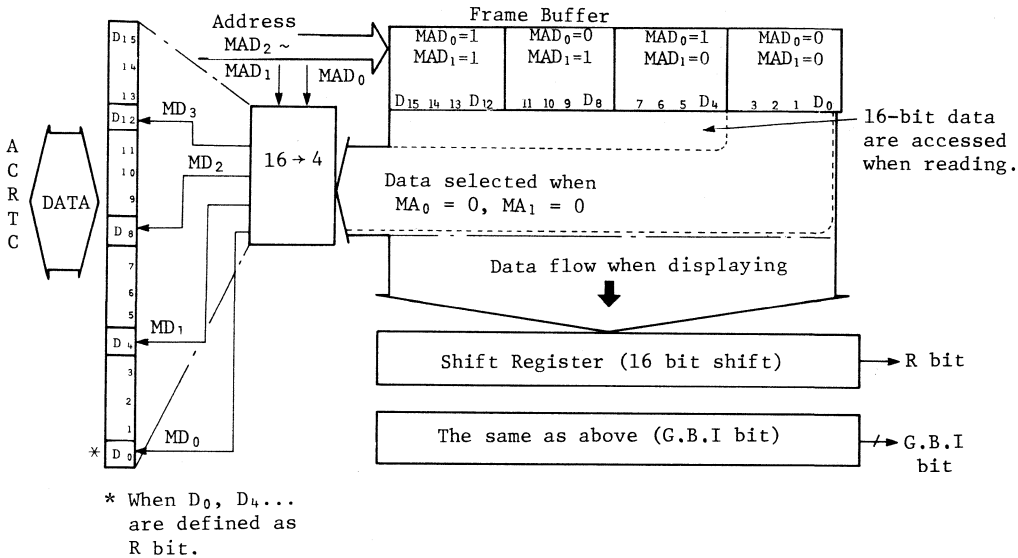


Fig. 1-5 Relation between ACRTC and Pixels (Frame Buffer Data)

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 × 400 rasters, 15 colors	
2	Dot clock	20.126 MHz	
3	2CLK	5.03 MHz	
4	ACRTC type No.	HD63484-6	
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	HD68000	
10	Memory type No.	HM48416-12 × 16 (128 KB)	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 : 16dots x 16rasters (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	HD68450 × 1 (8 MHz) 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.	

Note) Command DMA mode can't be used for the HD63484 R and S Mask Version.

2. INTERFACE WITH 16-BIT MPU

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

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

In this case, $\overline{CS}$ of the ACRTC is acquired by decoding the HD68000'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 dhows the HD68000 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 T3 cycle to inform the MPU of data output. The HD68000 detects $\overline{DTACK}$ being asserted in S4 cycle and negates $\overline{US}$, $\overline{UDS}$ and $\overline{LDS}$. If $\overline{DTACK}$ is not acknowledged in S4, the HD68000 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 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.

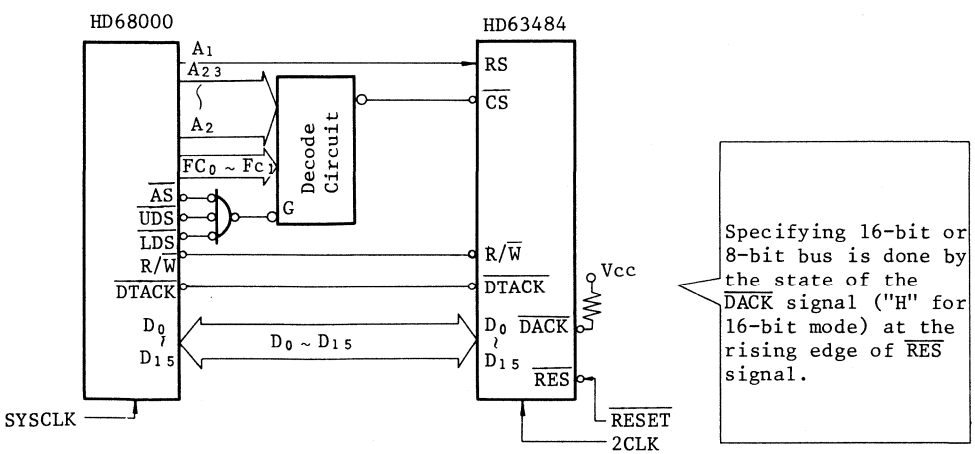


Fig. 2-1 Bus Connection Example with the HD68000 (16 bit Bus)

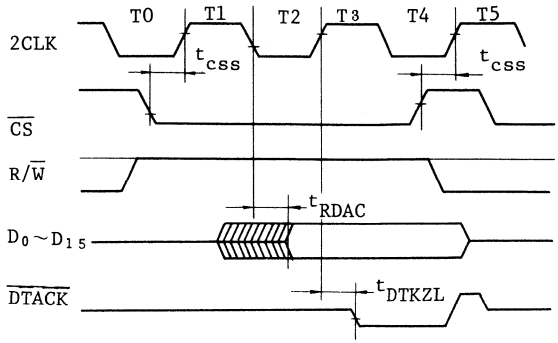


Fig. 2-2 ACRTC Read Cycle Timing: 16-bit Asynchronous Bus (MPU←ACRTC)

Note 1) When deciding the timing of $\overline{BERR}$ signal generation circuit, the phase difference between the HD68000 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 the $\overline{CS}$ assertion during interrupt acknowledge cycle.

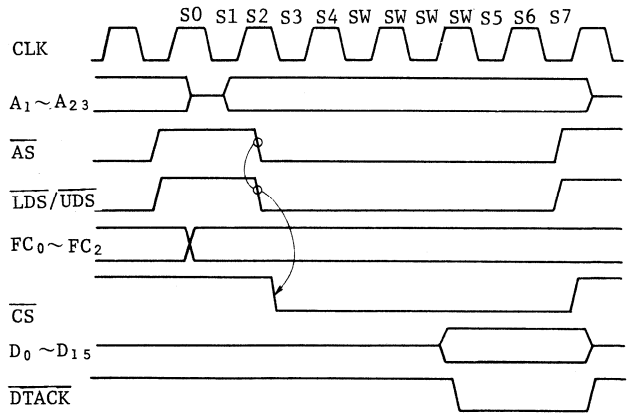


Fig. 2-3 HD68000 Read Cycle Timing

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 write cycle timing. The HD68000 asserts $\overline{UDS}$ and $\overline{LDS}$ in S4 cycle. After receiving $\overline{CS}$, ACRTC latches the 16-bit data and asserts $\overline{DTACK}$. The HD68000 detects $\overline{DTACK}$, terminates data send, and negates the bus control signals.

In the MPU mode, wait cycle will be inserted if the 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 are decreased.

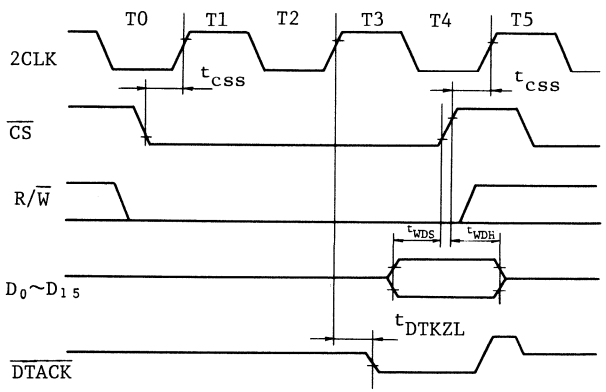


Fig. 2-4 ACRTC Write Cycle Timing: 16-bit Asynchronous Bus (MPU→ACRTC)

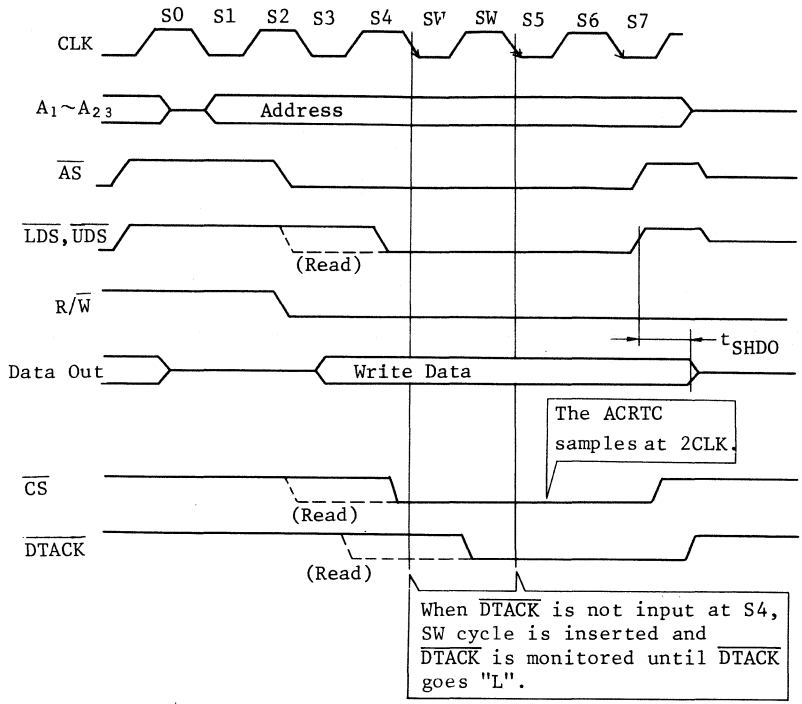


Fig. 2-5 HD68000 Write Timing

Note) In the HD68000 interface, The $\overline{CS}$ should not be generated only by decoding the address. 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.

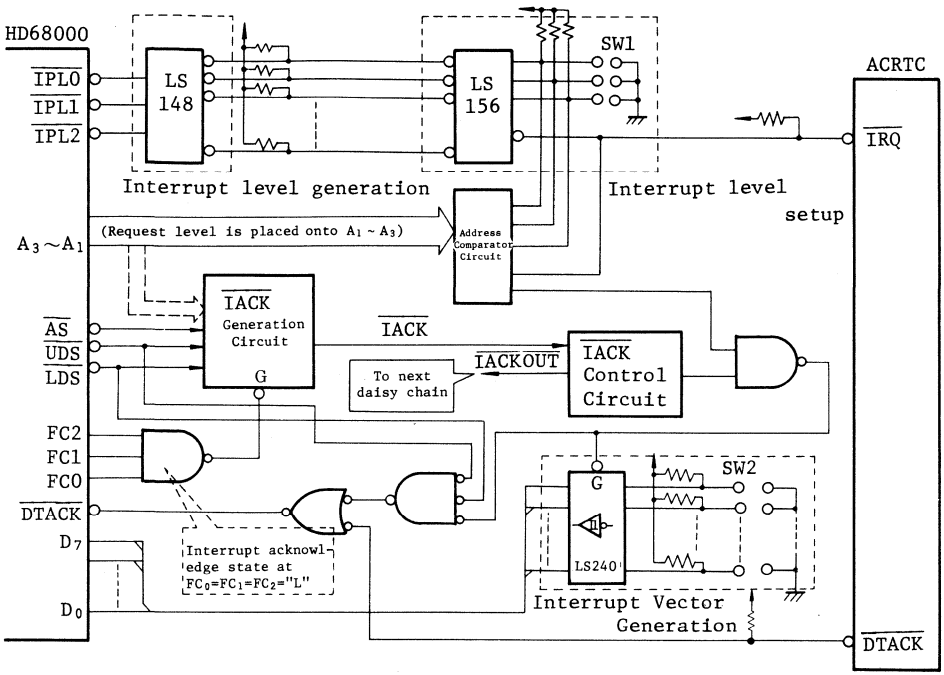


Fig. 2-6 The Example of Interrupt Signal Generation Circuit

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 by $\overline{IPL0}$, $\overline{IPL1}$, and $\overline{IPL2}$ lines. After completing the current instruction execution, the HD68000 performs an interrupt acknowledge sequence. At this time, the interrupt level is output onto A1 through A3 by HD68000. 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 to generate the interrupt vector, and $\overline{DTACK}$ signal.

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

An 8 bit vector is placed on D0 through D7 in the interrupt acknowledge cycle, but signals D8 through D15 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 asserts $\overline{UDS}$ during this cycle.

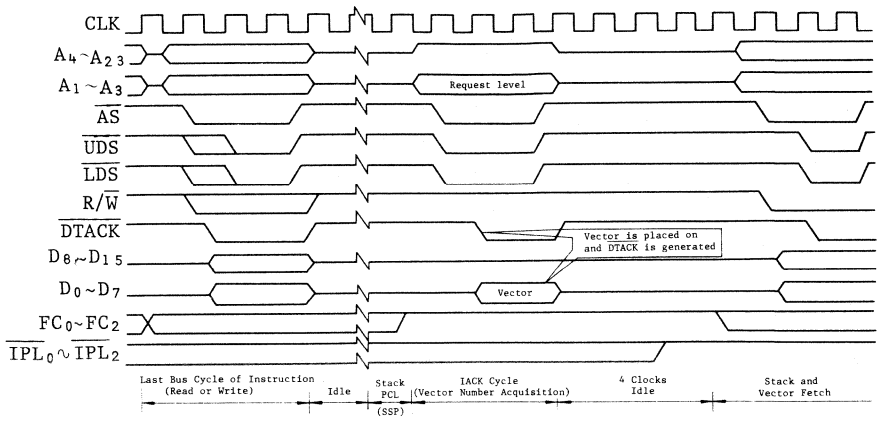


Fig. 2-7 Interrupt Acknowledge Timing

2.2 Connection to the HD68450: 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. Both modes are realized with the same hardware, and either can be selected by only setting the resisters 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 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 reads the ACRTC	MPU ← ACRTC	MPU → ACRTC	$R/\overline{W}\cdot\overline{CS}$ (ACRTC)
2	MPU	HD68000 writes the ACRTC	MPU → ACRTC	MPU → ACRTC	$\overline{R/\overline{W}\cdot\overline{CS}}$
3	MPU	Interruption acknowledge cycle (response from HD68000)	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.

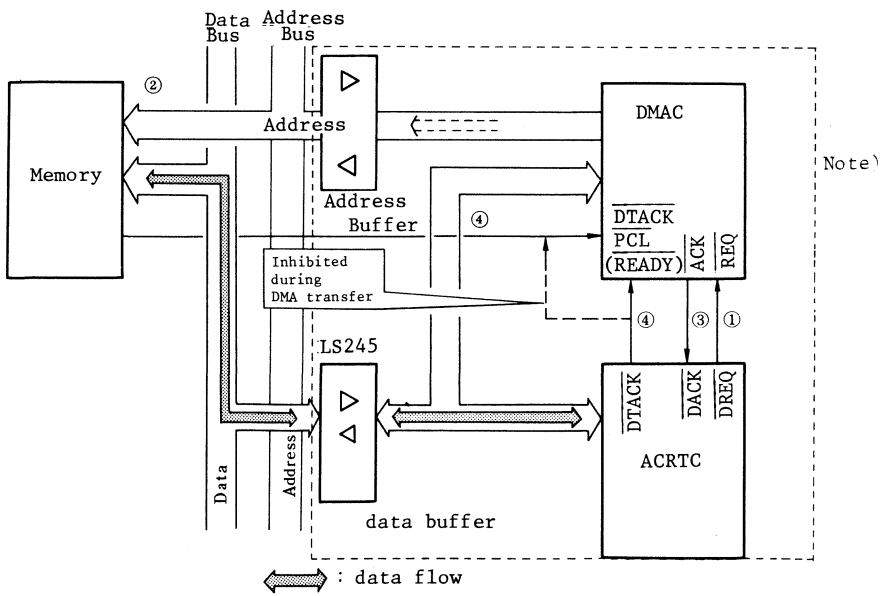


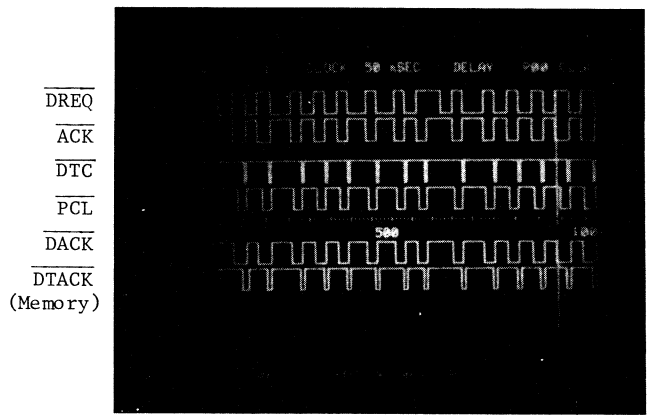
Fig. 2-8 DMA Data Transfer

Basic sequence

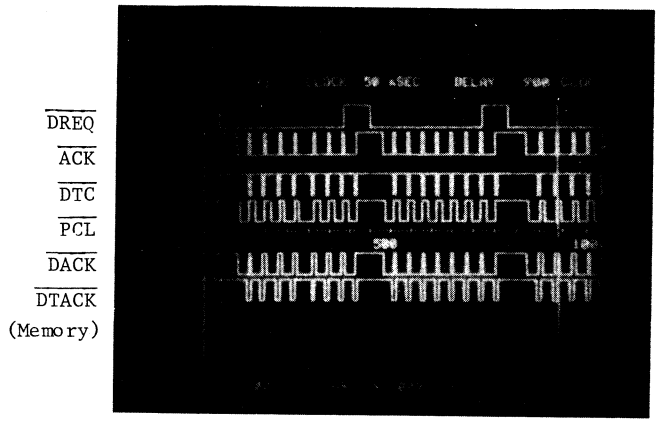
1 Transfer request is generated by the ACRTC.
2 DMAC acquires bus mastership and addresses the main memory.
3 $\overline{ACK}$ is output by the DMAC to access the ACRTC.
4 Memory returns $\overline{DTACK}$, and ACRTC applies $\overline{READY}$ ($\overline{DTACK}$) to $\overline{PCL}$ pin of the DMAC 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



(b) Burst Mode

Picture 2-1 DMA Transfer Timing

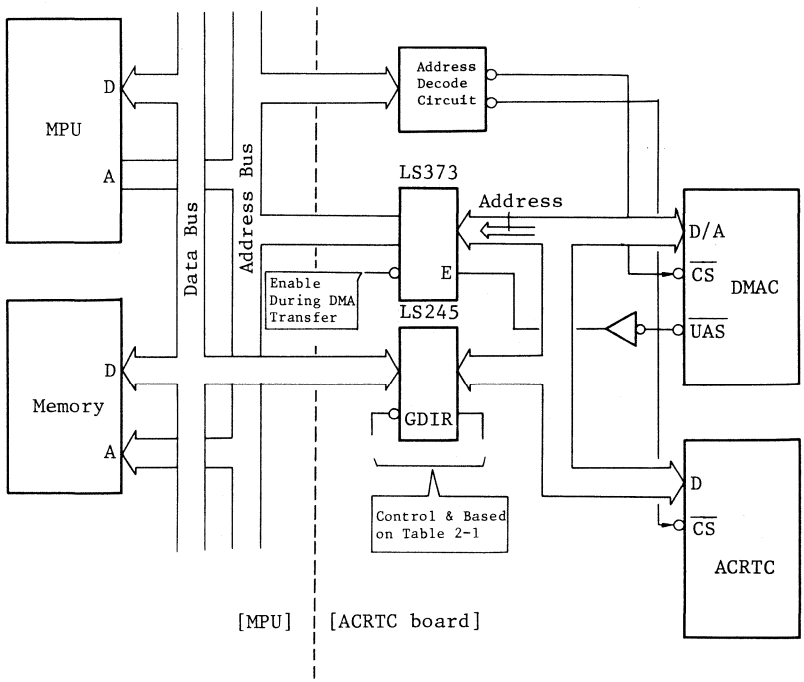


Fig. 2-9 Block Diagram of Bus Control in the DMA

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 address (A8 through A23: 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.

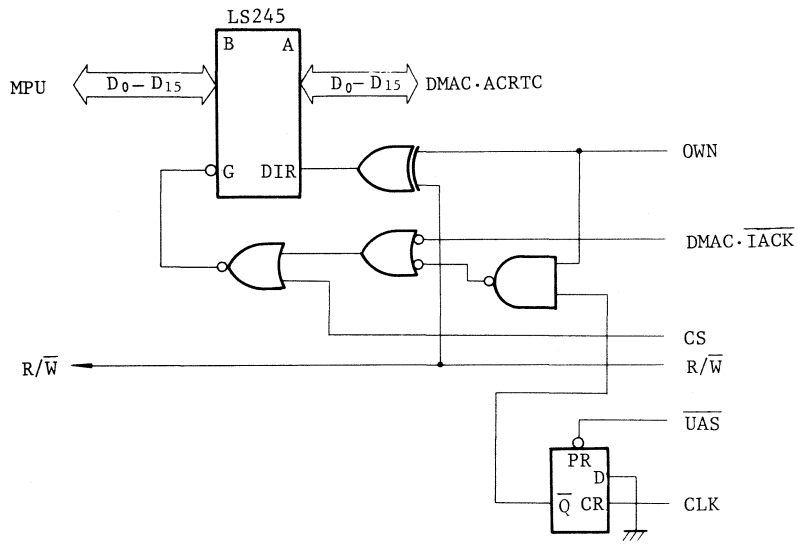


Fig. 2-10 The Example of Bus Control Circuit

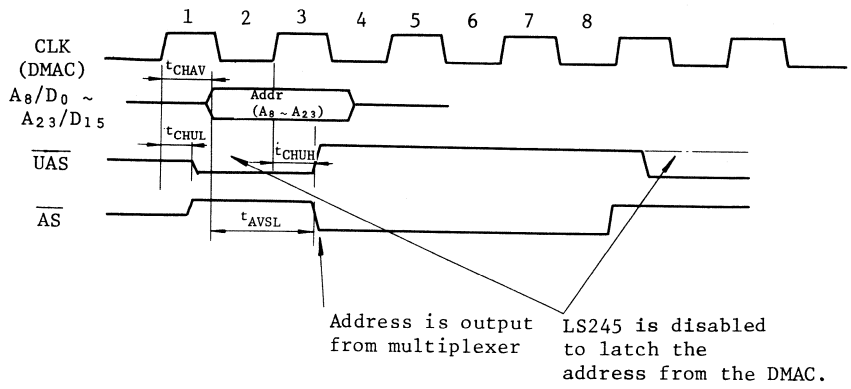


Fig. 2-11 Address Control Timing of the DMAC

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.

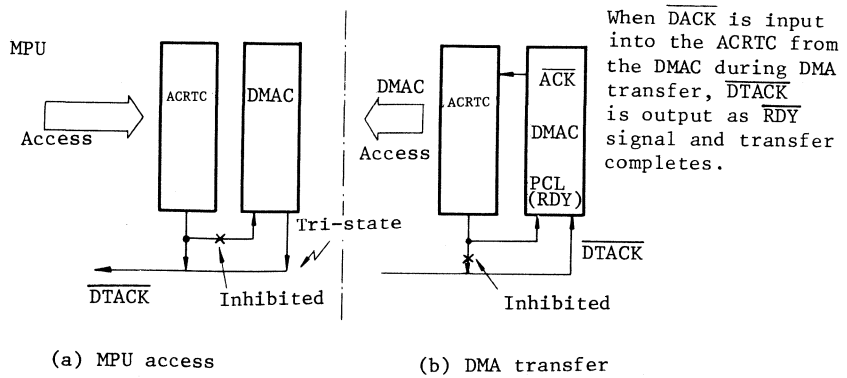


Fig. 2-12 Switching of the $\overline{DACK}$ signal

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 (tDWDH) 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.

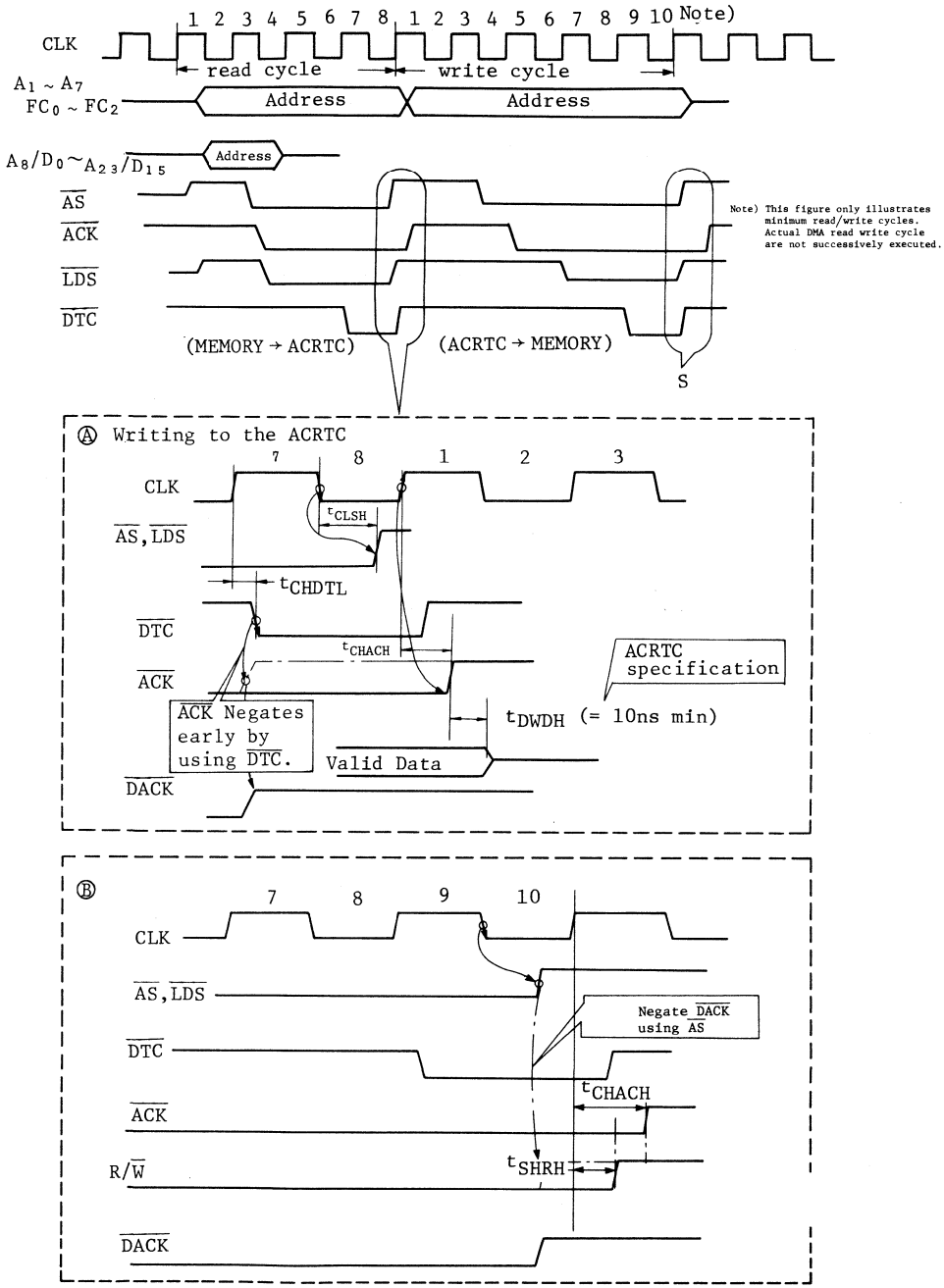


Fig. 2-13 DMAC Timing

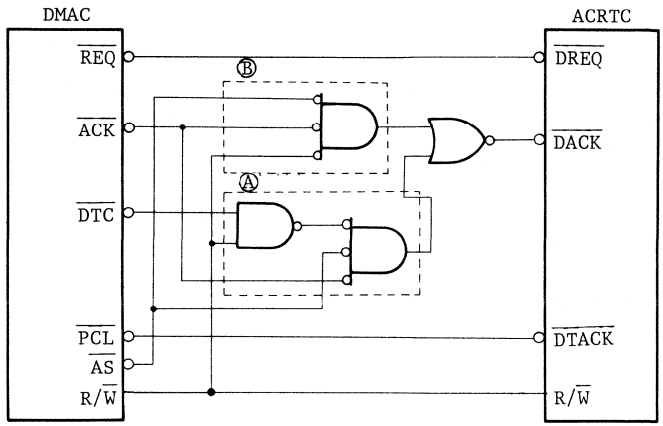


Fig. 2-14 Connection diagram for the DMAC and the ACRTC

3. INTERFACE WITH 8-BIT MPU

3.1 Connection to HD6809 (8 Bit Synchronous Bus)

When the $\overline{DACK}$ singleis "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 D0 through D7 are used, and high-order bytes D8 through D15 should not be connected. (ACRTC output "High" level on D8 ∿ D15.)

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 A1 through A15 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.

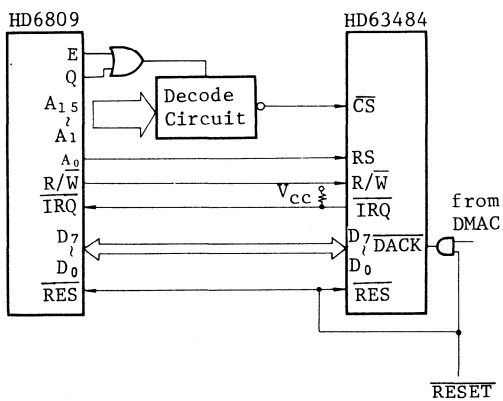


Fig. 3-1 ACRTC Bus Connection to HD6809

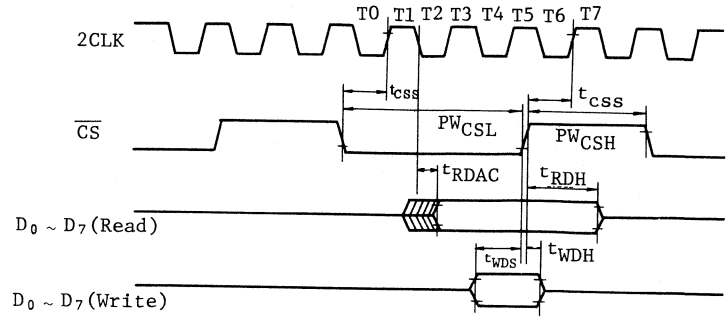


Fig. 3-2 MPU Read/Write Timing : 8-bit Synchronous Bus (MPU↔ACRTC)

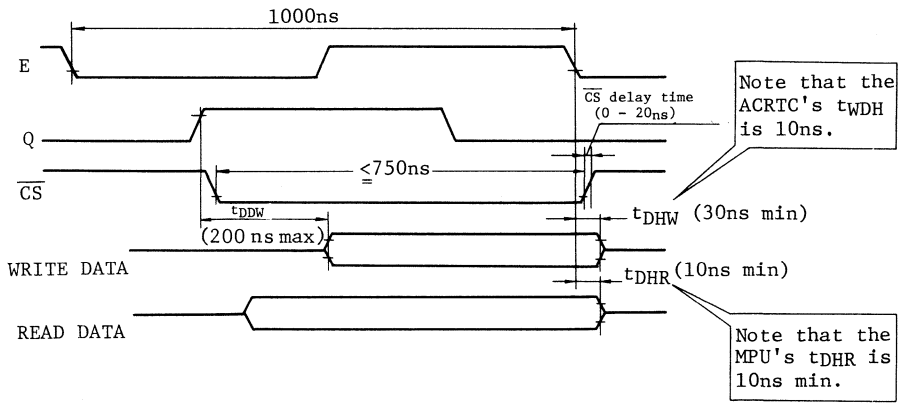


Fig. 3-3 Timing Chart of the HD6809 Read/Write Cycle

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.

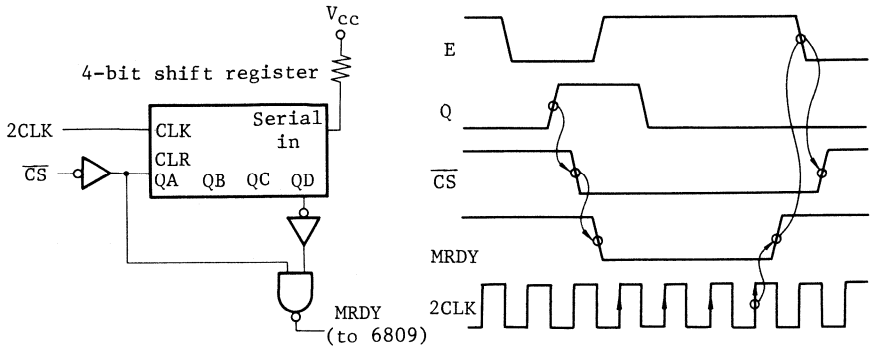


Fig. 3-4 Wait Circuit for the HD6809 and Timing

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.

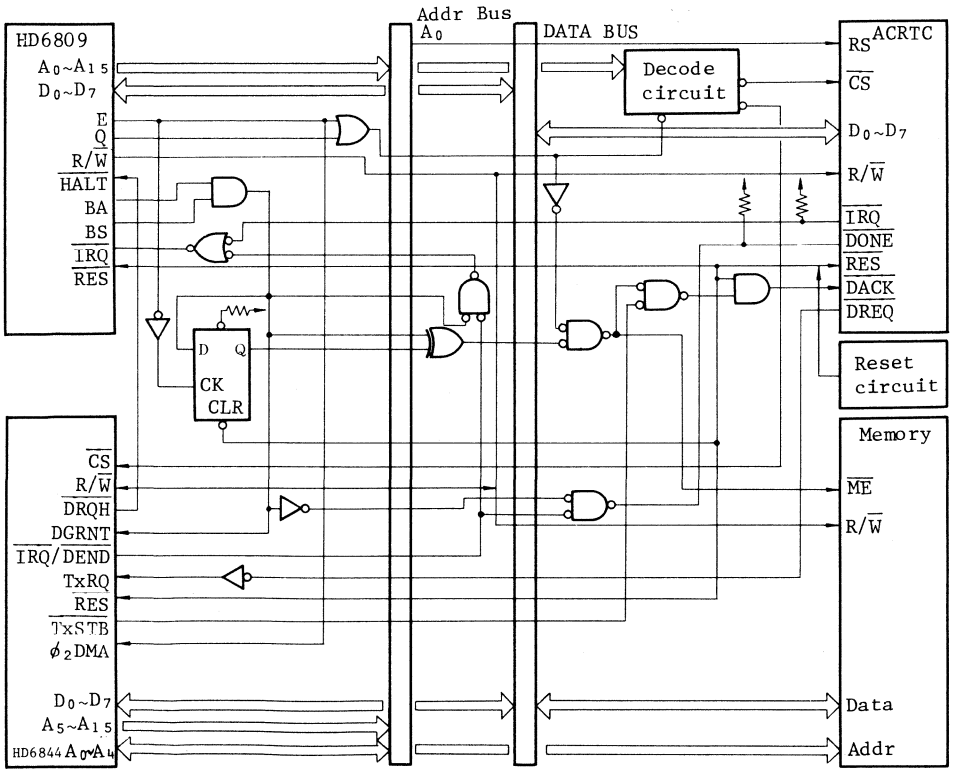


Fig. 3-5 8-Bit DMAC Interface Circuit

Note) Bus arbitration is performed as follows.

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

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

3 MPU receives $\overline{HALT}$. After completion of the current cycle, MPU disconnects itself from the bus, sets BA and BS to "H" to inform 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 $\overline{DACK}$ signal. For this purpose, the $\overline{TXSTB}$ signal from the DMAC is applied to the ACRTC $\overline{DACK}$ pin, as shown in Fig. 3-6. It should be noted that masking of the $\overline{TXSTB}$ is required for preventing the ACRTC from being improperly accessed during the bus arbitration period. Details are shown in Fig. 3-5.

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

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

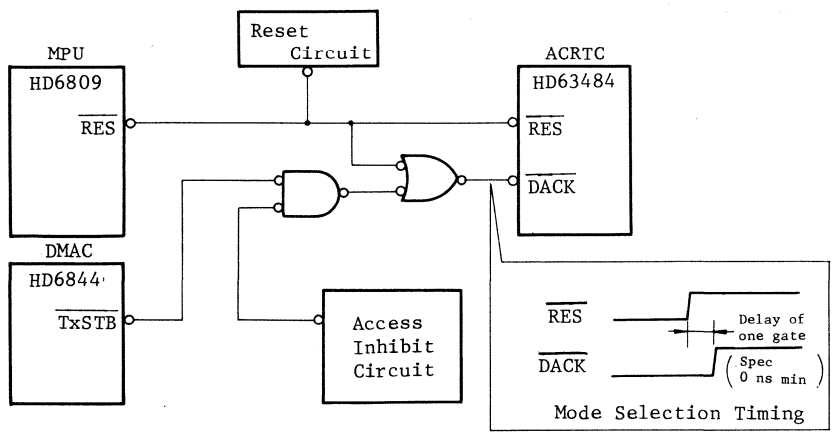


Fig. 3-6 Connection Circuit of $\overline{DACK}$ and $\overline{RES}$

4. CRT INTERFACE

4.1 Dot Clock, 2CLK, Load Signal Generation Circuit

The frequency of dot clock depends on a 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.

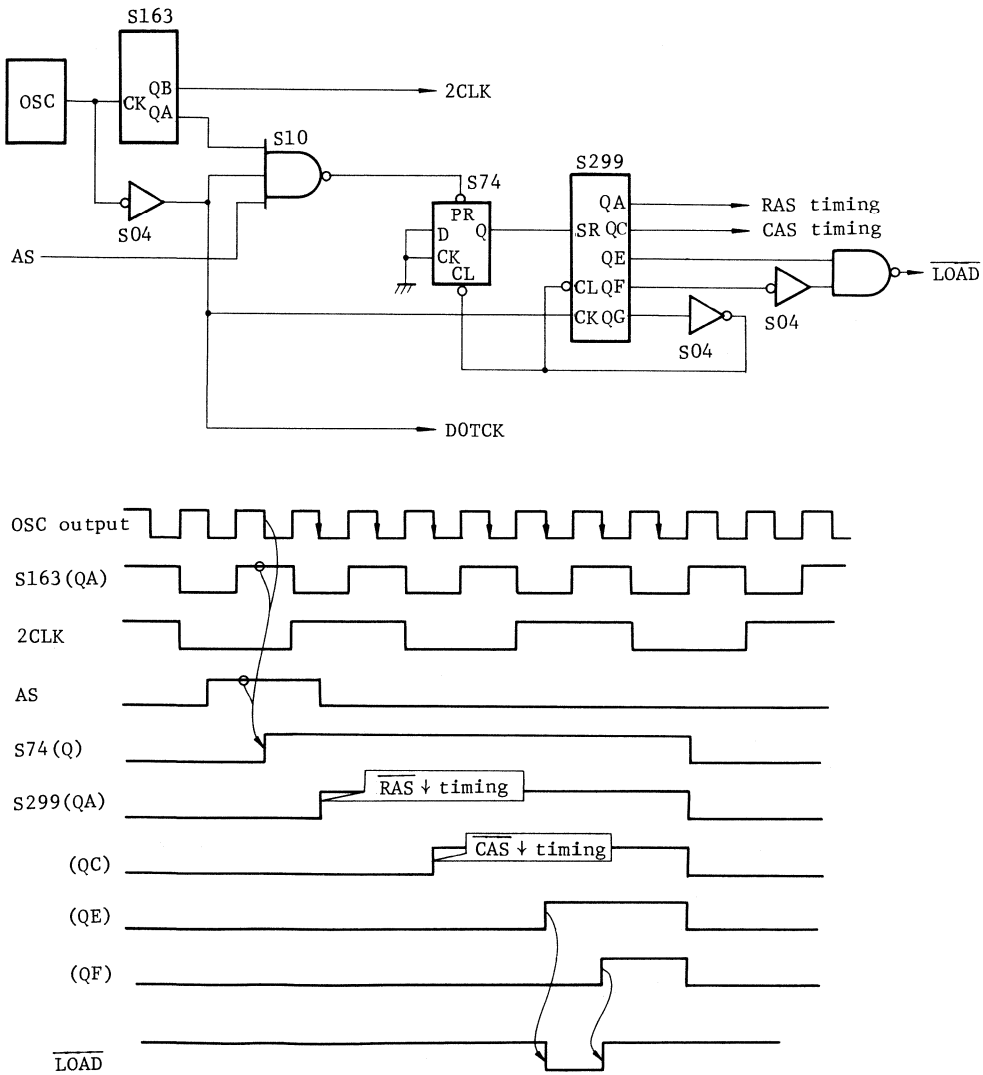


Fig. 4-1 Dot Clock, LOAD Signal Generation Circuit

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.

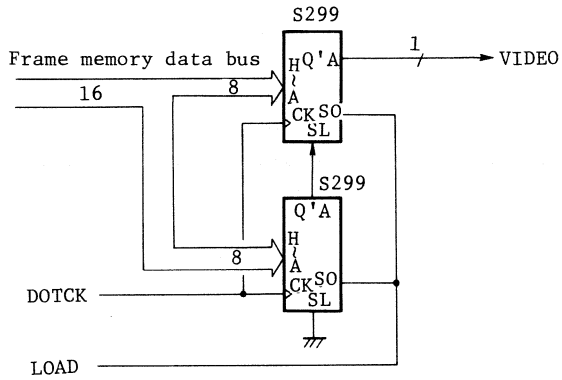


Fig. 4-2 Parallel/Serial Converting Circuit

(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 display 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.

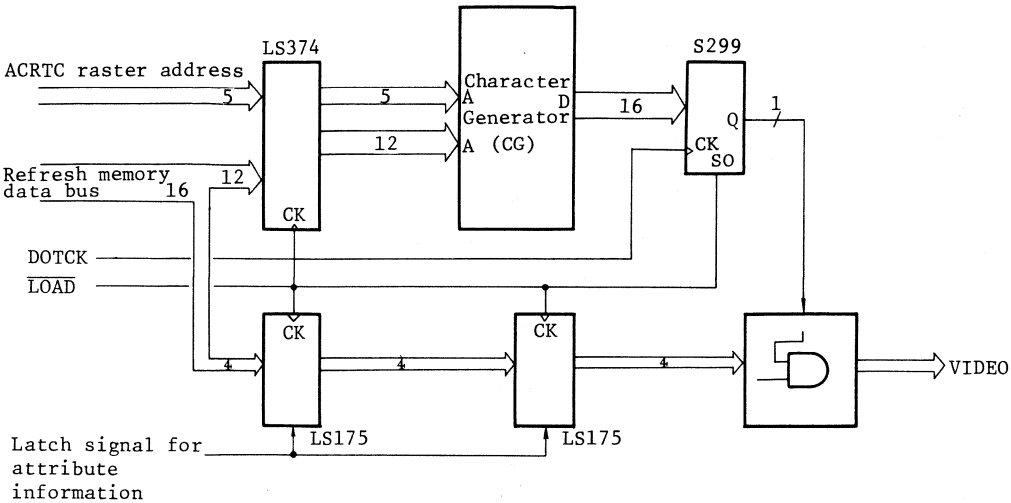


Fig. 4-3 An Example of Character Generating Circuit

(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 superimpose 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 nircuit.

DA1 (DUAL ACCESS 1 MODE)

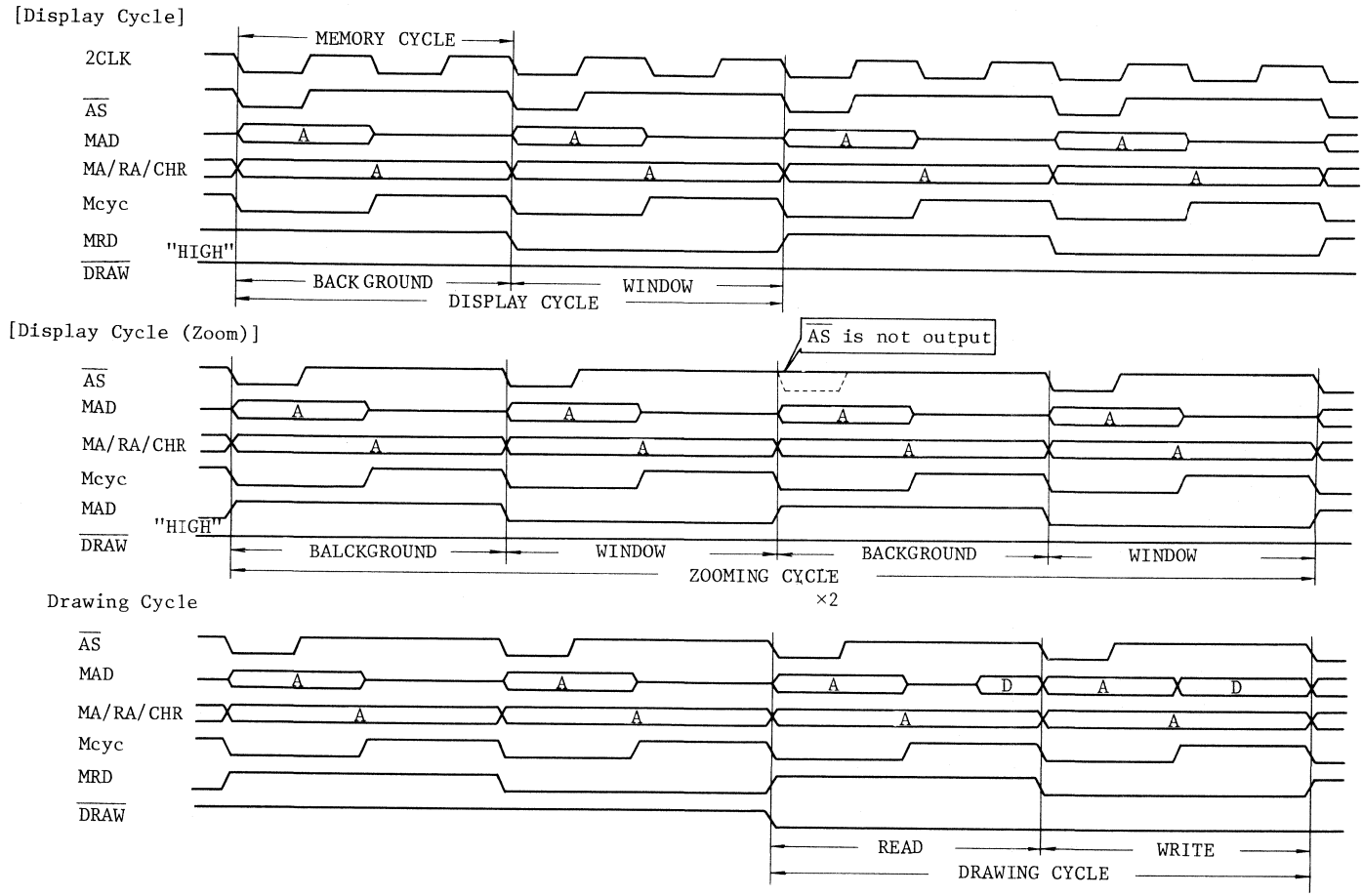


Fig. 4-4 Superimposed Access Mode (Dual Access Mode 1)

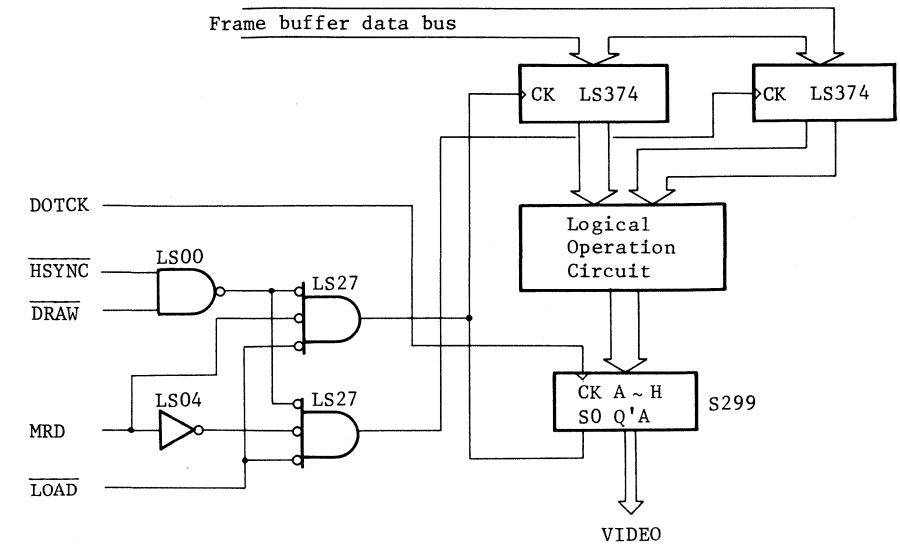


Fig. 4-5 Superimpose Circuit (1)

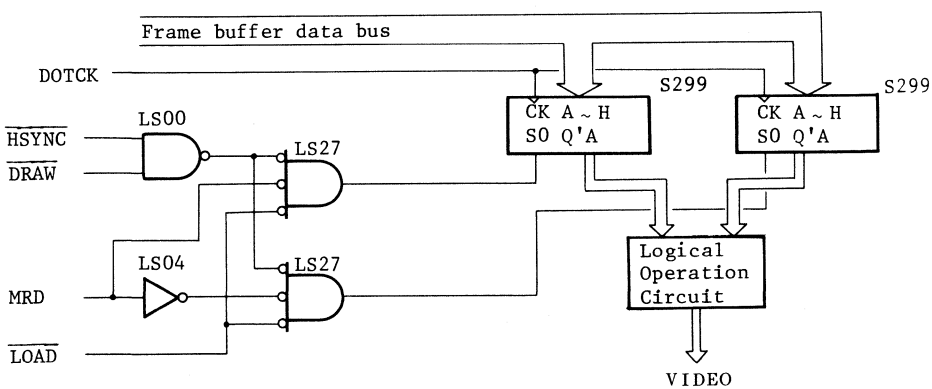


Fig. 4-6 Superimpose Circuit (2)

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 the one display cycle.

5.1.1 Frame Buffer for Graphic Display

In the case of graphic display, the data which is read out in the 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 used, and parallel to serial conversion of 16 dots are executed, the data number to be read out during the one display cycle is 64 bits (4 × 16).

On the other hand, the 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, DRAMs is the most commonly used from the view points of capacity and mounting area. The amount of memory chips is determined by the amount of data which is read out in one display period and the number of screens needed 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 page and nibble modes of DRAMs by implementing multiple access to the memory during one display period.

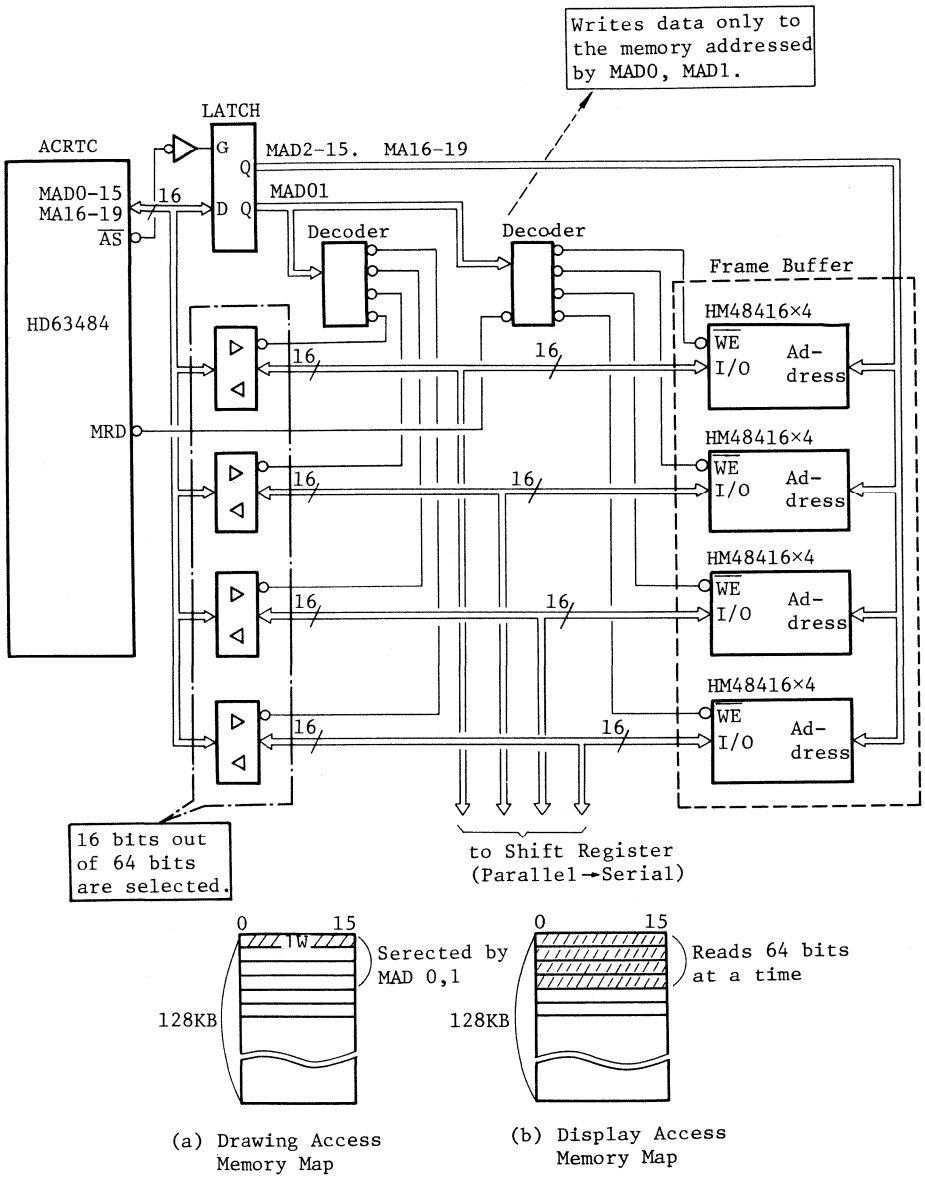


Fig. 5-1 The Example of Memory Organization Using the HM48416

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	512KB CRT 4 screens
HM50256	256K × 1bit	64	2MB CRT 16 screens
HM48416	16K × 4bit	16	128KB 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

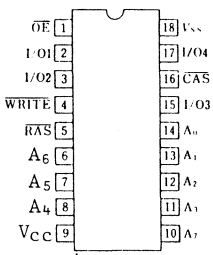
- **FEATURES**
- 16384-word x 4-bit Organization
- Single 5V (±10%)
- Low Power; 303mW Active, 20mW Standby
- High speed: Access Time 120ns/150ns/200ns (max)
- Page mode capability
- Output data controlled by $\overline{CAS}$, $\overline{OE}$
- TTL compatible
- 128 refresh cycles ($A_0 \sim A_6$, 2ms)

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

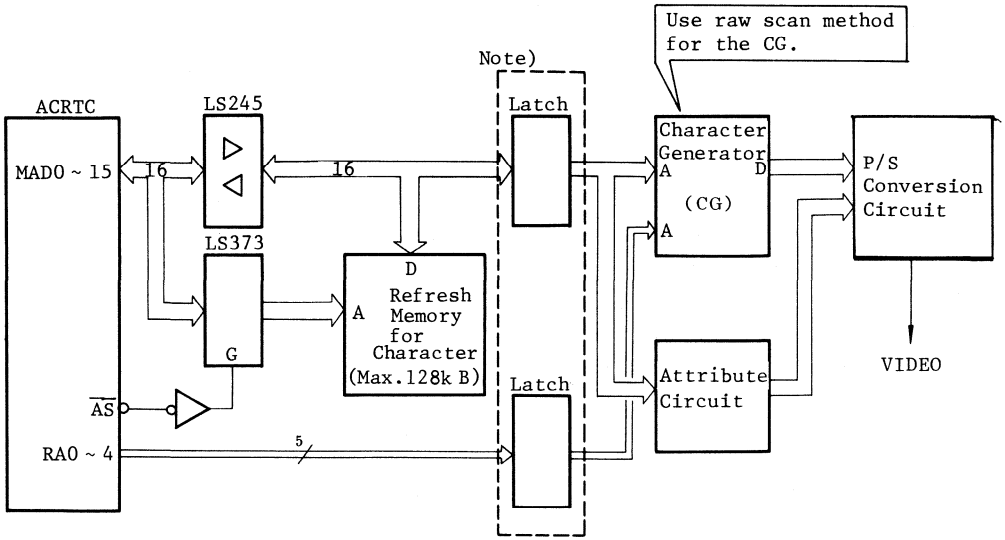
- **PIN ARRANGEMENT**



(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 128kB 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 implimented without using these latch.

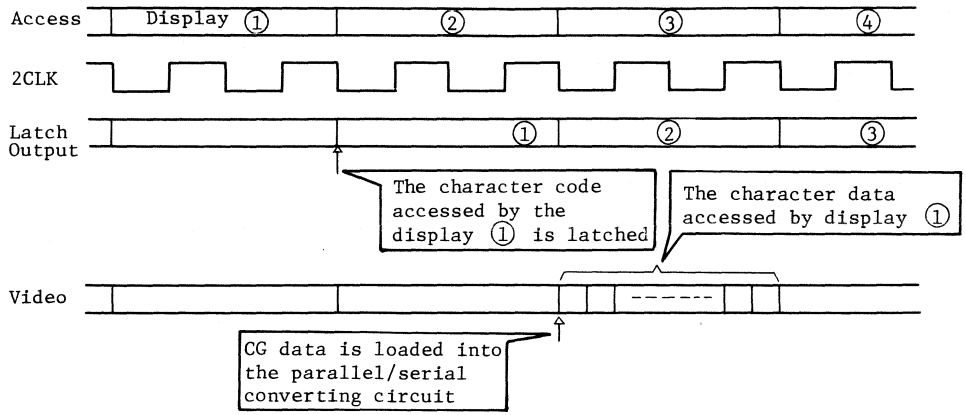
Fig. 5-2 Block Diagram of Character Display

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).

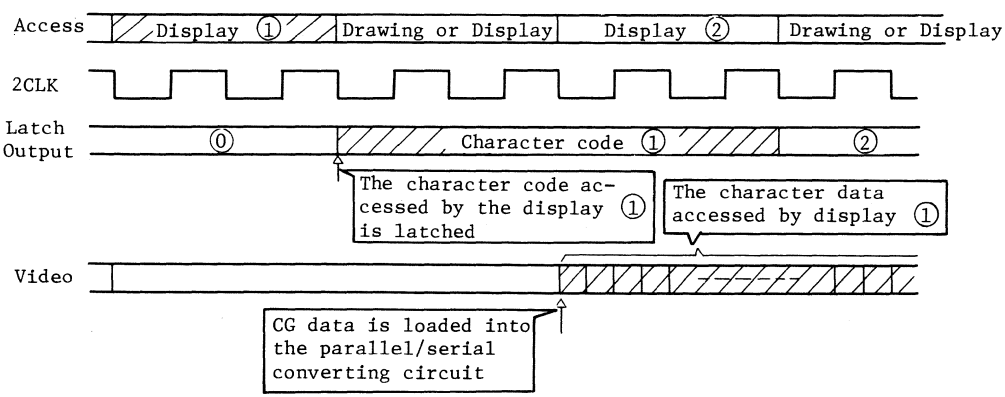
The display addresses which are output from MAD 0 - 15, are successively incremented by plus 1 starting from the stored in address of the display start address register (SAR). Therefore, character generator (CG) addresses to be displayed (character code) need to be stored in the refresh memory before 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

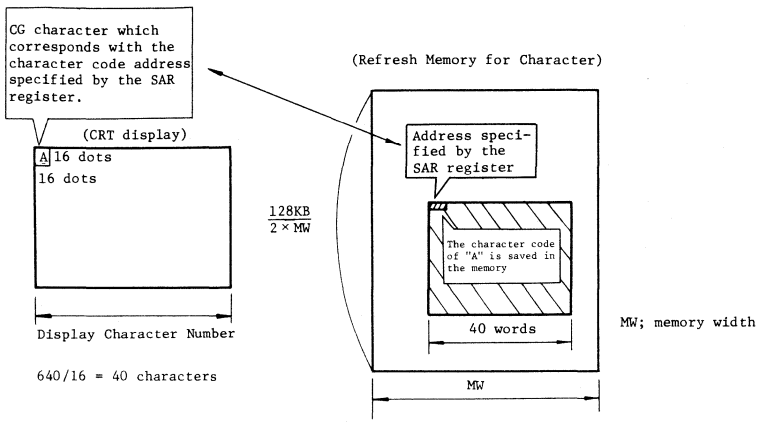


Fig. 5-4 Correspondence between the CRT Display and the Refresh Memory

Table 5-4 Memory Type and the Minimum Necessary Chips

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

Note 1) Calculates the CG address as 16bit.

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

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 dat 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 tMRDS of the ACRTC. This is because $\overline{CAS}$ falling with the same timing causes insufficient ACRTC read data setup time (tMRDS) when the 2CLK cycle time is shortened in high speed application.

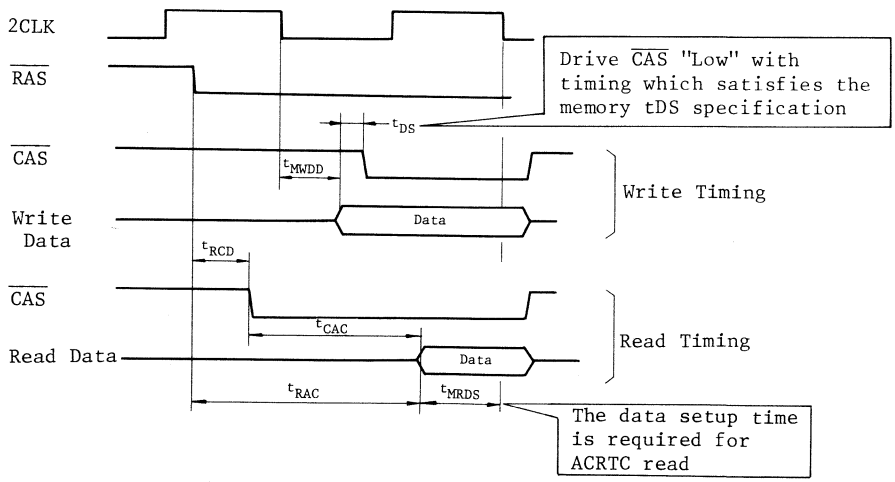


Fig. 5-5 Memory Read/Write Timing

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 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.

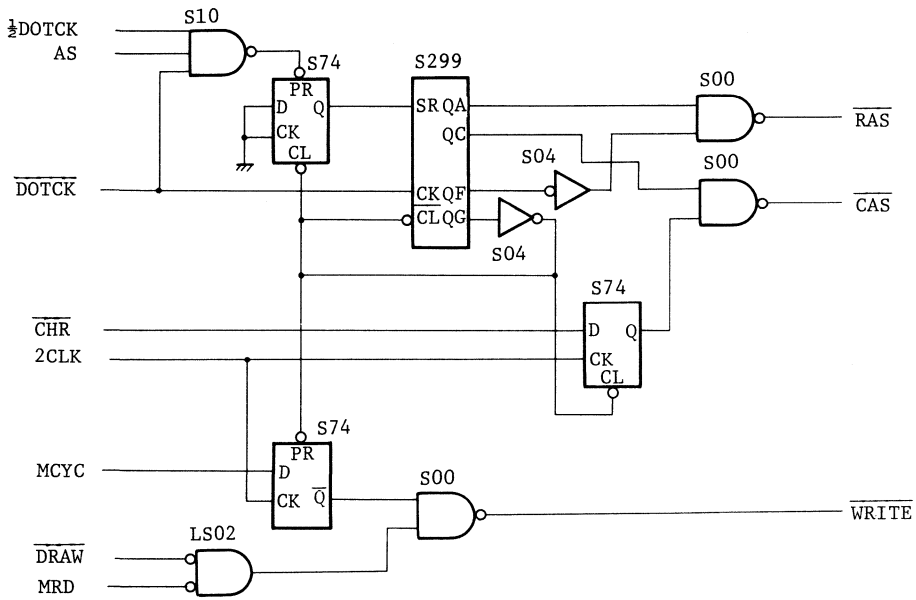


Fig. 5-6 DRAM Access Circuit

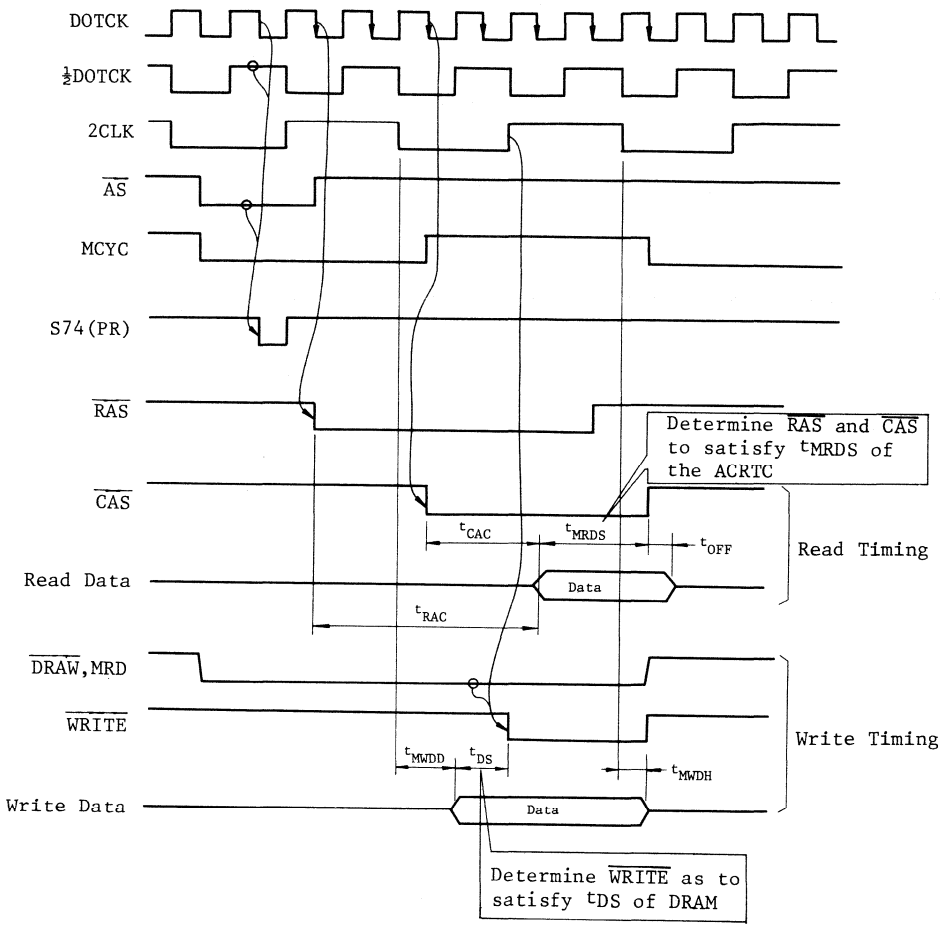
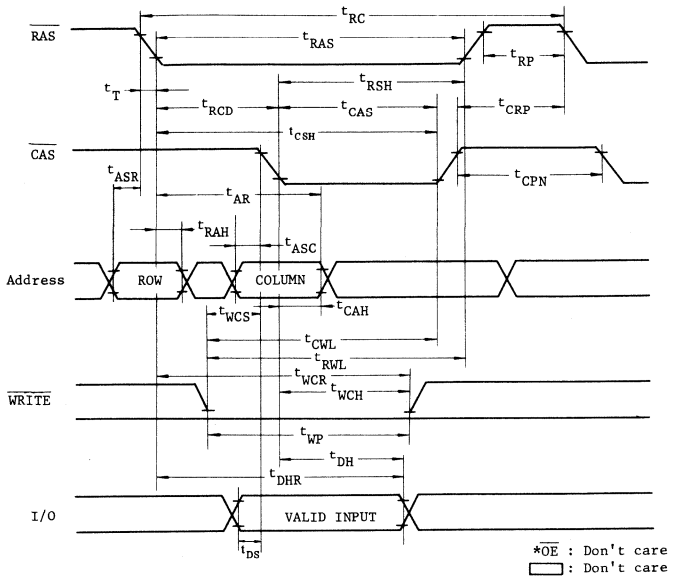


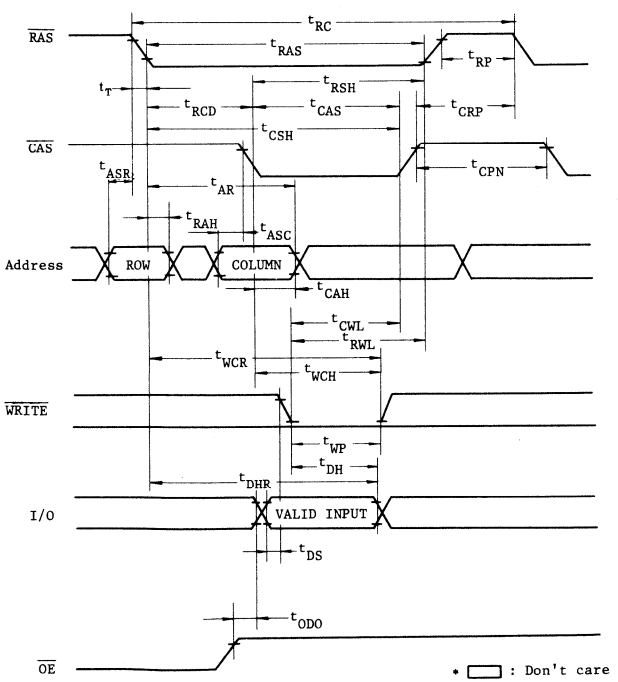
Fig. 5-7 DRAM Access Timing

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.

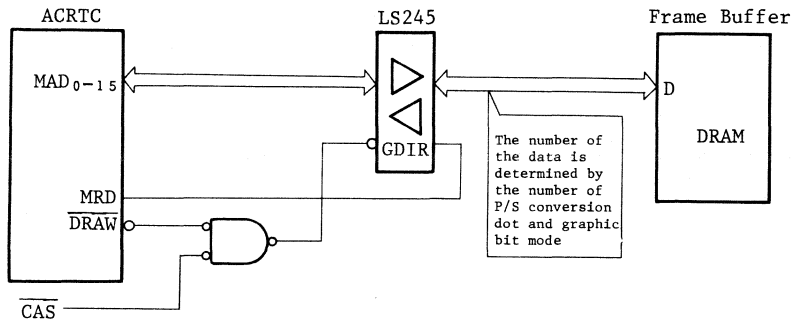


Fig. 5-8 Data Bus Control Circuit

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 2ms. The refresh frequency of the ACRTC is 2ms when $0A is set to HSW, (2ms/41.3 us) × 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 to be used 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 (tMRDS) 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 DH) must exceed the specified values.

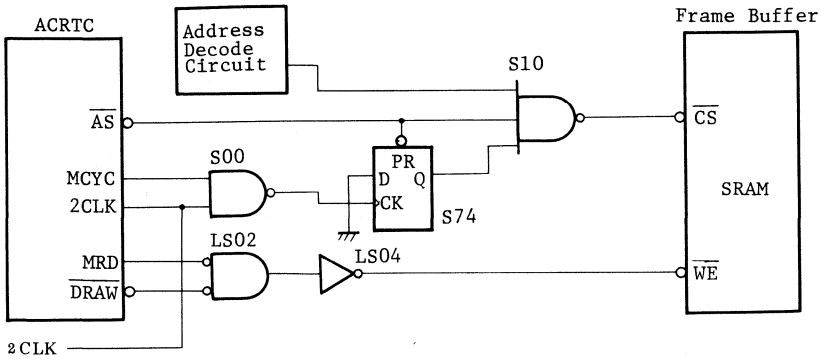


Fig. 5-9 The Example of SRAM Access Circuit

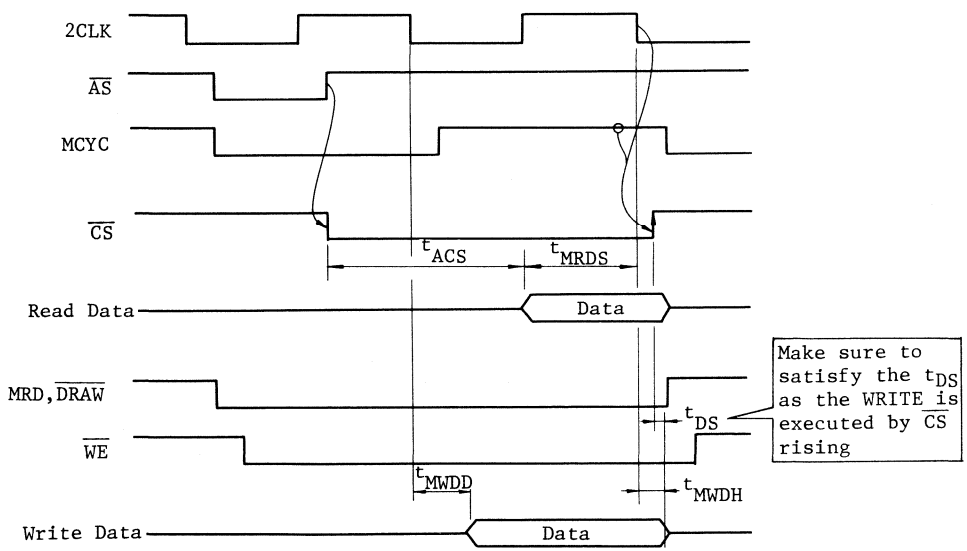


Fig. 5-10 SRAM Access Timing

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.

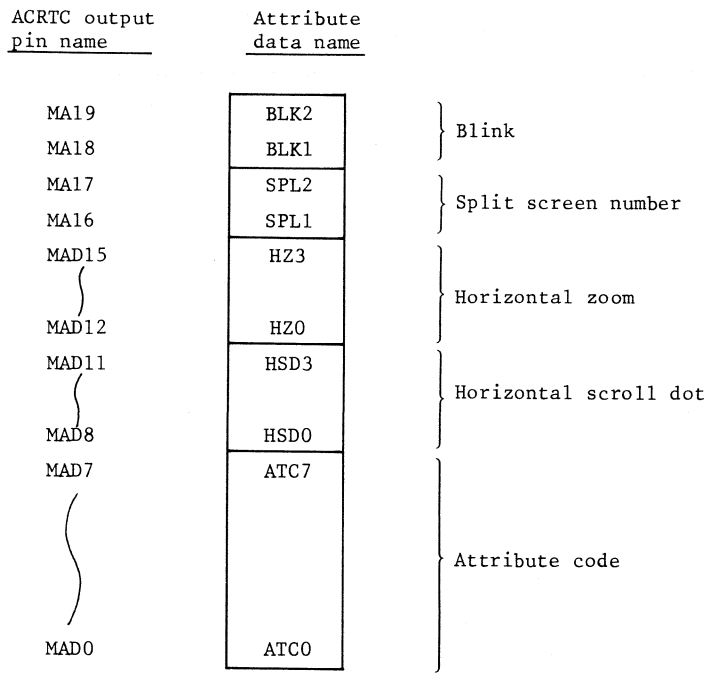


Fig. 6-1 Attribute Control Signal

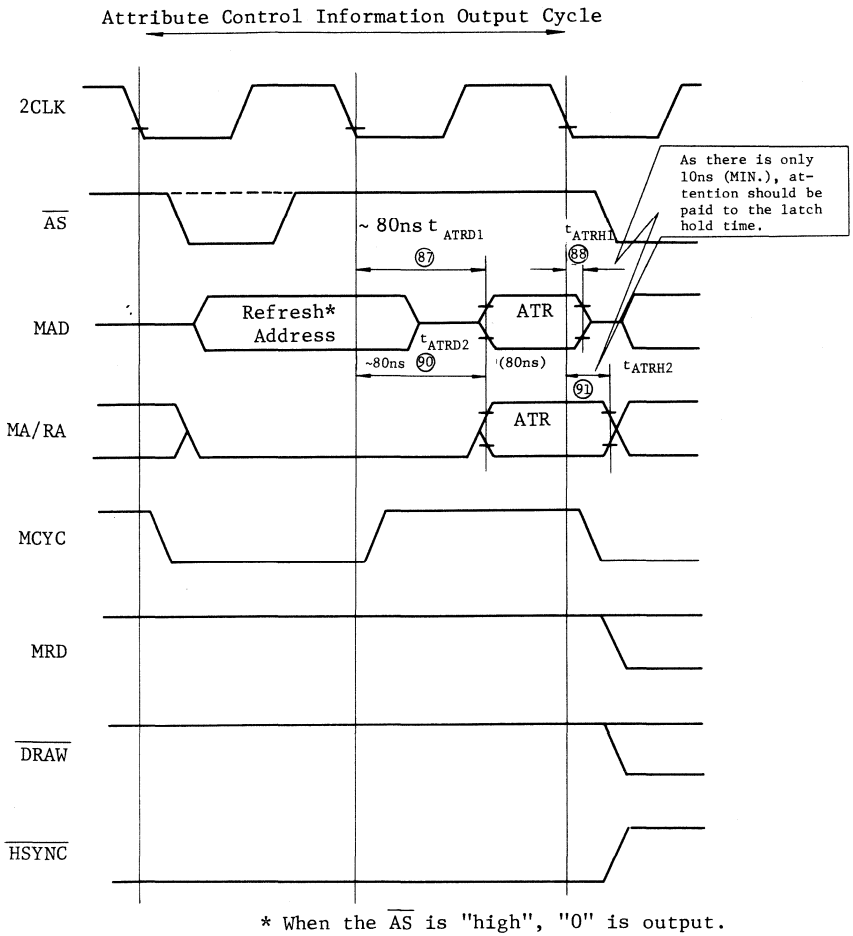


Fig. 6-2 Attribute Control Data Output Timing

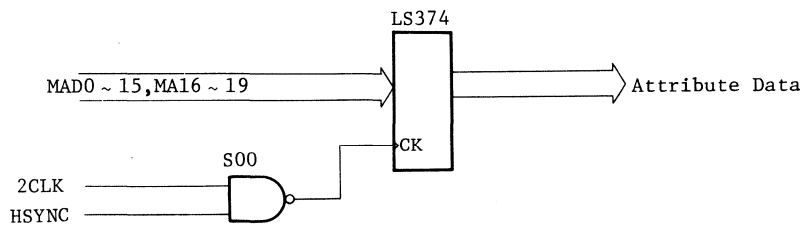


Fig. 6-3 Circuit to Latch Attribute Data

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 unite 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 03).

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.

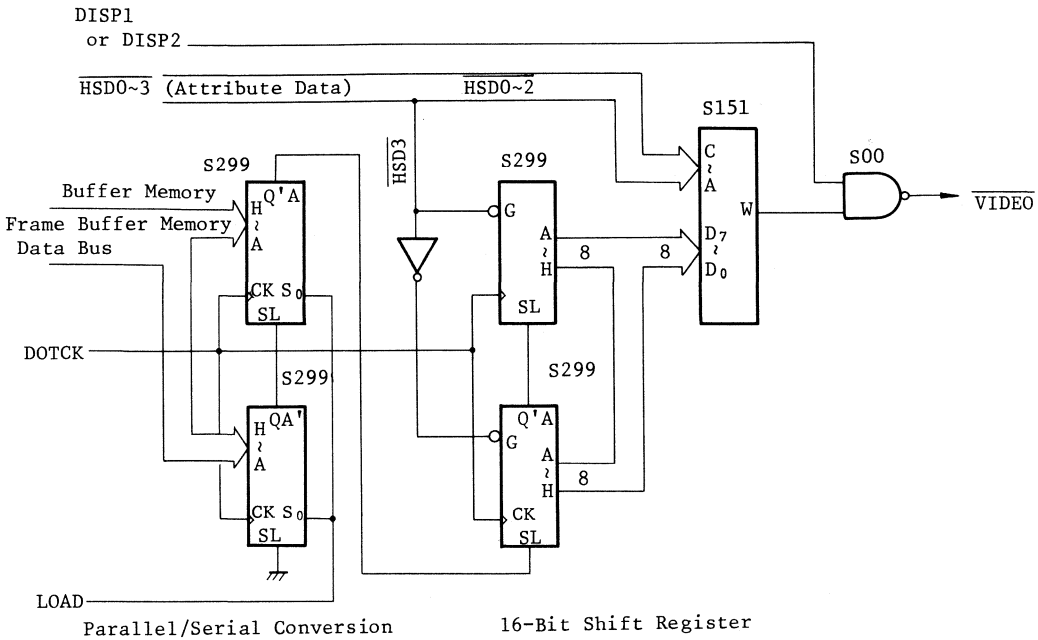


Fig. 6-4 Horizontal Smooth Scroll Circuite

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

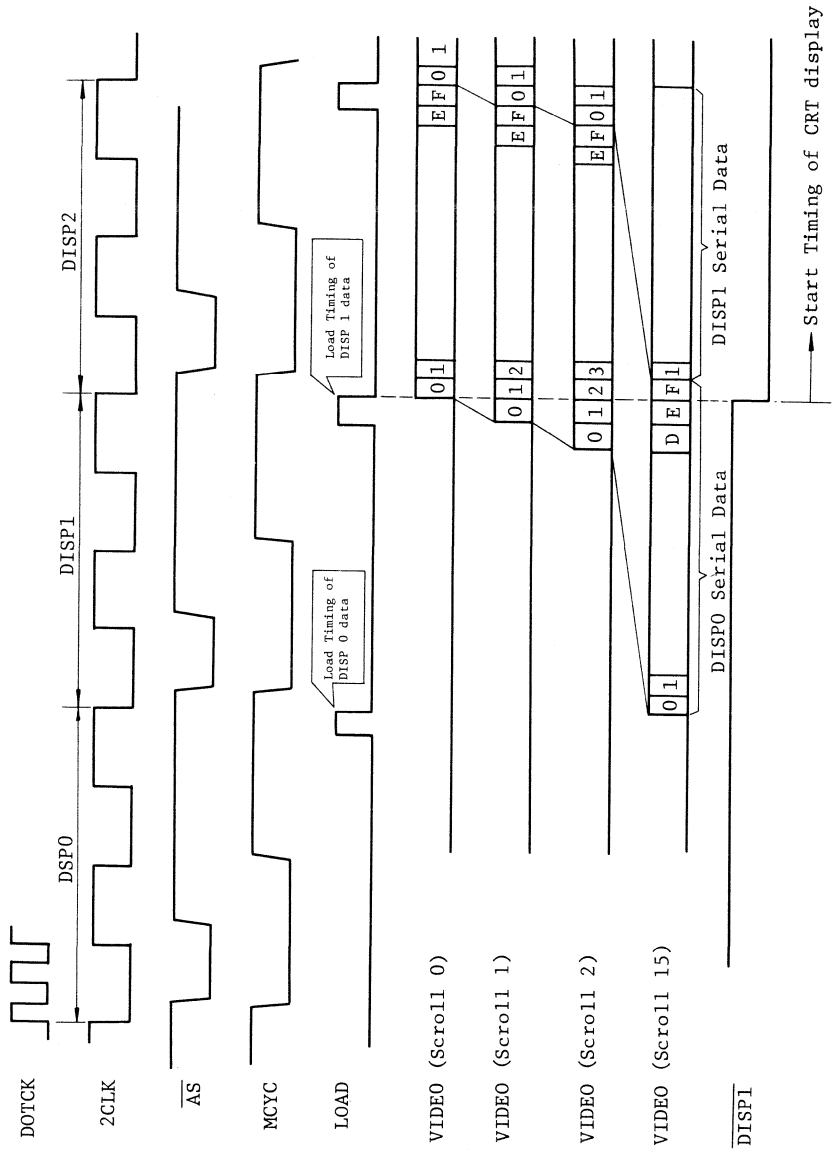
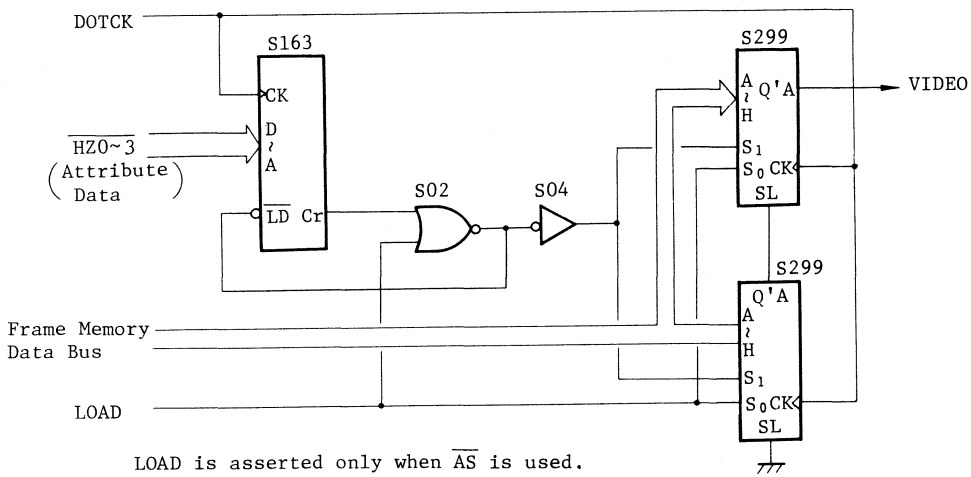


Fig. 6-5 Horizontal Smooth Scroll Timing of Base and Split Screens

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, 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

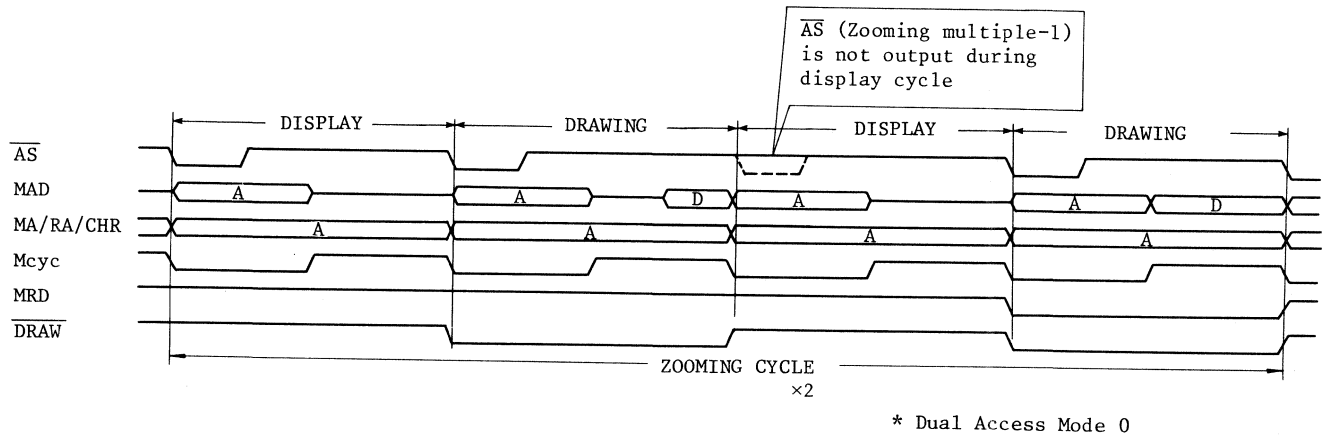


Fig. 6-7 Display Timing of Zoom (×2)

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".

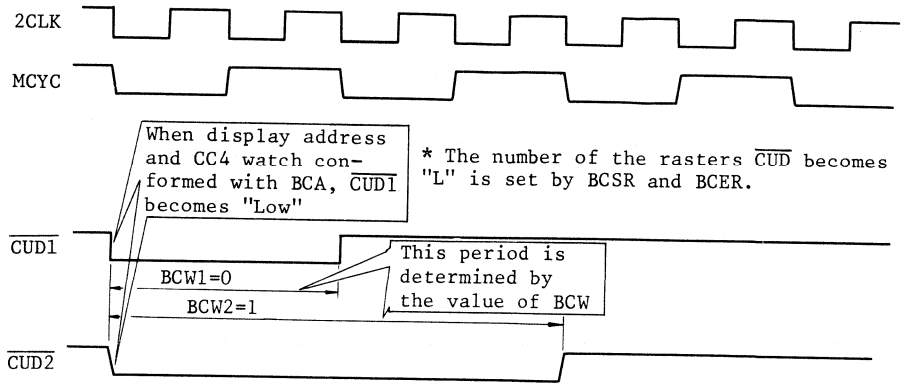
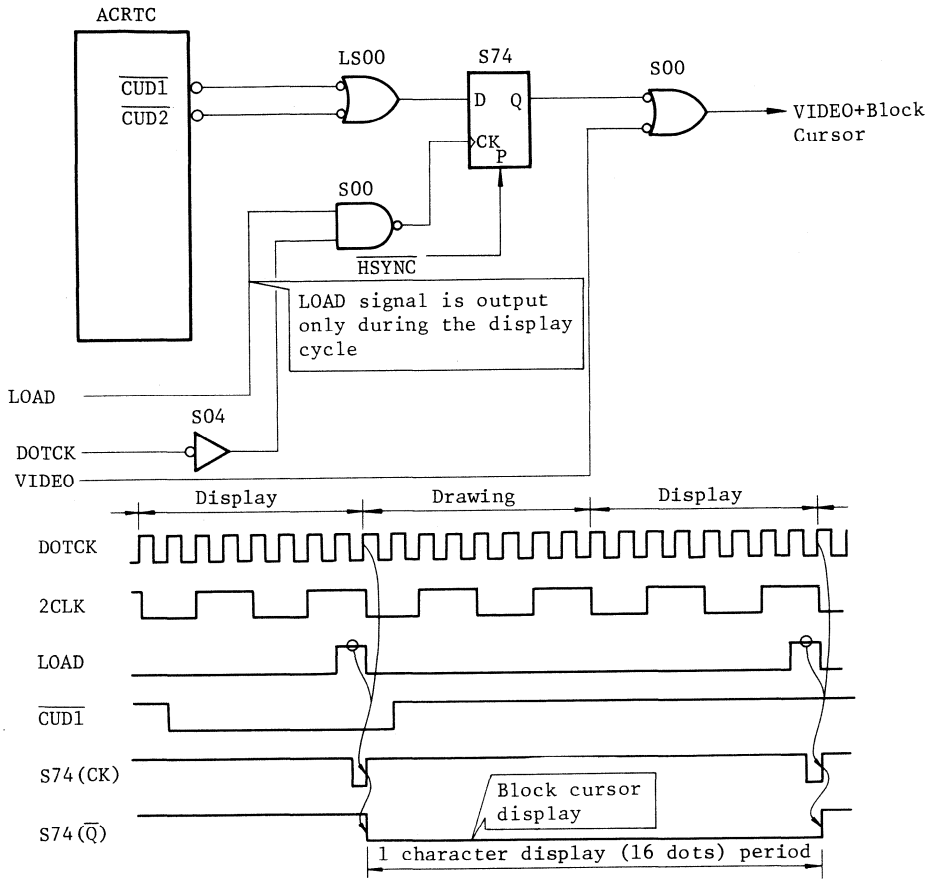


Fig. 6-8 Setting of $\overline{CUD}$ Signal Timing



* 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, the 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".

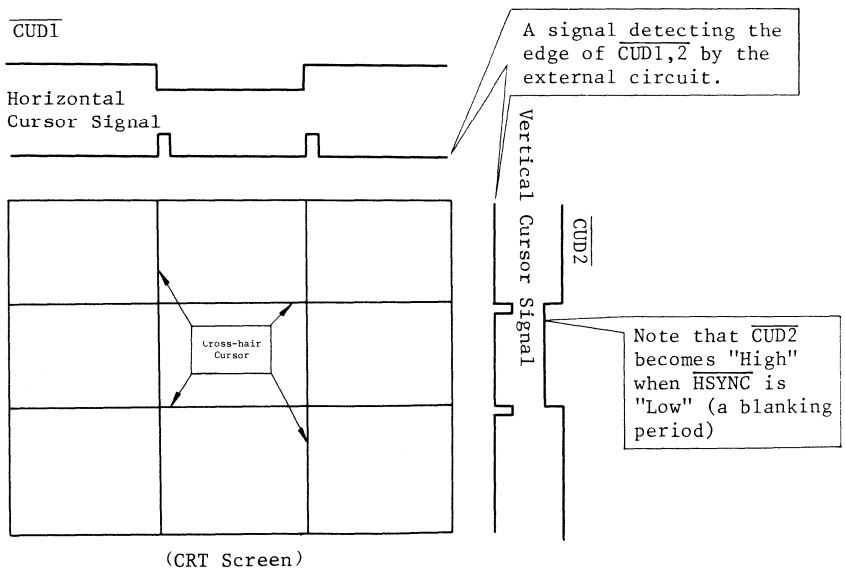


Fig. 6-10 Cross-hair Cursor Display

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 position horizontal position in the unit of dot (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)

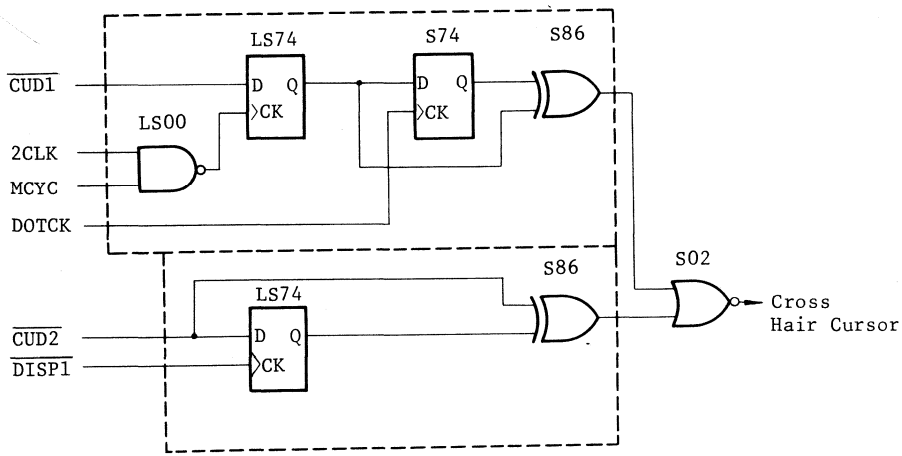
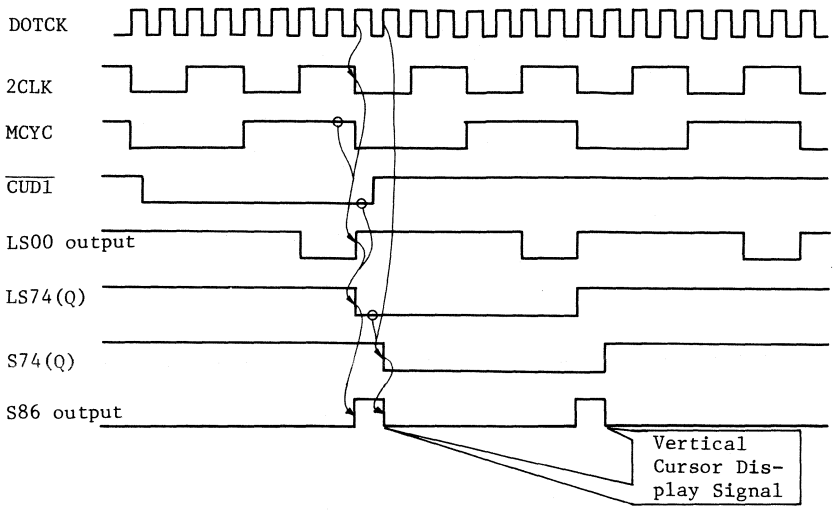
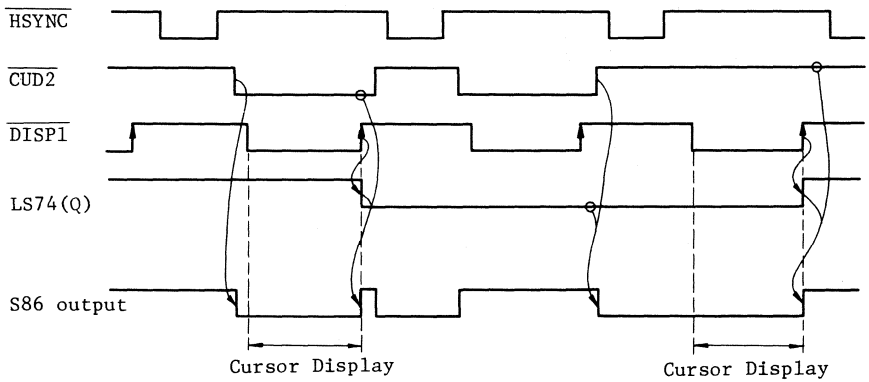


Fig. 6-11 Cross-hair Cursor Display Circuit



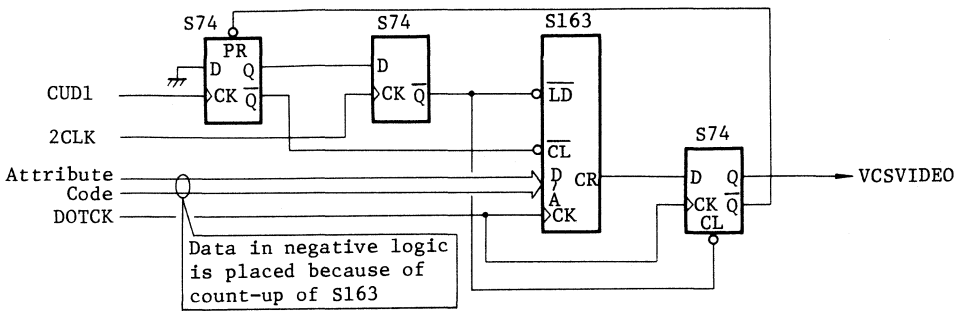
(a) Vertical Cursor Display Timing



(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.

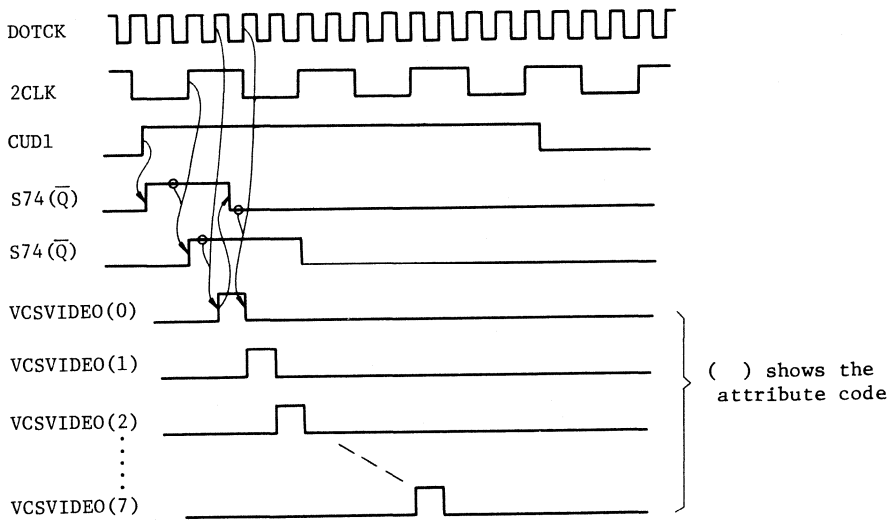
(b) Horizontal Cursor Display Timing

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 vertced 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.

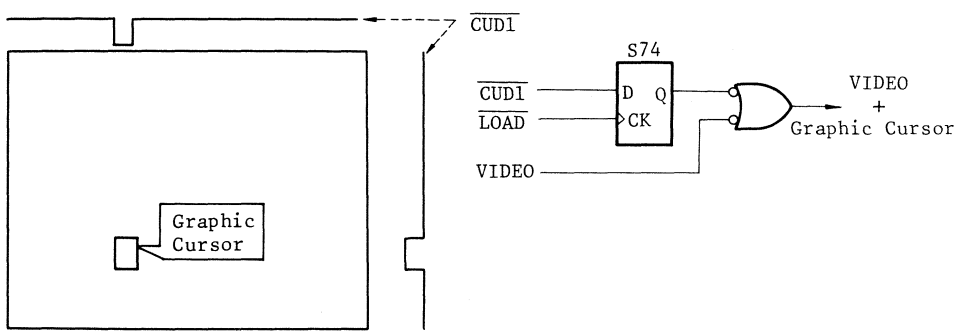


Fig. 6-14 Graphic Cursor Display Circuit

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.

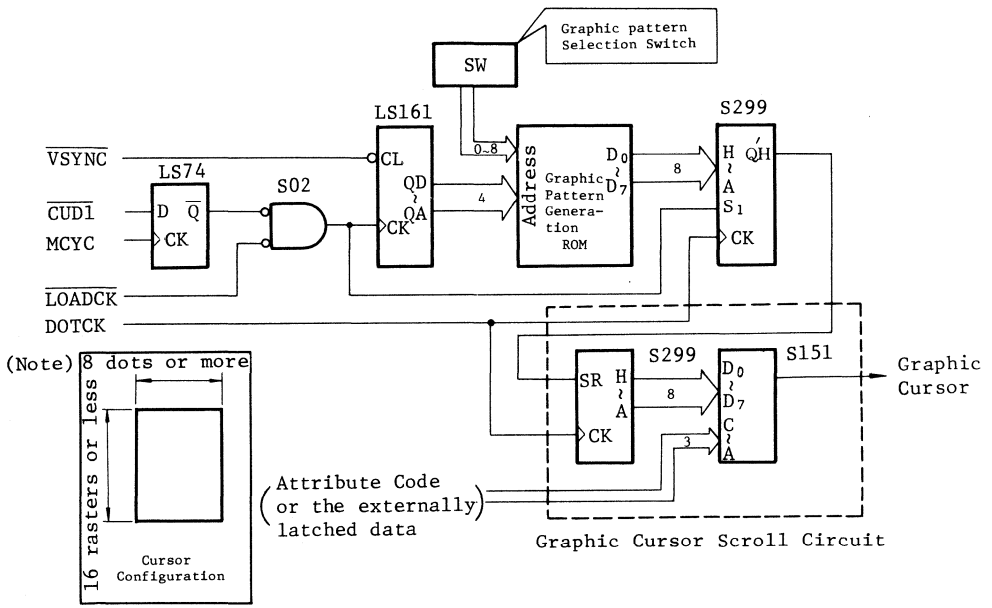
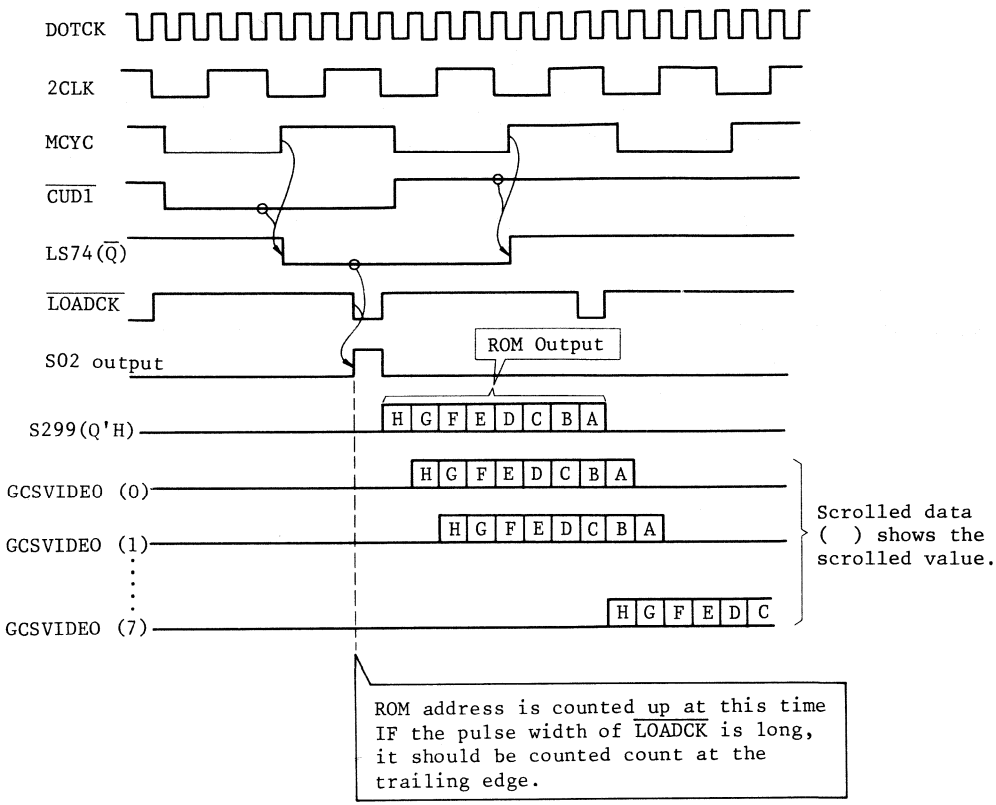


Fig. 6-15 Graphic Cursor Display Circuit



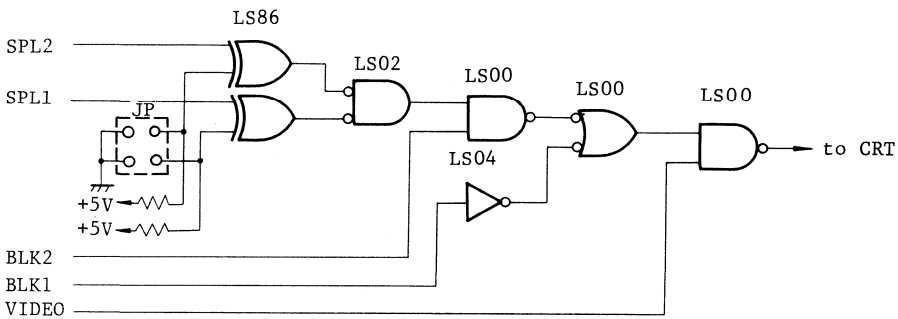
(Note) Total cursor width is ($\overline{CUD1}$ is "L") 8 dots.

Fig. 6-16 Graphic Cursor Display Timing

6.5 Blink and Split Control

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

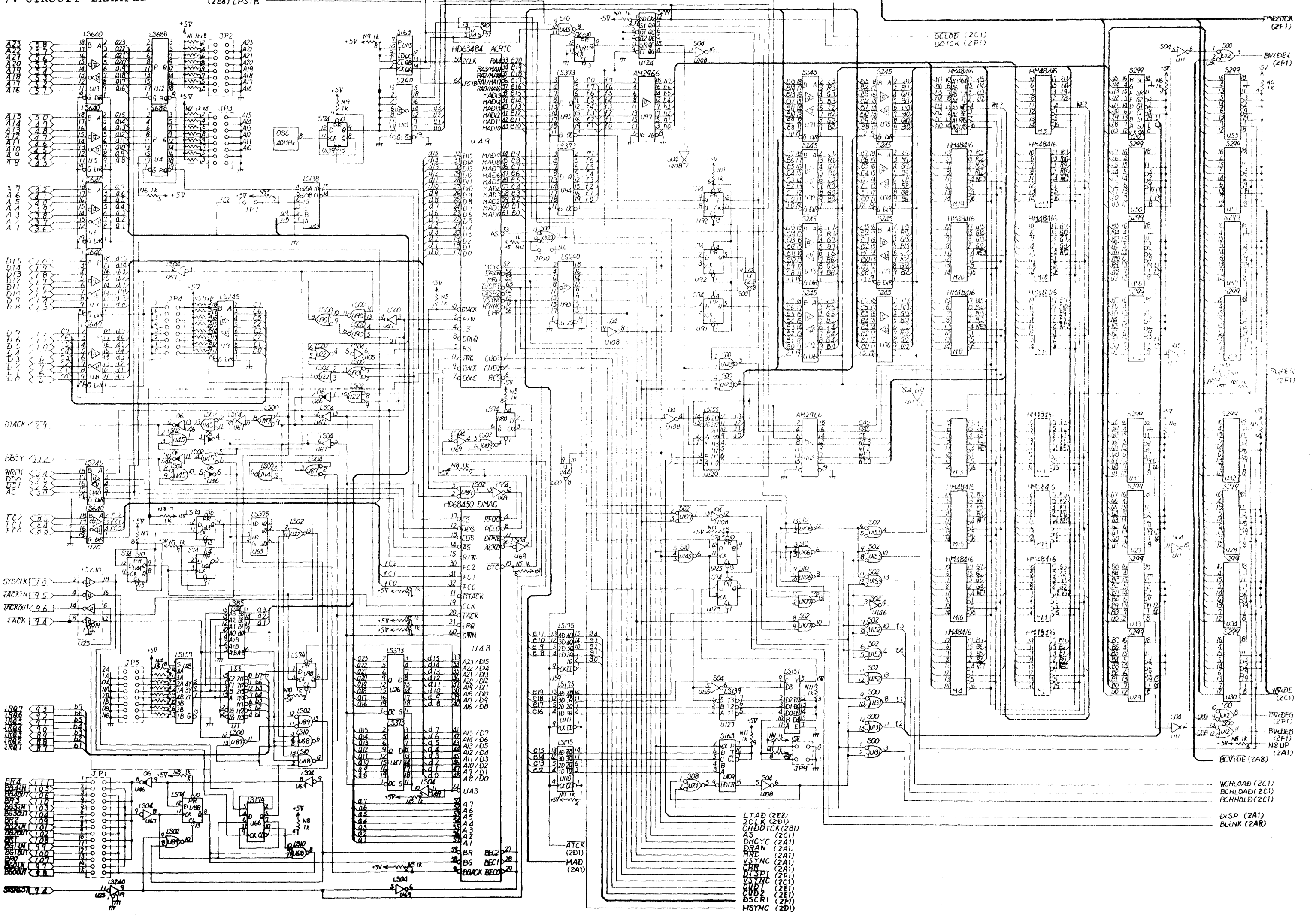
SPL1 and SPL2 output the split screen number which is currently displayed by ACRTC. However, the SPL can not indicate the window screen information. The BLK 1 and BLK2 are used to blink specific character. Also, by combining them with SPL performs the blink in units of screens. Fig. 6-17 shows the circuit to blink screens.



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

Fig. 6-17 Screen Blink Circuit

7. CIRCUIT EXAMPLE



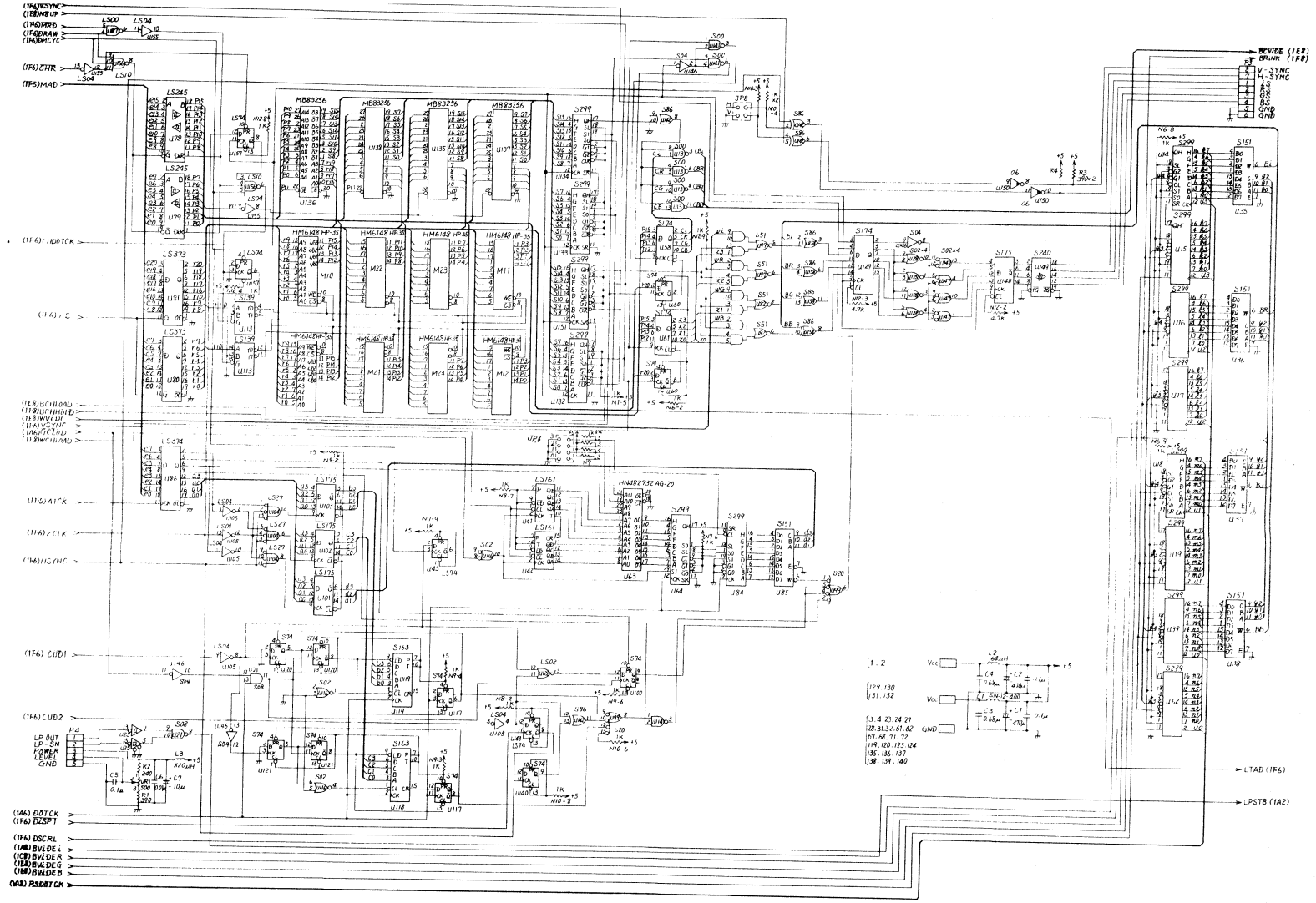


Fig. 7-1 (b) Application Circuit

VOLUME II
SOFTWARE

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.

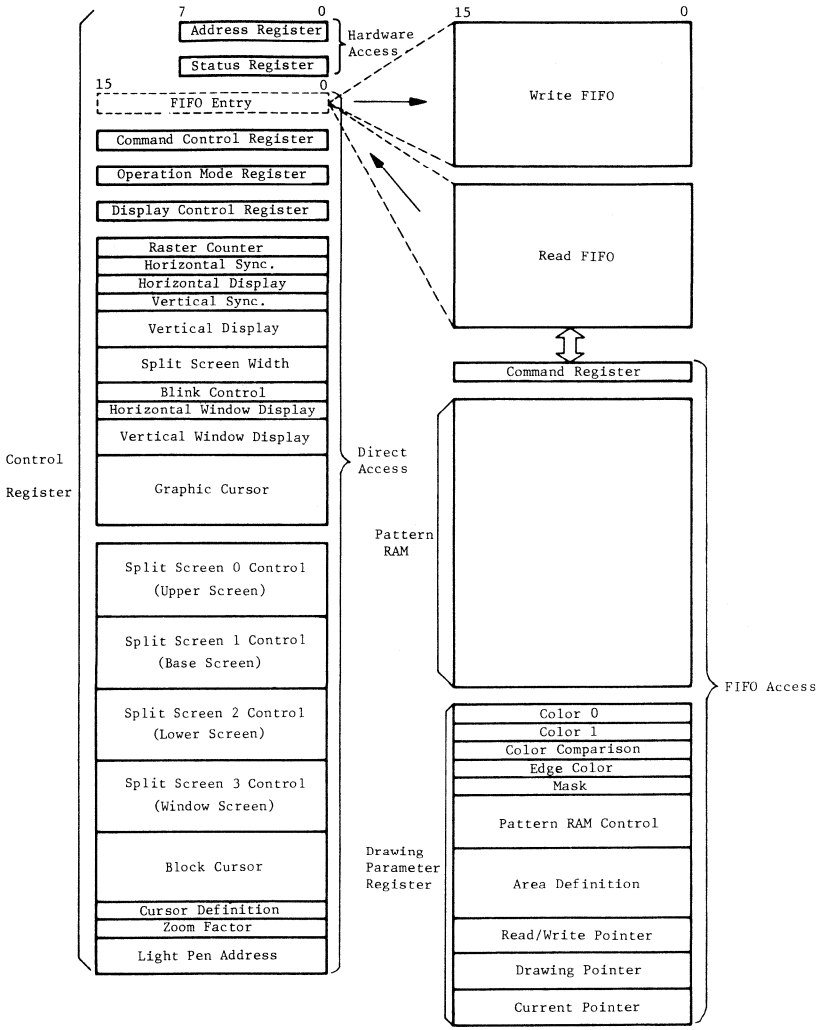


Fig. 1-1 Programming Model

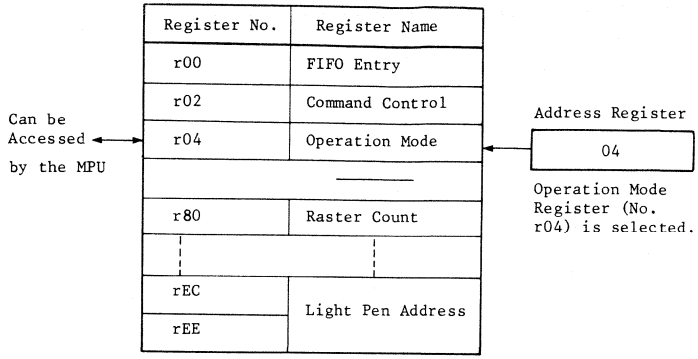


Fig. 1-2 Selection of a Control Register

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 to 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 of the Register Access: 8-bit interface

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

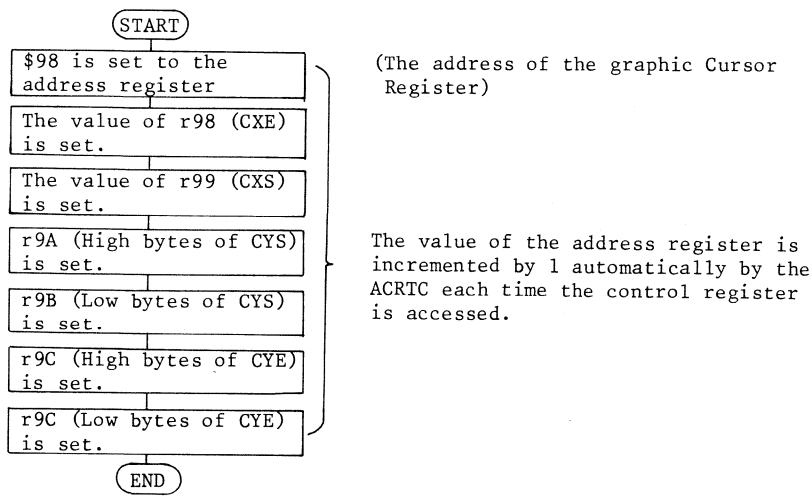


Fig. 1-3 Register Setup Flowchart Example (8-bit interface)

Example 2) Procedure of the Register Access: 16-bit interface

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

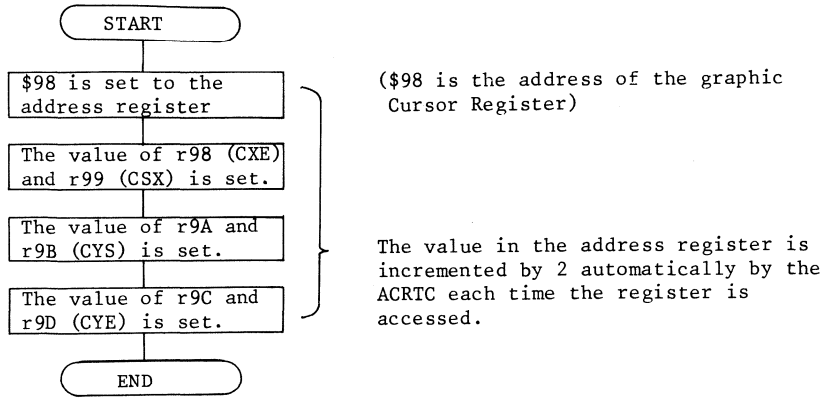


Fig. 1-4 Register Setup Flowchart Example (16-bit interface)

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 "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 X direction on the basis of the memory width set for each screen.

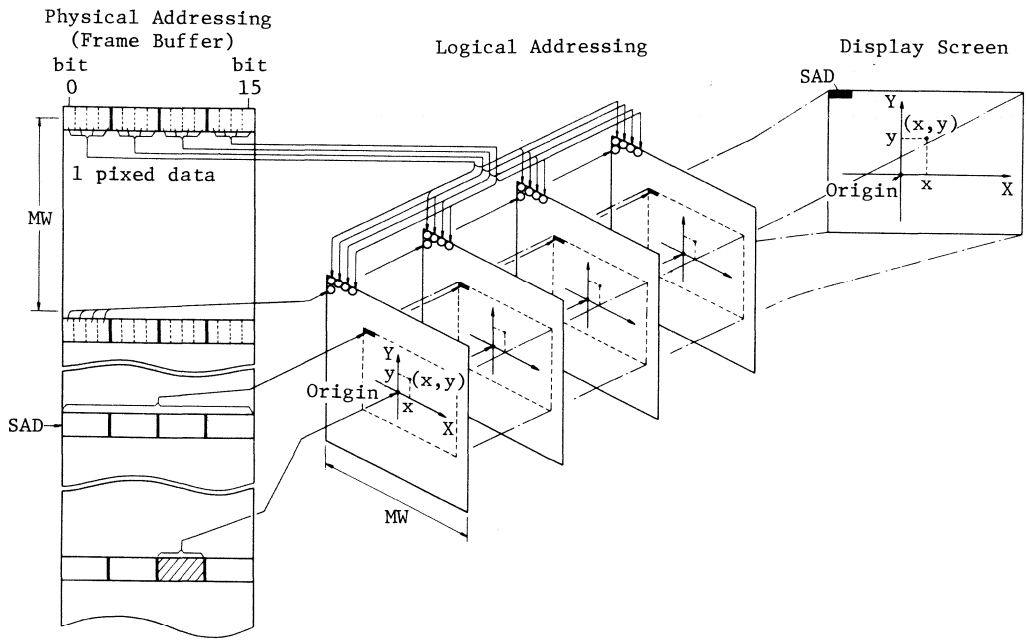


Fig. 1-5 Logical/Physical Addressing

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 the 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.

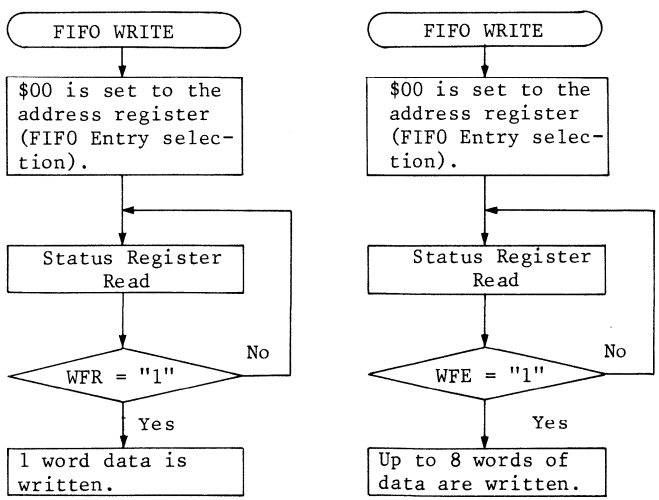
· Write FIFO Ready (WFR: bwt 1)
  The next word or byte can be written to the Write FIFO.

· 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

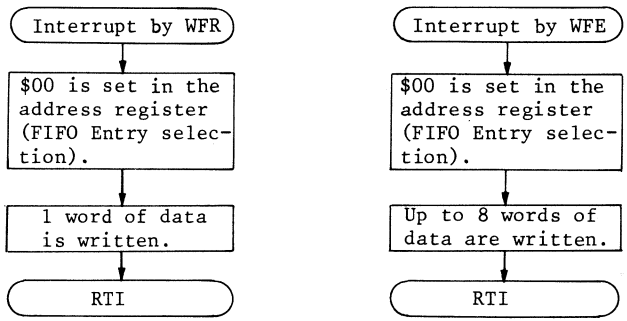


a) Checking Procedure of WFR b) Checking Procedure of WFE

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

In case a), it is confirmed by WFR that there is 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 of 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 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.

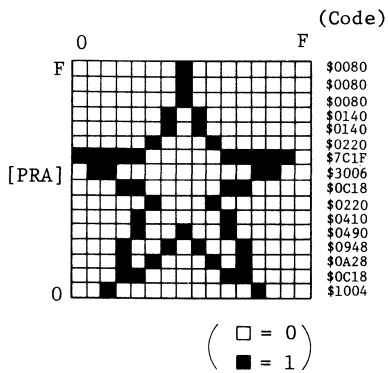


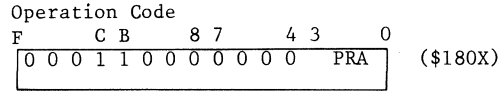
Fig. 1-8 Pattern RAM

<Mnemonic>

WPTN (PRA) n, D_1, D_2, --- , Dn

<Command Format>

Operation Code



Parameter

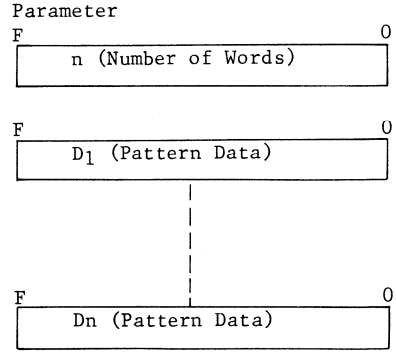


Fig. 1-9 WPTN Command

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.

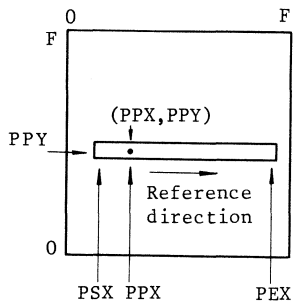


Fig. 1-10 Line Drawing Setting

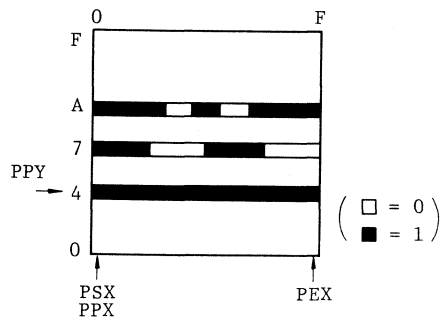


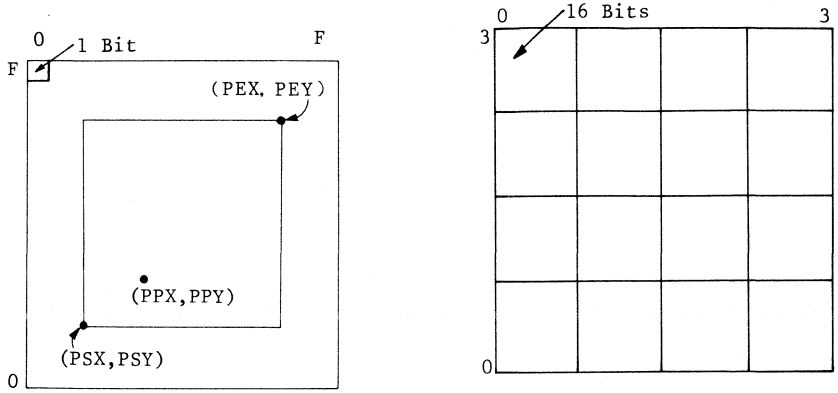
Fig. 1-11 Example of Line Type

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

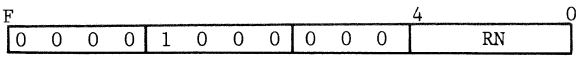
The Write Parameter Register (WPR) command is used to write the data to a specific drawing parameter register.

<Mnemonic>

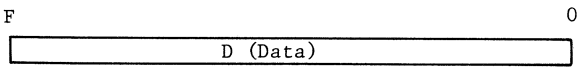
WPR (RN) D

<Command Format>

Operation Code



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

Example 1-2 WPR Command

1.5.2 Reading from the Drawing Parameter Registers

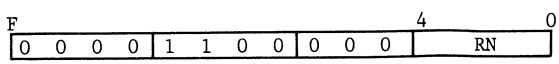
Read Parameter Register (RPR) command is used to read data from a specific drawing parameter register.

<Mnemonic>

RPR (RN)

<Command Format>

Operation Code



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.

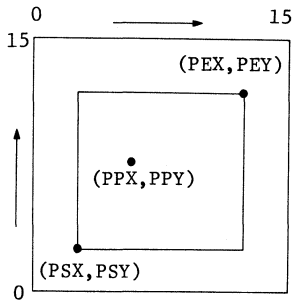


Fig. 1-13 Pattern RAM Setup

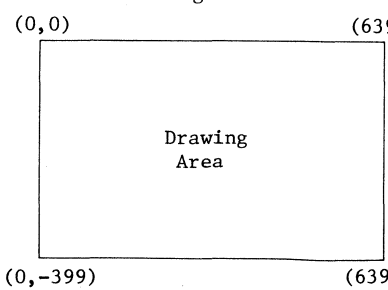
The magnification coefficient of the pattern RAM is set as the pattern zoom (PZX, PZY). The pattern is enlarged up to 16 times by setting between 0 and 15 for the zoom count.

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 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. The 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 lines, and one vertical cycle is the number of raster in one frame.

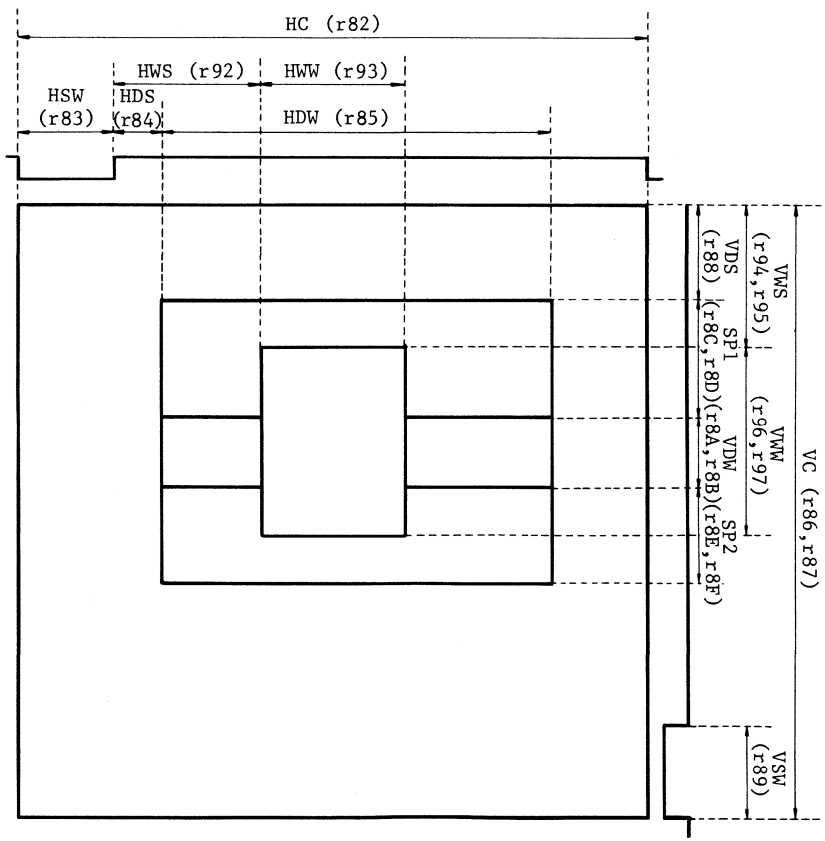


Fig. 1-16 Display Screen Specification

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,

$$Ts\ (ns) = \frac{T_H\ (\mu s) \times 1000}{Nd\ (dot)} \times Ns\ (dot) \quad : \text{ Single Access Mode}$$

$$Ts\ (ns) = \frac{T_H\ (\mu s) \times 1000}{Nd\ (dot)} \times Ns\ (dot) \times \frac{1}{2} \quad : \text{ Dual Access Mode}$$

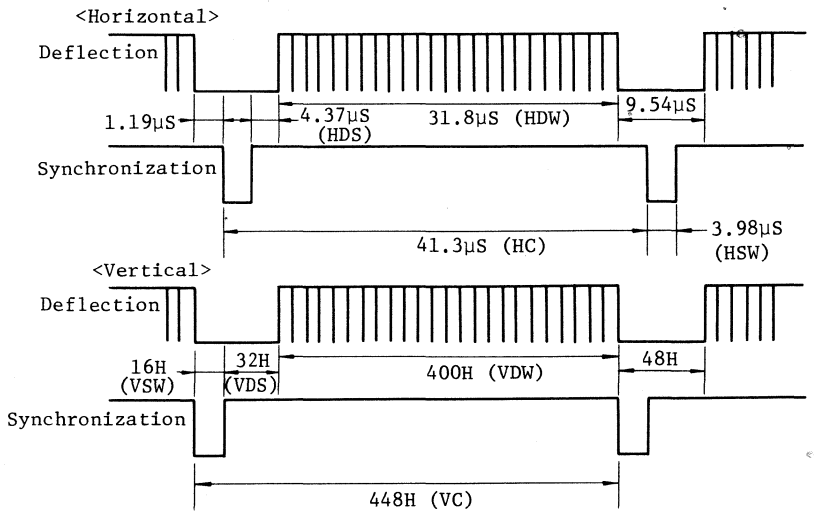


Fig. 1-17 CRT Specification

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.

$$Ts\ (ns) = \frac{31.8\ (\mu s) \times 1000}{640\ (dot)} \times 16\ (dot) = 795\ (ns) \quad : \text{ Single Access Mode}$$

$$Ts\ (ns) = \frac{31.8\ (\mu s) \times 1000}{640\ (dot)} \times 16\ (dot) \times \frac{1}{2} = 397.5\ (ns) \quad : \text{ Dual Access Mode}$$

460 ns

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.

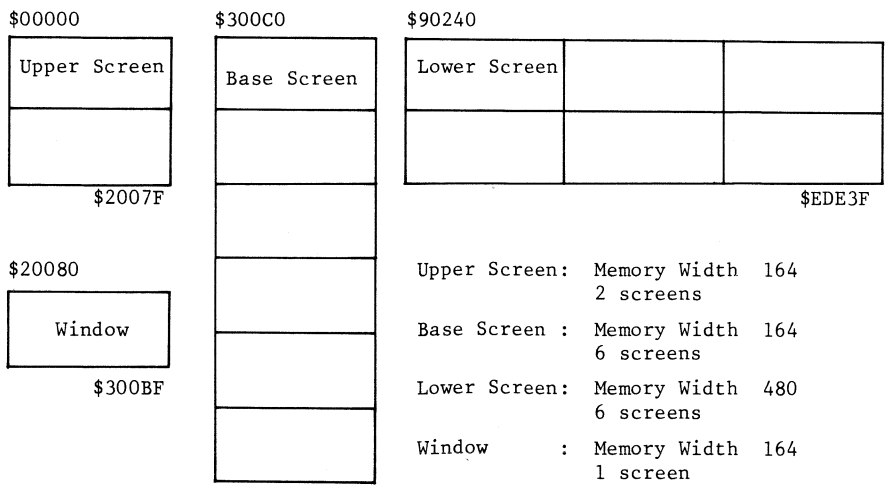


Fig. 1-18 Split Screens Configuration

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      $0F03         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

$\overline{CS}$	RS	R/$\overline{W}$	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		ATR								$4000
		-	r08 ↓ r7E	(undefined)		—	.....																—
		1	r80	Raster Count		RCR	.....				RC												—
		1/0	r82	Horizontal Sync.		HSR	HC								.....				HSW				$680A
			r84	Horizontal Display		HDR	HDS								HDW								$0B50
		1/0	r86	Vertical Sync.		VSR	.....				VC												$01C0
		1/0	r88	Vertical Display		VDR	VDS								.....				VSW				$2010
		1/0	r8A	Split Screen Width		SSW	.....				SP1												$0190
		1/0	r8C				.....				SP0												$0000
		1/0	r8E				.....				SP2												$0000
		1/0	r90	Blink Control		BCR	BON1				BOFF1				BON2				BOFF2				$0000
		1/0	r92	Horizontal Window Display		HWR	HWS								HWW								$0000
		1/0	r94	Vertical Window Display		VWR	.....				VWS												$0000
		1/0	r96				.....				VWW												$0000
		1/0	r98	Graphic Cursor		GCR	CXE								CXS								$0000
		1/0	r9A				.....				CYS												$0000
		1/0	r9C				.....				CYE												$0000
		-	r9E	(undefined)		—																	—
		-	rA0 ↓ rBE				.....																
0	1	1/0	rC0	Base Screen	Raster Addr.0	RAR0	.....				LRA0				.....				FRA0				$0000
		1/0	rC2		Memory Width 0	MWR0	CHR	.....							MW0								$00A4
		1/0	rC4		Start Addr.0	SAR0	.....				SDA0				.....				SA0H/SRA0				$0F00
		1/0	rC6				SA0L																$0000
		1/0	rC8		Raster Addr.1	RAR1	.....				LRA1				.....				FRA1				$0000
		1/0	rCA		Memory Width 1	MWR1	CHR	.....							MW1								$00A4
		1/0	rCC		Start Addr. 1	SAR1	.....				SDA1				.....				SA1H/SRA1				$0F03
		1/0	rCE				SA1L																$00C0
		1/0	rD0		Raster Addr.2	RAR2	.....				LRA2				.....				FRA2				$0000
		1/0	rD2		Memory Width 2	MWR2	CHR	.....							MW2								$01E0
		1/0	rD4		Start Addr.2	SAR2	.....				SDA2				.....				SA2H/SRA2				$0F09
		1/0	rD6				SA2L																$0240
		1/0	rD8		Raster Addr.3	RAR3	.....				LRA3				.....				FRA3				$0000
		1/0	rDA		Memory Width 3	MWR3	CHR	.....							MW3								$00A4
		1/0	rDC		Start Addr.3	SAR3	.....				SDA3				.....				SA3H/SRA3				$0002
		1/0	rDE				SA3L																$0080
		1/0	rE0	Block Cursor 1		BCUR1	BCW1				BCSR1				.....				BCER1				$0000
		1/0	rE2				BCA1																$0000
		1/0	rE4	Block Cursor 2		BCUR2	BCW2				BCSR2				.....				BCER2				$0000
		1/0	rE6				BCA2																$0000
		1/0	rE8	Cursor Definition		CDR	CM		CON1			COFF1			.....		CON2			COFF2			$0000
		1/0	rEA	Zoom Factor		ZFR	HZF				VZF				.....								$0000
		1	rEC	Light Pen Address		LPAR	.....								CHR	.....			LPAH				—
		1	rEE				LPAL																—
		-	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

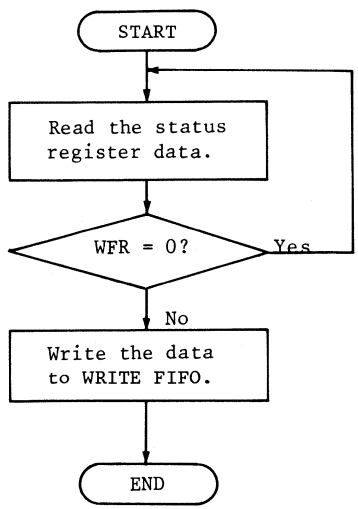


Fig. 2-2 One-Word Transfer Flowchart

2.2 Block Transfer

The program which continuously transfers plural commands and parameters to the ACRTC is shown below. The program transfers 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

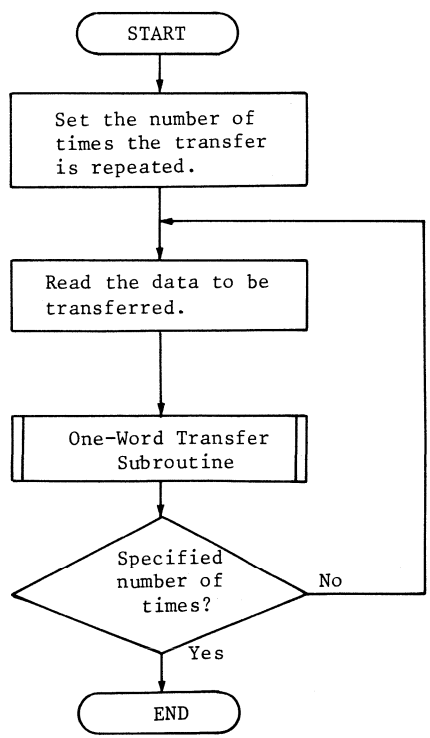


Fig. 2-4 Block Transfer Flowchart

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, $4030                   ]
        DC    WPR+$D, $0C00                   ]  Set Read/Write Pointer
        DC    CLR, $0000, 164, -400           :  Clear the screen
        DC    ORG, $4030, $0C00               :  Specify the drawing screen
        DC    WPTN, 16                        ]
        DC    -1, -1, -1, -1, -1, -1, -1, -1  ]  Write $FFFF to the pattern
        DC    -1, -1, -1, -1, -1, -1, -1, -1  ]  RAM.
        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     %0001000000000000
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     %1110010000000000
```

Fig. 2-6 ACRTC Command Table

3. DIRECTION 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 positions 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" is valid, thereby setting the pixel address in unit 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".

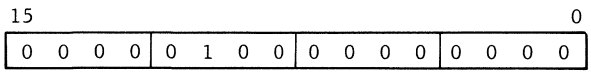
Fig. 3-1 shows an example of the ORG command on the base screen.

<Mnemonic>

ORG DPH, DPL

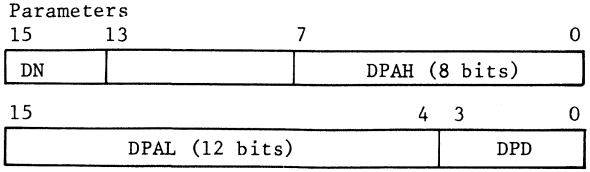
<Command Format>

Operation Code



DN	Screen No.
00	Upper Screen
01	Base Screen
10	Lower Screen
11	Window

Parameters

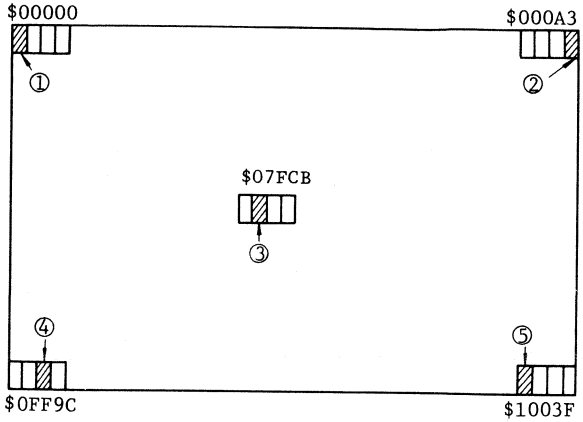


Note)

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

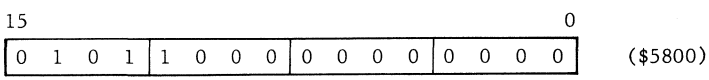
The "CLR" command is used to clear the drawing screen. The "CLR" command clears portions of the frame buffer area by writing a specified color code to the area to be cleared. The area is specified by the command parameters and read/write pointer in the drawing parameter register where the address is 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 unit of words, therefore, the color is specified by unit 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>

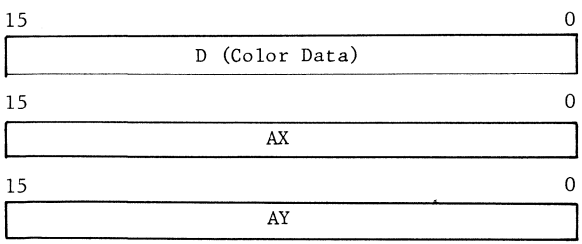
CLR D, AX, AY

<Command Format>

Operation Code



Parameters



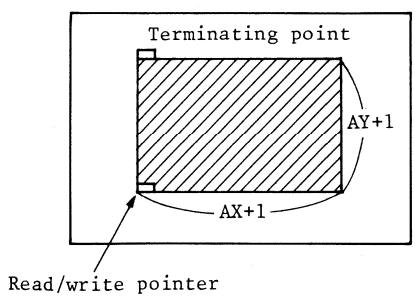


Fig. 3-3 CLR Command

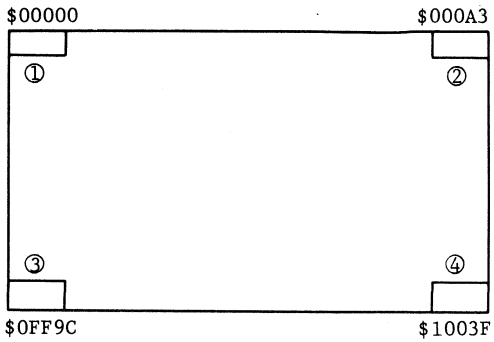


Fig. 3-4 Drawing Screen Example

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

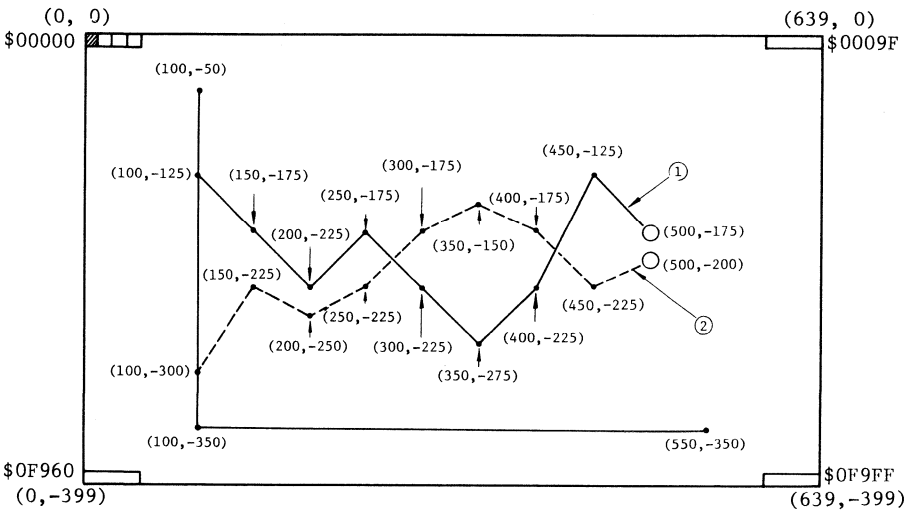
```
DC      7
DC      WPR + $C, $4010
DC      WPR + $D, $03F0
DC      CLR, $0000, -163, 399
```

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

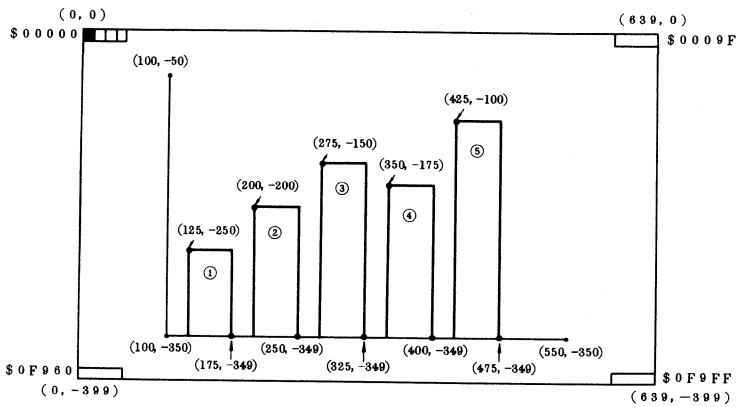
Fig. 3-5 Poly-Line Chart Example

```
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 a 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 a 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

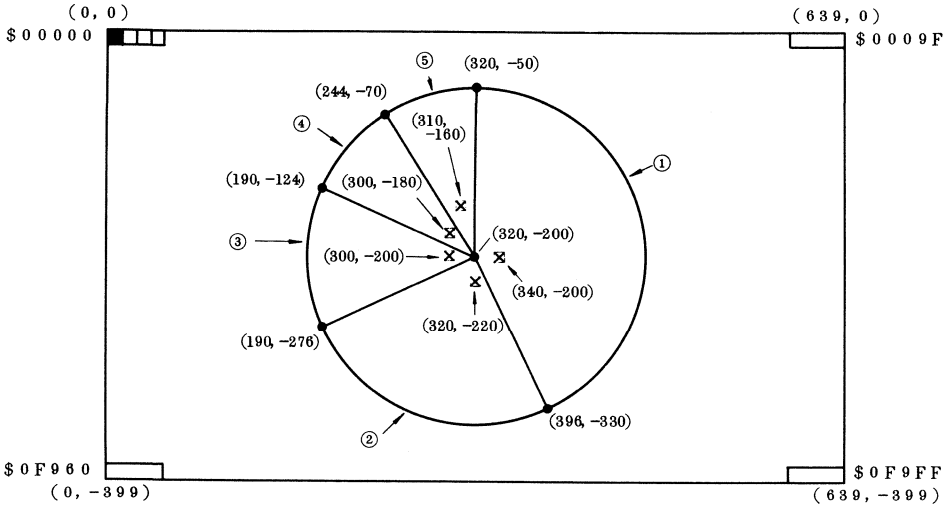
Fig. 3-7 Bar Chart Example

```
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 a 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 a 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 a 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 a 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 a 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       ]  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   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

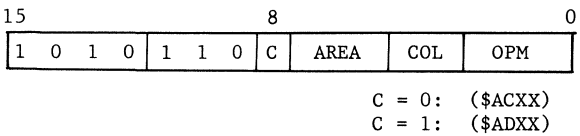
The "ELPS" command is used to draw ellipses. The parameter setup procedure necessary for ellipse drawing is shown below.

<Mnemonic>

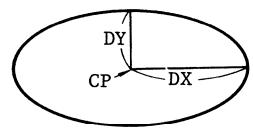
ELPS a,b, DX

<Command Format>

Operation Code



C = 0: ($ACXX)
C = 1: ($ADXX)



$a : b = DX^2 : DY^2$

Parameters

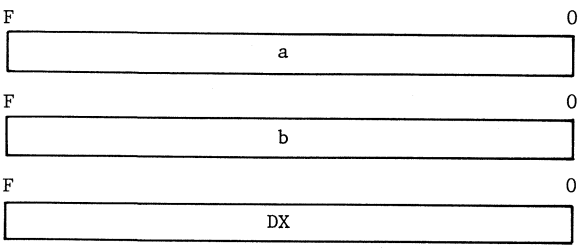


Fig. 3-11 ELPS Command

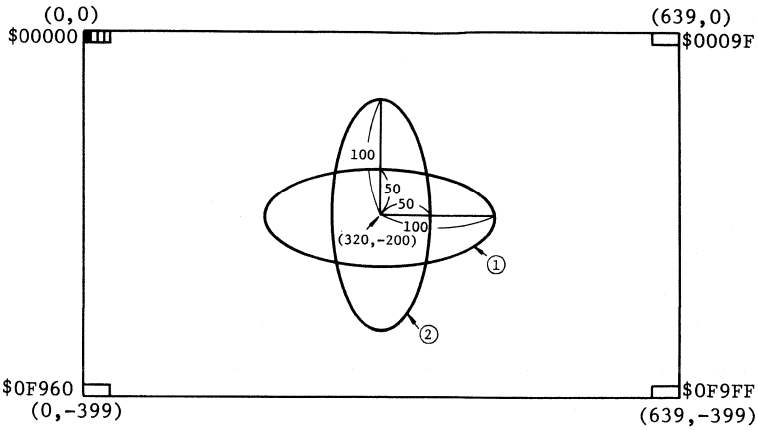
"a" and "b", the ratio of the square of the radius in X axis (DX) radius, and the Y axis (DY) radius, as well as "DX" value are set to the necessary parameters. Supposing DX = 10 and DY = 6, the value of "a" and "b" is calculated as follows:

$$DX = 10 \qquad DY = 6$$
$$(a : b) = (10^2 : 6^2) = (100 : 36) = (25 : 9)$$
$$a = 25, \quad b = 9$$

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 more 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 the ellipse 1.
          DC   ELPS, 1, 4, 50         :  Draw the 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 drawn 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.

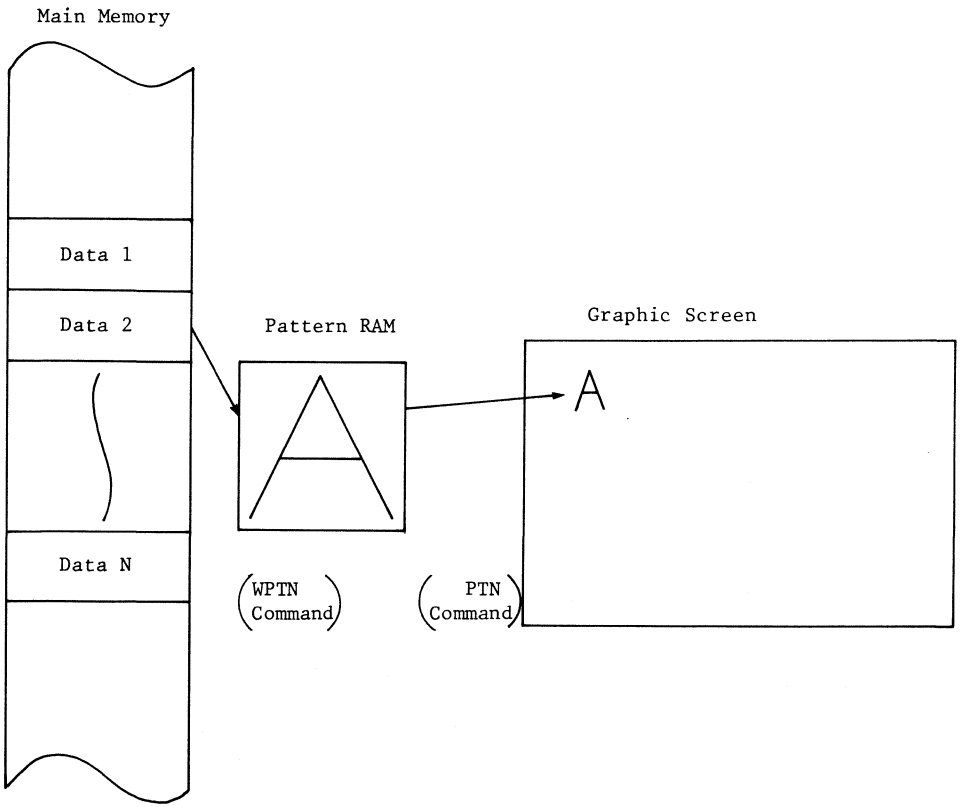


Fig. 3-14 Character Drawing Example

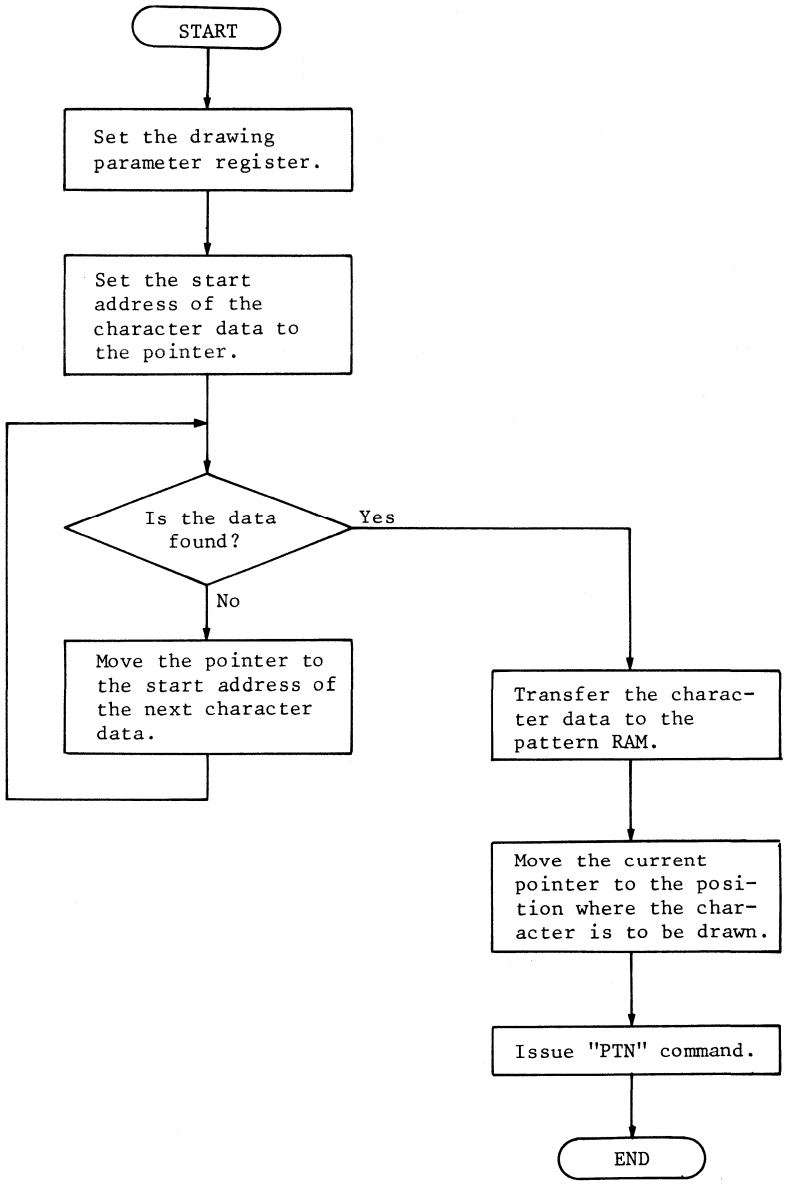


Fig. 3-15 Character Drawing Example Flowchart

```
*
*****************************
*                           *
*       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      $2330                    0
         DC      $0000, $03E0, $0410, $0808, $0808, $0808, $0808, $0808
         DC      $0808, $0808, $0808, $0808, $0808, $0410, $03E0, $0000
*
         DC      $2331                    1
         DC      $0000, $03E0, $0080, $0080, $0080, $0080, $0080, $0080
         DC      $0080, $0080, $0080, $0080, $00A0, $00C0, $0080, $0000
*
         DC      $2332                    2
         DC      $0000, $0FF8, $0808, $0008, $0010, $0020, $0040, $0180
         DC      $0200, $0400, $0800, $0808, $0808, $0410, $03E0, $0000
*
         DC      $2333                    3
         DC      $0000, $03E0, $0410, $0808, $0800, $0800, $0800, $0400
         DC      $0380, $0400, $0800, $0800, $0808, $0410, $03E0, $0000
*
         DC      $2334                    4
         DC      $0000, $07C0, $0100, $0100, $0FFC, $0104, $0108, $0108
         DC      $0110, $0120, $0120, $0140, $0140, $0180, $0100, $0000
*
         DC      $2335                    5
         DC      $0000, $03E0, $0410, $0808, $0800, $0800, $0800, $0808
         DC      $0418, $03E8, $0008, $0008, $0008, $0008, $07F8, $0000
*
         DC      $2336                    6
         DC      $0000, $03E0, $0410, $0808, $0808, $0808, $0808, $0808
         DC      $0418, $03E8, $0008, $0008, $0808, $0410, $03E0, $0000
*
         DC      $2337                    7
         DC      $0000, $0100, $0100, $0100, $0100, $0100, $0100, $0100
         DC      $0200, $0200, $0400, $0408, $0808, $0808, $0FF8, $0000
*
         DC      $2338                    8
         DC      $0000, $03E0, $0410, $0808, $0808, $0808, $0808, $0410
         DC      $03E0, $0410, $0808, $0808, $0808, $0410, $03E0, $0000
*
         DC      $2339                    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 paint-int (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: The color other than the data in the "EDG" register is used as the edge color.

<Mnemonic>

PAINT

<Command Format>

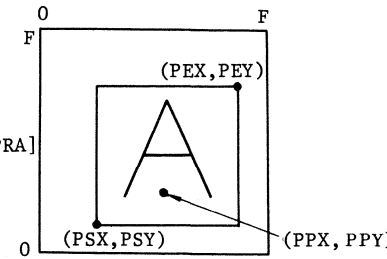
Operation Code

1	1	0	0	1	0	0	E	AREA	0	0	0	0	0

[PRA]

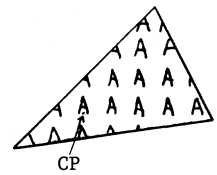
E=0: ($C8X0)
E=1: ($C9XX)

Pattern RAM



Note) COL and OPM is not programmable for the PAINT command. It is 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.

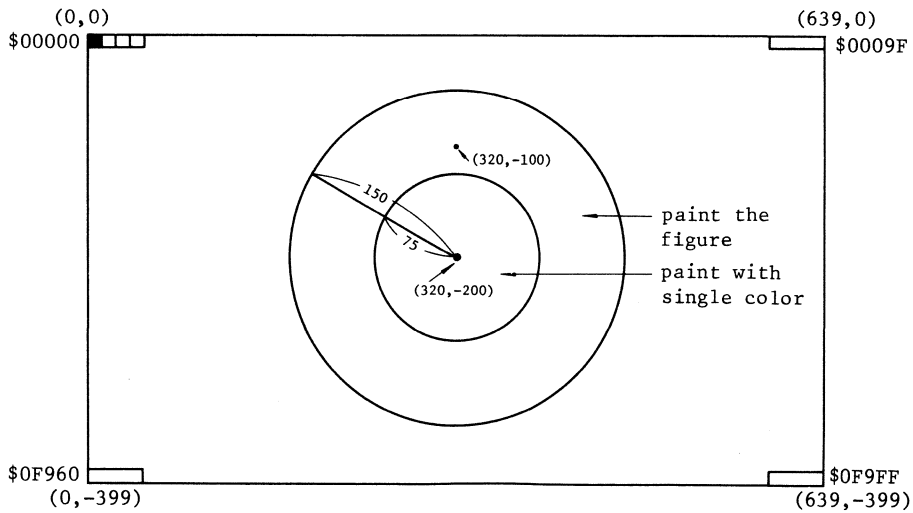
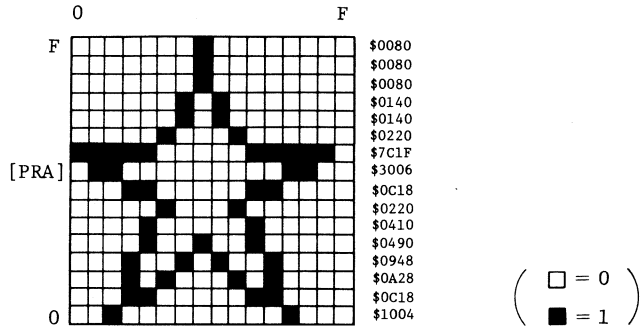


Fig. 3-19 Painting Example

Base Screen
4 bits / pixel mode
Memory Width: 160 words
Display Start Address: $0000



Pattern RAM Data

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.

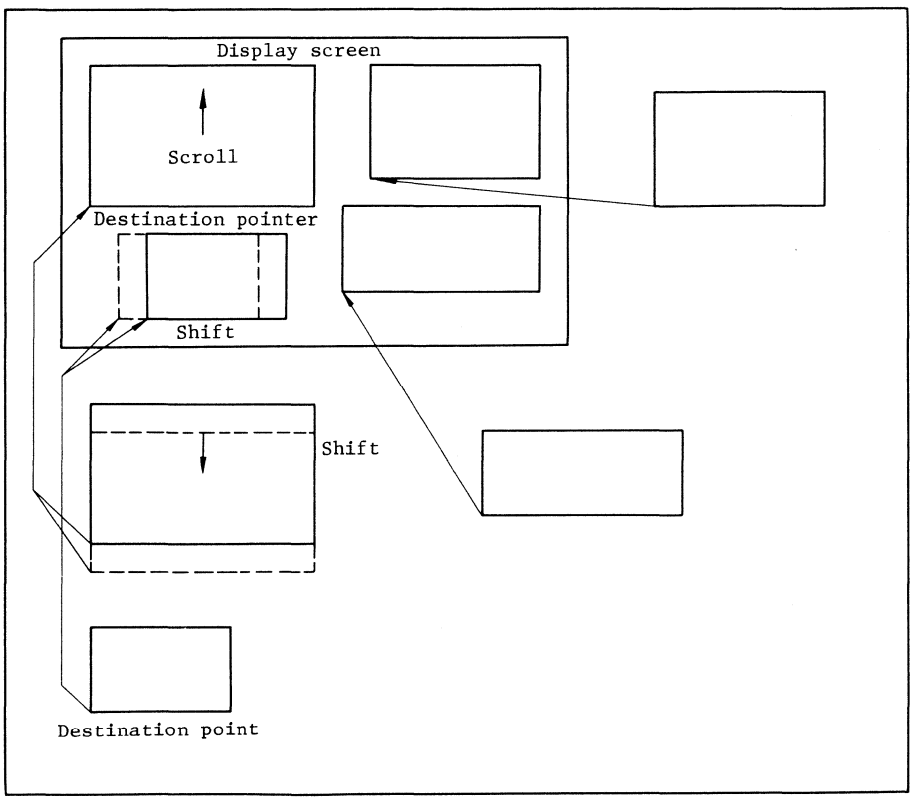


Fig. 3-21 Software Multi-window Support Example

4. SCREEN CONTROL

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.

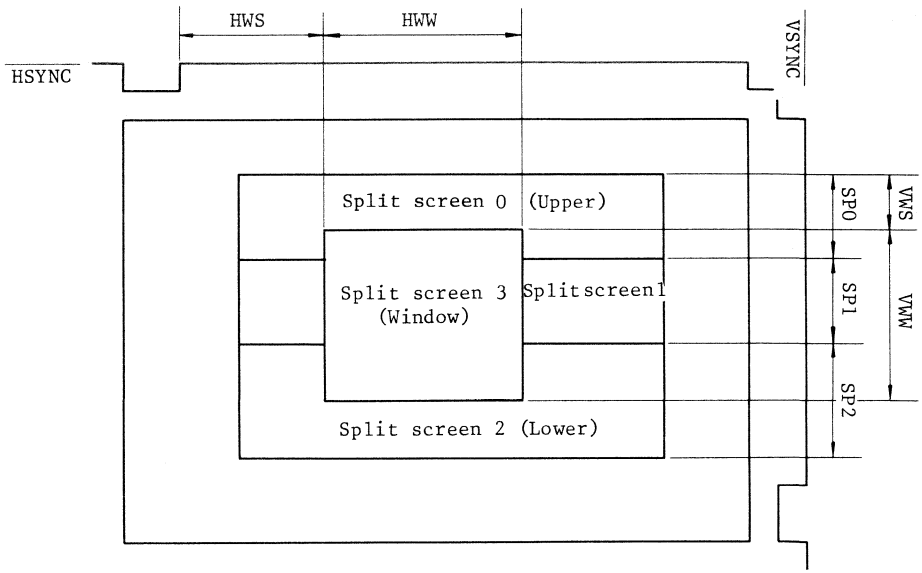


Fig. 4-1 Split Screen Configuration

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 ---------- (1)

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.

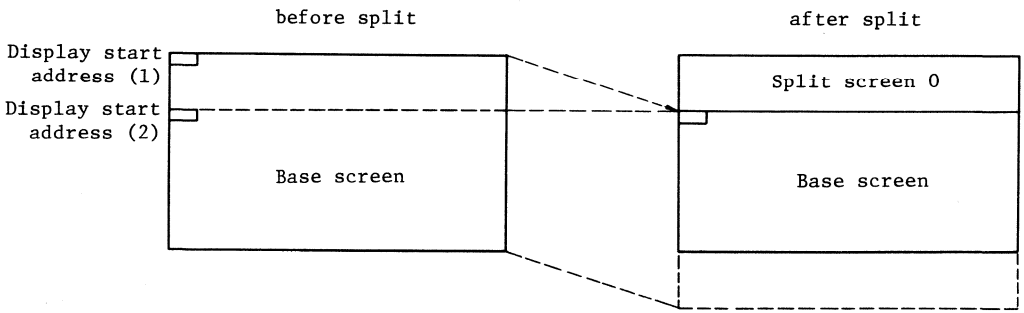


Fig. 4-2 Relation between Base Screen and Split Screen

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.

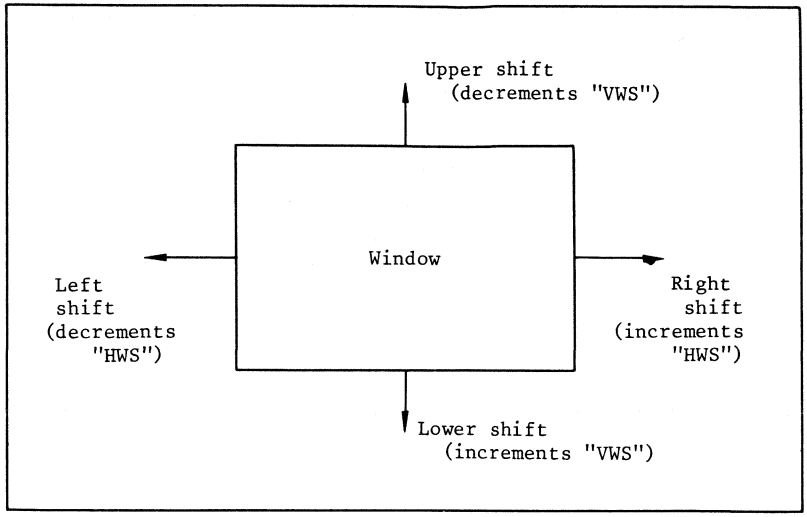


Fig. 4-3 Window Shift

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.

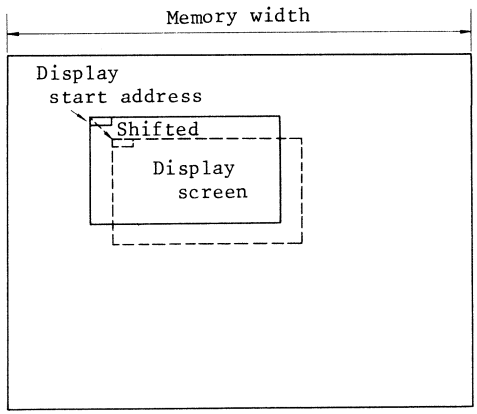


Fig. 4-4 Scroll and the Display Screen

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.

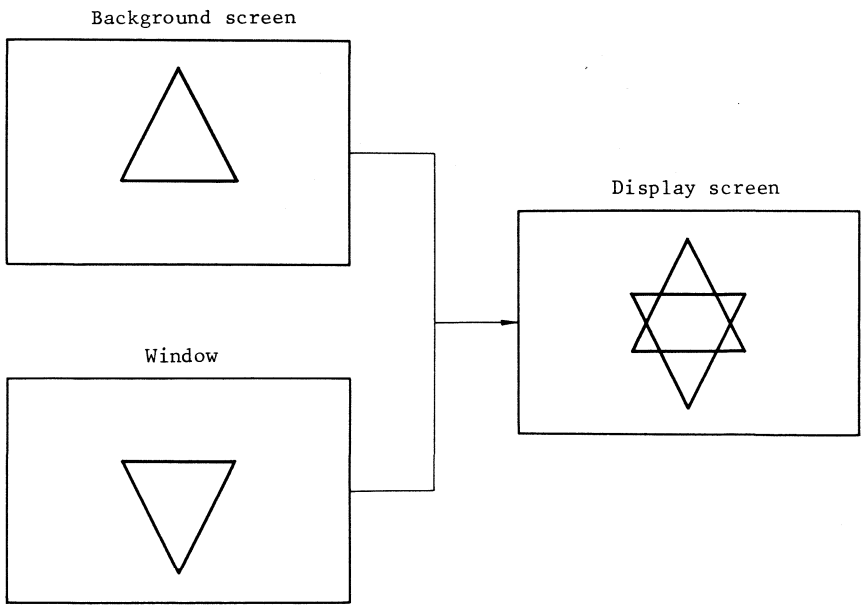


Fig. 4-9 Superimposing

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.

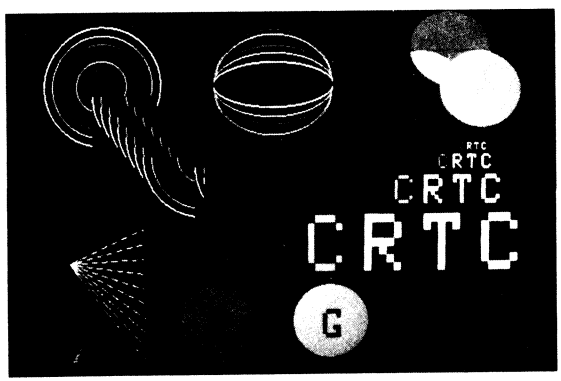


Fig. 5-1 Drawing Example (ACRTC)

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.

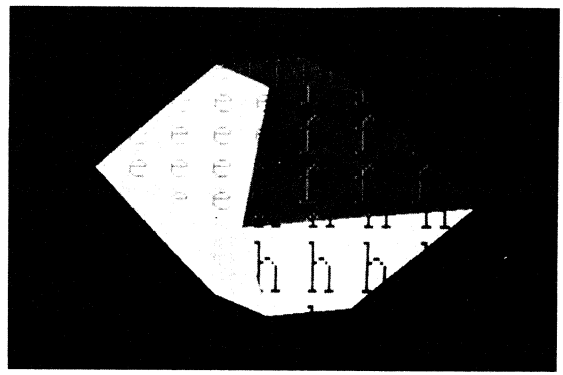


Fig. 5-2 Drawing Example (Painting the Polygons)

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.

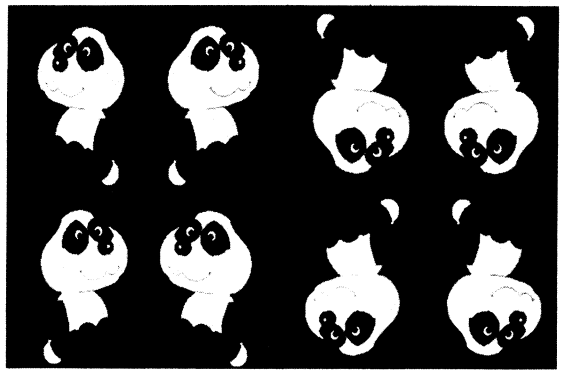


Fig. 5-3 Drawing Example (Panda Bear)

```
 1                                    OPT     CEX
 2                00A00000    ACRTC   EQU     $A00000
 3                            *
 4                            ****************************
 5                            *                          *
 6                            *   ACRTC COMMAND TABLE    *
 7                            *                          *
 8                            ****************************
 9                            *
10                00000400    ORG     EQU     %0000010000000000
11                00000800    WPR     EQU     %0000100000000000
12                00000C00    RPR     EQU     %0000110000000000
13                00001800    WPTN    EQU     %0001100000000000
14                00001C00    RPTN    EQU     %0001110000000000
15                00002400    DRD     EQU     %0010010000000000
16                00002800    DWT     EQU     %0010100000000000
17                00002C00    DMOD    EQU     %0010110000000000
18                            *
19                00004400    RD      EQU     %0100010000000000
20                00004800    WT      EQU     %0100100000000000
21                00004C00    MOD     EQU     %0100110000000000
22                00005800    CLR     EQU     %0101100000000000
23                00005C00    SCLR    EQU     %0101110000000000
24                00006000    CPY     EQU     %0110000000000000
25                00007000    SCPY    EQU     %0111000000000000
26                            *
27                00008000    AMOVE   EQU     %1000000000000000
28                00008400    RMOVE   EQU     %1000010000000000
29                00008800    ALINE   EQU     %1000100000000000
30                00008C00    RLINE   EQU     %1000110000000000
31                00009000    ARCT    EQU     %1001000000000000
32                00009400    RRCT    EQU     %1001010000000000
33                00009800    APLL    EQU     %1001100000000000
34                00009C00    RPLL    EQU     %1001110000000000
35                0000A000    APLG    EQU     %1010000000000000
36                0000A400    RPLG    EQU     %1010010000000000
37                0000A800    CRCL    EQU     %1010100000000000
38                0000AC00    ELPS    EQU     %1010110000000000
39                0000B000    AARC    EQU     %1011000000000000
40                0000B400    RARC    EQU     %1011010000000000
41                0000B800    AEARC   EQU     %1011100000000000
42                0000BC00    REARC   EQU     %1011110000000000
43                0000C000    AFRCT   EQU     %1100000000000000
44                0000C400    RFRCT   EQU     %1100010000000000
45                0000C800    PAINT   EQU     %1100100000000000
46                0000CC00    DOT     EQU     %1100110000000000
47                0000D000    PTN     EQU     %1101000000000000
48                0000E000    AGCPY   EQU     %1110000000000000
49                0000F000    RGCPY   EQU     %1111000000000000
50                            *
51  0 00000000 60000092               BRA     INIT
52                            *
53                            ******************************
54                            *                            *
55                            *   ACRTC INITIALIZE DATA    *
56                            *                            *
57                            ******************************
58                            *
```

Fig. 5-4 (1) (ACRTC)

```
68000 MACRO ASSEMBLER 1.0          ACRTC   .SA 12/13/84 10:09:38

 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:SPLIT SCREEN WIDTH  SP1 (BASE)
 64 0 0000000E 0000             DC      0                  R8C:SPLIT SCREEN WIDTH  SP0 (UPPER)
 65 0 00000010 0000             DC      0                  R8E: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 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. 5-4 (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                          *     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    #$82,ACRTC              R82 SELECT
                 0000
153 0 000000A0 343C000D              MOVE    #13,D2                  LOOP COUNTER -> D2
154 0 000000A4 33D900A00002  INIT1   MOVE    (A1)+,ACRTC+2           R82-R9D WRITE
155 0 000000AA 51CAFFF8              DBRA    D2,INIT1
156                          *
157 0 000000AE 33FC00C000A0          MOVE    #$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 33FC000600A0          MOVE    #$06,ACRTC              R06 SELECT
                 0000
```

Fig. 5-4 (3) (ACRTC)

```
68000 MACRO ASSEMBLER 1.0          ACRTC   .SA 12/13/84 10:09:38

169 0 000000E8 33D900A00002          MOVE      (A1)+,ACRTC+2          R06 WRITE
170                                  *
171                                  ****************************
172                                  *                          *
173                                  *       D  E  M  O   1      *
174                                  *                          *
175                                  ****************************
176                                  *
177 0 000000EE 43FA001A             DEMO1     LEA       DATA1(PC),A1
178                                  *
179 0 000000F2 6100FF5E                       BSR       CTWRTE
180 0 000000F6 6100FF5A                       BSR       CTWRTE
181                                  *
182 0 000000FA 203C0007FFFF         DELAY     MOVE.L    #$7FFFF,D0
183 0 00000100 048000000001         DD1       SUBI.L    #1,D0
184 0 00000106 66F8                           BNE       DD1
185                                  *
186 0 00000108 60E4                           BRA       DEMO1
187                                  *
188                                  ******************
189                                  *                *
190                                  *   D A T A  1   *
191                                  *                *
192                                  ******************
193                                  *
194                 00000140         X0        SET       320
195                 FFFFFF88         Y0        SET       -120
196                                  *
197 0 0000010A 000A                 DATA1     DC        10
198 0 0000010C 040040300C00                   DC        ORG,$4030,$0C00
199 0 00000112 080C4020                       DC        WPR+$C,$4020           R/W POINTER-$40200800
200 0 00000116 080D0800                       DC        WPR+$D,$0800
201 0 0000011A 5800000000A3                   DC        CLR,$0000,163,-800
                FCE0
202                                  *
203 0 00000122 0306                           DC        774
204 0 00000124 18000010                       DC        WPTN,16
205 0 00000128 040F04110A11                   DC        $040F,$0411,$0A11,$0A0F,$1111,$1111,$110F,$0000
                0A0F11111111
                110F0000
206 0 00000138 1E1111091905                   DC        $1E11,$1109,$1905,$010F,$0111,$1111,$1E0F,$0000
                010F01111111
                1E0F0000
207                                  *
208 0 00000148 0800CCCC                       DC        WPR,$CCCC              CL0 - $CCCC
209 0 0000014C 0801CCCC                       DC        WPR+1,$CCCC            CL1 - $CCCC
210 0 00000150 80000208FFD8                   DC        AMOVE,200+X0,80+Y0
211 0 00000156 AC00000A0007                   DC        ELPS,10,7,44
                002C
212 0 0000015E 0803CCCC                       DC        WPR+3,$CCCC            EDG - $CCCC
213 0 00000162 C800                           DC        PAINT
214 0 00000164 0800AAAA                       DC        WPR,$AAAA              CL0 - $AAAA
215 0 00000168 0801AAAA                       DC        WPR+1,$AAAA            CL1 - $AAAA
216 0 0000016C 80000208FF52                   DC        AMOVE,200+X0,-54+Y0
217 0 00000172 AC00000A0007                   DC        ELPS,10,7,44
                002C
218 0 0000017A 0803AAAA                       DC        WPR+3,$AAAA            EDG - $AAAA
219 0 0000017E C800                           DC        PAINT
```

Fig. 5-4 (4) (ACRTC)

```
220 0 00000180 08009999              DC      WPR,$9999              CL0 - $9999
221 0 00000184 08019999              DC      WPR+1,$9999            CL1 - $9999
222 0 00000188 80000208FECC          DC      AMOVE,200+X0,-188+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+X0,26+Y0
227 0 000001A2 E00001D2FFA2          DC      AGCPY,146+X0,26+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 800000C6FFA2          DC      AMOVE,-122+X0,26+Y0
233 0 000001D2 E00001D2FFA2          DC      AGCPY,146+X0,26+Y0,108,108
                   006C006C
234 0 000001DC 800000C6FF1C          DC      AMOVE,-122+X0,-108+Y0
235 0 000001E2 E00001D2FF1C          DC      AGCPY,146+X0,-108+Y0,108,108
                   006C006C
236 0 000001EC 800000C6FE96          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+X0,26+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+X0,26+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 800000C6FE96          DC      AMOVE,-122+X0,-242+Y0
249 0 00000242 E00001D2FFA2          DC      AGCPY,146+X0,26+Y0,108,108
                   006C006C
250 0 0000024C 800000C6FFA2          DC      AMOVE,-122+X0,26+Y0
251 0 00000252 E00001D2FE96          DC      AGCPY,146+X0,-242+Y0,108,108
                   006C006C
252 0 0000025C 800000C6FF1C          DC      AMOVE,-122+X0,-108+Y0
253 0 00000262 E00000C6FE96          DC      AGCPY,-122+X0,-242+Y0,108,108
                   006C006C
254                              *
255 0 0000026C 800000C6FF1C          DC      AMOVE,-122+X0,-108+Y0
256 0 00000272 E00100C6FFA2          DC      AGCPY+1,-122+X0,26+Y0,108,108
                   006C006C
257                              *
258 0 0000027C 8000014CFF1C          DC      AMOVE,12+X0,-108+Y0
259 0 00000282 E003014CFF1C          DC      AGCPY+3,12+X0,-108+Y0,108,108
                   006C006C
260                              *
261 0 0000028C 800000C6FFA2          DC      AMOVE,-122+X0,26+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 8000014CFE96          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 E001014CFE96          DC      AGCPY+1,12+X0,-242+Y0,108,108
                      006C006C
274 0 000002EC 800001B2FF79          DC      AMOVE,114+X0,-15+Y0
275 0 000002F2 E00101D2FE96          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      PTN+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      PTN+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      PTN+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 - $FFFF
318 0 000003BC AC00000A0007      DC      ELPS,10,7,70
                        0046
319 0 000003C4 80000140FFB5      DC      AMOVE,0+X0,45+Y0
320 0 000003CA 0800FFFF          DC      WPR,$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 0E111111011F      DC      $0E11,$1111,$011F,$0111,$0111,$110A,$0E04,$0000
                        011101111110A
                        0E040000
344 0 0000045E 80000208FF74      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 - $9999
350 0 00000478 D0000705          DC      PTN,7*256+5
351 0 0000047C 800001E8FF62      DC      AMOVE,168+X0,-38+Y0
352 0 00000482 0807F151          DC      WPR+7,$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)

```
411 0 0000058E 080771D1              DC      WPR+7,$71D1             PE,PZ
412 0 00000592 D0000F0B              DC      PTN,15*256+11
413 0 00000596 80000208FF40          DC      AMOVE,200+X0,-72+Y0
414 0 0000059C 080773D3              DC      WPR+7,$73D3             PE,PZ
415 0 000005A0 D0001F17              DC      PTN,31*256+23
416 0 000005A4 800001E8FEFE          DC      AMOVE,168+X0,-138+Y0
417 0 000005AA 080777D7              DC      WPR+7,$77D7             PE,PZ
418 0 000005AE D0003F2F              DC      PTN,63*256+47
419 0 000005B2 8000006EFF65          DC      AMOVE,-210+X0,-35+Y0
420 0 000005B8 E0000032FF6F          DC      AGCPY,-270+X0,-25+Y0,70,70
                   00460046
421 0 000005C2 80000078FF5B          DC      AMOVE,-200+X0,-45+Y0
422 0 000005C8 E0000032FF6F          DC      AGCPY,-270+X0,-25+Y0,70,70
                   00460046
423 0 000005D2 80000082FF51          DC      AMOVE,-190+X0,-55+Y0
424 0 000005D8 E0000032FF6F          DC      AGCPY,-270+X0,-25+Y0,70,70
                   00460046
425 0 000005E2 8000008CFF47          DC      AMOVE,-180+X0,-65+Y0
426 0 000005E8 E0000032FF6F          DC      AGCPY,-270+X0,-25+Y0,70,70
                   00460046
427 0 000005F2 80000096FF3D          DC      AMOVE,-170+X0,-75+Y0
428 0 000005F8 E0000032FF6F          DC      AGCPY,-270+X0,-25+Y0,70,70
                   00460046
429 0 00000602 800000A0FF33          DC      AMOVE,-160+X0,-85+Y0
430 0 00000608 E0000032FF6F          DC      AGCPY,-270+X0,-25+Y0,70,70
                   00460046
431 0 00000612 800000AAFF29          DC      AMOVE,-150+X0,-95+Y0
432 0 00000618 E0000032FF6F          DC      AGCPY,-270+X0,-25+Y0,70,70
                   00460046
433                            *
434 0 00000622 0800AAAA              DC      WPR,$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,$2333             PE,PZ
439 0 00000636 80000050FEA2          DC      AMOVE,-240+X0,-230+Y0
440 0 0000063C 880000AAFEA2          DC      ALINE,-150+X0,-230+Y0
441 0 00000642 80000050FEA2          DC      AMOVE,-240+X0,-230+Y0
442 0 00000648 880000AAFEB1          DC      ALINE,-150+X0,-215+Y0
443 0 0000064E 80000050FEA2          DC      AMOVE,-240+X0,-230+Y0
444 0 00000654 880000AAFEC0          DC      ALINE,-150+X0,-200+Y0
445 0 0000065A 80000050FEA2          DC      AMOVE,-240+X0,-230+Y0
446 0 00000660 880000AAFECF          DC      ALINE,-150+X0,-185+Y0
447 0 00000666 80000050FEA2          DC      AMOVE,-240+X0,-230+Y0
448 0 0000066C 880000AAFEDE          DC      ALINE,-150+X0,-170+Y0
449 0 00000672 80000050FEA2          DC      AMOVE,-240+X0,-230+Y0
450 0 00000678 880000AAFEED          DC      ALINE,-150+X0,-155+Y0
451 0 0000067E 80000050FEA2          DC      AMOVE,-240+X0,-230+Y0
452 0 00000684 880000AAFEFC          DC      ALINE,-150+X0,-140+Y0
453 0 0000068A 80000050FEA2          DC      AMOVE,-240+X0,-230+Y0
454 0 00000690 880000AAFF0B          DC      ALINE,-150+X0,-125+Y0
455 0 00000696 80000050FEA2          DC      AMOVE,-240+X0,-230+Y0
456 0 0000069C 880000AAFF1A          DC      ALINE,-150+X0,-110+Y0
457 0 000006A2 80000050FEA2          DC      AMOVE,-240+X0,-230+Y0
458 0 000006A8 880000AAFF29          DC      ALINE,-150+X0,-95+Y0
459 0 000006AE 0800DDDD              DC      WPR,$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 80000050FF06          DC        AMOVE,-240+X0,-130+Y0
 463 0 000006C0 880000AAFF29          DC        ALINE,-150+X0,-95+Y0
 464 0 000006C6 80000050FF06          DC        AMOVE,-240+X0,-130+Y0
 465 0 000006CC 880000AAFF1A          DC        ALINE,-150+X0,-110+Y0
 466 0 000006D2 80000050FF06          DC        AMOVE,-240+X0,-130+Y0
 467 0 000006D8 880000AAFF0B          DC        ALINE,-150+X0,-125+Y0
 468 0 000006DE 80000050FF06          DC        AMOVE,-240+X0,-130+Y0
 469 0 000006E4 880000AAFEFC          DC        ALINE,-150+X0,-140+Y0
 470 0 000006EA 80000050FF06          DC        AMOVE,-240+X0,-130+Y0
 471 0 000006F0 880000AAFEED          DC        ALINE,-150+X0,-155+Y0
 472 0 000006F6 80000050FF06          DC        AMOVE,-240+X0,-130+Y0
 473 0 000006FC 880000AAFEDE          DC        ALINE,-150+X0,-170+Y0
 474 0 00000702 80000050FF06          DC        AMOVE,-240+X0,-130+Y0
 475 0 00000708 880000AAFECF          DC        ALINE,-150+X0,-185+Y0
 476 0 0000070E 80000050FF06          DC        AMOVE,-240+X0,-130+Y0
 477 0 00000714 880000AAFEC0          DC        ALINE,-150+X0,-200+Y0
 478 0 0000071A 80000050FF06          DC        AMOVE,-240+X0,-130+Y0
 479 0 00000720 880000AAFEB1          DC        ALINE,-150+X0,-215+Y0
 480 0 00000726 80000050FF06          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                     00Q0B000    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                     00000C00
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)

```
68000 MACRO ASSEMBLER 1.0          PAINT   .SA 12/13/84 10:00:28

                                        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. 5-5 (1) (Painting the Polygons)

```
68000 MACRO ASSEMBLER 1.0          PAINT   .SA 12/13/84 10:00:28

  59 0 00000004 680A          INITTBL  DC       $680A                 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. 5-5 (2) (Painting the Polygons)

```
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 INITIALIZE      *
147                          *                         *
148                          ***************************
149                          *
150 0 00000094 43FAFF6E      INIT     LEA      INITTBL(PC),A1
151                          *
152 0 00000098 33FC008200A0           MOVE     #$82,ACRTC            R82 SELECT
                 0000
153 0 000000A0 343C000D               MOVE     #13,D2                LOOP COUNTER -> D2
154 0 000000A4 33D900A00002 INIT1     MOVE     (A1)+,ACRTC+2         R82-R9D WRITE
155 0 000000AA 51CAFFF8               DBRA     D2,INIT1
156                          *
157 0 000000AE 33FC00C000A0           MOVE     #$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 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 00000126 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 A000000030156           DC        APLG,3,342,-52,402,-75,301,-82
                FFCC0192FFB5
                012DFFAE
 213 0 0000015E 980000030068            DC        APLL,3,104,-160,272,-220,301,-82
                FF600110FF24
                012DFFAE
 214 0 0000016E 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,-286,303,-308
                FEE2012FFECC
```

Fig. 5-5 (4) (Painting the Polygons)

```
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
                094804900410
                02200C18
231  0 000001E8 30067C1F0220            DC      $3006,$7C1F,$0220,$0140,$0140,$0080,$0080,$0080
                014001400080
                00800080
232  0 000001F8 C800                    DC      PAINT
233                                     *
234  0 000001FA 002C                    DC      44
235  0 000001FC 0800AAAA                DC      WPR,$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 0000061809A4            DC      $0000,$0618,$09A4,$1064,$1058,$1040,$0040,$0020
                106410581040
                00400020
246  0 00000244 07FC00200020            DC      $07FC,$0020,$0020,$0210,$0410,$0220,$01C0,$0000
                021004100220
                01C00000
247  0 00000254 C800                    DC      PAINT
248                                     *
249  0 00000256 002C                    DC      44
250  0 00000258 0800BBBB                DC      WPR,$BBBB               CL0 - $BBBB
251  0 0000025C 0801BBBB                DC      WPR+1,$BBBB             CL1 - $BBBB
252  0 00000260 0803BBBB                DC      WPR+3,$BBBB             EDG - $BBBB
253  0 00000264 80000012DFFAE           DC      AMOVE,301,-82
254  0 0000026A A00000030192            DC      APLG,3,402,-75,536,-200,272,-220
                FFB50218FF38
                0110FF24
255  0 0000027A 80000192FFB3            DC      AMOVE,402,-77
256  0 00000280 0800DDDD                DC      WPR,$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 - $CCCC
 268  0 000002C0 80000068FF60          DC      AMOVE,104,-160
 269  0 000002C6 A00000030110          DC      APLG,3,272,-220,303,-308,241,-286
                    FF24012FFECC
                    00F1FEE2
 270  0 000002D6 8000006EFF5C          DC      AMOVE,110,-164
 271  0 000002DC 0800EEEE              DC      WPR,$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
ACCPY                   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                     00006000     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)

```
68000 MACRO ASSEMBLER 1.0          PANDA   .SA 12/13/84 10:23:59

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 33FC000600A0            MOVE      #$06,ACRTC            R06 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       DEM03
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)

```
313                                 **********************
314                                 *
315  0 000002BA 33FC00DC00A0 DW     MOVE      #$DC,ACRTC          RDC SELECT
                0000
316  0 000002C2 303900A00002        MOVE      ACRTC+2,D0
317  0 000002C8 4840                SWAP      D0
318  0 000002CA 303900A00002        MOVE      ACRTC+2,D0
319                                 *
320  0 000002D0 0480000000A4        SUBI.L    #164,D0
321  0 000002D6 60C4                BRA       UD
322                                 *
323                                 **********************
324                                 *                    *
325                                 *  DELAY - CTWRTE    *
326                                 *                    *
327                                 **********************
328                                 *
329  0 000002D8 48A72000 DTWRTE     MOVEM     D2,-(A7)
330  0 000002DC 3419                MOVE      (A1)+,D2
331  0 000002DE 3019     DTWR       MOVE      (A1)+,D0
332  0 000002E0 6100FD86            BSR       CWRITE
333  0 000002E4 6100000C            BSR       DDELAY
334  0 000002E8 51CAFFF4 DTEE       DBRA      D2,DTWR
335  0 000002EC 4C9F0004            MOVEM     (A7)+,D2
336  0 000002F0 4E75                RTS
337                                 *
338  0 000002F2 323C0001 DDELAY     MOVE      #1,D1
339  0 000002F6 61000016 DDEY       BSR       RDELAY
340  0 000002FA 51C9FFFA            DBRA      D1,DDEY
341  0 000002FE 4E75                RTS
342                                 *
343  0 00000300 323C0077 PDELAY     MOVE      #119,D1
344  0 00000304 61000008 PPD        BSR       RDELAY
345  0 00000308 51C9FFFA            DBRA      D1,PPD
346  0 0000030C 4E75                RTS
347                                 *
348                                 **********************
349                                 *                    *
350                                 *   RASTER  DELAY    *
351                                 *                    *
352                                 **********************
353                                 *
354  0 0000030E 48E78000 RDELAY     MOVEM.L   D0,-(A7)
355  0 00000312 33FC008000A0 RDEY   MOVE      #$80,ACRTC          R80 SELECT
                0000
356  0 0000031A 303900A00002        MOVE      ACRTC+2,D0
357  0 00000320 0C400191            CMPI      #401,D0
358  0 00000324 66EC                BNE       RDEY
359  0 00000326 4CDF0001            MOVEM.L   (A7)+,D0
360  0 0000032A 4E75                RTS
361                                 *
362                                 ********************
363                                 *                  *
364                                 *   D A T A 3      *
365                                 *                  *
366                                 ********************
367                                 *
368  0 0000032C 01AD    DATA3       DC        429
```

Fig. 5-6 (7) (Panda Bear)

```
 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,$0C80
 373 0 00000344 08050000              DC      WPR+5,0               PP,PZC
 374 0 00000348 08060000              DC      WPR+6,0               PS
 375 0 0000034C 08070000              DC      WPR+7,0               PE,PZ
 376 0 00000350 0800FFFF              DC      WPR,$FFFF             CL0 - $FFFF
 377 0 00000354 0801FFFF              DC      WPR+1,$FFFF           CL1 - $FFFF
 378 0 00000358 0803FFFF              DC      WPR+3,$FFFF           EDG - $FFFF
 379                          *
 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 B00000730000          DC      AARC,115,0,134,-115
                0086FF8D
 387 0 0000039E 8000005AFFB0          DC      AMOVE,90,-80
 388 0 000003A4 C800                  DC      PAINT
 389 0 000003A6 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 80000055FFD5          DC      AMOVE,85,-43
 393 0 000003B8 B0000051FFE2          DC      AARC,81,-30,75,-19
                004BFFED
 394 0 000003C2 B0000062FFCC          DC      AARC,98,-52,60,-40
                003CFFD8
 395 0 000003CC B0000046FFD8          DC      AARC,70,-40,75,-48
                004BFFD0
 396 0 000003D6 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,-66
                0050FFBE
 400 0 000003FA B0000051FFCD          DC      AARC,81,-51,95,-56
                005FFFC8
 401 0 00000404 B0000058FFCD          DC      AARC,88,-51,85,-43
                0055FFD5
 402 0 0000040E 80000054FFC9          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 00000476 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,-54,169,-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 B00000BCFF61            DC      AARC,188,-159,120,-160
                 0078FF60
437 0 000004EE B0000071FF5D            DC      AARC,113,-163,109,-157
                 006DFF63
438 0 000004F8 B0000060FF5E            DC      AARC,96,-162,86,-152
                 0056FF68
439 0 00000502 B0000048FF63            DC      AARC,72,-157,57,-152
                 0039FF68
440 0 0000050C B0000060FF52            DC      AARC,96,-174,52,-172
                 0034FF54
441 0 00000516 B0000048FF52            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,-166
                 008DFF5A
445 0 0000053E B0000084FF62            DC      AARC,132,-158,143,-156
                 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)

```
449 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 B000006BFF6F             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+$100
470                             *
471 0 000005E6 80000083FF45             DC      AMOVE,131,-187
472 0 000005EC B0000078FF5B             DC      AARC,120,-165,140,-167
                008CFF59
473 0 000005F6 80000083FF45             DC      AMOVE,131,-187
474 0 000005FC B000008BFF4D             DC      AARC,139,-179,143,-166
                008FFF5A
475 0 00000606 0803FFFF                 DC      WPR+3,$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 - $CCCC
479 0 00000616 0801CCCC                 DC      WPR+1,$CCCC             CL1 - $CCCC
480 0 0000061A 8000007AFF5B             DC      AMOVE,122,-165
481 0 00000620 B0000085FF62             DC      AARC,133,-158,142,-165
                008EFF5B
482 0 0000062A 8000008AFF58             DC      AMOVE,138,-168
483 0 00000630 88000087FF5C             DC      ALINE,135,-164
484 0 00000636 80000082FF56             DC      AMOVE,130,-170
485 0 0000063C 88000082FF5C             DC      ALINE,130,-164
486 0 00000642 8000005AFF6A             DC      AMOVE,90,-150
487 0 00000648 8800005CFF60             DC      ALINE,92,-160
488 0 0000064E 80000065FF6B             DC      AMOVE,101,-149
489 0 00000654 88000066FF64             DC      ALINE,102,-156
490 0 0000065A 8000004BFF72             DC      AMOVE,75,-142
491 0 00000660 8800004AFF68             DC      ALINE,74,-152
492 0 00000666 8000003CFF6D             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 10:23:59

 496 0 0000067A 8000005AFF8E            DC        AMOVE,90,-114
 497 0 00000680 B00000730000            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
                FF6A00B5
 502 0 0000069C 8000001EFE85            DC        AMOVE,30,-379
 503 0 000006A2 E00000B4FF3B            DC        AGCPY,180,-197,150,181
                009600B5
 504 0 000006AC 800000B4FE85            DC        AMOVE,180,-379
 505 0 000006B2 E000001EFF3B            DC        AGCPY,30,-197,150,181
                009600B5
 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      00000156
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                    00005C00
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)