

AGC-1

## GENERAL INFORMATION

First Edition  
August 1986

PART NO. 800106

FORCE COMPUTERS Inc./GmbH  
All Rights Reserved

This document shall not be duplicated, nor its contents used  
for any purpose, unless express permission has been granted.

Copyright by FORCE Computers®



**NOTE**

The information in this document has been carefully checked and is believed to be entirely reliable. FORCE COMPUTERS makes no warranty of any kind with regard to the material in this document, and assumes no responsibility for any errors that may appear in this document. FORCE COMPUTERS reserves the right to make changes without notice to this, or any of its products, to improve reliability, performance or design.

FORCE COMPUTERS assumes no responsibility for the use of any circuitry other than circuitry which is part of a product of FORCE COMPUTERS GmbH/Inc.

FORCE COMPUTERS does not convey to the purchaser of the product described herein any license under the patent rights of FORCE COMPUTERS GmbH/Inc. nor the rights of others.

**FORCE COMPUTERS Inc.**  
727 University Avenue  
Los Gatos, CA 95030  
U.S.A.

Phone : (408) 354 34 10  
Telex : 172465  
FAX : (408) 395 77 18

**FORCE COMPUTERS GmbH**  
Daimlerstrasse 9  
D-8012 Ottobrunn/Munich  
West Germany

Phone : (089) 600 91-0  
Telex : 524190 forc-d  
FAX : (089) 609 77 93

**FORCE COMPUTERS FRANCE Sarl**  
11, rue Casteja  
92100 Boulogne  
France

Phone : (1) 4620 37 37  
Telex : 206 304 forc-f  
Fax : (1) 4621 35 19

**FORCE Computers UK Ltd.**  
No. 1 Holly Court  
3 Tring Road  
Wendover  
Buckinghamshire HP22 6NR  
England

Phone : (0296) 625456  
Telex : 838033  
Fax : (0296) 624027



## **1. GENERAL INFORMATION TO THE 63484 ACRTC**

The ACRTC is a high performance graphics-controller, compatible to the 68000 family of microprocessors. The ACRTC concept is to incorporate major functionality with effective graphic commands.

High level command language increases performance and reduces software development costs. In this way, the ACRTC converts logical x - y coordinates to physical frame buffer addresses. It supports 38 commands, including LINE, RECTANGLE, POLYLINE, POLYGON, CIRCLE, ELLIPSE, ARC, ELLIPSE ARC, FILLED RECTANGLE, PAINT, PATTERN and COPY. An on-chip 32byte pattern RAM can be used for powerful graphic environments. Conditional drawing functions are useful for drawing patterns, colour mixing and software windowing. The drawing area control supports clipping and hitting. The ACRTC is able to control four hardware windows, zooming and smooth scrolling in both vertical and horizontal directions. The capability of displaying up to 256 colours and the maximum drawing speed of 2 million Pixel per second in monochrome and colour applications allows high performance CAD terminals to be used.

## Features of the 63484 ACRTC:

- Up to 4096 by 4096 bit map graphic display and/or 256 lines by 256 characters by 32 rasters character display.
- Separate bit map graphic (2Mbyte) and character (128Kbyte) address spaces with combined graphic/character display.
- Three horizontal split screens and one window screen. Size and position fully programmable.
- Independent horizontal and vertical smooth scroll for each screen.
- 1 to 16 zoom magnitude - independent x and y factors.
- Logical pixel specification as 1, 2, 4, 8 or 16 bits for monochrome, gray scale and colour displays.
- Programmable address increment supports frame buffer memory width up to 128 bits for video bit rates greater than 500 MHz.
- Unique interleaved access mode for "flashless" displays.
- ACRTC provides dynamic RAM refresh.
- Asynchronous bus interface, optimised for the 68000 MPU family and the 68450 DMAC (16bit data bus).
- Separate on-chip 16byte READ and WRITE FIFOs.
- Maskable interrupts including FIFO status.
- Full programmability of CRT timing signals.
- Three raster scan modes.
- Master or Slave synchronization to multiple ACRTCs or other video generating devices.
- Programmable cursor and display timing skew.
- Eight user definable video attributes.
- Light pen detection.

## **2. IMPLEMENTATION OF THE 63484 ACRTC ON THE AGC-1**

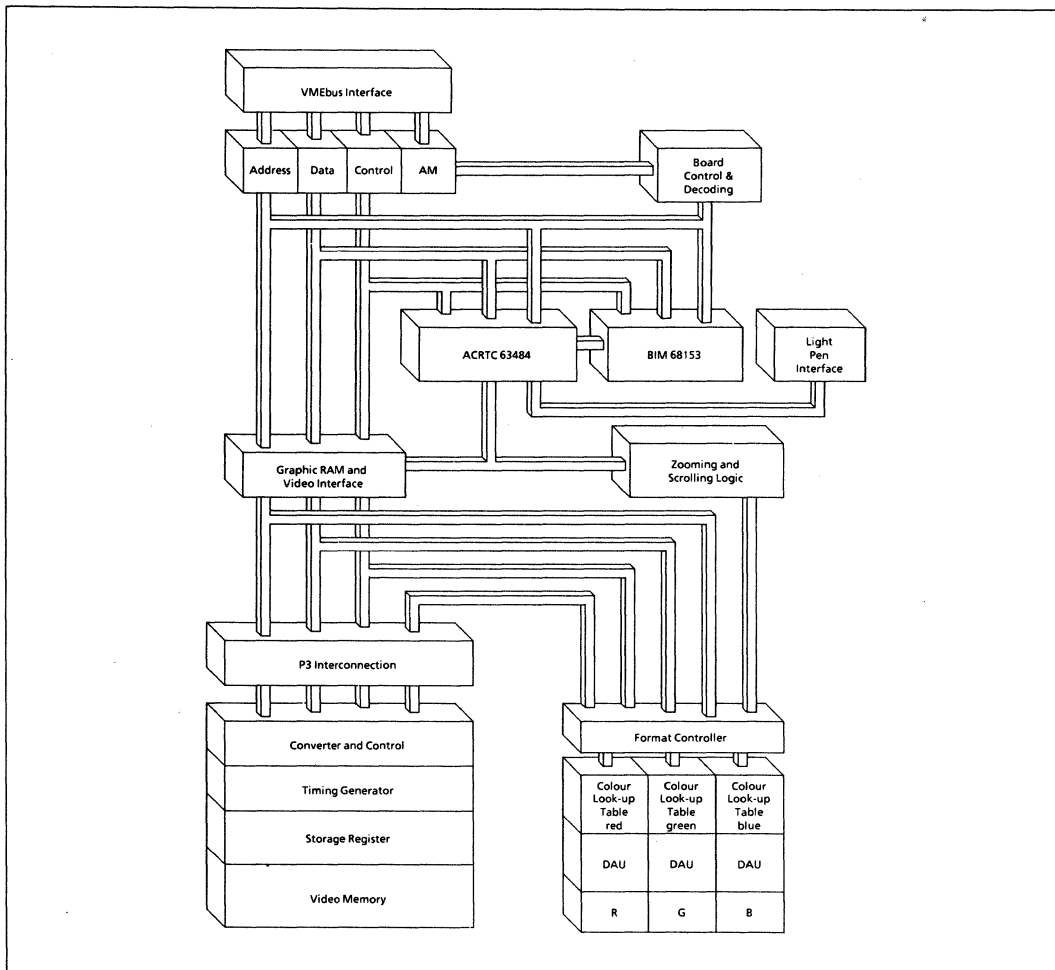
The AGC-1 is a highly integrated graphic-system on two double-euroboards, based on the 63484 ACRTC. With the optional SYS68K/AGC-1X board, it offers two bus interfaces and allows the board set to be used in single-processor as well as in high end multi-processor applications.

The board has a jumper selectable access address within the 16Mbyte address range. Also the VMEbus or VMXbus access is selectable. The on-board video RAM of 2Mbyte is dual ported and in this way directly accessible by either the processor or a DMA Controller for high speed display manipulation.

The Pixel clock of 64MHz offers resolutions up to 1024 X 800 non-interlaced with 4 bits per pixel. With the on-board colour palette, the user has the choice of up to 256 colours out of 16 million. With the implementation of the dual access mode, the ACRTC display and drawing accesses are interleaved and so the ACRTC reaches the maximum drawing speed without a "flashing" display. The complex attribute logic offers a great variety of display modes, e.g. blinking, inverse video, conditional blinking.

The AGC-1X board adds powerful character overlay with a separate character and attribute RAM, a software loadable character generator, the 68450 DMA Controller, which is used for VMEbus data transfers, two serial I/O interfaces for printer, mouse or digitizer and the VMXbus interface.

**Figure 2-1: Block Diagram of the SYS68K/AGC-1**





### **Features of the SYS68K/AGC-1:**

- 63484 ACRTC with a clock frequency of 8MHz.
- 3 x AM8151 graphic colour palette with 16/256 entries.
- 68153 Bus Interrupter Module for all local interrupts.
- 2Mbyte of video RAM directly accessible from the VMEbus
- R-, G-, B- and composite SYNC output.
- External synchronization input/output for other AGCs.
- Light pen interface.
- Fully buffered local address, data and control bus.
- VME/P1014 interface (16 bit).
- Software selectable interrupt request level and programmable interrupt vector.

The following table shows the general memory lay-out of the SYS68K/AGC-1:

The board start address (BBA) is jumper selectable in 256Kbyte steps.

Table 2-1: The Address Map

Start Address	End Address	Memory Area
BBA	BBA + \$33FFF	Occupied for the SYS68K/AGC-1X boards
BBA + \$34000	BBA + \$35FFF	BIM 68153
BBA + \$36000	BBA + \$37FFF	GCP 1 red
BBA + \$38000	BBA + \$39FFF	GCP 2 green
BBA + \$3A000	BBA + \$3BFFF	GCP 3 blue
BBA + \$3C000	BBA + \$3FFFF (\$3DFFF)	ACRTC 63484
BBA + \$3E000	BBA + \$3FFFF	Interrupt-Register VMX-Option
BBA + \$40000	BBA + \$23FFFF	Video-RAM

The Board Base Address is set by default during manufacturing to \$C00000 (Start Address) / \$E3FFFF (End Address).

# INSTALLATION

FORCE COMPUTERS Inc./GmbH  
All Rights Reserved

This document shall not be duplicated, nor its contents used  
for any purpose, unless express permission has been granted.

Copyright by FORCE Computers®



Please read the complete installation procedure before the board is installed in a VMEbus environment, to avoid malfunctions and damage to components.



T A B L E   O F   C O N T E N T S

1. GENERAL OVERVIEW..... 1-1

    1.1 The Function Switch Positions..... 1-2

    1.2 Connection of the RGB Monitor..... 1-4

2. INSTALLATION IN THE RACK..... 2-1

    2.1 Power On..... 2-2





## **1. GENERAL OVERVIEW**

This manual only describes the function of the SYS68K/AGC-1A and SYS68K/AGC-1B boards. For all other functions of the SYS68K/AGC-1X board, please refer to the SYS68K/AGC-1X User's Manual.

Easy installation of the AGC-1 board set is provided because of the default jumper settings. It may be necessary to modify the board base address (BBA) to adapt the board to the various applications.

Both the main board and the slave board are screwed together and have the correct measurements to be plugged into a standard VMEbus backplane. Only the AGC-1A board contains the VMEbus interface.

## 1.1 The Function Switch Positions

The board contains 2 function switches listed from the top to the bottom:

VME/VMX

RUN/HALT

The two positions of the switches are defined as "up" and "down".

The VMEbus interface will be enabled if all function switches are set to the position "down" (for first installation).

Please toggle each of the switches sometime before installing the board in the rack to detect any damage incurred during transportation.

## **1.2. Connection of the RGB-Monitor**

The terminal has to be connected to the BNC connectors on the main board (right board of the package).

The RGB outputs are compatible to the RS343 standard for connection with RGB monitors. The composite SYNC signal is mixed on all colour outputs so that mostly there is no need for an extra SYNC connection.

Monitors with separate SYNC input must be connected to the SYNC output of the AGC-1.

The SYNC output can be configured to have a positive or negative SYNC signal. This is specified by the jumper settings of B7 (SYNCSEL).

For jumper settings, please refer to section 4.11.4 in the Hardware User's Manual (Register 3).



## 2. Installation in the Rack

The main (AGC-1A) and the slave (AGC-1B) boards are screwed together and can immediately be mounted into a VME rack.

### Caution

- A) Switch off power before installing the board to avoid electrical damage to the components.
  
- b) The main board contains a special ejector (the handles).

The boards have to be plugged in and the screws of the front panel must be tightened to guarantee proper installation.

## 2.1. Power On

If the board is correctly installed, the switches are in the correct positions, the monitor is connected, the power for the VMEbus rack can be switched on.

If everything has worked, the green RUN LED and the yellow VME LED must turn on.

Now the AGC-1 is accessible from an VMEbus-MASTER under the selected board base address (\$C00000 is set by default during manufacturing).

For initialisation of the ACRTC, please refer to section 4.12 in the Hardware User's Manual (Register 3).

# HARDWARE

# USER'S MANUAL

FORCE COMPUTERS Inc./GmbH  
All Rights Reserved

This document shall not be duplicated, nor its contents used  
for any purpose, unless express permission has been granted.

Copyright by FORCE Computers®





# T A B L E O F C O N T E N T S

1.	GENERAL INFORMATION.....	1-1
2.	GENERAL OPERATION.....	2-1
3.	THE 63484 ACRTC DESCRIPTION.....	3-1
4.	ACCESS TO THE SYS68K/AGC-1.....	4-1
4.1	Board Base Address Selection.....	4-1
4.1.1	Standard Memmory Space.....	4-1
4.1.2	Short I/O Space.....	4-9
4.2	Board Size Selection.....	4-14
4.3	Address Modifier Selection.....	4-21
4.4	Functional Groups on the SYS68K/AGC-1.....	4-25
4.5	Access to the Devices on the SYS68K/AGC-1.....	4-26
4.6	Addressing the Colour Look-up Tables.....	4-27
4.6.1	Colour Switch Mode.....	4-31
4.6.2	Blink Switch Mode.....	4-35
4.7	Addressing the BIM 68153.....	4-38
4.7.1	Interrupt Structure of the SYS68k/AGC-1.....	4-38
4.7.2	Programming the BIM on the SYS68K/AGC-1.....	4-43
4.8	Addressing the Video Memory.....	4-44
4.9	Addressing the ACRTC 63484.....	4-53
4.9.1	Address Register.....	4-59
4.9.2	Status Register.....	4-60
4.9.3	FIFO Entry.....	4-62
4.9.4	Command Control Register.....	4-63
4.9.5	Operation Mode Register.....	4-66
4.9.6	Display Control Register.....	4-70
4.9.7	Raster Count Register.....	4-75
4.9.8	Horizontal Sync Register.....	4-76
4.9.9	Horizontal Display Register.....	4-78
4.9.10	Horizontal Window Display Register.....	4-78
4.9.11	Vertical Sync Register.....	4-80
4.9.12	Vertical Display Register.....	4-81
4.9.13	Vertical Window Display Register.....	4-83
4.9.14	Split Screen Width Register.....	4-85
4.9.15	Blink Control Register.....	4-87
4.9.16	Memory Width Register.....	4-88
4.9.17	Start Address Register.....	4-89
4.9.18	Zoom Factor Register.....	4-91
4.9.19	Light Pen Address Register.....	4-92
4.10	Drawing Control Register.....	4-93
4.10.1	Pattern RAM.....	4-95
4.10.2	Colour 0 Register.....	4-96
4.10.3	Colour 1 Register.....	4-96
4.10.4	Colour Comparison Register.....	4-97
4.10.5	Edge Colour Register.....	4-97
4.10.6	Mask Register.....	4-98
4.10.7	Pattern RAM Control Register.....	4-99
4.10.8	Area Definition Register.....	4-101
4.10.9	Read-Write Pointer.....	4-102
4.10.10	Drawing Pointer.....	4-103
4.10.11	Current Ponter.....	4-104

T A B L E O F C O N T E N T S Cont'd

5. COMMAND OVERVIEW.....	5-1
5.1 Register Access Commands.....	5-2
5.2 Data Transfer Commands.....	5-3
5.3 Modify Mode.....	5-4
5.4 Graphic Drawing Commands.....	5-5
5.5 Operation Mode.....	5-7
5.6 Colour Mode.....	5-8
5.7 Area Mode.....	5-9
6. MISCELLANEOUS.....	6-1
6.1 Miscellaneous Jumper Settings.....	6-1
6.2 Light Pen Interface.....	6-4
6.3 External Synchronization.....	6-6
6.4 Display Monitor Interface.....	6-8
7. CALCULATING THE SCREEN PARAMETERS.....	7-1
7.1 Horizontal Timing.....	7-4
7.2 Vertical Timing.....	7-7
7.3 Display Control RAM.....	7-9
7.4 Control Registers.....	7-12

## L I S T   O F   F I G U R E S

1-1:	Photo of the SYS68K/AGC-1A Board.....	1-2
2-1:	Block Diagram of the SYS68K/AGC-1.....	2-2
4-1:	Jumper Location Diagram of Address Selection.....	4-2
4-2:	Jumper Location Diagram of Short I/O Selection.....	4-11
4-3:	Jumper Location Diagram of Board Size Selection.....	4-15
4-4:	Jumper Location Diagram of AM-Code Selection.....	4-22
4-5:	Access to the Colour Look-up Tables.....	4-29
4-6:	Jumper Location Diagram of Switch Colour Mode.....	4-33
4-7:	Jumper Location Diagram of Blink Switch Mode.....	4-36
4-8:	Access to the BIM68153.....	4-39
4-9:	Interrupt Cycle Timing of the SYS68K/AGC-1.....	4-41
4-10:	VMEbus Access to the Video RAM.....	4-45
4-11:	ACRTC Drawing Access to the Video Ram.....	4-49
4-12:	Jumper Location Diagram of RAS/CAS Generation.....	4-51
4-13:	Access to the 63484 ACRTC.....	4-57
6-1:	Jumper Location Diagram of RAS/CAS Generation.....	6-2
6-2:	Jumper Location Diagram of M/S-Selection.....	6-7
6-3:	Jumper Location Diagram of Jumper B7 (SYNCSEL).....	6-9
7-1:	Display Screen Specification.....	7-3

## L I S T   O F   T A B L E S

4-1:	Jumper Settings for Board Base Address.....	4-4
4-2:	Memory Map for Standard Memory Access.....	4-8
4-3:	Jumper Settings for Short I/O Access.....	4-12
4-4:	Memory Map for Short I/O Access.....	4-13
4-5:	Jumper Settings for Board Size Selection.....	4-16
4-6:	Address Modifier Codes.....	4-19
4-7:	Jumper Settings for Address Modifier Codes.....	4-23
4-8:	Jumper Settings for Switch Colour Mode.....	4-34
4-9:	Jumper Settings for Blink Switch Mode.....	4-37
4-10:	Address Location of the BIM 68153.....	4-40
4-11:	Jumper Settings for RAS/CAS Generation.....	4-42
4-12:	Programming Model of the 63484 ACRTC.....	4-43
4-13A:	Time Values of Video Ram Read Cycle.....	4-46
4-13B:	Time Values of Video RAM Read Cycle.....	4-48
4-14:	Time Values of A ACRTC Drawing Cycle to the Video RAM...	4-50
4-15:	Jumper Settings of B14, B15, B16 and B17.....	4-52
4-16:	Programming Model.....	4-54
4-17:	Time Values of a VMEbus Access to the 63484 ACRTC.....	4-58
5-1:	Graphic Drawing Commands.....	5-6
6-1:	Jumper Settings for RAS/CAS Generation.....	6-3
6-2:	Pin Assignments of Connector P4.....	6-5



## 1. GENERAL INFORMATION

This high performance VMEbus-based GRAPHIC board combines the powerful graphic processor, the 63484 ACRTC with 2 Mbyte Video-RAM, 3 digital-to-analogue Converters and 3 colour-look-up tables with 256 entries each.

The implementation of the so called DUAL ACCESS MODE (see description of the 63484 ACRTC) allows Video-RAM accesses during display without 'flashing' display and so speeds up drawing operations.

The also built-in direct Video-RAM access via the VMEbus during display supports very fast updating or loading of the FRAME BUFFER.

Two graphic bit modes in conjunction with the colour-look-up tables make it possible to simultaneously display up to 256 colours out of a total range of 16 million.

A sophisticated attribute logic offers a wide variety of display manipulation, such as blinking of specified colours or objects and switching the look-up tables. Also supported is a smooth scroll in horizontal and vertical directions and independent zooming in both directions.

The local control, consisting of a Bus Interrupter Modul (BIM) offers software control for programming the ACRTC and updating the Frame Buffer or the colour-look-up tables during vertical retrace period.

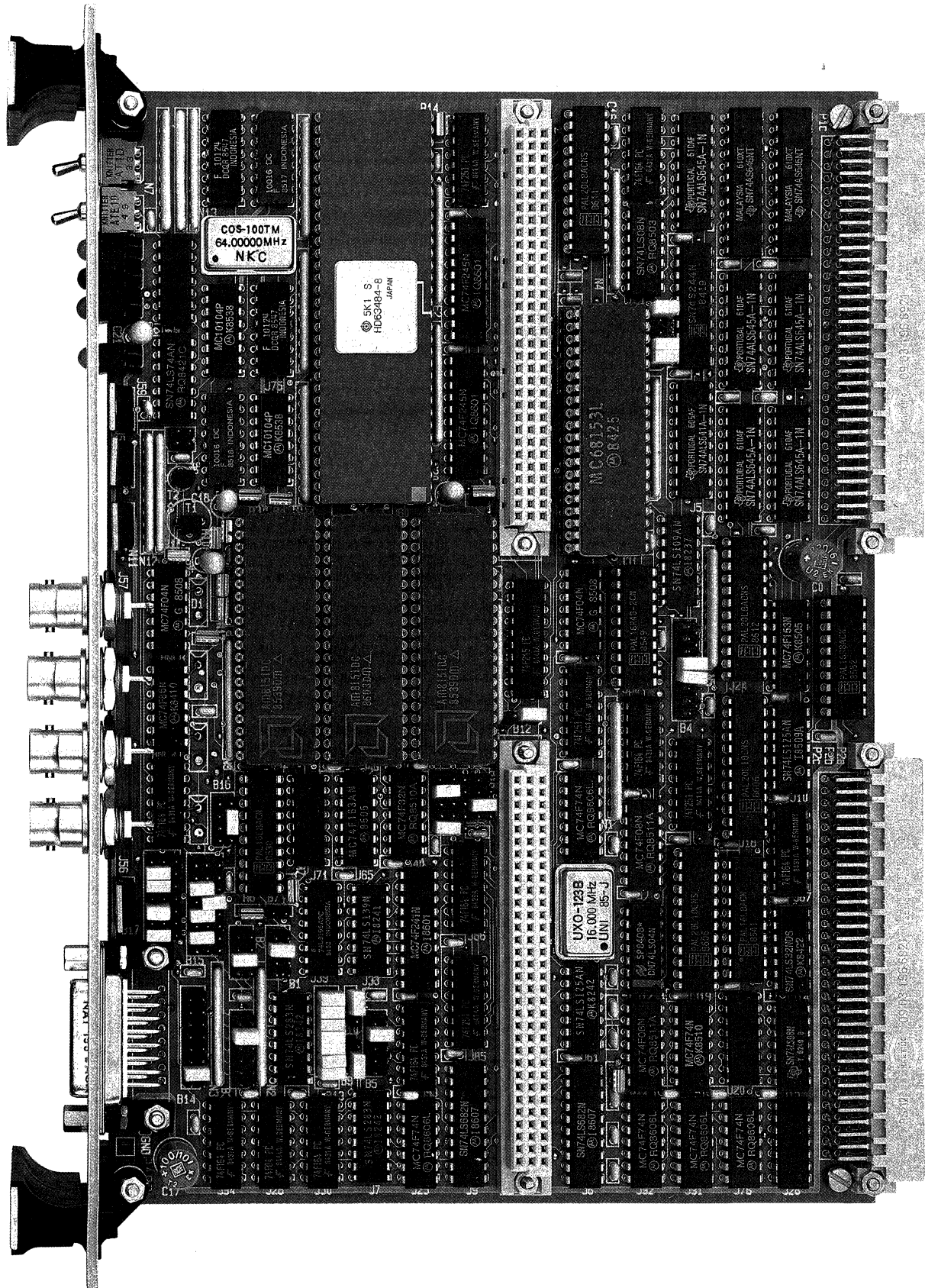
To provide all of these features, the AGC-1 consists of two double Eurocard boards, the main board and the slave board. On the main board the bus interface, the graphic processor, the colour-look-up tables and the Digital to Analogue coverters are installed.

The slave board contains the 2 Mbyte Video-RAM as well as the parallel-to-serial shift registers.

The photo of the SYS68K/AGC-1 shows the main board in detail, and Figure 1-1 outlines the block diagram of AGC-1.

This manual provides a general operating description of the SYS68K/AGC-1 hardware. Follow manufacturer's installation instructions for use and troubleshooting.

**Figure 1-1: Photo of the SYS68K/AGC-1A Board**



## 2. GENERAL OPERATION

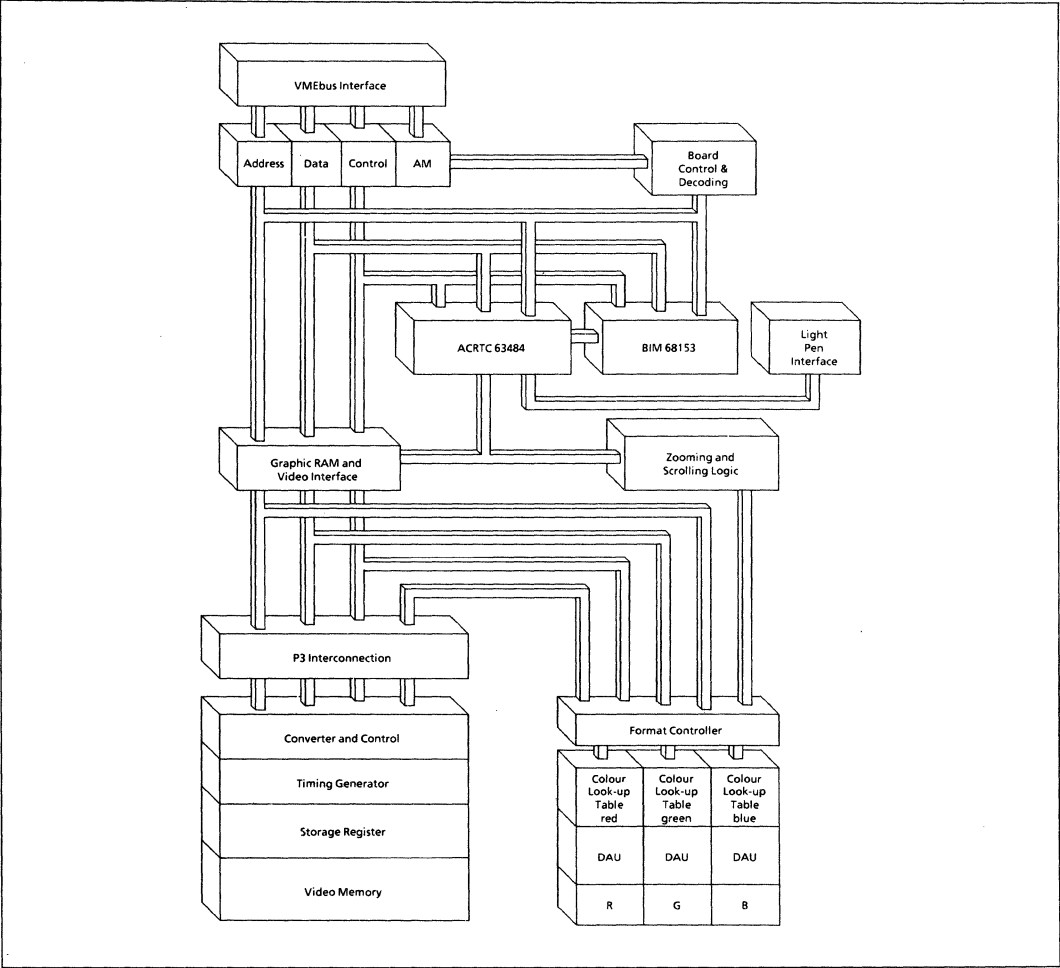
The SYS68K/AGC-1 board set contains the powerful 63484 ACRTC with 2 MByte of Video-RAM, three DAC's with colour look-up tables and powerful attribut capabilities.

One BIM (68153 Bus Interrupter Module) is used to support fully asynchronous operation to the VMEbus.

The SYS68K/AGC-1 boards are fully VMEbus Rev. C (IEEE P1014) compatible.

See Figure 2-1 for the general block diagram of the SYS68K/AGC-1 board set.

**Figure 2-1: Block Diagram of the SYS68K/AGC-1**





## Features of the SYS68K/AGC-1

- 63484 ACRTC with 8 MHz Clock Frequency.
- 2 Mbyte dynamic Video-RAM with 120ns access time for up to 2048 \* 2048 pixel (in 4 bit/pixel mode).
- Direct video memory access from the VMEbus during display time.
- Zoom logic for magnification in horizontal and vertical direction (up to 16).
- Smooth scroll logic (pixel-wise) for vertical and horizontal direction.
- Video interface bus for upgrading with character overlay.
- 3 Graphic Colour Palette AM8151 with 256 entries for each colour and 64 MHz Pixel clock frequency.
- Bus Interrupter Module for all local Interrupt sources.
- Interrupt handling via programmable interrupt vectors.
- Each VMEbus IRQ level can be enabled/disabled via software.
- Fully decoding of the address modifiers.
- Jumper selectable access address for short I/O (A16 mode) or standard memory (A24 mode).
- RUN/LOCAL switch for complete isolation from the VMEbus.
- Fully VMEbus Rev.C and IEEE P1014 compatible



### 3. THE 63484 ADVANCED GRAPHIC CONTROLLER DESCRIPTION

The 63484 ACRTC contains 38 high level commands including 23 drawing commands for fast generation of polylines, circles, ellipses, circle and ellipse arcs and filled areas. Fast copy and clear commands are very useful for animated graphics. The maximum drawing speed with 8 MHz clock is 2 Million Pixel/second.

He supports up to 2 MByte of video RAM and up to 128 KByte of character RAM.

The built-in bit pattern RAM allows the generation of patterns with 16 \* 16 pixel or line-types with 256 pixel pattern length.

Also supported are four hardware windows and functions such as zooming, scrolling, hiting and clipping.

The automatic conversion from logical X/Y coordinates to physical frame buffer addresses eases the programming of graphics.

The non-multiplexed address and data bus are fully asynchronous to allow optimized hardware interfacing to the used VMEbus.

Two 16 Byte FIFO`s (read an write) for commands and parameters speed up the accesses from a host processor as well as the DMA-interface for connection with a direct memory access controller.



#### 4. Access to the SYS68K/AGC-1

The SYS68K/AGC-1 can be accessed when it is inserted in the VMEbus motherboard and the toggle switches on the front panel are switched to RUN (green LED lights) and VME (yellow VME LED lights).

To access the board, the address bits A18-A23 must match the Board Base Address jumper setting and the Address Modifier Code must match the Address Modifier decoding jumper settings. The SYS68K/AGC-1 board is delivered with the default Board Base Address of \$C00000 and is accessible with any Address Modifier Code.

The SYS68K/AGC-1 board occupies an address range of 2.256 MBytes, beginning with the Board Base Address. Read, write and read-modify-write accesses are supported with byte and word operands.

An access to any location in the 2.256 MBytes address range is a legal bus cycle and no bus error will be forced.

##### 4.1 Board Base Address Selection

The Base Address of the SYS68K/AGC-1 board is jumper selectable mainly in two different ways described in Chapter 4.1.1 and 4.1.2.

##### 4.1.1 Board Base Address Selection in the STANDARD MEMORY SPACE

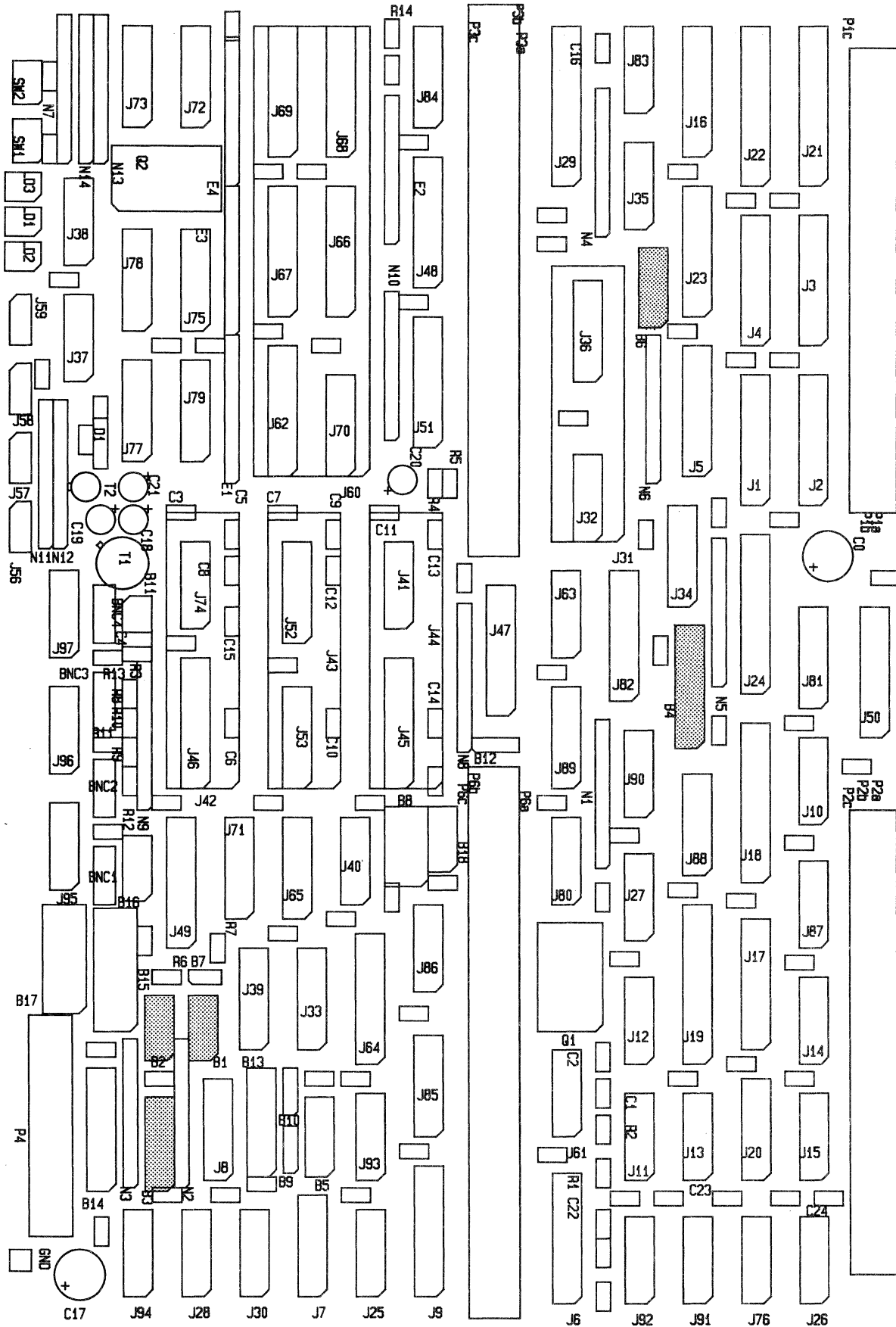
The Base Address of the SYS68K/AGC-1 is jumper selectable in 256kbyte steps within the 16 Mbyte address space. To select the board, the address on the address lines A18-A23 must match the base address jumper settings at jumperfield B3.

B3 is the jumper field defining the address bits A18 to A23 of the Board Base Address.

Fig. 4-1 displays the location of the jumper field on the PC board.

Table 4-1 shows the connection assignments and Table 4-2 shows the memory map for standard memory access.

Figure 4-1: Jumper Location Diagram A



Jumper Description:

B1 and B2 : BOARD SIZE SELECTION  
B3 : BOARD BASE ADDRESS SELECTION  
B4 : ADDRESS MODIFIER CODE SELECTION  
B6 : SHORT I/O ADDRESS SELECTION

Table 4-1: Jumper Settings for Board Base Address (BBA)

a) BBA=\$000000    b) BBA=\$040000    c) BBA=\$080000    d) BBA=\$0C0000

1 -- 12	1     12	1 -- 12	1     12
2 -- 11	2 -- 11	2     11	2     11
3 -- 10	3 -- 10	3 -- 10	3 -- 10
4 -- 9	4 -- 9	4 -- 9	4 -- 9
5 -- 8	5 -- 8	5 -- 8	5 -- 8
6 -- 7	6 -- 7	6 -- 7	6 -- 7

e) BBA=\$100000    f) BBA=\$140000    g) BBA=\$180000    h) BBA=\$1C0000

1 -- 12	1     12	1 -- 12	1     12
2 -- 11	2 -- 11	2     11	2     11
3 -- 10	3     10	3     10	3     10
4 -- 9	4 -- 9	4 -- 9	4 -- 9
5 -- 8	5 -- 8	5 -- 8	5 -- 8
6 -- 7	6 -- 7	6 -- 7	6 -- 7

i) BBA=\$200000    j) BBA=\$240000    k) BBA=\$280000    l) BBA=\$2C0000

1 -- 12	1     12	1 -- 12	1     12
2 -- 11	2 -- 11	2     11	2     11
3 -- 10	3 -- 10	3 -- 10	3 -- 10
4     9	4     9	4     9	4     9
5 -- 8	5 -- 8	5 -- 8	5 -- 8
6 -- 7	6 -- 7	6 -- 7	6 -- 7

m) BBA=\$300000    n) BBA=\$340000    o) BBA=\$380000    p) BBA=\$3C0000

1 -- 12	1     12	1 -- 12	1     12
2 -- 11	2 -- 11	2     11	2     11
3     10	3     10	3     10	3     10
4     9	4     9	4     9	4     9
5 -- 8	5 -- 8	5 -- 8	5 -- 8
6 -- 7	6 -- 7	6 -- 7	6 -- 7

q) BBA=\$400000    r) BBA=\$440000    s) BBA=\$480000    t) BBA=\$4C0000

1 -- 12	1     12	1 -- 12	1     12
2 -- 11	2 -- 11	2     11	2     11
3 -- 10	3 -- 10	3 -- 10	3 -- 10
4 -- 9	4 -- 9	4 -- 9	4 -- 9
5     8	5     8	5     8	5     8
6 -- 7	6 -- 7	6 -- 7	6 -- 7



Table 4-1: Jumper Settings for Board Base Address (BBA)

u) BBA=\$500000 v) BBA=\$540000 w) BBA=\$580000 x) BBA=\$5C0000

1 -- 12	1 12	1 -- 12	1 12
2 -- 11	2 -- 11	2 11	2 11
3 10	3 10	3 10	3 10
4 -- 9	4 -- 9	4 -- 9	4 -- 9
5 8	5 8	5 8	5 8
6 -- 7	6 -- 7	6 -- 7	6 -- 7

y) BBA=\$600000 z) BBA=\$640000 A) BBA=\$680000 B) BBA=\$6C0000

1 -- 12	1 12	1 -- 12	1 12
2 -- 11	2 -- 11	2 11	2 11
3 -- 10	3 -- 10	3 -- 10	3 -- 10
4 9	4 9	4 9	4 9
5 8	5 8	5 8	5 8
6 -- 7	6 -- 7	6 -- 7	6 -- 7

C) BBA=\$700000 D) BBA=\$740000 E) BBA=\$780000 F) BBA=\$7C0000

1 -- 12	1 12	1 -- 12	1 12
2 -- 11	2 -- 11	2 11	2 11
3 10	3 10	3 10	3 10
4 9	4 9	4 9	4 9
5 8	5 8	5 8	5 8
6 -- 7	6 -- 7	6 -- 7	6 -- 7

G) BBA=\$800000 H) BBA=\$840000 I) BBA=\$880000 J) BBA=\$8C0000

1 -- 12	1 12	1 -- 12	1 12
2 -- 11	2 -- 11	2 11	2 11
3 -- 10	3 -- 10	3 -- 10	3 -- 10
4 -- 9	4 -- 9	4 -- 9	4 -- 9
5 -- 8	5 -- 8	5 -- 8	5 -- 8
6 7	6 7	6 7	6 7

K) BBA=\$900000 L) BBA=\$940000 M) BBA=\$980000 N) BBA=\$9C0000

1 -- 12	1 12	1 -- 12	1 12
2 -- 11	2 -- 11	2 11	2 11
3 10	3 10	3 10	3 10
4 -- 9	4 -- 9	4 -- 9	4 -- 9
5 -- 8	5 -- 8	5 -- 8	5 -- 8
6 7	6 7	6 7	6 7

Table 4-1: Jumper Settings for Board Base Address (BBA, Jumper B3)

O) BBA=\$A00000    P) BBA=\$A40000    Q) BBA=\$A80000    R) BBA=\$AC0000

1 -- 12	1     12	1 -- 12	1     12
2 -- 11	2 -- 11	2     11	2     11
3 -- 10	3 -- 10	3 -- 10	3 -- 10
4     9	4     9	4     9	4     9
5 -- 8	5 -- 8	5 -- 8	5 -- 8
6     7	6     7	6     7	6     7

S) BBA=\$B00000    T) BBA=\$B40000    U) BBA=\$B80000    V) BBA=\$BC0000

1 -- 12	1     12	1 -- 12	1     12
2 -- 11	2 -- 11	2     11	2     11
3     10	3     10	3     10	3     10
4     9	4     9	4     9	4     9
5 -- 8	5 -- 8	5 -- 8	5 -- 8
6     7	6     7	6     7	6     7

W) BBA=\$C00000    X) BBA=\$C40000    Y) BBA=\$C80000    Z) BBA=\$CC0000

1 -- 12	1     12	1 -- 12	1     12
2 -- 11	2 -- 11	2     11	2     11
3 -- 10	3 -- 10	3 -- 10	3 -- 10
4 -- 9	4 -- 9	4 -- 9	4 -- 9
5     8	5     8	5     8	5     8
6     7	6     7	6     7	6     7

default set-up

1) BBA=\$D00000    2) BBA=\$D40000    3) BBA=\$D80000    4) BBA=\$DC0000

1 -- 12	1     12	1 -- 12	1     12
2 -- 11	2 -- 11	2     11	2     11
3     10	3     10	3     10	3     10
4 -- 9	4 -- 9	4 -- 9	4 -- 9
5     8	5     8	5     8	5     8
6     7	6     7	6     7	6     7

5) BBA=\$E00000    6) BBA=\$E40000    7) BBA=\$E80000    8) BBA=\$EC0000

1 -- 12	1     12	1 -- 12	1     12
2 -- 11	2 -- 11	2     11	2     11
3 -- 10	3 -- 10	3 -- 10	3 -- 10
4     9	4     9	4     9	4     9
5     8	5     8	5     8	5     8
6     7	6     7	6     7	6     7

Table 4-1: Jumper Settings for Board Base Address (BBA)

9) BBA=\$F00000	a) BBA=\$F40000	b) BBA=\$F80000	c) BBA=\$FC0000
1 -- 12	1    12	1 -- 12	1    12
2 -- 11	2 -- 11	2    11	2    11
3    10	3    10	3    10	3    10
4    9	4    9	4    9	4    9
5    8	5    8	5    8	5    8
6    7	6    7	6    7	6    7

The following table shows the memory lay-out of AGC-1 for Standard Memory Access:

The Board Base Address (BBA) is jumper selectable in 256K steps.

Table 4-2: MEMORY MAP FOR STANDARD MEMORY ACCESS

Start Address	End Address	Memory Area
BBA	BBA + \$35FFF	BIM 68153 13.824 times
BBA + \$36000	BBA + \$37FFF	GCP 1 red 16 times
BBA + \$38000	BBA + \$39FFF	GCP 2 green 16 times
BBA + \$3A000	BBA + \$3BFFF	GCP 3 blue 16 times
BBA + \$3C000	BBA + \$3FFFF	ACRTC 63484 4096 times
BBA + \$40000	BBA + \$23FFFF	Video-RAM 1 time

#### 4.1.2 Board Base Address Selection for SHORT I/O MEMORY

The second way of accessing the SYS68K/AGC-1 is by using the SHORT I/O address range.

All devices except the Video RAM area can be accessed via SHORT I/O addressing.

In this case the Board Size has to be defined to \$000000 (see Table 4-5)

The SYS68K/AGC-1 then occupies 3,5 Kbyte of the 64 Kbyte SHORT I/O memory.

The Board Start Address is then jumper selectable in 4 Kbyte steps.

For SHORT I/O addressing the address bits A12-A15 must match the Board Base Address for SHORT I/O (B6) and the Address Modifier Code must match the Address Modifier decoding jumper settings for SHORT I/O (see Section 4.2).

B6 is the jumper field defining bits 12 to 15 of the Board Base Address.

Fig.4-2 shows the location of the jumper area B6 on the PC board.

Table 4-3 gives all possible jumper settings.

The default jumper setting of B6 is for STANDARD MEMORY access. Please be careful, especially that jumper position 5,6 is disconnected in this case.

Table 4-4 gives the memory map for short I/O access.



Figure 4-2 Jumper Location Diagram B

B6 : BOARD BASE ADDRESS SELECTION FOR SHORT I/O MEMORY

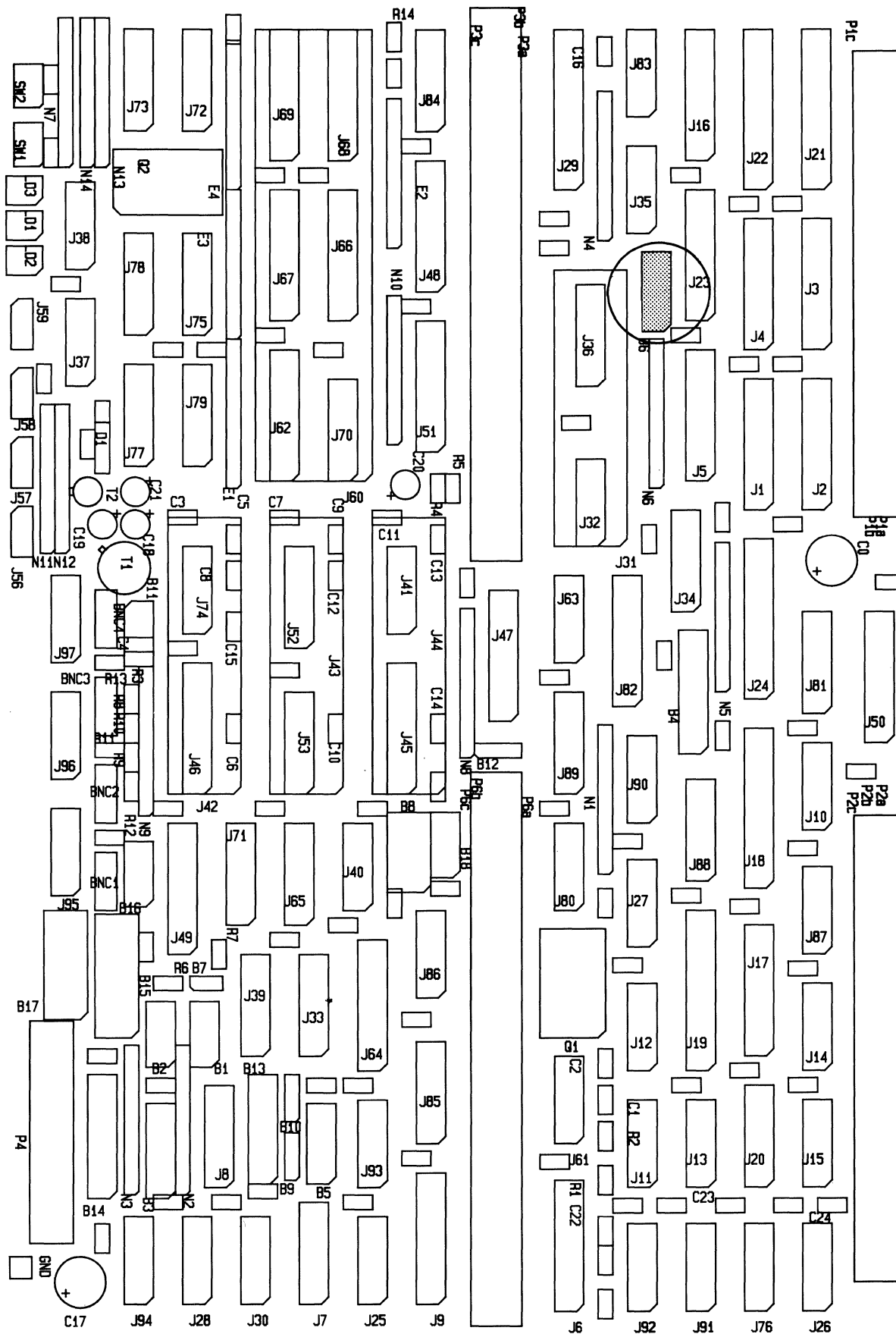


Table 4-3: Board Base Address Selection for SHORT I/O Memory

a) STANDARD MEMORY    b) BBA=\$FF0000    c) BBA=\$FF1000    d) BBA=\$FF2000

1 -- 10  
2 -- 9  
3 -- 8  
4 -- 7  
5    6

1 -- 10  
2 -- 9  
3 -- 8  
4 -- 7  
5 -- 6

1 -- 10  
2 -- 9  
3 -- 8  
4    7  
5 -- 6

1 -- 10  
2 -- 9  
3    8  
4 -- 7  
5 -- 6

Default Condition

e) BBA=\$FF3000

1 -- 10  
2 -- 9  
3    8  
4    7  
5 -- 6

f) BBA=\$FF4000

1 -- 10  
2    9  
3 -- 8  
4 -- 7  
5 -- 6

g) BBA=\$FF5000

1 -- 10  
2    9  
3 -- 8  
4    7  
5 -- 6

h) BBA=\$FF6000

1 -- 10  
2    9  
3    8  
4 -- 7  
5 -- 6

i) BBA=\$FF7000

1 -- 10  
2    9  
3    8  
4    7  
5 -- 6

j) BBA=\$FF8000

1    10  
2 -- 9  
3 -- 8  
4 -- 7  
5 -- 6

k) BBA=\$FF9000

1    10  
2 -- 9  
3 -- 8  
4    7  
5 -- 6

l) BBA=\$FFA000

1    10  
2 -- 9  
3    8  
4 -- 7  
5 -- 6

m) BBA=\$FFB000

1    10  
2 -- 9  
3    8  
4    7  
5 -- 6

n) BBA=\$FFC000

1    10  
2    9  
3 -- 8  
4 -- 7  
5 -- 6

o) BBA=\$FFD000

1    10  
2    9  
3 -- 8  
4    7  
5 -- 6

p) BBA=\$FFE000

1    10  
2    9  
3    8  
4 -- 7  
5 -- 6

d) BBA=\$FFF000

1    10  
2    9  
3    8  
4    7  
5 -- 6



The following table shows the memory lay-out of AGC-1 for Short I/O Memory Access:

The Board Base Address (BBA) is jumper selectable in 4K steps using the short I/O decoding.

Table 4-4: Memory Map for Short I/O Access

Start Address	End Address	Memory Area
BBA	BBA + \$5FF	BIM 68153 96 times
BBA + \$600	BBA + \$7FF	GCP 1 red 1 time
BBA + \$800	BBA + \$9FF	GCP 2 green 1 time
BBA + \$A00	BBA + \$BFF	GCP 3 blue 1 time
BBA + \$C00	BBA + \$DFF	ACRTC 63484 128 times
only accessible via ACRTC 63484		Video-RAM

## 4.2 Board Size Selection

For special applications it could be necessary to decrease the memory space occupied by the SYS68K/AGC-1 board. This could be done by setting the jumpers B1 and B2. In doing this it is possible to fade out the Video Memory or parts of it. Nevertheless the Video Memory can be accessed via the 63484 ACRTC.

The Board Size can be decreased in 64 Kbyte steps.

Fig. 4-3 displays the locations of the jumper fields on the PC board.

Table 4-5 shows the possible jumper settings for the Board Size (BS)



Table 4-5 Board Size Selection Jumper Settings

a) BS=\$240000	b) BS=\$230000	c) BS=\$220000	d) BS=\$210000
B1 1 -- 8 2 -- 7 3 -- 6 4 -- 5	1 8 2 7 3 -- 6 4 -- 5	1 -- 8 2 7 3 -- 6 4 -- 5	1 8 2 -- 7 3 -- 6 4 -- 5
B2 1 -- 8 2 7 3 -- 6 4 -- 5	1 -- 8 2 7 3 -- 6 4 -- 5	1 -- 8 2 7 3 -- 6 4 -- 5	1 -- 8 2 7 3 -- 6 4 -- 5
Default set-up			
e) BS=\$200000	f) BS=\$1F0000	g) BS=\$1E0000	h) BS=\$1D0000
B1 1 -- 8 2 -- 7 3 -- 6 4 -- 5	1 8 2 7 3 6 4 5	1 -- 8 2 7 3 6 4 5	1 8 2 -- 7 3 6 4 5
B2 1 -- 8 2 7 3 -- 6 4 -- 5	1 8 2 -- 7 3 -- 6 4 -- 5	1 8 2 -- 7 3 -- 6 4 -- 5	1 8 2 -- 7 3 -- 6 4 -- 5
i) BS=\$1C0000	j) BS=\$1B0000	k) BS=\$1A0000	l) BS=\$190000
B1 1 -- 8 2 -- 7 3 -- 6 4 5	1 8 2 7 3 6 4 5	1 -- 8 2 7 3 6 4 5	1 8 2 -- 7 3 6 4 5
B2 1 8 2 -- 7 3 -- 6 4 -- 5	1 8 2 -- 7 3 -- 6 4 -- 5	1 8 2 -- 7 3 -- 6 4 -- 5	1 8 2 -- 7 3 -- 6 4 -- 5
m) BS=\$180000	n) BS=\$170000	o) BS=\$160000	p) BS=\$150000
B1 1 -- 8 2 -- 7 3 -- 6 4 5	1 8 2 7 3 6 4 -- 5	1 -- 8 2 7 3 6 4 -- 5	1 8 2 -- 7 3 6 4 -- 5
B2 1 8 2 -- 7 3 -- 6 4 -- 5	1 8 2 -- 7 3 -- 6 4 -- 5	1 8 2 -- 7 3 -- 6 4 -- 5	1 8 2 -- 7 3 -- 6 4 -- 5

Table 4-5 Board Size Selection Jumper Settings

q) BS=\$140000	r) BS=\$130000	s) BS=\$120000	t) BS=\$110000
B1 1 -- 8 2 -- 7 3 -- 6 4 -- 5	1 8 2 7 3 -- 6 4 -- 5	1 -- 8 2 7 3 -- 6 4 -- 5	1 8 2 -- 7 3 -- 6 4 -- 5
B2 1 8 2 -- 7 3 -- 6 4 -- 5	1 8 2 -- 7 3 -- 6 4 -- 5	1 8 2 -- 7 3 -- 6 4 -- 5	1 8 2 -- 7 3 -- 6 4 -- 5
u) BS=\$100000	v) BS=\$0F0000	w) BS=\$0E0000	x) BS=\$0D0000
B1 1 -- 8 2 -- 7 3 -- 6 4 -- 5	1 8 2 7 3 6 4 5	1 -- 8 2 7 3 6 4 5	1 8 2 -- 7 3 6 4 5
B2 1 8 2 -- 7 3 -- 6 4 -- 5	1 -- 8 2 -- 7 3 -- 6 4 -- 5	1 -- 8 2 -- 7 3 -- 6 4 -- 5	1 -- 8 2 -- 7 3 -- 6 4 -- 5
y) BS=\$0C0000	z) BS=\$0B0000	A) BS=\$0A0000	B) BS=\$090000
B1 1 -- 8 2 -- 7 3 6 4 5	1 8 2 7 3 -- 6 4 5	1 -- 8 2 7 3 -- 6 4 5	1 8 2 -- 7 3 -- 6 4 5
B2 1 -- 8 2 -- 7 3 -- 6 4 -- 5	1 -- 8 2 -- 7 3 -- 6 4 -- 5	1 -- 8 2 -- 7 3 -- 6 4 -- 5	1 -- 8 2 -- 7 3 -- 6 4 -- 5
C) BS=\$080000	D) BS=\$070000	E) BS=\$060000	F) BS=\$050000
B1 1 -- 8 2 -- 7 3 -- 6 4 5	1 8 2 7 3 6 4 -- 5	1 -- 8 2 7 3 6 4 -- 5	1 8 2 -- 7 3 6 4 -- 5
B2 1 -- 8 2 -- 7 3 -- 6 4 -- 5	1 -- 8 2 -- 7 3 -- 6 4 -- 5	1 -- 8 2 -- 7 3 -- 6 4 -- 5	1 -- 8 2 -- 7 3 -- 6 4 -- 5

Table 4-5 Board Size Selection Jumper Settings

G) BS=\$040000 H) BS=\$030000 I) BS=\$020000 J) BS=\$010000

B1	1 -- 8	1 8	1 -- 8	1 8
	2 -- 7	2 7	2 -- 7	2 -- 7
	3 -- 6	3 -- 6	3 -- 6	3 -- 6
	4 -- 5	4 -- 5	4 -- 5	4 -- 5

B2	1 -- 8	1 -- 8	1 -- 8	1 -- 8
	2 -- 7	2 -- 7	2 -- 7	2 -- 7
	3 -- 6	3 -- 6	3 -- 6	3 -- 6
	4 -- 5	4 -- 5	4 -- 5	4 -- 5

K) BS=\$000000

B1	1 -- 8
	2 -- 7
	3 -- 6
	4 -- 5

B2	1 -- 8
	2 -- 7
	3 -- 6
	4 -- 5

Table 4-6: Address Modifier Codes

HEX Code	Address Modifier						Function
	5	4	3	2	1	0	
3F	H	H	H	H	H	H	Standard Supervisory Block Transfer
3E	H	H	H	H	H	L	Standard Supervisory Program Access
3D	H	H	H	H	L	H	Standard Supervisory Data Access
3C	H	H	H	H	L	L	Reserved
3B	H	H	H	L	H	H	Standard Non-Privileged Block Transfer
3A	H	H	H	L	H	L	Standard Non-Privileged Program Access
39	H	H	H	L	L	H	Standard Non-Privileged Data Access
38	H	H	H	L	L	L	Reserved
37	H	H	L	H	H	H	Reserved
36	H	H	L	H	H	L	Reserved
35	H	H	L	H	L	H	Reserved
34	H	H	L	H	L	L	Reserved
33	H	H	L	L	H	H	Reserved
32	H	H	L	L	H	L	Reserved
31	H	H	L	L	L	H	Reserved
30	H	H	L	L	L	L	Reserved
2F	H	L	H	H	H	H	Reserved
2E	H	L	H	H	H	L	Reserved
2D	H	L	H	H	L	H	Short Supervisory Access
2C	H	L	H	H	L	L	Reserved
2B	H	L	H	L	H	H	Reserved
2A	H	L	H	L	H	L	Reserved
29	H	L	H	L	L	H	Short Non-Privileged Access
28	H	L	H	L	L	L	Reserved
27	H	L	L	H	H	H	Reserved
26	H	L	L	H	H	L	Reserved
25	H	L	L	H	L	H	Reserved
24	H	L	L	H	L	L	Reserved
23	H	L	L	L	H	H	Reserved
22	H	L	L	L	H	L	Reserved
21	H	L	L	L	L	H	Reserved
20	H	L	L	L	L	L	Reserved

L = low signal level

H = high signal level

cont'd ...

HEX Code	Address Modifier						Function
	5	4	3	2	1	0	
1F	L	H	H	H	H	H	User Defined
1E	L	H	H	H	H	L	User Defined
1D	L	H	H	H	L	H	User Defined
1C	L	H	H	H	L	L	User Defined
1B	L	H	H	L	H	H	User Defined
1A	L	H	H	L	H	L	User Defined
19	L	H	H	L	L	H	User Defined
18	L	H	H	L	L	L	User Defined
17	L	H	L	H	H	H	User Defined
16	L	H	L	H	H	L	User Defined
15	L	H	L	H	L	H	User Defined
14	L	H	L	H	L	L	User Defined
13	L	H	L	L	H	H	User Defined
12	L	H	L	L	H	L	User Defined
11	L	H	L	L	L	H	User Defined
10	L	H	L	L	L	L	User Defined
0F	L	L	H	H	H	H	Extended Supervisory Block Transfer
0E	L	L	H	H	H	L	Extended Supervisory Program Access
0D	L	L	H	H	L	H	Extended Supervisory Data Access
0C	L	L	H	H	L	L	Reserved
0B	L	L	H	L	H	H	Extended Non-Privileged Block Transfer
0A	L	L	H	L	H	L	Extended Non-Privileged Program Access
09	L	L	H	L	L	H	Extended Non-Privileged Data Access
08	L	L	H	L	L	L	Reserved
07	L	L	L	H	H	H	Reserved
06	L	L	L	H	H	L	Reserved
05	L	L	L	H	L	H	Reserved
04	L	L	L	H	L	L	Reserved
03	L	L	L	L	H	H	Reserved
02	L	L	L	L	H	L	Reserved
01	L	L	L	L	L	H	Reserved
00	L	L	L	L	L	L	Reserved

L = low signal level

H = high signal level



### 4.3 Address Modifier Code Selection

The SYS68K/AGC-1 has a jumper selectable Address Modifier Code. Table 4-6 shows the VMEbus AM codes. Each allowed AM Code or combinations of allowed AM Codes can be set. The selection is coded by jumper settings in the field at B4.

Fig. 4-4 shows the location of the jumper field on the PC board.

Table 4-6 shows the VMEbus AM Codes.

Table 4-7 shows the jumper selectable connections.

When selecting the SHORT I/O AM Codes, the Base Address jumper selection on B3 is ignored. The Base Address for SHORT I/O access is jumpered with the jumper field B6. Please be make sure that the board size jumper setting is set to \$00000 for SHORT I/O access.



**TABLE 4-7: Address Modifier Code Selection**

a) AM Code= 3E	b) AM Code= 3D	c) AM Code= 3A	d) AM Code= 39
1 -- 16	1 -- 16	1 -- 16	1 -- 16
2 -- 15	2 -- 15	2 -- 15	2 -- 15
3 -- 14	3 -- 14	3 -- 14	3 -- 14
4 -- 13	4 -- 13	4 -- 13	4 -- 13
5 -- 12	5 -- 12	5 -- 12	5 -- 12
6 -- 11	6 -- 11	6 -- 11	6 -- 11
7 -- 10	7 -- 10	7 -- 10	7 -- 10
8 -- 9	8 -- 9	8 -- 9	8 -- 9

a) AM Code= 2D	b) AM Code= 29	c) AM Code= 1F	d) AM Code= 1E
1 -- 16	1 -- 16	1 -- 16	1 -- 16
2 -- 15	2 -- 15	2 -- 15	2 -- 15
3 -- 14	3 -- 14	3 -- 14	3 -- 14
4 -- 13	4 -- 13	4 -- 13	4 -- 13
5 -- 12	5 -- 12	5 -- 12	5 -- 12
6 -- 11	6 -- 11	6 -- 11	6 -- 11
7 -- 10	7 -- 10	7 -- 10	7 -- 10
8 -- 9	8 -- 9	8 -- 9	8 -- 9

a) AM Code= 1D	b) AM Code= 1C	c) AM Code= 1A	d) AM Code= 19
1 -- 16	1 -- 16	1 -- 16	1 -- 16
2 -- 15	2 -- 15	2 -- 15	2 -- 15
3 -- 14	3 -- 14	3 -- 14	3 -- 14
4 -- 13	4 -- 13	4 -- 13	4 -- 13
5 -- 12	5 -- 12	5 -- 12	5 -- 12
6 -- 11	6 -- 11	6 -- 11	6 -- 11
7 -- 10	7 -- 10	7 -- 10	7 -- 10
8 -- 9	8 -- 9	8 -- 9	8 -- 9

a) AM Code= 18	b) AM Code= 17	c) AM Code= 16	d) AM Code= 15
1 -- 16	1 -- 16	1 -- 16	1 -- 16
2 -- 15	2 -- 15	2 -- 15	2 -- 15
3 -- 14	3 -- 14	3 -- 14	3 -- 14
4 -- 13	4 -- 13	4 -- 13	4 -- 13
5 -- 12	5 -- 12	5 -- 12	5 -- 12
6 -- 11	6 -- 11	6 -- 11	6 -- 11
7 -- 10	7 -- 10	7 -- 10	7 -- 10
8 -- 9	8 -- 9	8 -- 9	8 -- 9

Table 4-7: ADDRESS MODIFIER CODE SELECTION

a) AM Code= 14	b) AM Code= 3D,39	c) AM Code= 2D,29	d) AM Code= 3A,39,38
1 16	1 -- 16	1 16	1 -- 16
2 15	2 -- 15	2 15	2 15
3 14	3 -- 14	3 14	3 -- 14
4 13	4 13	4 -- 13	4 13
5 -- 12	5 12	5 -- 12	5 -- 12
6 -- 11	6 -- 11	6 -- 11	6 -- 11
7 10	7 10	7 -- 10	7 -- 10
8 -- 9	8 -- 9	8 -- 9	8 -- 9
	default set-up		
a) AM Code= 3E,3D,3C	b) AM Code= 3E,3A	c) AM Code= 3E,3F,3E,3D,3C,3B,3A,39,38	d) AM Code= don't care
1 16	1 -- 16	1 -- 16	1 -- 16
2 -- 15	2 -- 15	2 -- 15	2 -- 15
3 -- 14	3 -- 14	3 -- 14	3 -- 14
4 13	4 13	4 13	4 -- 13
5 -- 12	5 -- 12	5 -- 12	5 -- 12
6 -- 11	6 11	6 -- 11	6 -- 11
7 -- 10	7 -- 10	7 -- 10	7 -- 10
8 -- 9	8 -- 9	8 -- 9	8 -- 9

#### 4.4 Functional Groups on the SYS68K/AGC-1

The SYS68K/AGC-1 board is considered to consist of the following functional groups:

- Graphics Processor
- 3 Colour Look-up Tables
- Video Memory

The Graphics Processor consists of:

- the ACRTC 63484
- two interrupt channels of the BIM

A Colour Look-up Table consists of:

- 256 Byte of colour value RAM

The Video Memory consists of:

- 2M Byte of DRAM

#### 4.5 Access to the Devices on the SYS68K/AGC-1 Board

There are six addressable devices installed :

ACRTC	63484
BIM	68153
GCP-red	8151
GCP-green	8151
GCP-blue	8151
Video-RAM	

All user accessible registers and RAM areas are directly addressable on the SYS68K/AGC-1.

All devices and the RAM area have a 16 bit wide data bus.

The BIM and the three GCP`s are accessible only on the lower byte (D0-D7).

Read, write and read-modify-write accesses are supported. (the limitations are documented in the device data sheets).

The access addresses of accessible Registers and RAM areas on the board are given in Chapter 4.1.

## 4.6 Addressing of the Colour Look-up Tables

The 3 Colour Look-up tables for red, green and blue are fully independent of each other, and they are all identical in their functions.

The 256 colour entries provide 256 colours (in 8 bit/pixel mode) or 16 different tables with 16 colours (in 4bit/pixel mode). This makes it possible to display 256 colours (16 colours in 4 bit/pixel mode) out of 16.777216 million.

The colour look-up tables are addressed in word mode. The lower byte contains the colour value, the upper byte is always \$FF. A byte-wide addressing is possible but then every even address represents a colour entry.

The bit value of a given pixel points on the respective colour entry of the colour look-up tables.

Example: The look-up table for the RED colour is located at address \$BFA000

### 4 bit/pixel mode

Pixel value	Look-up table address
\$04	\$BFA008
\$0A	\$BFA014
\$0F	\$BFA01E

### 8 bit/pixel mode

Pixel value	Look-up table address
\$10	\$BFA020
\$55	\$BFA0AA
\$FF	\$BFA1FE

To avoid a 'flickering' display the colour look-up tables should be loaded before the initialisation of the ACRTC or during the vertical retrace period. This could be done Interrupt-driven or by polling the Raster Count register of the ACRTC (see Chapter 4.7.1 and 4.9.7).

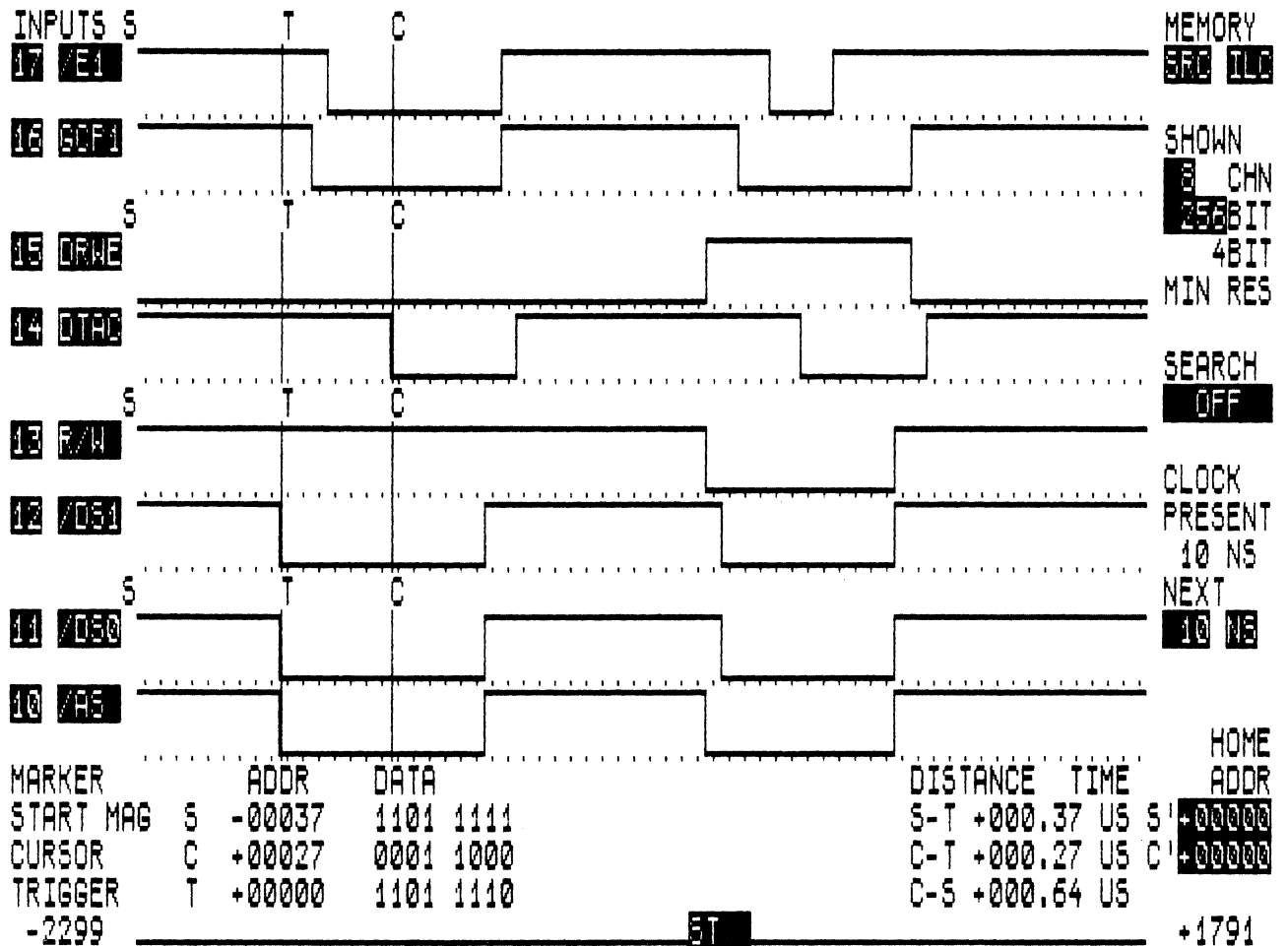
Figure 4-5 shows the timing diagram of an access to the colour look-up table access.

This page is intentionally left blank.



Figure 4-5: Access to the Colour Look-up Table

READ CYCLE FOLLOWED BY WRITE CYCLE



Characteristics	min.	max.
AS low to DTACK low	250ns	310ns
DS low to DTACK low	230ns	290ns
AS low to GCPEN1 low	110ns	150ns
DS low to GCPEN1 low	110ns	150ns
Access Cycle Time AS low to DTACK high	510ns	630ns

#### 4.6.1 Colour Switch Mode

In the 4 bit/pixel mode the look-up tables can be switched in the horizontal retrace period (see Chapter 4.9.8). By doing this it is possible to display 256 different colours in one picture using the 4 bit mode. The restriction is that only one colour look-up table set-up per line can be used. The switching is done by setting the attribut bits 0-3 in the Display Control Register of the ACRTC to 1 (see Chapter 4.9.6).

This switch colour mode can be chosen by the jumper settings of B18.

Fig. 4-6 shows the jumper location on the board.

Table 4-8 gives the correct jumper settings.





Table 4-8: Jumper Settings of B18 (SCM)

a) SCM = 0

Number of tables = 1  
DEFAULT

1	8
2	7
3	6
4	5

b) SCM=1

Number of tables = 2

1	8
2	7
3	6
4	-- 5

c) SCM = 2

Number of tables = 4

1	8
2	7
3	-- 6
4	-- 5

d) SCM=3

Number of tables = 8

1	8
2	-- 7
3	-- 6
4	-- 5

e) SCM = 5

Number of tables = 16

1	-- 8
2	-- 7
3	-- 6
4	-- 5

#### 4.6.2 Blink Switch Mode

In this mode the Blink Feature of the ACRTC is supported (see Chapter 4.9.14). Depending on the Blink Mode of the ACRTC the Colour Look-up table is switched and so certain symbols or areas of the frame may appear in another colour. This feature can be used in the 4 bit/pixel mode. In the 8 bit/pixel mode the number of bits/pixel will be reduced respectively (see Chapter 4.9.6).

The Blink Switch Mode is enabled through the jumper settings of B8.

Fig. 4-7 shows the jumper location on the board.

Table 4-9 gives the correct jumper settings.

Figure 4-7: Jumper Location Diagram of B8 (Blink Switch Mode)

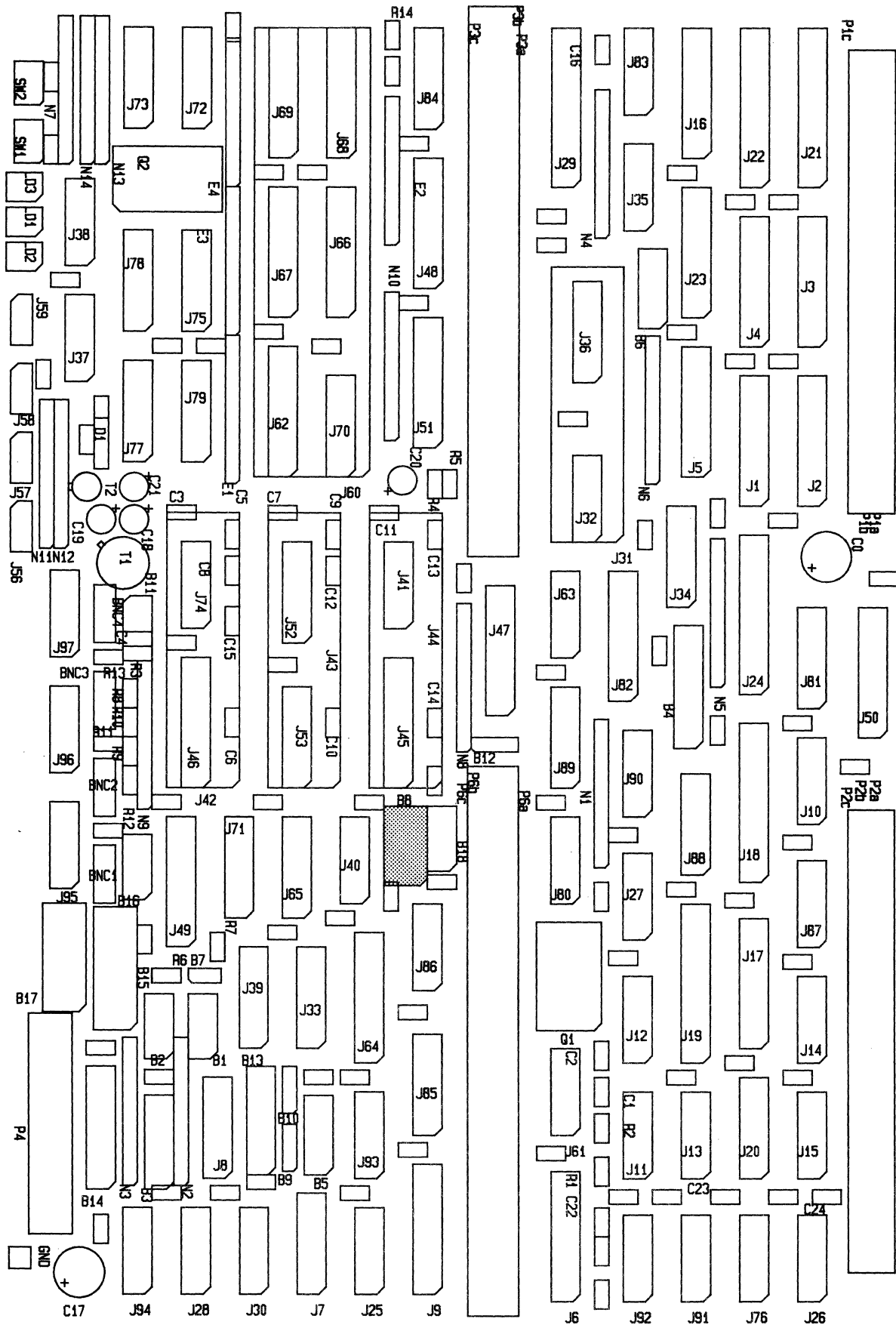




Table 4-9: Jumper Settings of B8 (BCM)

a) BCM = 0

BLINK1 disabled  
BLINK2 disabled

1	--	2	3
4	--	5	6
7		8	9
10	--	11	12
13	--	14	15

b) BCM=1

BLINK1 enabled  
BLINK2 disabled

1		2	--	3
4	--	5		6
7		8		9
10	--	11		12
13	--	14		15

c) BCM = 2

BLINK1 disabled  
BLINK2 enabled

1	--	2		3
4		5	--	6
7		8		9
10	--	11		12
13	--	14		15

d) BCM=1

BLINK1 enabled  
BLINK2 enabled

1		2	--	3
4		5	--	6
7		8		9
10	--	11		12
13	--	14		15

Default Condition

## 4.7 Addressing the BIM 68153

### 4.7.1 Interrupt Structure of the SYS68K/AGC-1

The VMEbus supports interrupting with seven interrupt request lines IRQ1-IRQ7. The seven lines represent seven interrupt request levels. IRQ1 has the lowest priority, IRQ7 has the highest. More than one board can access the same IRQ line at the same time, one board may request interrupts on several levels simultaneously.

When the interrupt handler enters the interrupt acknowledge cycle to honour an interrupt request, it asserts the /IACK line, puts the level code on the address lines A1-A3 and drives /AS low.

The interrupters have to decide which one has been acknowledged by the interrupt handler.

The VMEbus defines a Daisy Chain structure for the interrupters. Each interrupter has an /IACKIN and an /IACKOUT pin. The /IACKOUT pin of a slot is connected to the /IACKIN pin of the next slot on the motherboard. When the board has an interrupt pending on the level being acknowledged, it puts the Interrupt Vector onto the data bus and asserts the /DTACK line. The interrupt acknowledge cycle terminates when the interrupt handler clears the /AS line.

When an interrupter receives the /IACKIN input signal, and does not have an interrupt pending on the level being acknowledged, it asserts its /IACKOUT output signal.

The SYS68K/AGC-1 board can drive an interrupt request on each of the 7 IRQ lines of the VMEbus. Several lines may be asserted simultaneously.

Interrupt requests can be generated by the status report of the ACRTC as well as on the VSYNC phase of the display.

By the last for example, it is possible to load new colour look-up-tables during the vertical retrace period so as to avoid a 'flashing' display.

The two not used channels of the BIM are reserved for expansion with the SYS68K/AGC-1X board.

The interrupt level and the interrupt vector is software programmable for both interrupts.

Figure 4-8 shows the BIM access timing diagram.

Figure 4-9 shows the Interrupt request timing.

Figure 4-8: Access to the BIM 68153

READ CYCLE FOLLOWED BY WRITE CYCLE

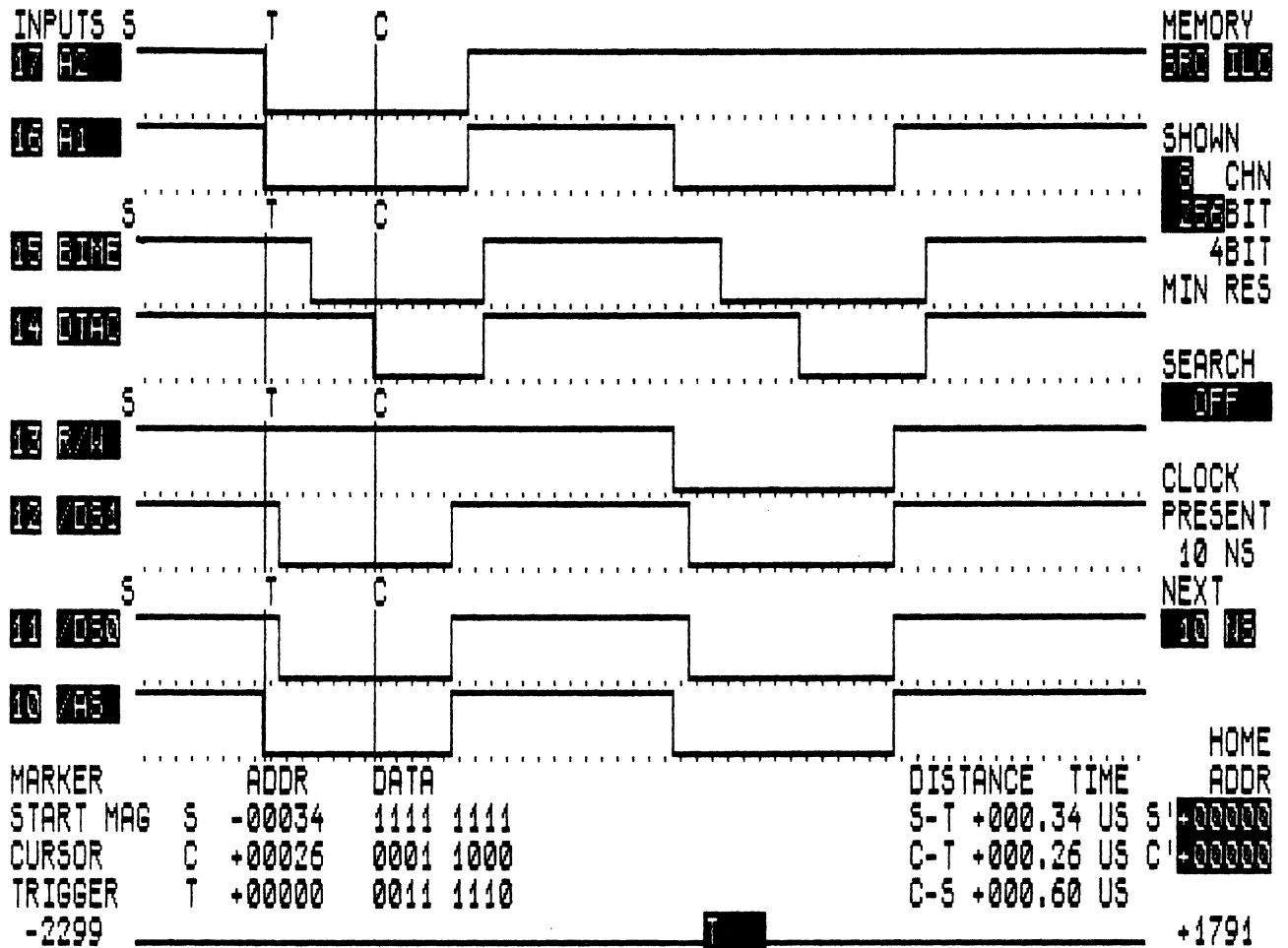


Table 4-10: Access Times for a Read Cycle to the BIM 68153

Characteristics	min.	max.
AS low to DTACK low	260ns	300ns
DS low to DTACK low	230ns	270ns
AS low to BIMEN low	90ns	110ns
DS low to BIMEN low	70ns	90ns
READ Cycle Time AS low to DTACK high	580ns	640ns
WRITE Cycle Time AS low to DTACK high	570ns	590ns

Figure 4-9: Interrupt Cycle Timing of the BIM 68153

VSYNC REQUESTS FOR INTERRUPT

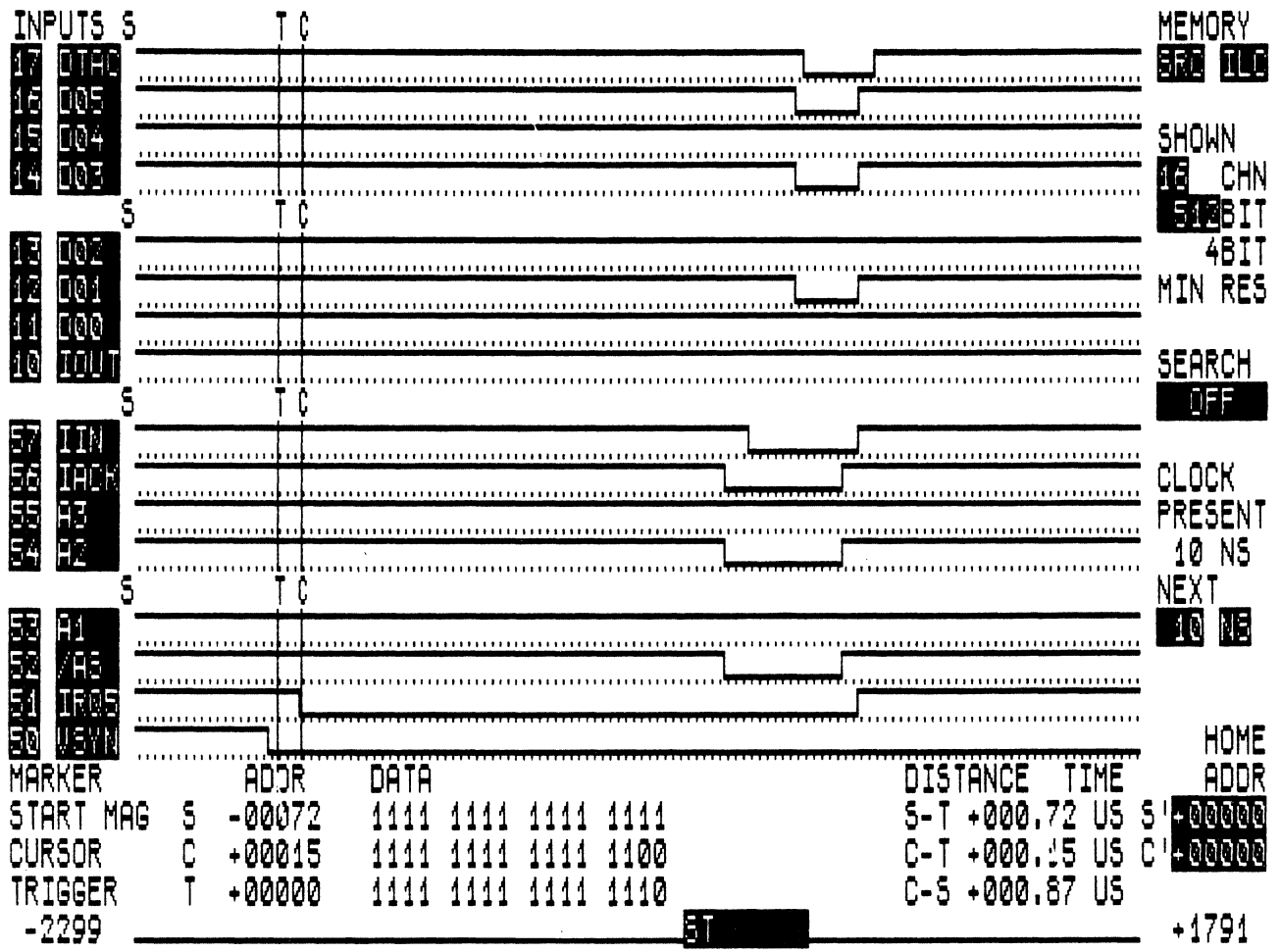


Table 4-11: Access Times of a Interrupt Acknowledge Cycle

Characteristics	min.	max.
VSYNC low to IRQ low	150ns	180ns
IACK low to DTACK low	370ns	430ns
IACKIN low to DTACK low	250ns	270ns
IACK low to IRQ high	650ns	710ns
IACKOUT high to AS high	10ns	20ns
IACK Cycle Time IACK low to DTACK high	730ns	790ns

#### 4.7.2 Programming the BIM (68153) on the SYS68K/AGC-1

The Bus Interrupter Modul (BIM) on the board transmits Interrupt Request Signals of the ACRTC and the VSYNC signal to the VMEbus.

The reset state of the BIM blocks all interrupting activity from the board to the VMEbus. The initialisation allows the software selection of the interrupt level for each interrupt signal individually. The interrupt vector is supplied by the BIM and therefore the BIM Control Register have to be programmed for internal vector generation.

The BIM is accessed word by word, but only the lower 8 data bits contain the register values, the upper bits are always \$FF.

Table 4-12 shows the address locations of the respective registers.

Please refer to the Data Sheet of the BIM 68153 for programming the control and vector registers.

Table 4-12: Address Location of the BIM 68153

BBA + \$0000	CONTROL REGISTER 1	reserved
BBA + \$0002	CONTROL REGISTER 2	ACRTC Interrupt
BBA + \$0004	CONTROL REGISTER 3	VSYNC Interrupt
BBA + \$0006	CONTROL REGISTER 4	reserved
BBA + \$0008	VECTOR REGISTER 1	reserved
BBA + \$000A	VECTOR REGISTER 2	ACRTC
BBA + \$000C	VECTOR REGISTER 3	VSYNC
BBA + \$000E	VECTOR REGISTER 4	reserved

## 4.8 Addressing the Video Memory

The Video Memory consists of 2M Byte contiguous dynamic Memory. The correct RAS-, CAS- and WRITE- Timing is selected through the jumper settings of B14, B15, B16, B17. All jumper settings of this jumperfield must not be modified.

The correct jumper settings are given in Table 4-11.

The memory organisation is word by word, but byte accesses are although possible. Depending on the chosen pixel mode, up to 4 pixel can be modified with one Read-Modify-Write Access.

The video memory is accessible in two ways via the ACRTC or direct via the VMEbus interface.

The access through the ACRTC is always possible, the access via the VMEbus is only supported if the board is jumpered for Standard Memory Access and the Video Memory is not blanked out (see Chapter 4.1).

The implementation of the DUAL ACCESS MODE (see hardware manual ACRTC 63484) together with the `dual port` mechanism of the video RAM avoids a `flickering` display when there are accesses during the display period.

The video memory area is located at address BBA + \$40000.

Fig. 4-10A and B shows the timing for the VMEbus access and Table 4-13A and B lists the time values.

The timing for the ACRTC access is shown in Fig.4-11 and the corresponding time values are outlined in Table 4-14.

Fig. 4-12 shows the jumper locations of B14, B15, B16 and B17.

Table 4-15 lists the correct jumper settings.



Figure 4-10: VMEbus Access to the Video RAM READ CYCLE

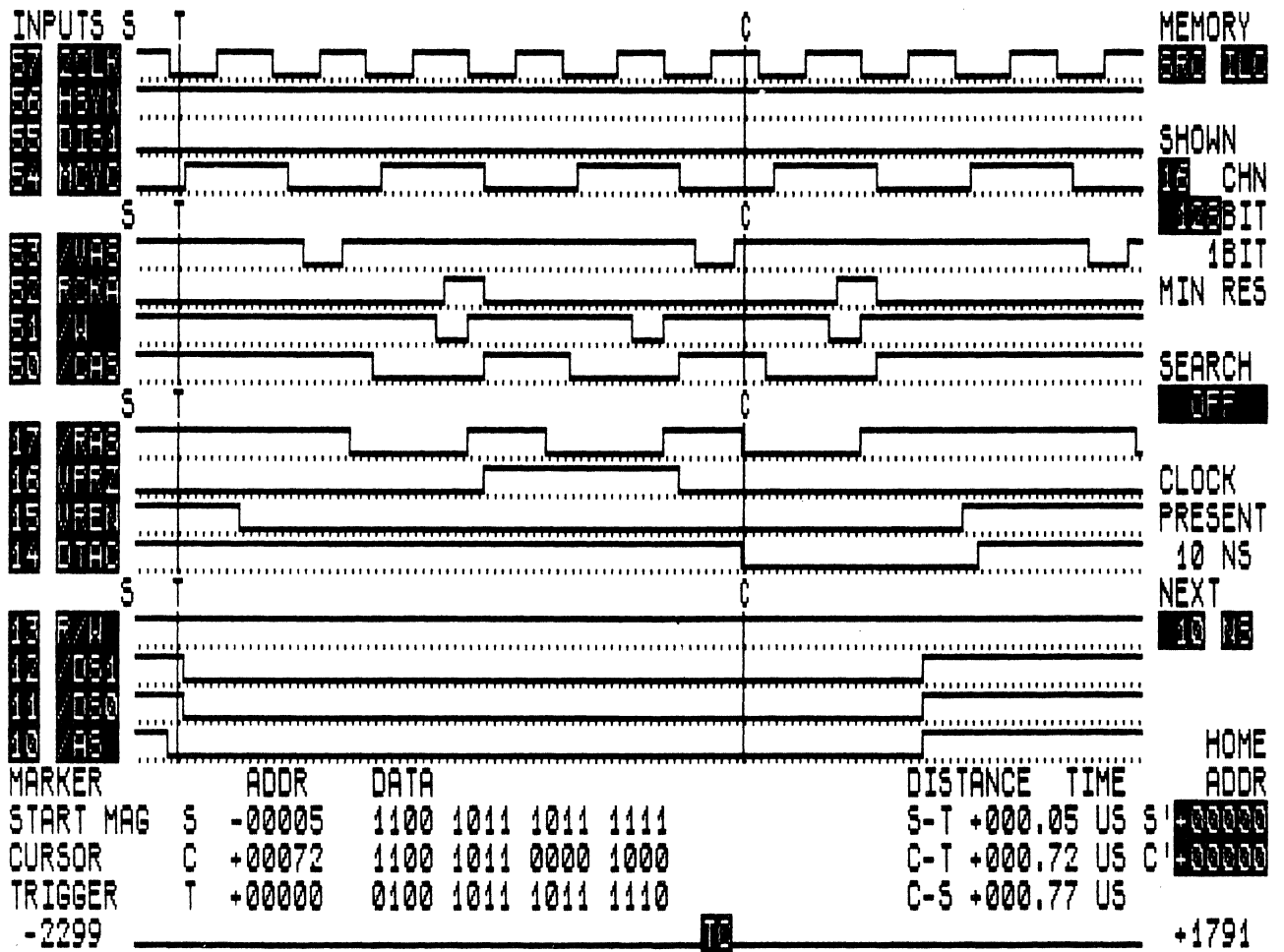


Table 13A: Time Values of a Video RAM Read Cycle

Characteristics	min.	max.
AS low to DTACK low	750ns	1000ns
DS low to DTACK low	730ns	980ns
AS low to RAS low	480ns	700ns
DS low to RAS low	460ns	680ns
READ Cycle Time AS low to DTACK high	1020ns	1330ns

Figure 4-10: VMEbus Access to the Video RAM WRITE CYCLE

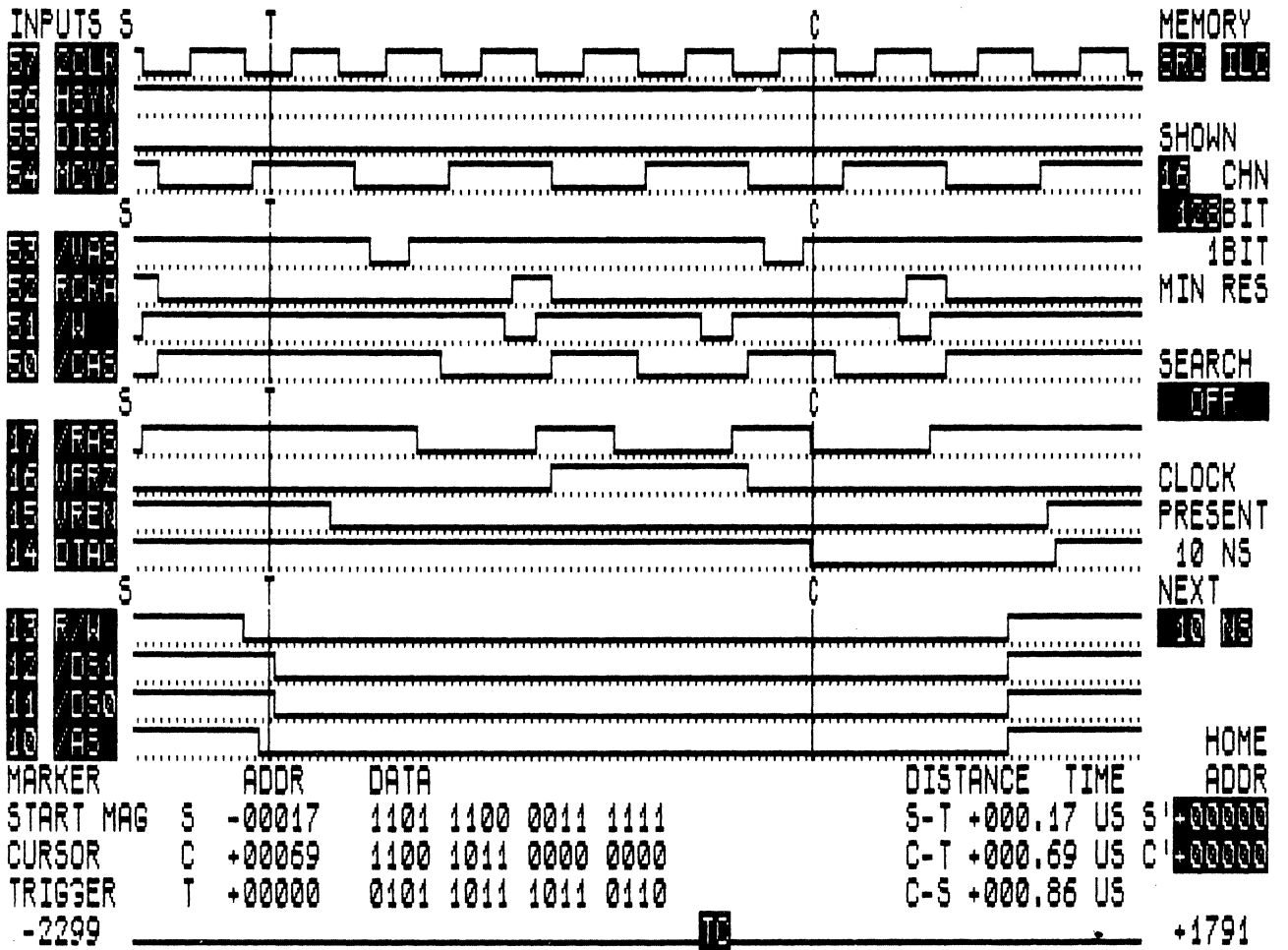


Table 4-13B: Time Values of a Video RAM Write Cycle

Characteristics	min.	max.
AS low to DTACK low	700ns	1180ns
DS low to DTACK low	680ns	1160ns
AS low to RAS low	700ns	1140ns
DS low to RAS low	680ns	1120ns
WRITE Cycle Time AS low to DTACK high	1010ns	1250ns

Figure 4-11: ACRTC Drawing Access to the Video RAM

READ-MODIFY-WRITE CYCLE

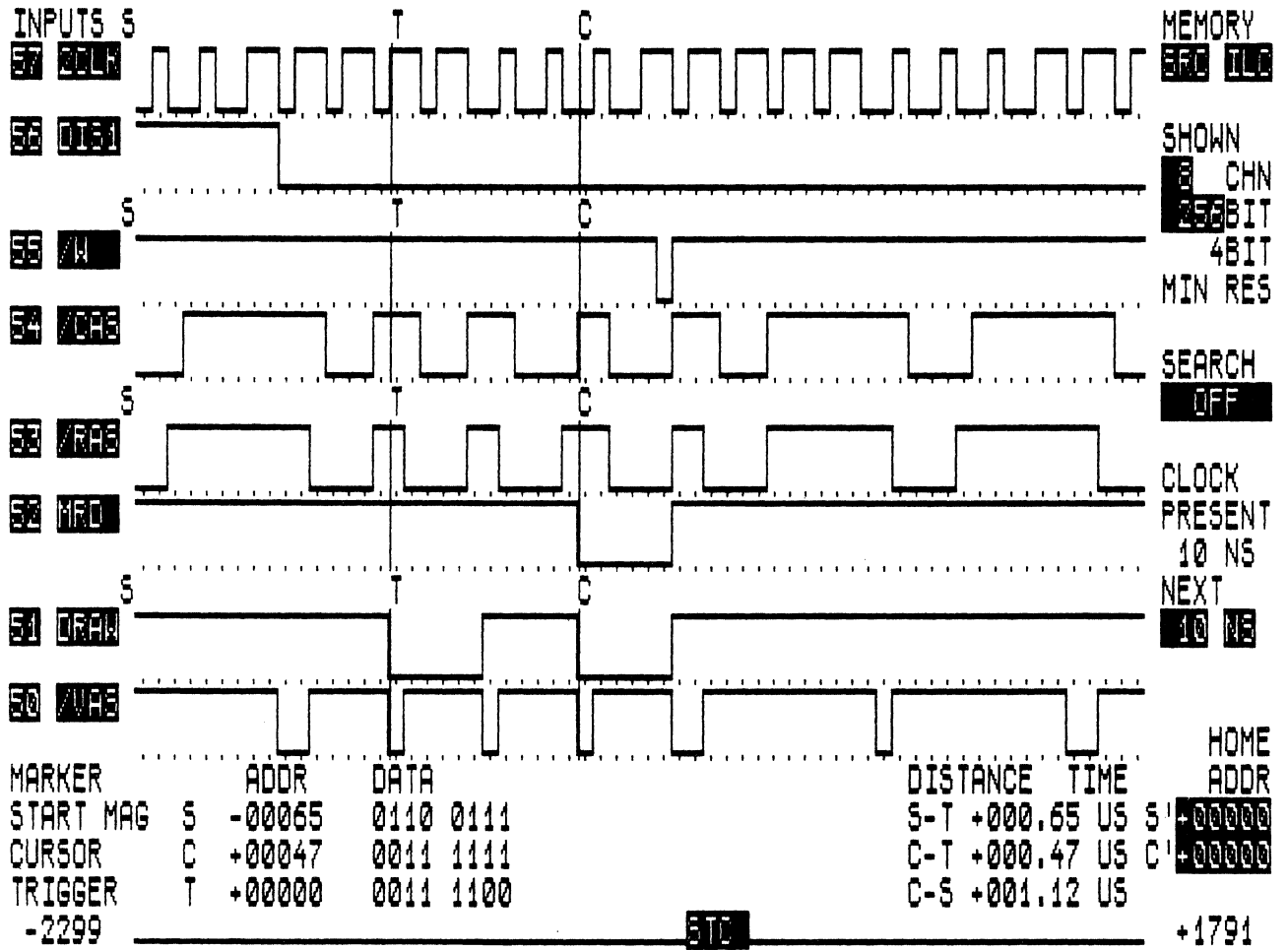


Table 4-14: Time Values of A ACRCCT Drawing Cycle to the Video RAM

Characteristics	min.	max.
VAS low to RAS low	30ns	60ns
VAS low to CAS low	50ns	100ns
RAS low to CAS low	20ns	40ns
CAS low to WRITE low	90ns	110ns
WRITE Pulse Duration	30ns	60ns
RAS Recovery Time	70ns	120ns
RMW Cycle Time	710ns	730ns



Table 4-15: Jumper Settings of B14, B15, B16 AND B17

JUMPER B14

DEFAULT CONDITION

1 16  
2 15  
3 -- 14  
4 13  
5 12  
6 11  
7 10  
8 -- 9

JUMPER B15

DEFAULT CONDITION

1 2 3  
4 5 6  
7 8 9  
10 11 -- 12  
13 -- 14 15  
16 17 18  
19 20 21  
22 23 24

JUMPER B16

DEFAULT CONDITION

1 -- 8  
2 7  
3 6  
4 5

JUMPER B17

DEFAULT CONDITION

1 2 3  
4 5 6  
7 8 9  
10 -- 11 12  
13 14 15  
16 17 18  
19 20 -- 21



#### 4.9 Addressing the ACRTC 63484

The ACRTC incorporates more than 200 byte of internal control registers and control RAM which is accessible by the host processor.

There is a distinction between two ways of accessing these registers. At first there are the direct addressable registers, the STATUS Register, the ADDRESS Register and the FIFO ENTRY. The STATUS and the ADDRESS register are distinguished by the R/W-Signal. The Status register is read-only and the Address registers are write only. The addresses of these registers are located at address BBA + \$3C000 for Standard Memory access (see Chapter 4-1).

The second group are the indirect addressable registers, which are partitioned in two groups.

##### a) Directly Accesible Register

These registers are accessed by writing the register address into the ADDRESS register and reading and writing the data over the FIFO ENTRY.

##### b) FIFO Accessible Register

These registers are accessed by writing the FIFO address (\$00) into the ADDRESS register. After operation, the READ or WRITE PARAMETER REGISTER command with the respective register address written into the FIFO.

The programming model is shown in Table 4-16 and the time values are listed in Table 4-16.

Figure 4-13 shows the access timing from the VMEbus to the 63484 ACRTC.

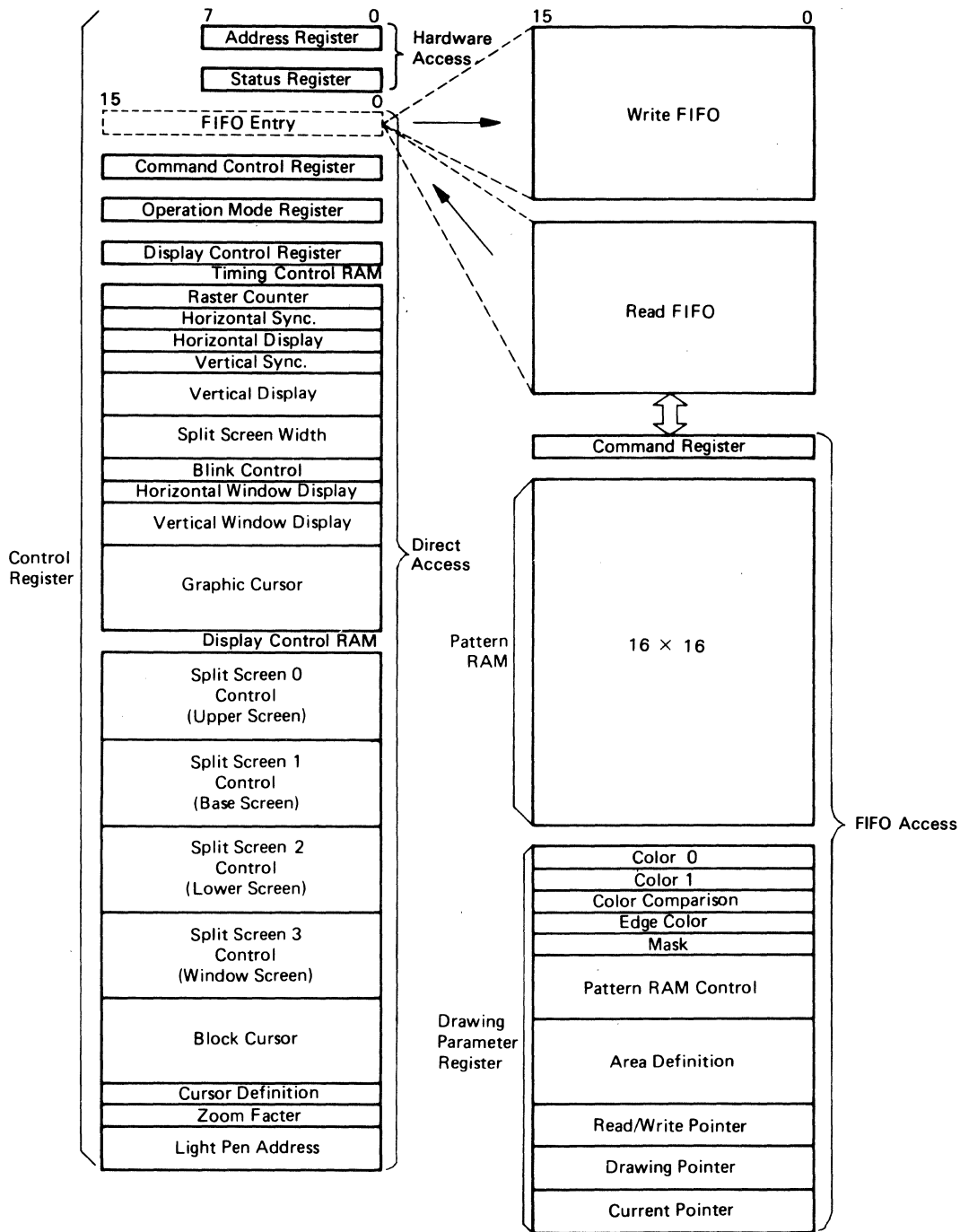
For the detailed register descriptions in this chapter, the following terminology is used.

For directly accessible registers, the register address is shown as `rNN` where NN is interpreted as an 8 bit hexadecimal value. For example, the Zoom Factor register address is \$EA hexadecimal, so the ZFRs register address is shown as `rEA`.

For FIFO accessible Drawing Parameter registers, the register address is shown as `PrNN`. For example, the Colour Comparison register is addressed as parameter register \$2 hex, so the CMP register address is shown as `Pr02`.

When register diagrams are shown, unused bits will be marked with X. Unless stated otherwise, unused bits may be freely written with any value, and that value will be returned on subsequent reads of the register.

**Table 4-16: PROGRAMMING MODEL**



**Table 4-16: (con't) Programming Model**

CS	RS	RW	Reg. No.	Register Name	Abbr.	DATA (H)								DATA (L)											
						15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
1	-	-	-	-	-	.....																			
0	0	0	AR	Address Register	AR	.....																			
0	0	1	SR	Status Register	SR	.....																			
0	1	0	r00	FIFO Entry	FE	.....																			
1	0	0	r02	Command Control	CCR	ABT	PSE	DDM	CDM	DRC	GBM	CRE	ARE	CEE	LPE	RFE	RRE	WRE	WEE						
1	0	0	r04	Operation Mode	OMR	M/S	STR	ACP	WSS	CSK	DSK	RAM	GAI	ACM	RSM										
1	0	0	r06	Display Control	DCR	DSP	SE1	SE0	SE2	SE3	A T R														
-	-	-	r08	(undefined)	-	.....																			
1	0	0	r80	Raster Count	RCR	.....										R C									
1	0	0	r82	Horizontal Sync.	HSR	H C										H S W									
1	0	0	r84	Horizontal Display	HDR	H D S										H D W									
1	0	0	r86	Vertical Sync.	VSR	.....										V C									
1	0	0	r88	Vertical Display	VDR	V D S										V S W									
1	0	0	r8A	Split Screen Width	SSW	.....										S P 1									
1	0	0	r8C	Split Screen Width	SSW	.....										S P 0									
1	0	0	r8E	Split Screen Width	SSW	.....										S P 2									
1	0	0	r90	Blink Control	BCR	BON1				BOFF1				BON2				BOFF2							
1	0	0	r92	Horizontal Window Display	HWR	H W S										H W W									
1	0	0	r94	Vertical Window Display	VWR	.....										V W S									
1	0	0	r96	Vertical Window Display	VWR	.....										V W W									
1	0	0	r98	Graphic Cursor	GCR	C X E										C X S									
1	0	0	r9A	Graphic Cursor	GCR	.....										C Y S									
1	0	0	r9C	Graphic Cursor	GCR	.....										C Y E									
-	-	-	rA0	(undefined)	-	.....																			
0	1	0	rC0	Raster Addr. 0	RAR0	L R A 0										F R A 0									
1	0	0	rC2	Upper Screen Memory Width 0	MWR0	CHR	.....										M W 0								
1	0	0	rC4	Upper Screen Start Addr. 0	SAR0	S D A 0										S A 0 L									
1	0	0	rC6	Upper Screen Start Addr. 0	SAR0	S D A 0										S A 0 L									
1	0	0	rC8	Base Screen Raster Addr. 1	RAR1	L R A 1										F R A 1									
1	0	0	rCA	Base Screen Memory Width 1	MWR1	CHR	.....										M W 1								
1	0	0	rCC	Base Screen Start Addr. 1	SAR1	S D A 1										S A 1 L									
1	0	0	rCE	Base Screen Start Addr. 1	SAR1	S D A 1										S A 1 L									
1	0	0	rD0	Lower Screen Raster Addr. 2	RAR2	L R A 2										F R A 2									
1	0	0	rD2	Lower Screen Memory Width 2	MWR2	CHR	.....										M W 2								
1	0	0	rD4	Lower Screen Start Addr. 2	SAR2	S D A 2										S A 2 L									
1	0	0	rD6	Lower Screen Start Addr. 2	SAR2	S D A 2										S A 2 L									
1	0	0	rD8	Window Raster Addr. 3	RAR3	L R A 3										F R A 3									
1	0	0	rDA	Window Memory Width 3	MWR3	CHR	.....										M W 3								
1	0	0	rDC	Window Start Addr. 3	SAR3	S D A 3										S A 3 L									
1	0	0	rDE	Window Start Addr. 3	SAR3	S D A 3										S A 3 L									
1	0	0	rE0	Block Cursor 1	BCUR1	B C W 1				B C S R 1				B C A 1				B C E R 1							
1	0	0	rE2	Block Cursor 1	BCUR1	B C W 1				B C S R 1				B C A 1				B C E R 1							
1	0	0	rE4	Block Cursor 2	BCUR2	B C W 2				B C S R 2				B C A 2				B C E R 2							
1	0	0	rE6	Block Cursor 2	BCUR2	B C W 2				B C S R 2				B C A 2				B C E R 2							
1	0	0	rE8	Cursor Definition	CDR	C M				CON1				COFF1				CON2				COFF2			
1	0	0	rEA	Zoom Factor	ZFR	H Z F				V Z F				.....											
1	0	0	rEC	Light Pen Address	LPAR	.....										L P A L									
1	0	0	rEE	Light Pen Address	LPAR	.....										L P A L									
1	0	0	rFO	(undefined)	-	.....																			
-	-	-	rFE	(undefined)	-	.....																			

Note : 1 ..... "High" level  
0 ..... "Low" level

Register No.	Read/Write	Name of Register	Abbr.	Data (H)								Data (L)							
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pr00	R/W	Color 0	CLO	CLO															
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															
Pr12	R	Current Pointer **	CP	X															
Pr13				Y															
Pr14	-	.....	-	.....															
Pr15	-	.....	-	.....															

..... Always set to "0"

\*\* ..... Set binary complements for negative values of X and Y axis.

This page is intentionally left blank.

Figure 4-13: Access to the 63484 ACRTC

READ CYCLE FOLLOWED BY WRITE CYCLE

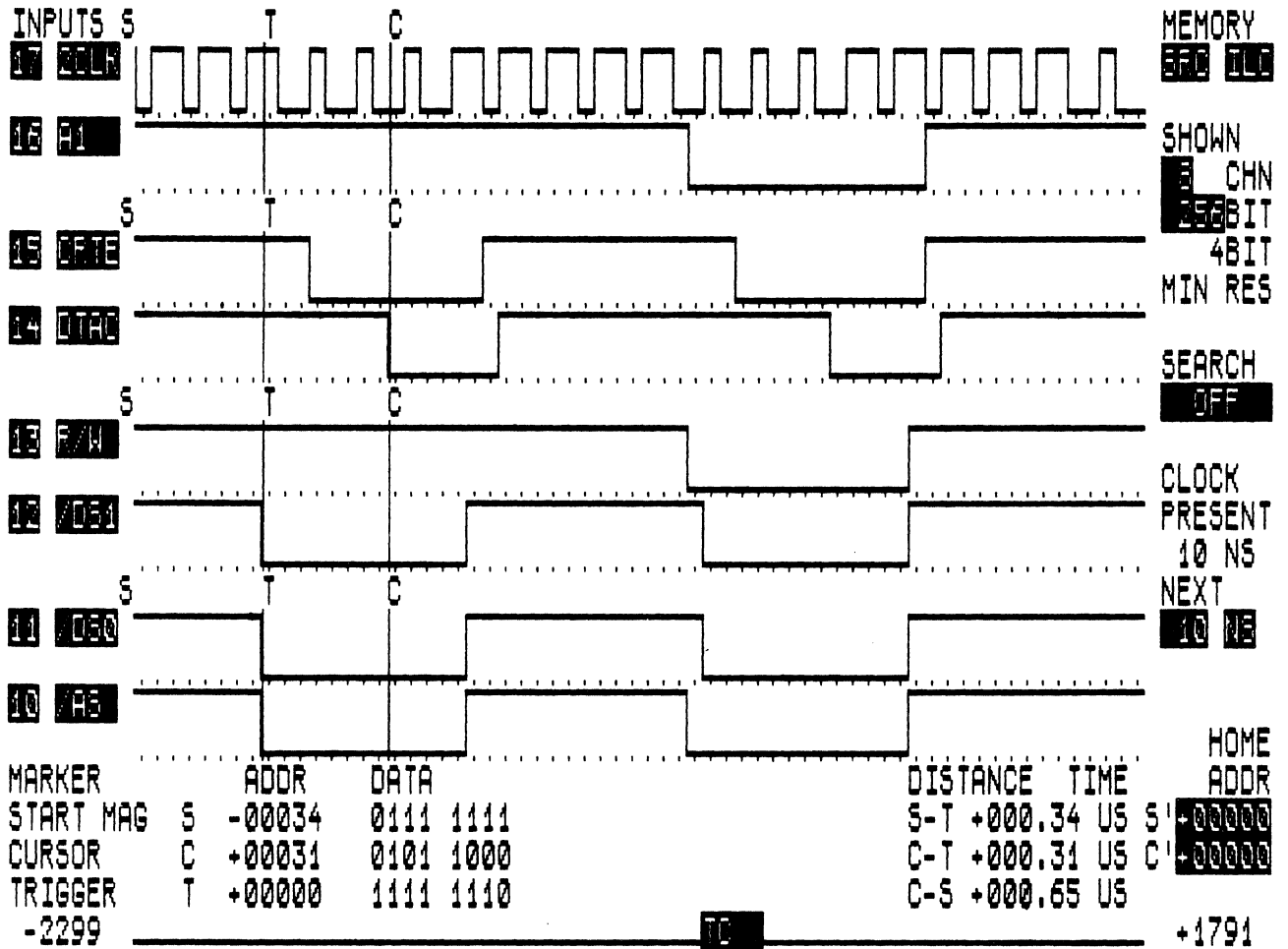
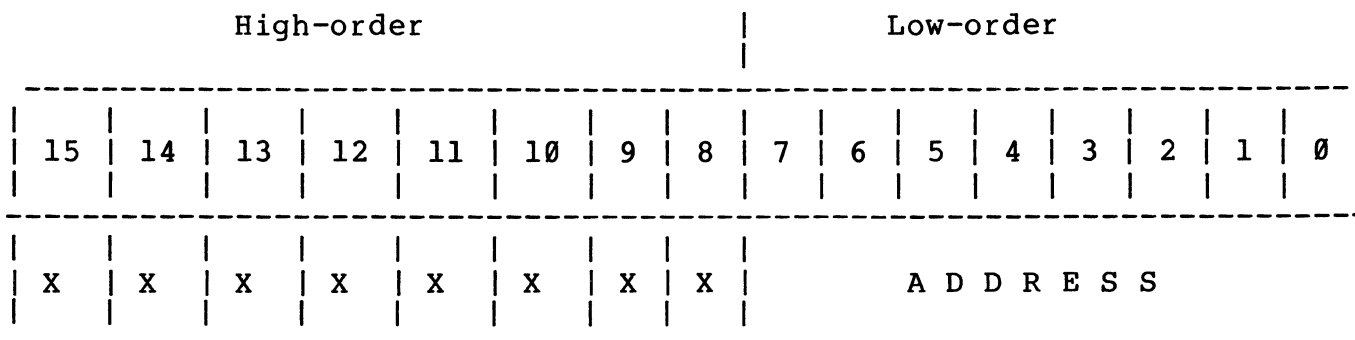


Table 4-17: Time values of a VMEbus access to the 63484 ACRC

Characteristics	min.	max.
AS low to DTACK low	310ns	370ns
DS low to DTACK low	280ns	360ns
AS low to CRTEN low	90ns	110ns
DS low to CRTEN low	70ns	90ns
READ Cycle Time AS low to DTACK high	660ns	690ns
WRITE Cycle Time As low to DTACK high	610ns	690ns

#### 4.9.1 ADDRESS Register (AR) Write Only



The AR is a write only register used to specify the address (\$0-\$FF) of the ACRTC control register to be accessed.

AR should be loaded with 0 to access the READ and WRITE FIFO's.

The Timing Control RAM and Display Control RAM occupy the register address space from r80-r9F and rC0-rEF respectively.

To support block move type initialisation/access of these registers, reads and writes to the register address space r80-rFF results in automatic incrementing of the AR. Therefore, the programmer need not explicitly address each register for sequential access.

AR is not incremented for accesses from r00 to r7F.

#### 4.9.2 Status Register (SR) Read Only

High-order								Low-order							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X	CER	ARD	CED	LPD	RFF	RFR	WFR	WFE

The SR is a read-only register containing 8 bits which reflect the state of internal status flags. If enabled by an interrupt enable bit in the CCR, a 1 bit in the corresponding SR flag will cause an interrupt to be generated.

After a hardware reset, the CED, WFE and WFR bits are set to 1 and all other bits are reset to 0 (\$FF23). The status register bits are briefly described below:

\* **Command Error Flag (CER: bit 7)**

CER set to 1 indicates that the ACRTC has detected an undefined command or invalid parameter.  
CED is cleared by setting the ABT bit in the CCR = 1.

\* **Area Detect Flag (ARD: bit 6)**

ARD is set to 1 depending on the AREA mode programmed for ACRTC graphic drawing commands. The ARD flag allows the host to detect whether the ACRTC has performed clipping or hitting during graphic drawing.  
ARD is cleared by execution of the RPR (Read Parameter Register) command or by setting the ABT bit in the CCR = 1.

\* **Command End (CED: bit 5)**

CED set to 1 indicates that the ACRTC is able to accept a new command.  
CED is cleared by writing a command to the write FIFO.

\* **Light Pen Detect (LPD: bit 4)**

LPD set to 1 indicates that the light pen strobe has occurred and the Light Pen Address Register contains the latched address.  
LPD is cleared by reading the LPAR or setting the ABT bit in CCR to 1.



\* **Read FIFO Full (RFF: bit 3)**

RFF set to 1 indicates that the read FIFO is full (contains 8 words of data).

RFF is cleared by reading at least one word from the FIFO or setting the ABT bit in CCR to 1.

\* **Read FIFO Ready (RFR: bit 2)**

RFR set to 1 indicates that the read FIFO contains one or more words of data.

RFR is cleared by reading all data from the read FIFO.

\* **Write FIFO Ready (WFR: bit 1)**

WFR set to 1 indicates that the write FIFO is not full, and host writes can occur. WFR is also set to 1 when the ABT bit in CCR is set to 1.

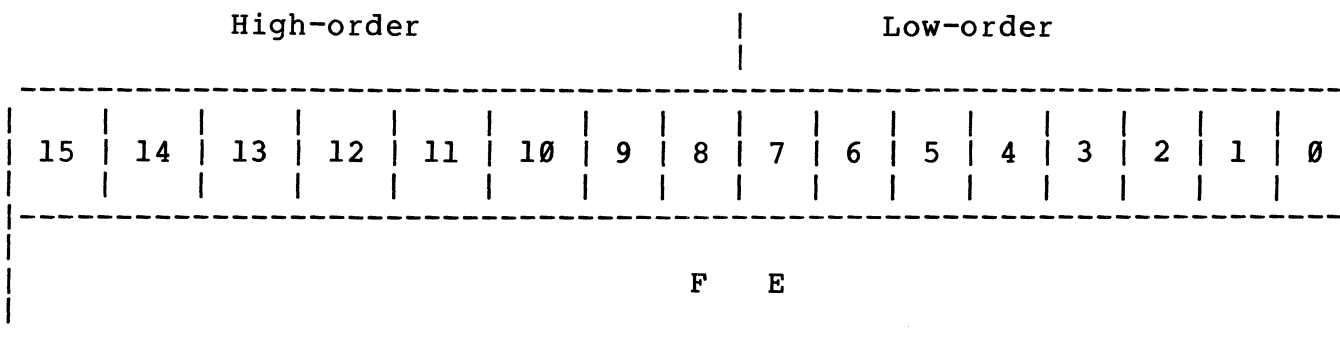
WFR is cleared when the write FIFO contains 8 words of data.

\* **Write FIFO Empty (WFE: bit 0)**

WFE set to 1 indicates that the write FIFO is empty. WFE is also set to 1 when the ABT bit in CCR is set to 1.

WFE is cleared when a 16 bit word data is written to the write FIFO.

### 4.9.3 FIFO Entry (FE: r00-r01)



When the AR contains the FIFO Entry address (r00), reads and writes to the ACRTC utilize the corresponding 8 word read or write FIFO.

In DMA transfer mode, the read and write FIFO's are selected regardless of the contents of AR and AR remains unchanged.

#### 4.9.4 Command Control Register (CCR: r02-r03)

High-order								Low-order								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
ABT	PSE	DDM	0	DRC		G	B	M	CRE	ARE	CEE	LPE	RFE	RRE	WRE	WEE

CCR controls command processing and enabling and disabling of interrupt requests. The 8 interrupt enable bits in the lower byte of CCR correspond directly to the 8 status flags in the Status register.

After hardware reset, the ABT bit is initialised to 1 and all other CCR bits are initialised to 0.

#### \* **Abort (ABT: bit 15)**

ABT	FUNCTION
0	ACRTC command execution is enabled. When ABT is changed from 0 to 1, the ACRTC cannot access the FIFO's until the Host issues a command.
1	ACRTC command execution is aborted and the read/write FIFOs are cleared. The Status register is set to \$23. Setting the ABT bit to 1 is equivalent to hardware reset assertion.

#### \* **Pause (PSE: bit 14)**

PSE	FUNCTION
0	ACRTC command execution is resumed.
1	ACRTC command execution is halted until PSE is reset to 0. ACRTC DMA (Data and Parameter) is halted until PSE is reset to 0.

\* DATA DMA Mode (DDM: bit 13)

DDM	FUNCTION
0	Data DMA transfer mode is disabled.
1	Data DMA transfer mode is enabled. Whether DMA is burst or cycle steal mode is determined by DRC (bit 11). DDM must be set before DMA data transfer commands are issued.

\* DMA Request Control (DRC: bit 11)

DRC	FUNCTION
0	Burst Mode: /DREQ is designated as a level signal. A maximum of 8 words data is transferred per DMA request.
1	Cycle Steal Mode: /DREQ is designated as a pulse signal. /DREQ is output for each word.

\* Graphic Bit Mode (GBM: bit 10 - bit 8)

GBM	Mode	# of colours	Pixel/word	
000	1 bit/pixel	1	16	not supported
001	2 bit/pixel	4	8	not supported
010	4 bit/pixel	16	4	supported
011	8 bit/pixel	256	2	supported
100	16 bit/pixel	65536	1	not supported
101				
111				

I N V A L I D

## Interrupt Enable Bit (IE: bit 7 - bit 0)

An interrupt is generated when an event flag in the Status register and the corresponding interrupt enable bit are both set to 1.

Bit		Name	Set to 1 enables interrupt for ...
7	Command Error	CRE	Command Error
6	Area Detect	ARE	Clipping and Hitting detection
5	Command End	CEE	Command Termination
4	Light Pen Detect	LPE	Light Pen Strobe asserted
3	Read FIFO Full	RFE	Read FIFO Full
2	Read FIFO Ready	RRE	Read FIFO Ready
1	Write FIFO Ready	WRE	Write FIFO Ready
0	Write FIFO Empty	WEE	Write FIFO Empty

#### 4.9.5 Operation Mode Register (OMR: r04-r05)

High-order							Low-order								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M/S	STR	ACP	0	CSK	DSK	0	0	1	1	ACM	RSM				

OMR determines major operating parameters and modes of the ACRTC. The two most significant bits (M/S and STR) are reset to 0 and all other bits are unaffected by a hardware reset.

##### \* Master/Slave (M/S: bit 15)

M/S defines whether the ACRTC operates as a master or slave when combined with other ACRTCs or video generating devices.

M/S

FUNCTION

0 Slave Mode:

ACRTC internal operations are reset on the rising edge of the /EXSYNC input.

1 Master Mode:

/EXSYNC is defined as an output. For non-interlace modes, the /EXSYNC output timing is the same as /VSYNC output timing. For interlace modes, the /EXSYNC output timing is generated by the /VSYNC output for the odd field.

##### \* Start (STR: bit 14)

The STR bit is used to start and stop ACRTC operation. Initializing of registers which control basic ACRTC operation should only be performed when STR is reset to 0.

STR

FUNCTION

0 ACRTC display and drawing operations are halted. The DRAM refresh address is output on the MAD lines and the internal time base for CRT control signals is reset.

1 ACRTC starts display and drawing operations. Drawing commands halted when STR was reset to 0 are resumed.



\* Display Skew (DSK: bit 9 - bit 8)

DSK defines the /DISP1, /DISP2 delay in units of memory cycles independent of video memory access mode.

DSK	FUNCTION
00	No skew.
01	/DISP1, /DISP2 are skewed by one memory cycle.
10	/DISP1, /DISP2 are skewed by two memory cycles.
11	/DISP1, /DISP2 are skewed by three memory cycles.

\* Access Mode (ACM: bit 3 - bit 2)

The ACRTC provides three frame buffer access modes - Single, Interleaved and Superimposed. Only the last two are supported by the SYS68K/AGC-1.

ACM	FUNCTION
10	Interleaved Access Mode (Double Access Mode 0) : The video memory is accessed twice every display cycle. Display and Drawing cycles are interleaved during each phase of the display cycle. The window has the highest priority and overlaps the Background screens.
11	Superimposed Access Mode (Double Access Mode 1) : The video memory is accessed twice every display cycle. The first phase accesses the Background screen, the second phase accesses the Window screen. In this case Background and Window have equal priority and are superimposed. This mode is only supported with the AGC-1X-board.

In Interleaved and Superimposed access modes the horizontal display width of the Background screen and the Window screen must be even. Also, for these modes, the relation between the starting position of the horizontal display on the Background screen and the starting position of the horizontal display on the window screen must be even number/even number or odd number/odd number.



\* Raster Scan Mode (RSM: bit 1 - bit 0)

RSM selects the ACRTC raster scan mode. The Interlaced Sync Mode simply repeats each raster address for both the odd and the even field. The Interlaced Sync & Video Mode displays alternate even and odd rasters on alternate even and odd fields. Note that for Interlaced modes the refresh frequency for a given dot on the screen is one-half that of the Non-Interlaced mode. Interlaced modes normally require a more persistent phosphor to avoid a flickering display.

RSM	FUNCTION
00	Non-Interlace Mode
01	
10	Interlace Sync Mode
11	Interlace Sync & Video Mode

#### 4.9.6 Display Control Register (DCR: r06-r07)

High-order								Low-order							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DSP	SE1	SE0	SE2	SE3	A T T R I B U T E										

DCR controls ACRTC screen organisation and 8 bits of user defined video attributes.

Logically, the ACRTC has a Background screen (Upper, Base, Lower) and a Window screen.

DCR allows the screens to be enabled, disabled and blanked. If the Upper, Lower and Window screens are disabled, they do not have to be defined.

#### \* DISP Signal Control (DSP: bit 15)

0 DSP defines the output mode of the /DISP1 and /DISP2 display timing signals.

DSP

FUNCTION

0 DSP1 is driven active low during the display period of the Background screen (combined horizontal and vertical display). /DISP2 is controlled similarly for the window screen.

This mode is supported with the AGC-1X board (optional available).

1 /DISP1 is driven active low during the horizontal display of both the Background and the Window screens.

/DISP2 is driven active low during the vertical display period of both the Background and the Window screen.

\* Split Enable 1 (SE1: bit 14)

SE1 allows the Base screen to be blanked. Base screen drawing can occur when the Base screen is blanked since video memory display access is suppressed. Note that the Base screen parameters must be defined, even when the Base screen is always blanked.

SE1

FUNCTION

- 0 The ACRTC inhibits the the display enable timing and the display address outputs associated with the Base screen. The area of the Base screen, though blanked, remains on the CRT screen.
- 1 The ACRTC outputs display enable timing and display addresses for the Base screen.

\* Split Enable 0 (SE0: bit 13 - bit 12)

SE0 allows the Upper split screen to be enabled, disabled and blanked. If always disabled, the Upper screen parameters need not be defined.

SE0

FUNCTION

- 0X The ACRTC disables the Upper screen. Therefore, the Background screen contains two parts maximum - the Base and the Lower screen. The Base screen is moved upward by the number of rasters in the disabled Upper screen.
- 10 The display enable timing outputs and display address outputs are inhibited for the Upper screen. The area of the Upper screen, though blanked, remains on the screen.
- 11 The ACRTC outputs display enable timing and display addresses for the Upper screen.

\* Split Enable 2 (SE2: bit 11 - bit 10)

SE2 allows the Lower split screen to be enabled, disabled and blanked. If always disabled, the Lower screen parameters need not be defined.

SE2

FUNCTION

- 0X The ACRTC disables the Lower screen. Therefore, the Background screen contains two parts maximum - the Base and the Upper screen. The Base screen is extended downward by the number of rasters in the disabled Lower screen.
- 10 The display enable timing outputs and display address outputs are inhibited for the Lower screen. The area of the Lower screen, though blanked, remains on the screen.
- 11 The ACRTC outputs display enable timing and display addresses for the Lower screen.

\* Split Enable 3 (SE3: bit 9 - bit 8)

SE3 allows enabling, disabling and blanking of the Window screen. When disabled or blanked, the overlapped Background screens are displayed.

SE3

FUNCTION

- 0X The ACRTC disables the Window screen and overlapped Background screens are displayed. If always disabled, the Window parameters need not be defined.
- 10 The display enable timing outputs and display address outputs are inhibited for the Window screen. The area of the Window screen, though blanked, remains on the screen. For superimposed access modes (only supported with the AGC-1X board), the overlapped Background screens are displayed.
- 11 The ACRTC outputs display enable timing and display addresses for the Window screen.

\* Attribute Control (ATR: bit 7 - bit 0)

These 8 bits can be freely programmed as user defined video attributes. They are output at the beginning of each raster and so programmed dynamically, ATR allows video attributes to be controlled on a raster by raster basis.

On the SYS68K/AGC-1 these bits are used to control the following functions :

ATT0 - ATT3 : Colour Look-up Table Control (CTC: bit 7 - bit 5)

These bits are used for the colour look-up table switching in the 4 bit/pixel mode.

CTC	FUNCTION
0000	Colour look-up table 0 is used for display
0001	Colour look-up table 1 is used for display
0010	Colour look-up table 2 is used for display
0011	Colour look-up table 3 is used for display
0100	Colour look-up table 4 is used for display
0101	Colour look-up table 5 is used for display
0110	Colour look-up table 6 is used for display
0111	Colour look-up table 7 is used for display
1000	Colour look-up table 8 is used for display
1001	Colour look-up table 9 is used for display
1010	Colour look-up table 10 is used for display
1011	Colour look-up table 11 is used for display
1100	Colour look-up table 12 is used for display
1101	Colour look-up table 13 is used for display
1110	Colour look-up table 14 is used for display
1111	Colour look-up table 15 is used for display

#### ATT4 - ATT6: Display Mode Control (DMC: bit 4 - bit 6)

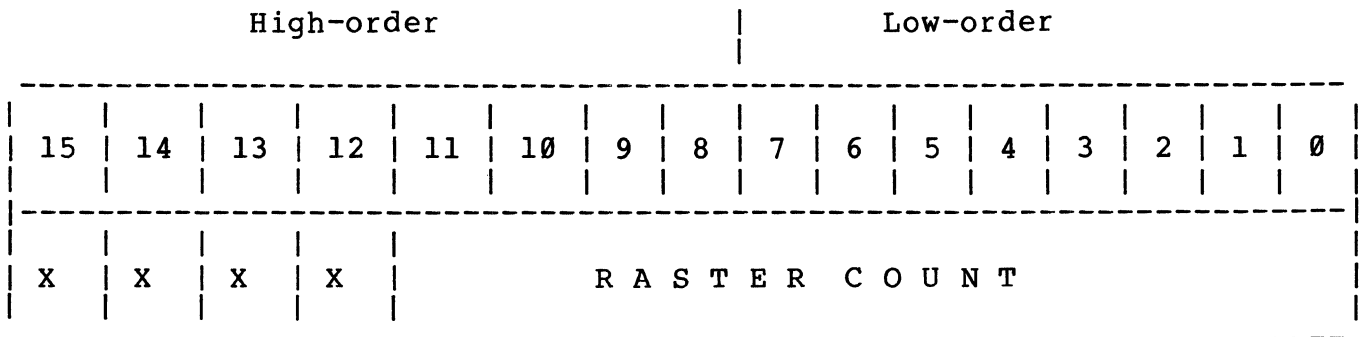
The DMC bits control the way of displaying the pixels on the screen.

DMC	FUNCTION
000	8 bit/pixel mode : In this mode 256 colours are displayable simultaneously. The maximum screen resolution is 800 x 600 pixel with 50 Hz noninterlaced.
001	4 bit/pixel mode 1 : In this mode 16 colours are displayed simultaneously. The maximum screen resolution is 1024 x 800 pixel with 60 Hz noninterlaced.
010	4 bit/pixel mode 2 : This mode enables the SWITCH COLOR and BLINK SWITCH mode respectively (see Chapter 4.6)
011	8 bit/pixel mode 2 : In this mode the 8 bit video data can be blanked out bit by bit and replaced through ATT0 - ATT3 or BLINK (i.e. 6 bit video data + ATT0 + ATT1 for colour look-up table switching).
100	   Reserved for expansion with the SYS68K/AGC-1X 
111	

#### ATT7: Smooth Scroll Bit 5 (SS5: bit 7)

This bit, together with the SDA bits in the Start Address Register (see Chapter 4.9.16) is used for horizontal smooth scroll in the 4 bit/pixel mode. It represents the least significant bit of the Start Dot Address.

#### 4.9.7 Raster Count Register (RCR: r80-r81)



RCR is a read-only register which contains the number of the raster currently being scanned on the CRT. Note that the initial RCR value after hardware reset is undefined. If RCR read operation is desired, the HSW should be set greater than or equal to 3. RCR should only be read when HSYNC is high. RCR is updated depending on the ACRTC raster scan modes as shown.

##### Non-Interlace:

RCR starts counting at 0 and increments by 1 sequentially.

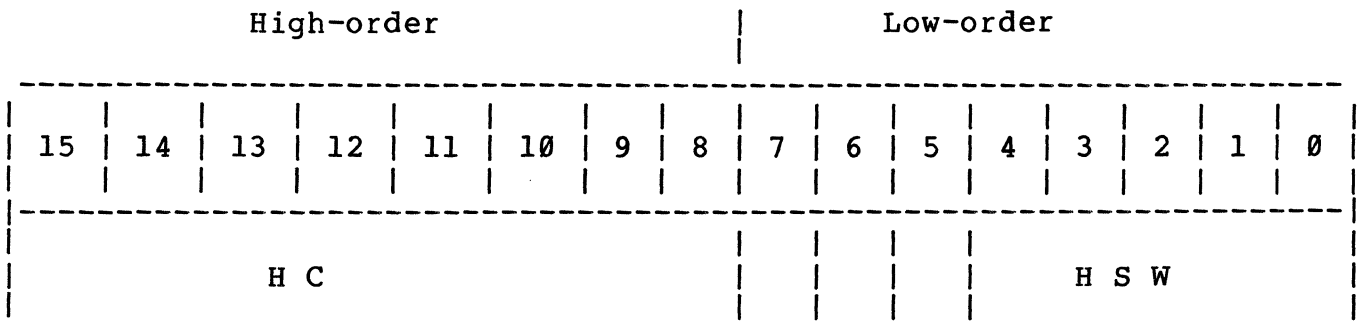
##### Interlace Sync:

RCR starts counting at 0 and increments by 1 sequentially in both the even and the odd fields. Because a dummy raster is added to the even field, the maximum raster number for the even field is one greater than for that for the odd field.

##### Interlace Sync & Video:

RCR starts counting at 0 in the even field and at 1 in the odd field, and incremented by 2 sequentially in both fields. The even field always has even raster numbers and the odd field always has odd raster numbers. A dummy raster is added to the even field as in Interlace Sync mode.

4.9.8 Horizontal Sync Register (RCR: r82-r83)



HSR defines the Horizontal Cycle (HC) and the Horizontal Sync Width (HSW).

**\* Horizontal Cycle (HC: bit 15 - bit 8)**

HC specifies the horizontal scan time (including the horizontal retrace period) in units of memory cycles (MC). On the SYS68K/AGC-1 one memory cycle is equivalent to 250 ns. HC is set depending on the specifications of the CRT display device. If H memory cycles are to be specified, HC should be set to H-1. When using interlaced scan modes, H should be an even number.

HC	Memory Cycle No.	Time
00000000	1	0.25 usec
00000001	2	0.50 usec
11111110	255	63.75 usec
11111111	256	64.00 usec



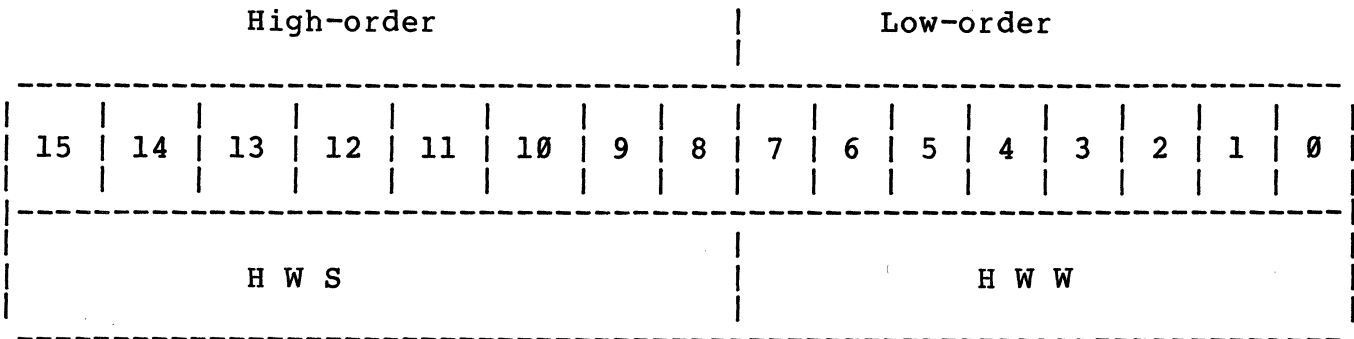
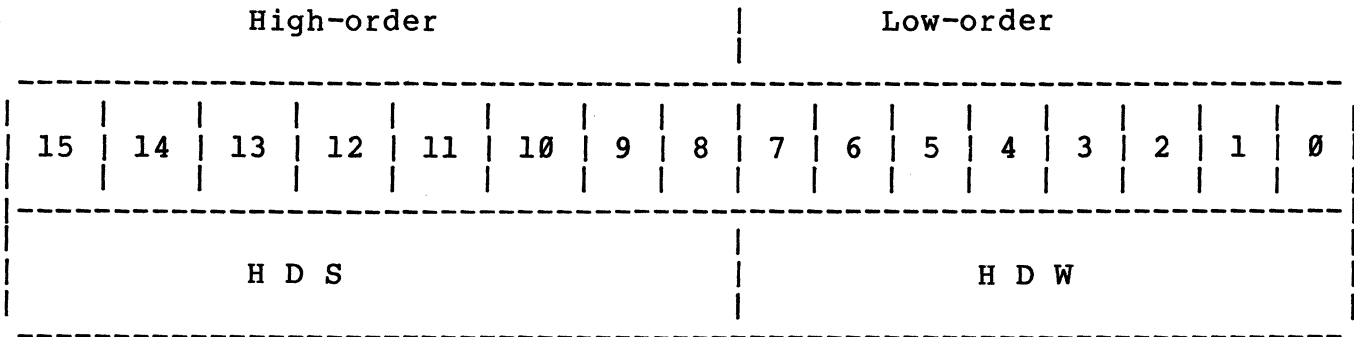
\* Horizontal Sync Width (HSW: bit 4 - bit 0)

HSW specifies the /HSYNC active low time in units of memory cycles. HSW is set depending on the specifications of the CRT display device. Valid values for HSW are 2 - 31. When using the RCR register, HSW must be 3 or greater.

HSW	Memory Cycle No.	Time
00010	2	0.50 usec
00011	3	0.75 usec
11110	30	7.50 usec
11111	31	7.75 usec

**4.9.9 Horizontal Display Register (HDR: r84-r85)**

**Horizontal Window Display Register (HWR: r92 - r93)**



HDR specifies the horizontal display start position and horizontal display width in units of memory cycles (1 MC = 250ns).

HWR specifies the horizontal Window start position and horizontal Window width in units of memory cycles.

**\* Horizontal Display Start (HDS: r84)**

HDS defines the interval between the rising edge of /HSYNC and the horizontal display starting point in units of memory cycles. If the Horizontal Display Start is HS memory cycles, HDS should be set to HS-1.

\* Horizontal Window Start (HWS: r92)

HWS defines the interval between the rising edge of /HSYNC and the horizontal Window display starting point in units of memory cycles. If the horizontal window display starting point is HS memory cycles, HWS should be set to HS-1.

HWS/HWS	Memory Cycle No.	Time
00000000	1	0.25 usec
00000001	2	0.50 usec
11111110	255	63.75 usec
11111111	256	64.00 usec

\* Horizontal Display Width (HDW: r85)

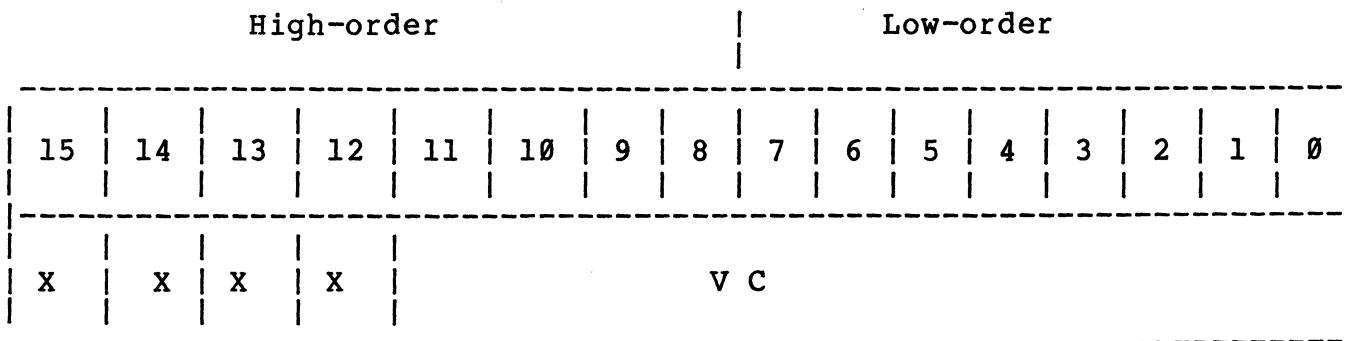
HDW defines the display period for one raster in units of memory cycles. If the horizontal display width is HW memory cycles, HDW should be set to HW-1.

\* Horizontal Window Width (HWW: r93)

HDW defines the window display period for one raster in units of memory cycles. If the horizontal window display width is HW memory cycles, HWW should be set to HW-1.

HDW/HWW	Memory Cycle No.	Time
00000000	1	0.25 usec
00000001	2	0.50 usec
11111110	255	63.75 usec
11111111	256	64.00 usec

#### 4.9.10 Vertical Sync Register (VSR: r86-r87)



VSR defines the period of the vertical scan cycle in units of rasters.

**\* Vertical Cycle (VC: bit 11 - bit 0)**

VC defines the vertical scan cycle period (including vertical retrace) in units of rasters. VC is set depending on the specifications of the CRT display device. The way VC is programmed depends on the ACRTC raster scan mode. VC should be programmed with a non-zero value.

**Non-Interlace Mode**

When the number of rasters is V, VC is set to V.

**Interlace Sync Mode**

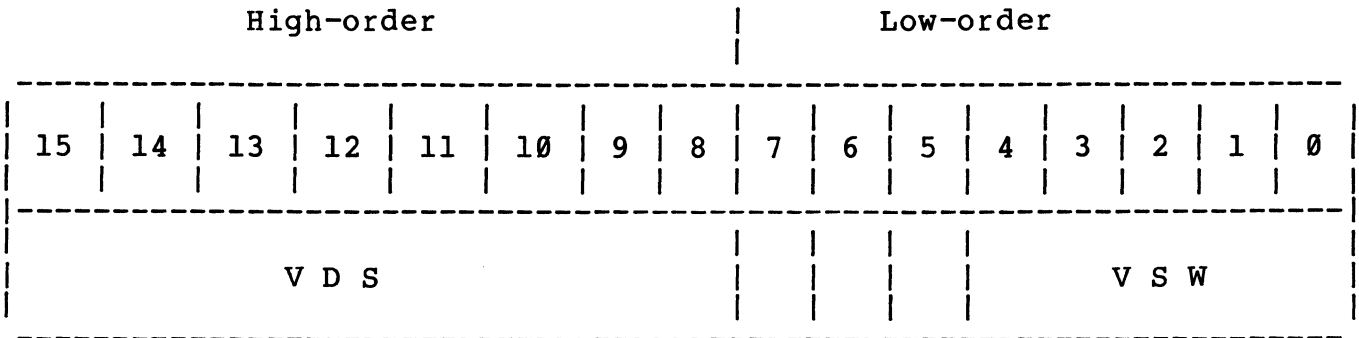
When the number of rasters in one field (odd or even) is V, VC is set to V. The total rasters in one frame is 2V+1 due to one dummy raster operation.

**Interlace Sync & Video Mode**

When the number of rasters in one frame (even field + odd field + dummy raster) is V, VC is set to V.

VC	Number of rasters
000000000001	1
000000000010	2
111111111110	4094
111111111111	4095

4.9.11 Vertical Display Register (VDR: r88-r89)



VDR defines the vertical sync width and vertical display start and width in units of rasters.

\* **Vertical Display Start (VDS: r88)**

VDS defines the period from the rising edge of /VSYNC to the vertical display start position in units of rasters. If the vertical display start position is the VS raster, VDS is set to VS-1. The way to program VDS depends on ACRTC raster scan modes as described for VSR (r86 - r87).

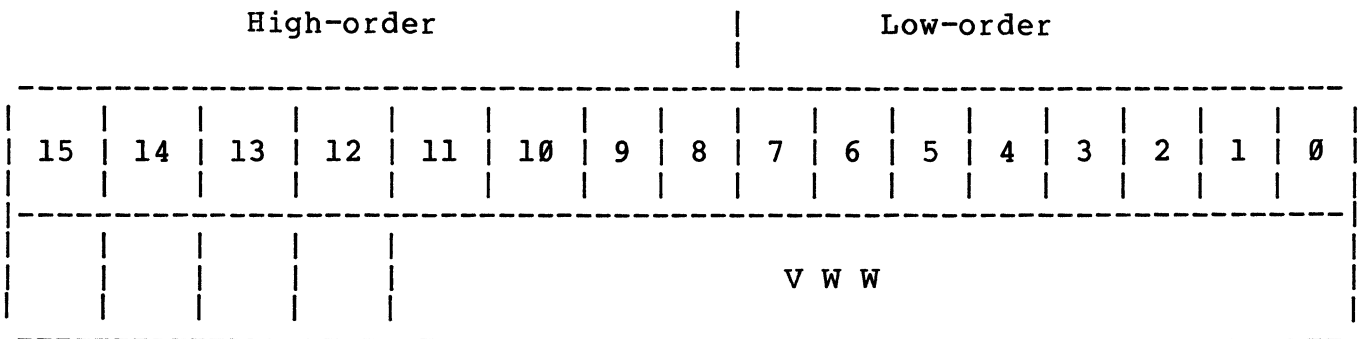
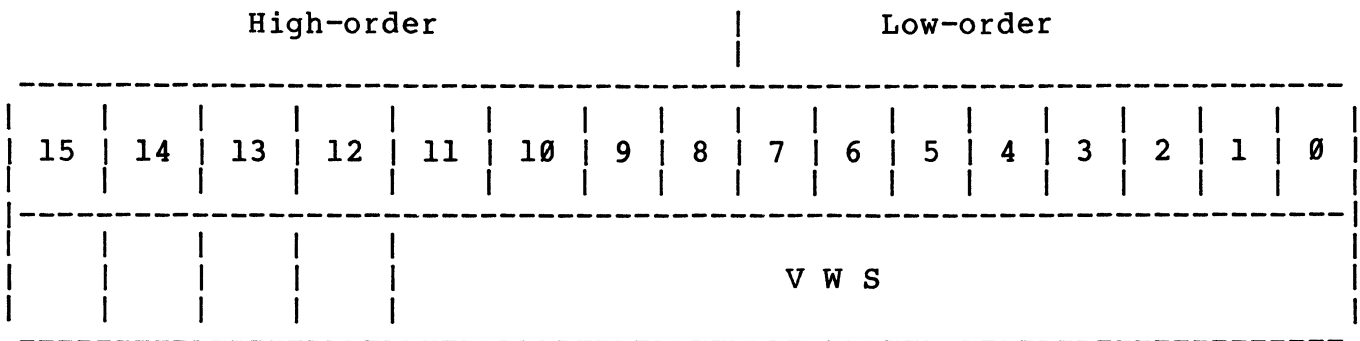
VDS	Number of rasters
00000000	1
00000001	2
11111110	255
11111111	256

\* Vertical Sync Width (VSW: r89 bit 4 - bit 0)

VSW defines the /VSYNC low pulse width in units of rasters. VSW is set depending on the CRT display device specifications. VSW should be set to a non-zero value.

VSW	Number of rasters
00001	1
00010	2
11110	30
11111	31

4.9.12 Vertical Window Display Register (VDR: r94-r97)



VWR is a read/write register that defines the vertical Window start position and width in units of rasters.

\* **Vertical Window Start (VWS: r94 - r95)**

VWS defines the period from the rising edge of /VSYNC to the vertical Window start position in units of rasters. When the vertical window start position is the VS raster, VWS is set to VS-1. Note that VWS must be greater or equal to VDS.

VC	Number of rasters
00000000000000	1
00000000000001	2
11111111111110	4095
11111111111111	4096

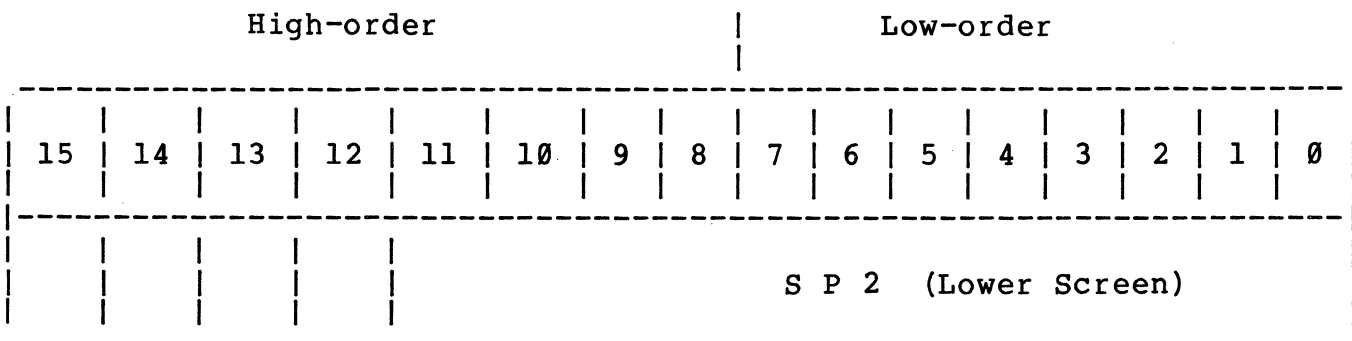
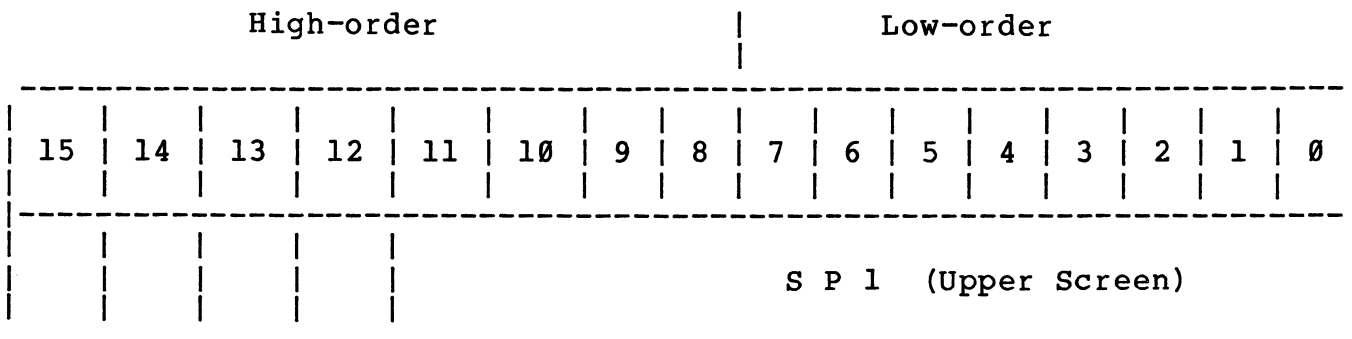
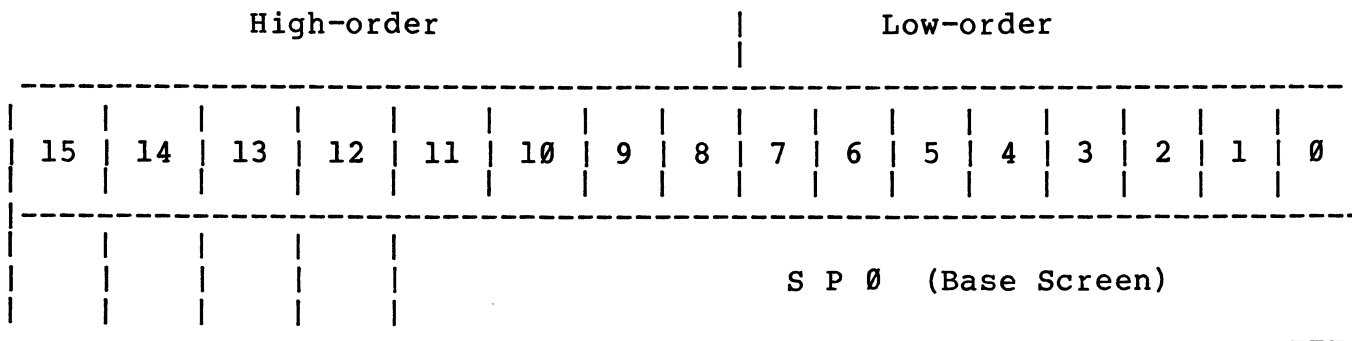
\* Vertical Window Width (VWW: r96 - r97)

VWW defines the vertical display period of the window screen in units of rasters. When the vertical window width is VW rasters, VWW is set to VW.

VC	Number of rasters
0000000000001	1
0000000000010	2
1111111111110	4094
1111111111111	4095



4.9.13 Split Screen Width Register (VDR: r8A-r8F)



SSW defines the vertical display width of the Upper (split screen 0), Base (split screen 1) and Lower (split screen 2) screens.

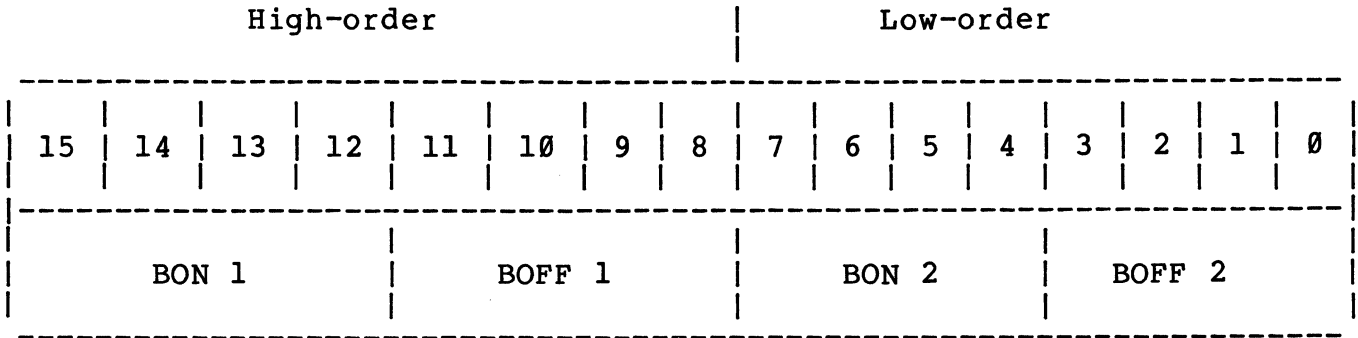
\* Split Screen Width (SP0: r8A - r8B bit 11 - bit 0)  
 (SP1: r8C - r8D bit 11 - bit 0)  
 (SP2: r8E - r8F bit 11 - bit 0)

SP0, SP1 and SP2 define the vertical display period of the Upper, Base and Lower screens respectively in units of rasters. If the vertical screen width is SW rasters, SP0/SP1/SP2 are set to SW.

SP1/SP2	Number of rasters
00000000000001	1
00000000000010	2
1111111111110	4094
1111111111111	4095

For the Base screen SP0 = 0 also can be used.

4.9.14 Blink Control Register (VDR: r90-r91)



BCR defines the blink on and blink off period for the Blink 1 and Blink 2 video attributes.

- \* **Blink On** (BON 1: r90 bit 15 - bit 12)  
(BON 2: r91 bit 7 - bit 4)

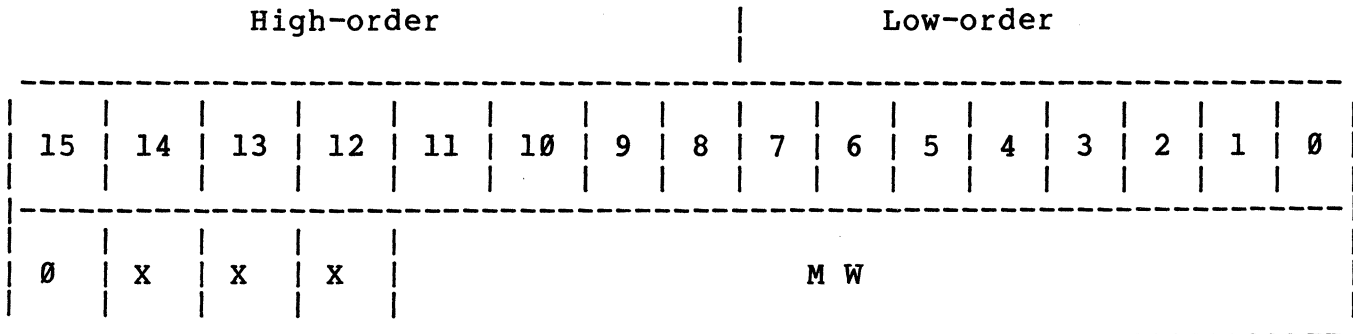
BON 1/2 defines the BLINK 1/2 attribute active high (on) period. The unit is 4 field periods. BLINK 1/2 is always low (OFF) when BON 1/2 = 0 is programmed.

- \* **Blink Off** (BOFF 1: r90 bit 11 - bit 8)  
(BOFF 2: r91 bit 3 - bit 0)

BOFF 1/2 defines the BLINK 1/2 attribute active low (OFF) period. The unit is 4 field periods. BLINK 1/2 is always high (on) when BON 1/2 = 0 are programmed.

BON 1/2 BOFF 1/2	Blink high/low Level No. of Fields
0 0 0 0	BLINK 1/2 always high
0 0 0 1	8
0 0 1 0	12
0 0 1 1	16
1 1 1 0	60
1 1 1 1	64

4.9.15 Memory Width Register (MWR0: rC2 - rC3) Upper Screen  
 (MWR1: rCA - rCB) Base Screen  
 (MWR2: rD2 - rD3) Lower Screen  
 (MWR3: rDA - rDB) Window Screen



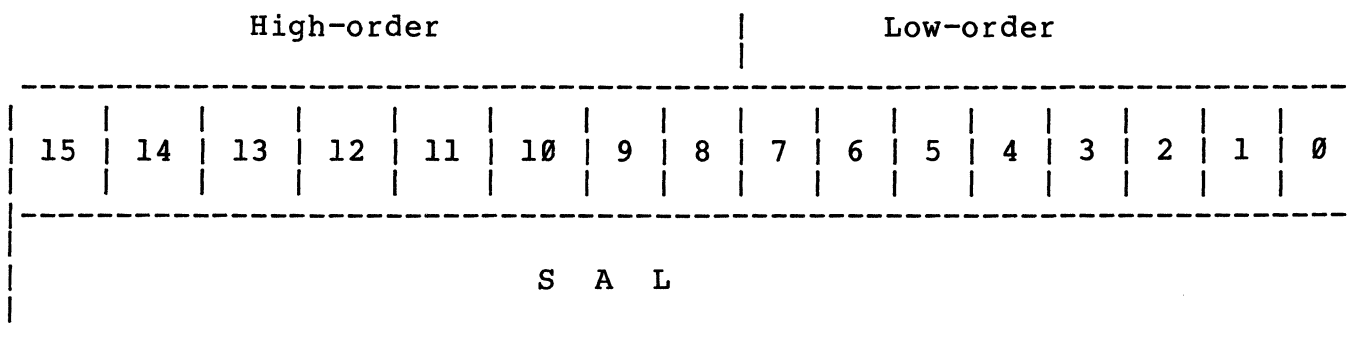
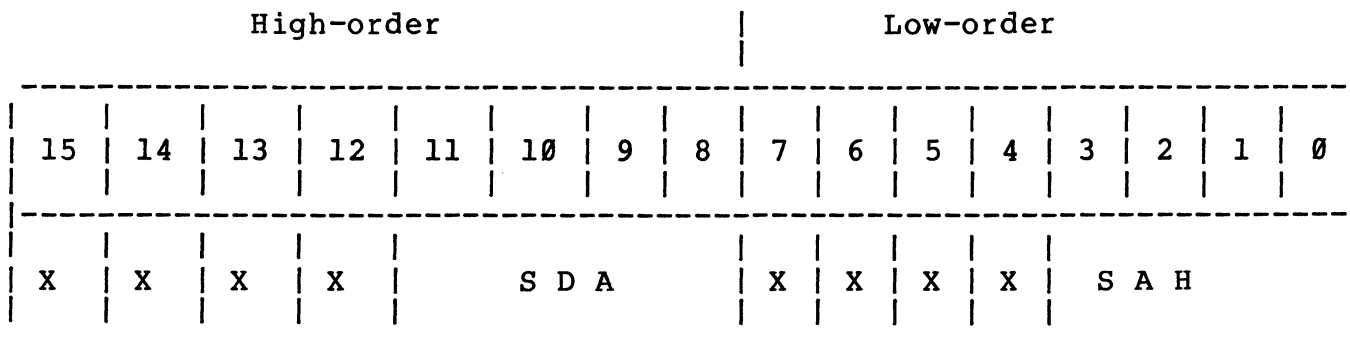
MWR defines the number of physical 16 bit frame buffer words which comprise all logical pixel X addresses for a single Y address. For example, if a screen is defined with 1024 logical pixel range in the X direction (X may vary from 0 to 1023), and 4 bits per pixel are assumed, that screens MWR value should be 256.

MWR should be greater than or equal to the Horizontal Display Width (HDW - r85). MWR must be greater than HDW to perform horizontal smooth scroll. MWR maximum value is 4095.

\* **Memory Width (MW: bit 11 - bit 0)**

MW	No. of Words
000000000000	0
000000000001	1
111111111110	4094
111111111111	4095

**4.9.16 Start Address Register** (SAR0: rC4 - rC7) Upper Screen  
 (SAR1: rCC - rCF) Base Screen  
 (SAR2: rD4 - rD7) Lower Screen  
 (SAR3: rDC - rDF) Window Screen



SAR defines the first frame buffer address for each screen. SAR0-3 apply to screens 0 - 3, the Upper, Base, Lower and Window screens respectively.

The screens have a 1M byte by 16 bit physical address space. SAR can take on any address. The memory addresses will 'wraparound' to 0 when the physical address space limit is reached independent of the split screen position.

\* **Start Address Low (SAL: bit 15 - bit 0)**

SAL contains the least significant 16 bits of the 20 bit start address.

\* **Start Address High (SAH: bit 3 - bit 0)**

SAH provides the most significant 4 bits of the 20 bit start address.

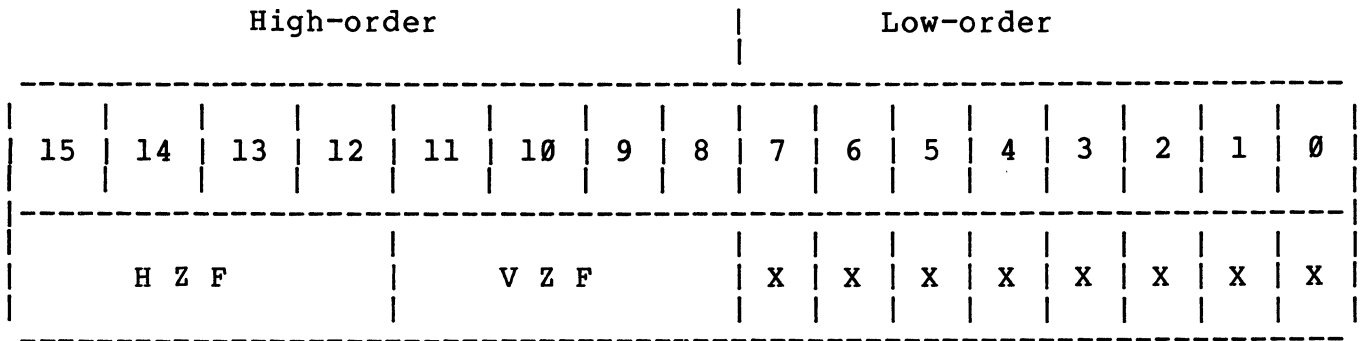
Vertical smooth scroll is done by simply increment or decrement the start address. The number of words is the same as defined in the Memory Width Register (MWR)

\* Start Dot Address (SDA: bit 11 - bit 8)

SDA is used to define a start dot horizontal offset for the horizontal smooth scroll circuit on the SYS68K/AGC-1. In the 8 bit/pixel mode SDA can vary between 0 and 15. This value corresponds with the number of pixels read out in one memory cycle. In the 4 bit/pixel mode, the number of pixel read out during one memory cycle is 32 and therefore the least significant bit to scroll pixel by pixel is provided by attribut bit 7 in the Display Control Register (DCR r06 - r07).

SDA	Number of pixels offset	
	8 bit/pixel	4 bit/pixel
0000	0	0
0001	1	2
0010	2	4
0011	3	6
0100	4	8
0101	5	10
0110	6	12
0111	7	14
1000	8	16
1001	9	18
1010	10	20
1011	11	22
1100	12	24
1101	13	26
1110	14	28
1111	15	30

#### 4.9.17 Zoom Factor Register (ZFR: rEA - rEB)



ZFR determines the horizontal and vertical multipliers (1-16) for zooming up. Zooming can only be applied to the Base screen. HZF and VZF should be set to 0 for no-zoom and 16 for 16 times zoom.

Note that zooming and scrolling horizontally together only can be done due to the following equations:

$$\frac{\text{No. pixel/raster}}{4 * (\text{HZF} + 1)} = N \quad (4 \text{ bit/pixel mode})$$

$$\frac{\text{No. pixel/raster}}{2 * (\text{HZF} + 1)} = N \quad (8 \text{ bit/pixel mode})$$

With N must be an integer value.

\* **Horizontal Zoom Factor (HZF: bit 15 - bit 12)**

HZF defines the horizontal zoom factor in units of memory cycles. The ACRTC will output a single display address HZF times.

\* **Vertical Zoom Factor (VZF: bit 11 - bit 8)**

VZF defines the vertical zoom factor. The ACRTC performs the vertical zoom by modifying its video memory address so that multiples of the same raster data are displayed.

VZF/HZF	Magnitude = HZF + 1
0 0 0 0	1
0 0 0 1	2
1 1 1 0	15
1 1 1 1	16

**4.9.18 Light Pen Address Register (LPAR: rEC - rEF)**

High-order								Low-order							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X	0	X	X	X				

High-order								Low-order							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L P A L															

LPAR is a read only register. When the ACRTC LPSTB input is asserted, the current display address is latched into the LPAR. The value in LPAR will differ from the actual display address under the light pen depending on various hardware delay times. Thus, the LPAR value should be adjusted by host software depending on system configuration.

**\* Light Pen Address High (LPAH: rED bit 3 - bit 0)**

LPAH is only valid if bit 7 = 0 and contains the most significant 4 bits of the 20 bit graphic screen display address.

**\* Light Pen Address Low (LPAL: rEE - rEF)**

LPAL contains the least significant bits of the 20 bit graphic screen display address.



#### 4.10 Drawing Control Registers

The ACRTC refers to a number of registers during graphic drawing operations.

- a) Pattern RAM
- b) Drawing Parameter Registers
  - Colour 0 Register (CL0)
  - Colour 1 Register (CL1)
  - Colour Comparison Register (CMP)
  - Edge Colour Register (EDG)
  - Mask Register (MASK)
  - Pattern RAM Control Register (PRC)
  - Area Definition Register (ADR)
  - Read/Write Pointer (RWP)
  - Drawing Pointer (DP)
  - Current Pointer (CP)

The Pattern RAM is accessed using the Read and Write Pattern (RPTN, WPTN) commands (see Appendix I). The Drawing Parameter Registers are accessed using the Read and Write Parameter Register (RPR, WPR) commands.

Table 4-13 shows the Drawing Parameter Register locations.

CS	RS	RW	Reg. No.	Register Name	Abbr.	DATA (H)								DATA (L)											
						15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
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	0	r00	FIFO Entry	FE	F E																			
1	0	0	r02	Command Control	CCR	ABT	PSE	DDM	CDM	DRC	GBM				CRE	ARE	CEE	LPE	RFE	RRE	WRE	WEE			
1	0	0	r04	Operation Mode	OMR	M/S	STR	ACP	WSS	CSK	DSK		RAM	GAI		ACM		RSM							
1	0	0	r06	Display Control	DCR	DSP	SE1	SE0	SE2	SE3	A T R														
-	-	-	r08	(undefined)	-	.....																			
1	0	0	r80	Raster Count	RCR	.....								R C											
1	0	0	r82	Horizontal Sync.	HSR	.....								H C				H S W							
1	0	0	r84	Horizontal Display	HDR	.....								H D S				H D W							
1	0	0	r86	Vertical Sync.	VSR	.....								V C											
1	0	0	r88	Vertical Display	VDR	.....								V D S				V S W							
1	0	0	r8A	Split Screen Width	SSW	.....								S P 1											
1	0	0	r8C			.....								S P 0											
1	0	0	r8E			.....								S P 2											
1	0	0	r90	Blink Control	BCR	BON1				BOFF1				BON2				BOFF2							
1	0	0	r92	Horizontal Window Display	HWR	.....								H W S				H W W							
1	0	0	r94	Vertical Window Display	VWR	.....								V W S											
1	0	0	r96			.....								V W W											
1	0	0	r98	Graphic Cursor	GCR	.....								C X E				C X S							
1	0	0	r9A			.....								C Y S											
1	0	0	r9C			.....								C Y E											
-	-	-	r9E	(undefined)	-	.....																			
-	-	-	rA0	(undefined)	-	.....																			
1	0	1	rC0	Raster Addr.0	RAR0	.....								L R A 0				F R A 0							
1	0	1	rC2	Upper Memory Width 0	MWR0	CHR	.....								M W 0				S A 0 L						
1	0	1	rC4	Start Addr.0	SAR0	.....								S D A 0				S A 0 H / S R A 0							
1	0	1	rC6	Base Screen	RAR1	.....								L R A 1				F R A 1							
1	0	1	rC8			Memory Width 1	MWR1	CHR	.....								M W 1				S A 1 L				
1	0	1	rCA			Start Addr. 1	SAR1	.....								S D A 1				S A 1 H / S R A 1					
1	0	1	rCC	Lower Screen	RAR2	.....								L R A 2				F R A 2							
1	0	1	rCE			Memory Width 2	MWR2	CHR	.....								M W 2				S A 2 L				
1	0	1	rD0			Start Addr.2	SAR2	.....								S D A 2				S A 2 H / S R A 2					
1	0	1	rD2	Window	RAR3	.....								L R A 3				F R A 3							
1	0	1	rD4			Memory Width 3	MWR3	CHR	.....								M W 3				S A 3 L				
1	0	1	rD6			Start Addr.3	SAR3	.....								S D A 3				S A 3 H / S R A 3					
1	0	1	rD8	Block Cursor 1	BCUR1	B C W 1				B C S R 1				B C A 1				B C E R 1							
1	0	1	rE0			B C W 2				B C S R 2				B C A 2				B C E R 2							
1	0	1	rE2	Block Cursor 2	BCUR2	B C W 2				B C S R 2				B C A 2				B C E R 2							
1	0	1	rE4			B C W 2				B C S R 2				B C A 2				B C E R 2							
1	0	1	rE6	Cursor Definition	CDR	C M				CON1				COFF1				CON2				COFF2			
1	0	1	rE8			H Z F				V Z F				.....				.....							
1	0	1	rEA	Zoom Factor	ZFR	.....																			
1	0	1	rEC	Light Pen Address	LPAR	.....								CHR				L P A H							
1	0	1	rEE			.....								L P A L											
-	-	-	rF0	(undefined)	-	.....																			
-	-	-	rFE	(undefined)	-	.....																			

Note 1 ..... "High" level  
0 ..... "Low" level

Table 4-13: Drawing Parameter Registers

#### 4.10.1 Pattern RAM

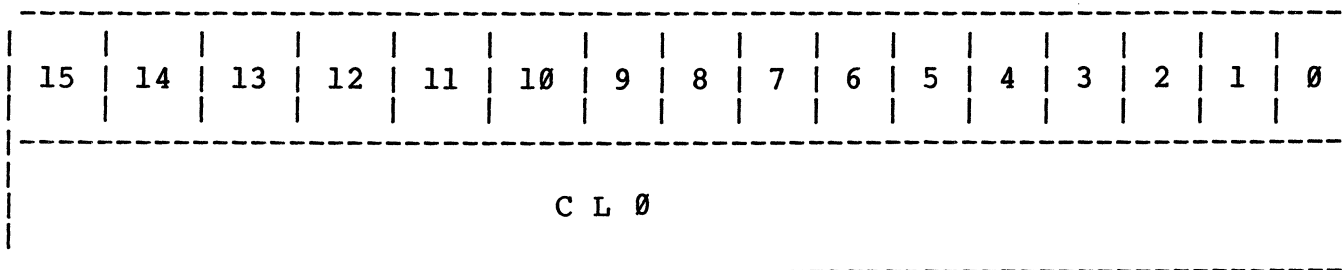
The ACRTC contains a 32 byte pattern RAM. The Pattern RAM is used for pre-defining data for the graphic drawing operations.

A 16 by 16 bit pattern (or 16 sets of 16 by 1 bit) can be stored in the Pattern RAM as a binary representation of screen data. In this case, a two entry colour 'palette' corresponding to 0 and 1 data values is defined using the Colour 0 (CL0) and Colour 1 (CL1) registers.

To store colour patterns in the Pattern RAM it is divided into four equal segments of either 4 by 4 bit patterns or 4 sets of 4 by 1 bit patterns. In this case, during drawing the colour coded contents of the Pattern RAM are directly written to the video memory. The particular segment used is defined by the Pattern RAM Control register (PRC).

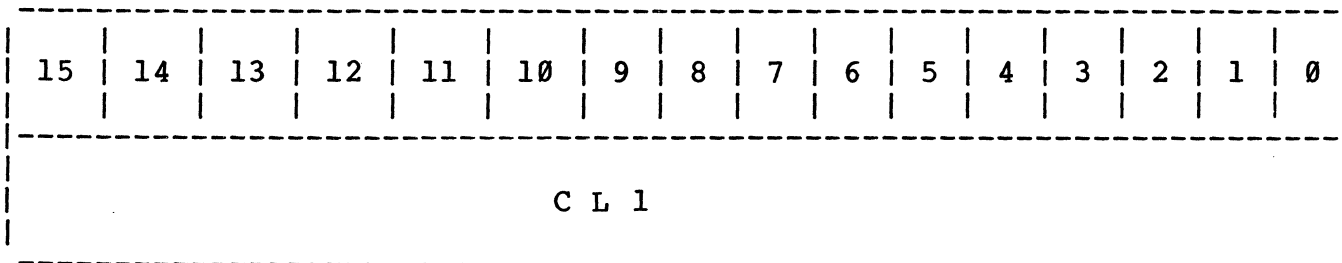
When multiple drawing commands use a common pattern, pattern continuity can be achieved by adjusting the pattern scanning pointer.

#### 4.10.2 Colour 0 Register (CL0: Pr00)



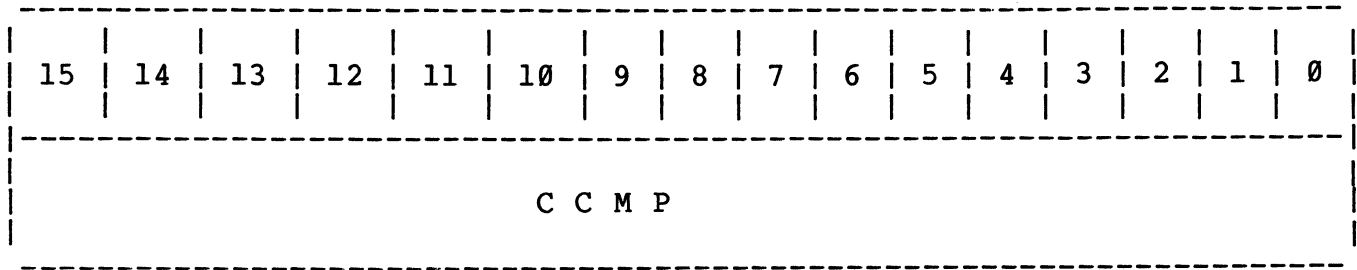
When logical drawing data = 0, the contents of CL0 are stored in the video memory. The value of CL0 corresponds with the bits/pixel mode used. For example in 4 bits/pixel mode CL0 contains the colour value for 4 pixel. If all pixel should be painted in the same colour, it is necessary to store the respective colour 4 times in the Colour 0 register.

#### 4.10.3 Colour 1 Register (CL1: Pr01)



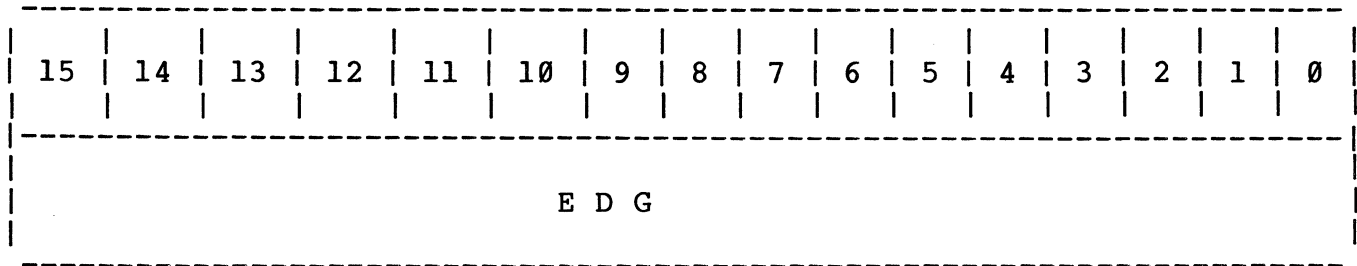
When logical drawing data = 1, the contents of CL1 are stored in the video memory. The value of CL1 corresponds with the bits/pixel mode used. For example in 4 bits/pixel mode CL1 contains the colour value for 4 pixel. If all pixel should be painted in the same colour, it is necessary to store the respective colour 4 times in the Colour 1 register.

#### 4.10.4 Colour Comparison Register (CMP: Pr02)



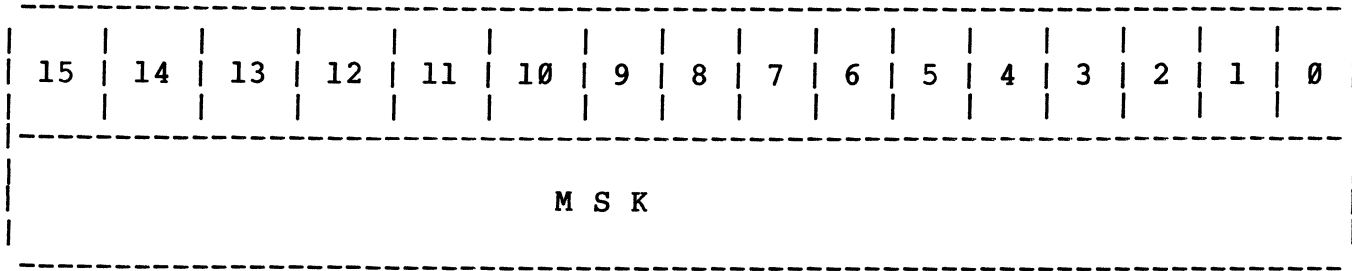
CMP defines a comparison colour for use with conditional drawing operations. Conditional drawing applies various logical comparisons between the drawing data and CCMP to determine if drawing should occur (refer to Appendix J-3). The value of CMP corresponds with the bits/pixel mode used and has to be programmed as the Colour registers respectively.

#### 4.10.5 Edge Colour Register (EDG: Pr03)



EDG defines the boundary edge colour for use with the PAINT command. In one mode, the edge is defined as the colour contained in the EDG. In another mode, the edge is defined as any colour except the colour contained in the EDG register. The values of EDG corresponds with the bits/pixel mode used and has to be programmed as the Colour registers respectively.

4.10.6 Mask Register (MASK: Pr04)



When performing data transfer and drawing of the video memory, MSK is used to mask bits upon which drawing and other logical operations should not be performed. If MSK bit is 0, the corresponding video memory bit is excluded from any logical operation. The values of EDG corresponds with the bits/pixel mode used and has to be programmed as the Colour registers respectively.

4.10.7 Pattern RAM Control Register (PRC: Pr05 - Pr07)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P P Y				P Z C Y				P P X				P P Y			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P S Y				0	0	0	0	P S X				0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P E Y				P Z Y				P E X				P Z X			

PRC specifies the size of the patterns used for drawing and the start point within the Pattern RAM for the pattern scan. The pattern size can be independently specified in X and Y dimensions (maximum 16 by 16 bits).

- \* Pattern Start X (PSX: Pr06 bit 7 - bit 4)  
 Pattern Start Y (PSY: Pr06 bit 15 - bit 12)

PSX and PSY specify the pattern scan starting point horizontal and vertical addresses respectively. These should be set between 0-15 for Colour register direct drawing and between 0-3 for Pattern RAM direct drawing.

- \* Pattern End X (PEX: Pr07 bit 7 - bit 4)  
Pattern End Y (PEY: Pr07 bit 15 - bit 8)

PEX and PEY specify the pattern scan ending point horizontal and vertical addresses respectively. These should be set between 0-15 for Colour register indirect drawing and between 0-3 for Pattern RAM direct drawing.

- \* Pattern Zoom X (PZX: Pr07 bit 3 - bit 0)  
Pattern Zoom Y (PZY: Pr07 bit 11 - bit 8)

PZX and PZY specify the magnification coefficient applied to the contents of the Pattern RAM. PZX, PZY = 0 specifies no magnification, while PZX, PZY = \$F specifies by 16 magnification.

- \* Pattern Zoom Count X (PZCX: Pr05 bit 3 - bit 0)  
Pattern Zoom Count Y (PZCY: Pr05 bit 11 - bit 8)

PZCX and PZCY specify the initial magnification counter values in the horizontal and vertical dimensions respectively. Normally, PZCX and PZCY should be set to 0.

- \* Pattern Pointer X (PPX: Pr05 bit 7 - bit 4)  
Pattern Pointer Y (PPY: Pr05 bit 15 - bit 12)

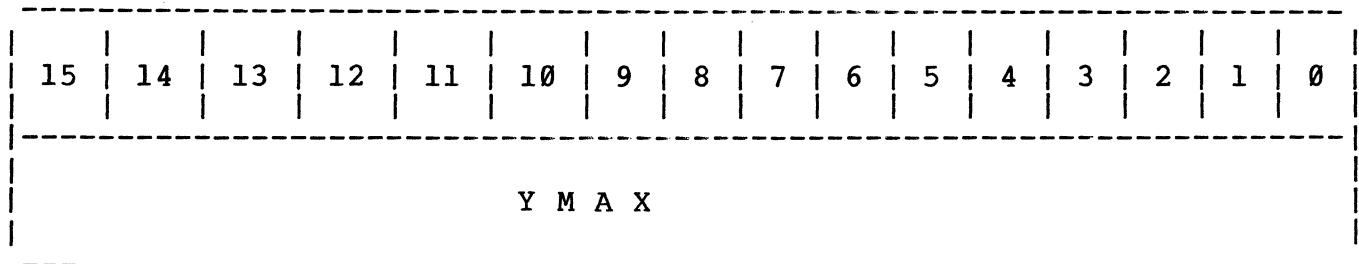
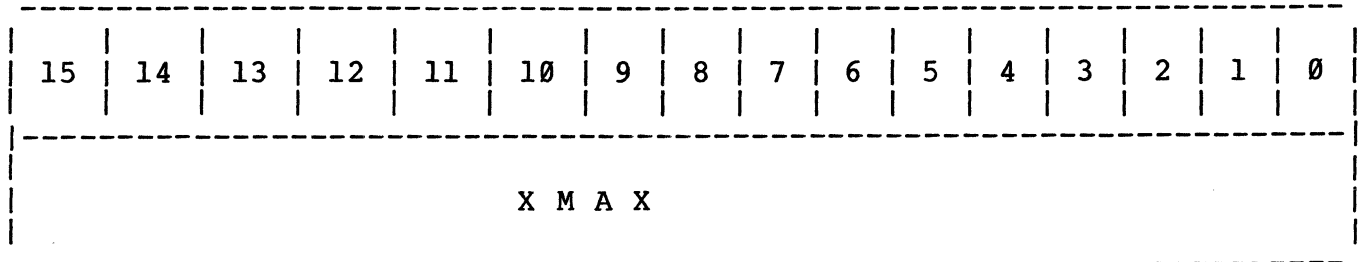
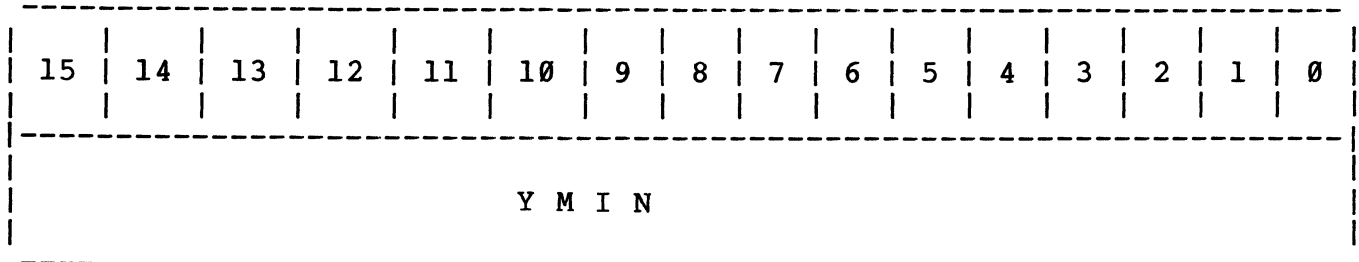
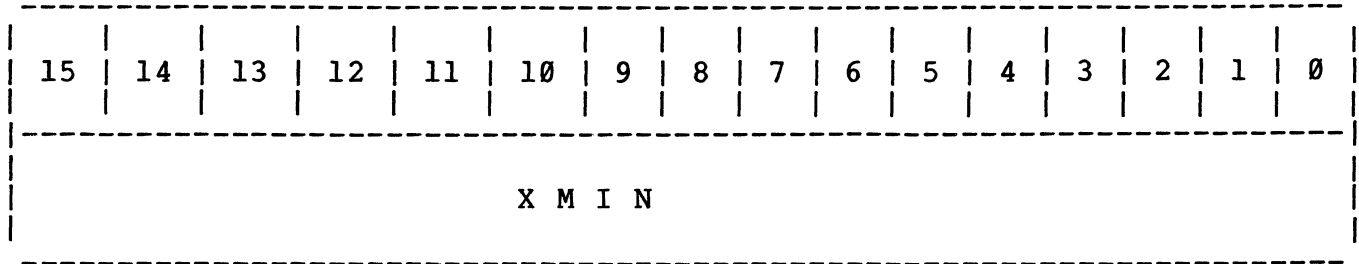
The current reference point within the Pattern RAM is specified by PPX and PPY. When using PSX, PSY to define a pattern scan starting point, the following relationships must be maintained:

$$PSX = PPX = PEX$$

and  $PSY = PPY = PEY$



4.10.8 Area Definition Register (ADR: Pr08 - Pr0B)



ADR is used to define a drawing area using logical X-Y addresses relative to the origin defined with the ORG command. The ACRTC will check logical drawing addresses against ADR depending on the AREA mode specified in the graphic drawing command.

#### 4.10.9 Read Write Pointer (RWP: Pr0C - Pr0D)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D N		X	X	X	X	X	X	R W P H							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W P L												X	X	X	X

RWP specifies a 20 bit physical video memory address for use with the data transfer commands.

#### \* Display Number (DN: Pr0C bit 15 - bit 14)

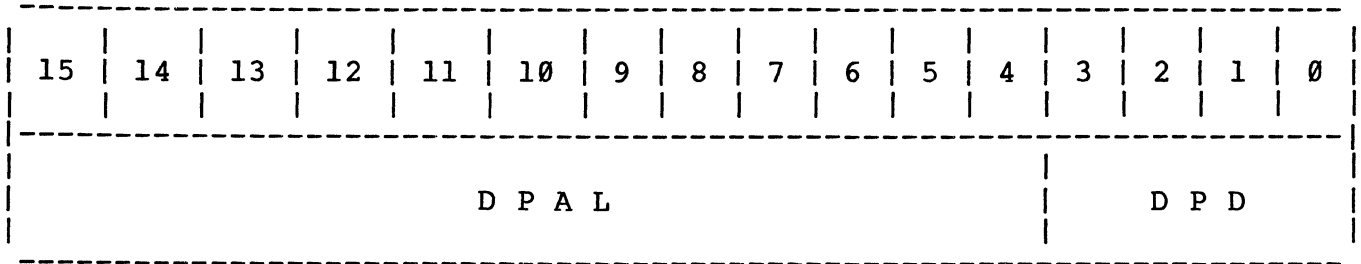
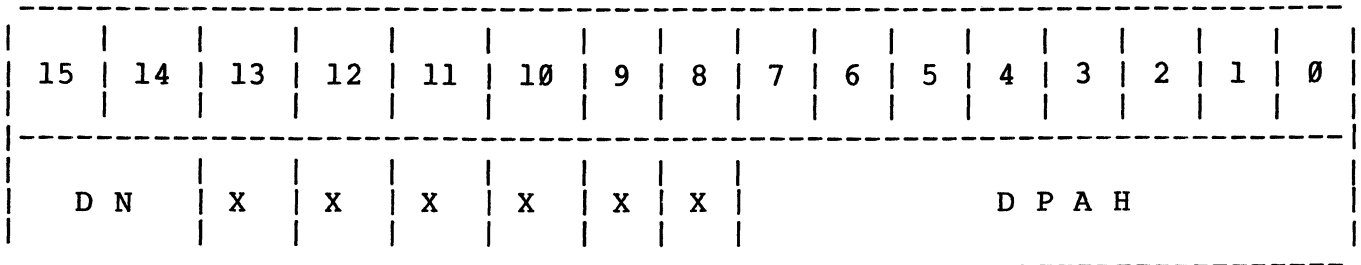
DN specifies the logical screen containing the data to be transferred.

DN	Functions
00	Upper Screen
01	Base Screen
10	Lower Screen
11	Window Screen

#### \* Read Write Pointer High (RWPH: Pr0C bit 7 - bit 0) Read Write Pointer Low (RWPL: Pr0D bit 15 - bit 4)

RWPH and RWPL define the initial 20 bit video memory address used with the data transfer commands.

**4.10.10 Drawing Pointer (RWP: Pr0C - Pr0D)**



The ACRTC uses DP for containing the physical drawing address calculated during drawing commands. When executing a drawing command, DP is updated as the Current Pointer (CP), specifying the current logical X-Y drawing address, is moved.

**\* Display Number (DN: Pr10 bit 15 - bit 14)**

DN specifies the screen for graphic drawing. Interpretation is the same as DN in the Read Write Pointer (RWP) register.

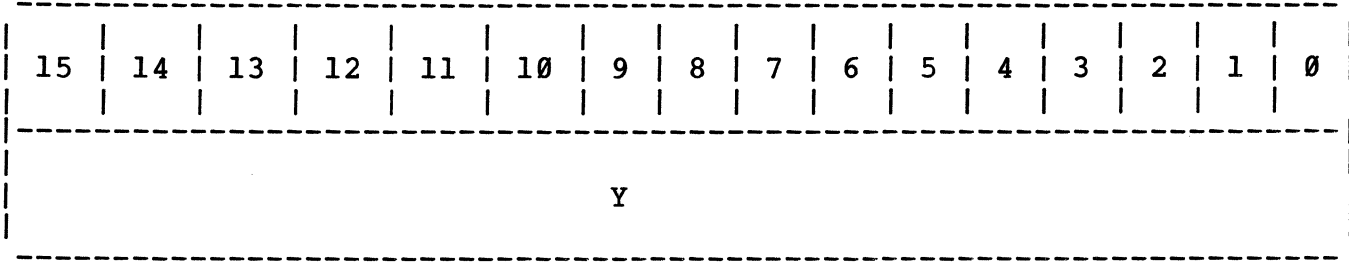
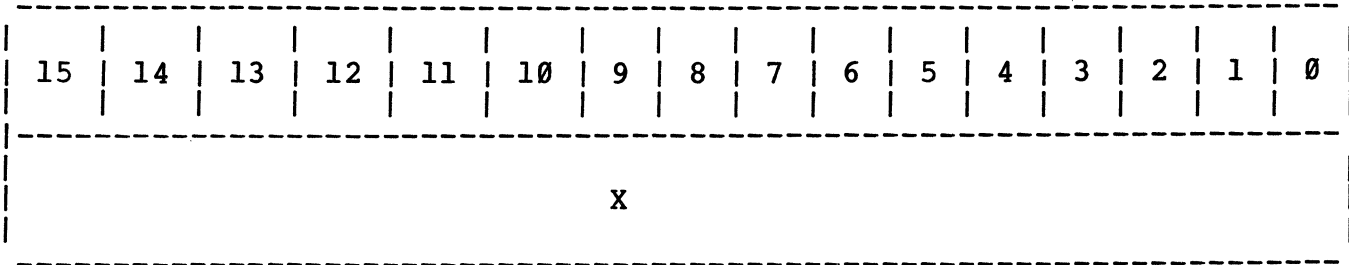
**\* Drawing Pointer Address High (DPAH: Pr10 bit 7 - bit 0)  
Drawing Pointer Address Low (DPAL: Pr11 bit 15 - bit 4)**

DPAH and DPAL specify the 20 bit physical drawing pointer address.

**\* Drawing Pointer Dot (DPD: Pr11 bit 3 - bit 0)**

DPD specifies the physical pixel address to locate a logical pixel within the 16 bit word addressed by DPAH, DPAL. Interpretation depends on the specified relationship between logical pixels and physical frame buffer bits as determined by the Graphics Bit Mode (GBM). In the 4 bits/pixel mode, DPD specifies 1 of 4 logical pixels using the most significant 2 bits of DPD. The 2 least significant bits are not used. In the 8 bits/pixel mode DPD specifies 1 of 2 logical pixels using the most significant bit of DPD.

4.10.11 Current Pointer (CP: Pr12 - Pr13)



CP specifies the logical X-Y coordinates of the current drawing address. As drawing proceeds, the ACRTC calculates the physical frame buffer address for each X-Y addressed logical pixel. The physical address corresponding to CP is stored in the Drawing Pointer (DP) register. Two's-complement format is used to indicate positive and negative values.

## 5. Command Overview

The ACRTC interprets and processes commands issued by the host. These commands are classified into three groups.

1. Register Access Commands
2. Data Transfer Commands
3. Graphic Drawing Commands

ACRTC commands consist of a 16 bit op-code, optionally followed by 1 or more 16 bit parameters.

Commands and parameters can be issued to the ACRTC in the following ways:

**\* Software Polling (WFR, WFE interrupts disabled)**

- a) The host program checks the Status Register for Write FIFO Ready (WFR) flag = 1, and then writes one word of command or parameters.
- b) The host program checks the Status Register for Write FIFO Empty (WFE) flag = 1, and then writes one to eight words of commands or parameters.

**\* Interrupt Driven (WFR, WFE interrupts enabled)**

- a) The host WFR interrupt service routine writes one word of command or parameters.
- b) The host WFE interrupt service routine writes one to eight words of commands or parameters.

In the specific case of Register Access Commands and an initially empty write FIFO, host writes need not be synchronized to the write FIFO status. The ACRTC can fetch and execute these commands faster than the host can issue them.

## 5.1 Register Access Commands

Registers associated with the Drawing processor (the Pattern RAM and the Drawing Parameter Registers) are accessed through the read and write FIFO's using Register Access Commands.

Command	Function
ORG	Initialize the relation between the origin point in the X-Y coordinates and the physical address.
WPR	Write into the parameter register
RPR	Read parameter register
WPTN	Write into pattern RAM
RPTN	Read the pattern RAM

For command codes and optionally parameters please refer to Appendix J-1.

## 5.2 Data Transfer Commands

Data Transfer Commands are used to move blocks of data between the host system memory and the ACRTC video memory or within the video memory itself. Before issuing these commands, a physical 20 bit frame buffer address must be specified in the Read Write Pointer (RWP) register.

The DMA Data Transfer Commands (DRD, DWT and DMOD) are used to send large amounts of data between system and video RAM. The programmer specifies the command and the X and Y logical pixel dimensions of the video memory data block. The ACRTC will automatically control an external DMAC (installed on the optionally available SYS68K/AGC-1X board) to request data transfers via the read and write FIFO's.

Note that DMA data transfers can be performed without an external DMAC, i.e. under host program control. In this case, the data DMA handshaking (DREQ, DACK and DONE) signals are disabled by resetting the DDM bit in the CCR to 0. After issuing a DMA transfer command, the host reads or writes the appropriate data to the ACRTC FIFO's under program control. The programmer must ensure that the amount of data transferred equals the amount specified as parameters to the command. Also note that the ACRTC will go into an indefinite wait state after the last transfer of a DRD command. Then, the command should be aborted (by setting the ABT bit in the CCR to 1) and the next command issued.

Command	Function
DRD	DMA read of the video memory
DWT	DMA write into the video memory
DMOD	DMA modify of the video memory data (bit maskable)
RD	One word read from the video memory
WT	One word write into the video memory
MOD	One word modify of the video memory (bit maskable)
CLR	Clear of video memory area
SCLR	Clear of video memory area (bit maskable)
CPY	Copy of video memory area into another area
SCPY	Copy of video memory area into another area (bit maskable)

### 5.3 Modify Mode

The DMOD, MOD, SCLR and SCPY commands allow 4 types of bit level logical operations to be applied to video memory data. The modify mode is encoded in the lower two bits (MM) of these op-codes. The bit positions within each video memory word to be modified are selectable using the MASK register (MSK). Bits masked with 1 are modifiable, those masked with 0 are not.

MM	Modify Mode
00	Replace video memory data with command parameter data
01	OR video memory data with command parameter data and rewrite to the video memory.
10	AND video memory data with command parameter data and rewrite to the video memory.
11	EOR video memory with command parameter data and rewrite to the video memory.

Refer to Appendix J-2 for examples showing the use of the REPLACE, OR, AND and EOR modify modes.



## 5.4 Graphic Drawing Commands

The ACRTC has 23 separate graphic drawing commands. Graphic drawing is performed by modifying the contents of the video memory based upon microcoded drawing algorithms in the ACRTC Drawing processor.

Most coordinate parameters for graphic drawing commands are specified using logical X-Y addressing. The complex task of translating a logical pixel address to a linear video memory word address, and further selecting the appropriate sub-field of the word (for example, a given logical pixel in 4 bits/pixel mode might reside in bits 8-11 of a video memory word) is performed at high speed by ACRTC hardware.

Most instructions allow specification of X-Y coordinates with either absolute or relative X-Y coordinates. In both cases, two complement numbers are used to represent positive and negative values.

### a) Absolute Coordinate Specification

The screen address (X,Y) is specified in units of logical pixels relative to an origin point defined with the ORG command.

### b) Relative Coordinate Specification

The screen address (dX,dY) is specified in units of logical pixels relative to the current drawing pointer (CP) position.

A graphic drawing command consists of an 8 bit command code, an Area Mode specifier (3 bits), a Colour Mode specifier (2 bits) and an Operation Mode specifier (3 bits).

The Area Mode allows versatile clipping and hitting detection. A drawing area can be defined, and should drawing operations attempt to enter or leave that area, a number of programmable actions can be taken by the ACRTC.

The Colour Mode determines whether the Pattern RAM is used indirectly to select the Colour Registers or is directly used as the colour information.

The Operation Mode defines one of eight logical operations to be performed between the video memory read data and the colour data in the Pattern RAM to determine the drawing data to be rewritten to the video memory.

Table 5-1 shows the Graphic Drawing Commands.

Please refer to Appendix J-1 for detailed information.

Table 5-1: Graphic Drawing Commands

Command	Function
AMOVE	Movement of current points
RMOVE	
ALINE	Drawing of straight lines
RLINE	
ARCT	Drawing of rectangles
RRCT	
APLL	Drawing of polylines
RPLL	
APLG	Drawing of polygones
RPLG	
CRCL	Drawing of circles
ELPS	Drawing of ellipses
AARC	Drawing of arcs
RARC	
AEARC	Drawing of ellipse arcs
REARC	
AFRCT	Painting of rectagle areas (Tiling)
RFRCT	
PAINT	Painting of arbitrary areas (Tiling)
DOT	Making of dots
PTN	Drawing of basic patterns (rotation angle: 45 )
AGCPY	Graphic copy between video memories (rotation angle: 90 /mirror turnover)
RGCPY	

## 5.5 Operation Mode

The OPM bits of the Graphic Drawing Command specify the logical drawing condition.

OPM	Operation Mode
0 0 0	REPLACE: Replaces the video memory data with the colour data.
0 0 1	OR: ORs the video memory data with the colour data. The result is rewritten to the video memory.
0 1 0	AND: ANDs the video memory data with the colour data. The result is rewritten to the video memory.
0 1 1	EOR: EORs the video memory data with the colour data. The result is rewritten to the video memory.
1 0 0	CONDITIONAL REPLACE (P = CMP): When the video memory data at the drawing position (P) is equal to the comparison colour (CMP), the video memory is replaced with the colour data.
1 0 1	CONDITIONAL REPLACE (P <> CMP): When the video memory data at the drawing position (P) is not equal to the comparison colour (CMP), the video memory data is replaced with the colour data.
1 1 0	CONDITIONAL REPLACE (P < CL): When the video memory data at the drawing position (P) is less than the colour register data (CL), the video memory data is replaced with the colour data.
1 1 1	CONDITIONAL REPLACE (P > CL): When the video memory data at the drawing position (P) is greater than the colour register data (CL), the video memory data is replaced with the colour data.

Refer to Appendix J-3 to show examples of each of the eight operation modes. In these examples, 4 bits/logical pixel is assumed.

## 5.6 Colour Mode (COL)

The COL bits specify the source of the drawing colour data as directly or indirectly (using the Colour Registers) determined by the contents of the Pattern RAM.

COL	Colour Mode
0 0	When Pattern RAM data = 0, Colour Register 0 is used. When Pattern RAM data = 1, Colour Register 1 is used.
0 1	When Pattern RAM data = 0, drawing is suppressed. When Pattern RAM data = 1, Colour Register 0 is used.
1 0	When Pattern RAM data = 0, drawing is suppressed. When Pattern RAM data = 1, Colour Register 1 is used.
1 1	Pattern Ram contents are directly used as colour data.

The Colour Mode chooses the source for colour information based on the contents (0 or 1) of a particular bit in the 16 bit by 16 bit Pattern RAM (see Appendix J-4). A sub-pattern is specified by programming the Pattern RAM Control Register (PRC) with the start (PSX, PSY) and end (PEX, PEY) points which define the diagonal of the sub-pattern. Furthermore, a specific starting point for Pattern RAM scanning is specified by PPX and PPY.

Normally, the colour registers (CL) should be loaded with one colour data based on the number of bits per pixel. For example, if 4 bits/pixel are used, the 4 bit colour pattern (e.g. 0001) should be replicated four times in the colour register, i.e.

Colour Register = 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1

By doing this, the colour changes due to changing dot address are avoided.

## 5.7 Area Mode (AREA)

Prior to drawing, a drawing area may be defined (Area Definition Register). Then, during Graphics Drawing Operation the ACRTC will check if the drawing point is attempting to enter or exit the defined drawing area. Based on eight Area Modes, the ACRTC will take appropriate action for clipping and hitting.

AREA	Drawing Area Mode
0 0 0	Drawing is executed without Area checking.
0 0 1	When attempting to exit the Area, drawing is stopped and the ARD (Area Detect) and CED (Command End) flags are set
0 1 0	Drawing is suppressed outside the Area - drawing operation continues and the ARD flag is not set.
0 1 1	Drawing operation is suppressed outside the Area - drawing operation continues and the ARD flag is set.
1 0 0	Same as AREA = 0 0 0.
1 0 1	When attempting to enter the Area, drawing operation is stopped and the ARD and CED flags are set.
1 1 0	Drawing is suppressed inside the Area - drawing operation continues and the ARD flag is not set.
1 1 1	Drawing is suppressed inside the Area - drawing operation continues and the ARD flag is set.

Refer to Appendix J-5 for an example of the execution of a CRCL command using various Area Modes. It is assumed, that the Area Definition Register has been loaded to define the Area bounded by Xmin, Ymin and Xmax, Ymax.



## 6. Miscellaneous

### 6.1 Miscellaneous Jumper Settings

The RAS/CAS and WRITE timing for the DRAMs of the video memory is specified through the jumper settings of B13, B14, B15, B16 and B17. B5 controls the synchronization between ACRTC and host accesses to the video RAM.

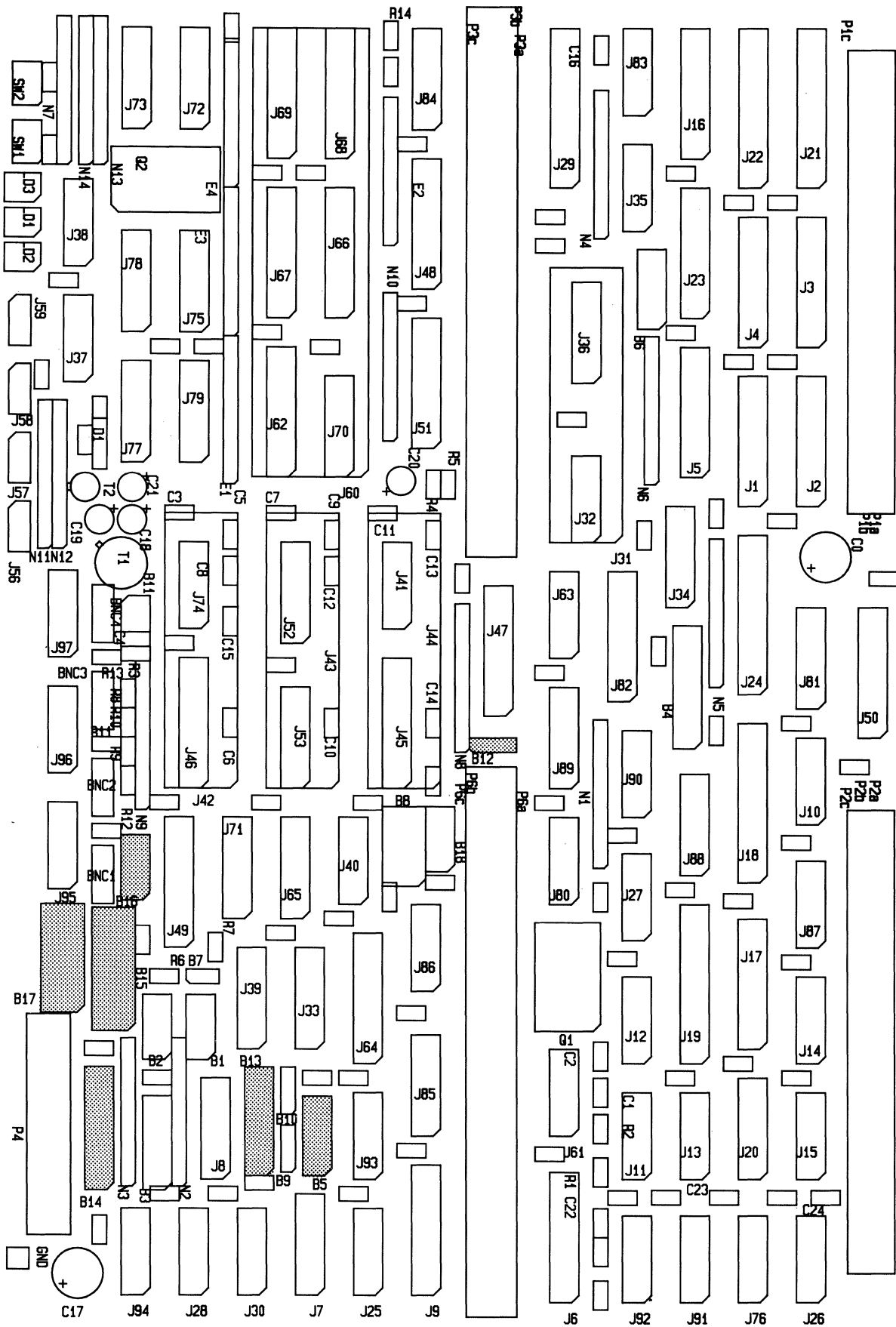
Note: These jumpers have a default setting and are not user definable.

Jumper B12 has a default setting for enabling horizontal smooth scroll and zooming together (Scroll Mode 1). When using large zoom factors, it could occur that the display jumps to the right side of the display monitor. This phenomenon can only be suppressed when horizontal smooth scroll and zooming are not used together. This mode is selected through jumper B12 (Scroll Mode 2). In the scroll mode 2 scrolling is only possible with no magnification.

Fig. 6-1 shows the jumper locations of B5 and B12 - B17 on the SYS68K/AGC-1.

Table 6-1 lists the default jumper settings.

Figure 6-1: Jumper Location Diagram D





Jumper B12			Jumper B12		
Scroll Mode 1			Scroll Mode 2		
1---	2	3	1	2---	3
Default Set-up					

Jumper B5	Jumper B13	Jumper B14		
10---	1	14	16	1
9	2	13	15	2
8	3	12	14----	3
7	4	11	13	4
6	5	5---	12	5
	6	9	11	6
	7	8	10	7
			9----	8

Jumper B15	Jumper B17	Jumper B16		
1	2	3	8----	1
4	5	6	7	2
7	8	9	6	3
10	11---	12	5	4
13---	14	15		
16	17	18		
19	20	21		
22	23	24		

Table 6-1: Jumper Settings of B5, B12, B13, B14, B15, B16, B17

## 6.2 Light Pen Interface

The Light pen interface is also realized on the SYS68K/AGC-1. It can be connected via the 15 pin DSUB connector (P4) on the front panel. The light pen must generate a positive or negative TTL-compatible pulse. Jumper B9 specifies the polarity of that pulse.

Jumper B9	Jumper B9
positive Pulse	negative Pulse
1---2 3	1 2---3
Default Set-up	

A light pen strobe pulse will occur when the CRT electron beam passes under the light pen during display refresh. When these pulse occur, the contents of the ACRTC display refresh address counter which then will be latched into the Light Pen Address register along with the logical screen (character or graphics) designator.

The various system and ACRTC delays will cause the latched address to differ slightly from the actual light pen position. The light pen address can be corrected using software, based on the system specific delays. Or, if the application does not require the highest light pen resolution, software can 'bound' the light pen address by specifying a range of values associated with a given area of the screen.

Table 6-2 shows the pin assignments of connector P4.

Figure 6-2 lists the jumper location of B9.

Note: Only Pins 7 and 8 have to be connected. All other pins must not be connected. These pins are reserved for future enhancements and for usage of multiple AGC-1.

Table 6-2: Pin Assignment of Connector P4

DISP1	1	9	GND
/HSYNC	2	10	64M
CLK31	3	11	/64M
2CLK	4	12	NC
/EXSYNC	5	13	NC
	6	14	NC
+5V	7	15	NC
Light Pen	8		

### 6.3 External Synchronisation

The SYS68K/AGC-1 allows the synchronization of multiple AGC's (up to 3). The ACRTC may be programmed as a single master or as one of a number of slave devices.

To synchronize multiple AGC's, connect them via P4 (Pin 1 - 5), select through Jumper B10 the Board as a master or slave device, set jumper B11 for internal or external pixel clock and program the ACRTC respective (in a system with 3 AGC's there exists one master and two slaves).

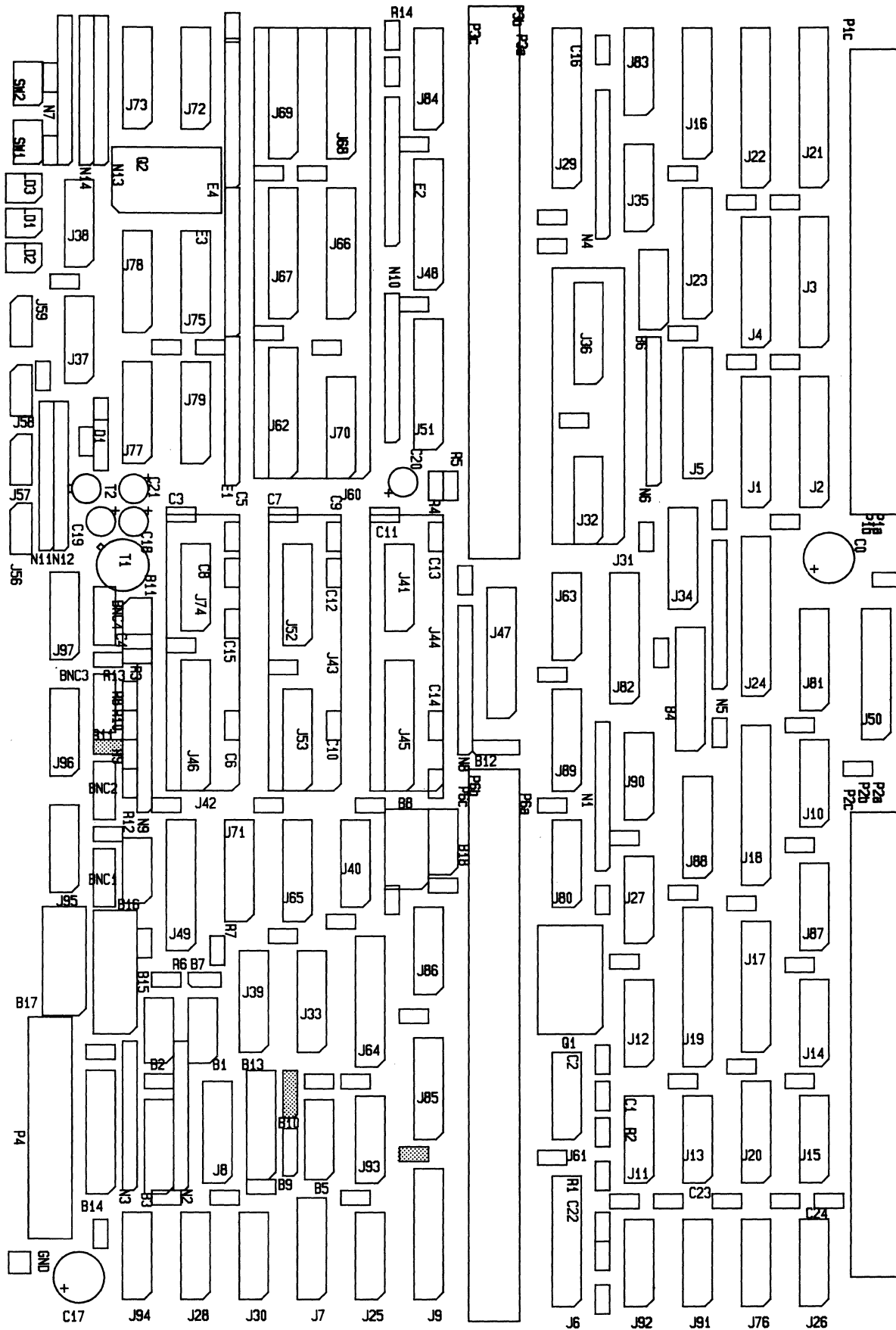
Jumper B10	Jumper B10
Master	Slave
1---2 3	1 2---3
Default Set-up	

Jumper B11	Jumper B11
Master	Slave
internal Clock	external Clock
3---4	3 4
2---1	2 1
Default Set-up	

Fig. 6-2 shows the jumper locations of B10 and B11 on the board.

Table 6-2 lists the pin assignments of connector P4.

Figure 6-2: Jumper Location Diagram of B10 (Light Pen) and B11



#### 6.4 Display Monitor Interface

The display monitor is connected with the SYS68K/AGC-1 over the BNC-connectors on the front panel. A composite Sync-signal is mixed on the RGB-outputs and easing the interfacing. The outputs are capable of driving 75 Ohm lines compatible to RS 434. Monitors with separate SYNC-input should be connected with the composite SYNC-output of the SYS68K/AGC-1. The polarity of the SYNC-signal is jumper selectable through jumper B7.

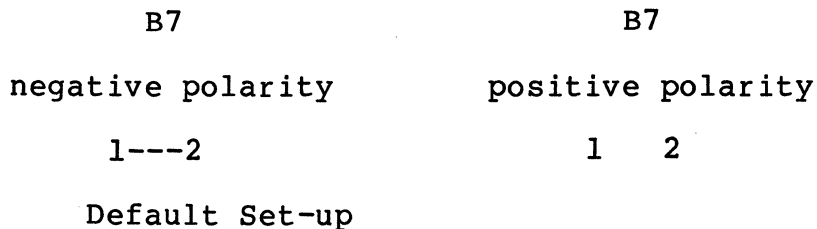
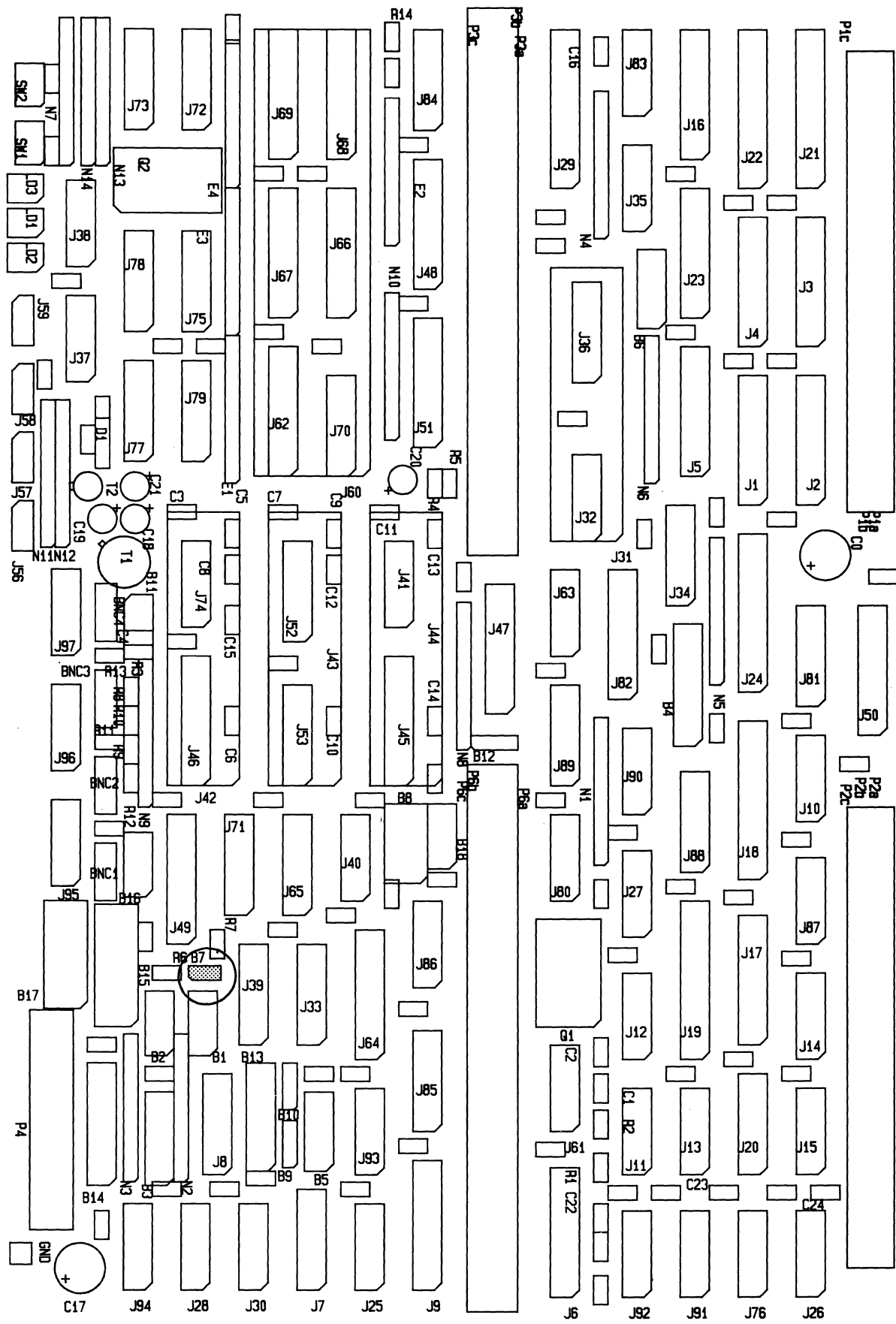


Fig. 6-3 shows the jumper location of B7 on the board.

Figure 6-3: Jumper Location Diagram of B7 (SYNCSEL)







## 7. Calculating the Screen Parameters

This chapter gives a brief explanation of calculating the screen parameters for a given display monitor and initializing the SYS68K/AGC-1 (see also Chapter 8). Before starting the configuration of the timing registers, it is necessary to completely specify the requirements of the display monitor hardware and system design.

Two fundamental points are :

- (1) All horizontal values used by the ACRTC are in units of memory cycles.
- (2) All vertical values are in units of scan lines (rasters)

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

The timing control RAM (see Chapter 4.9) holds the values that time and configure the display screen.

Fig. 7-1 shows how the display screen is specified in terms of the register values.

For clarification, there follows a worked example. Only the Base Screen will be implemented.

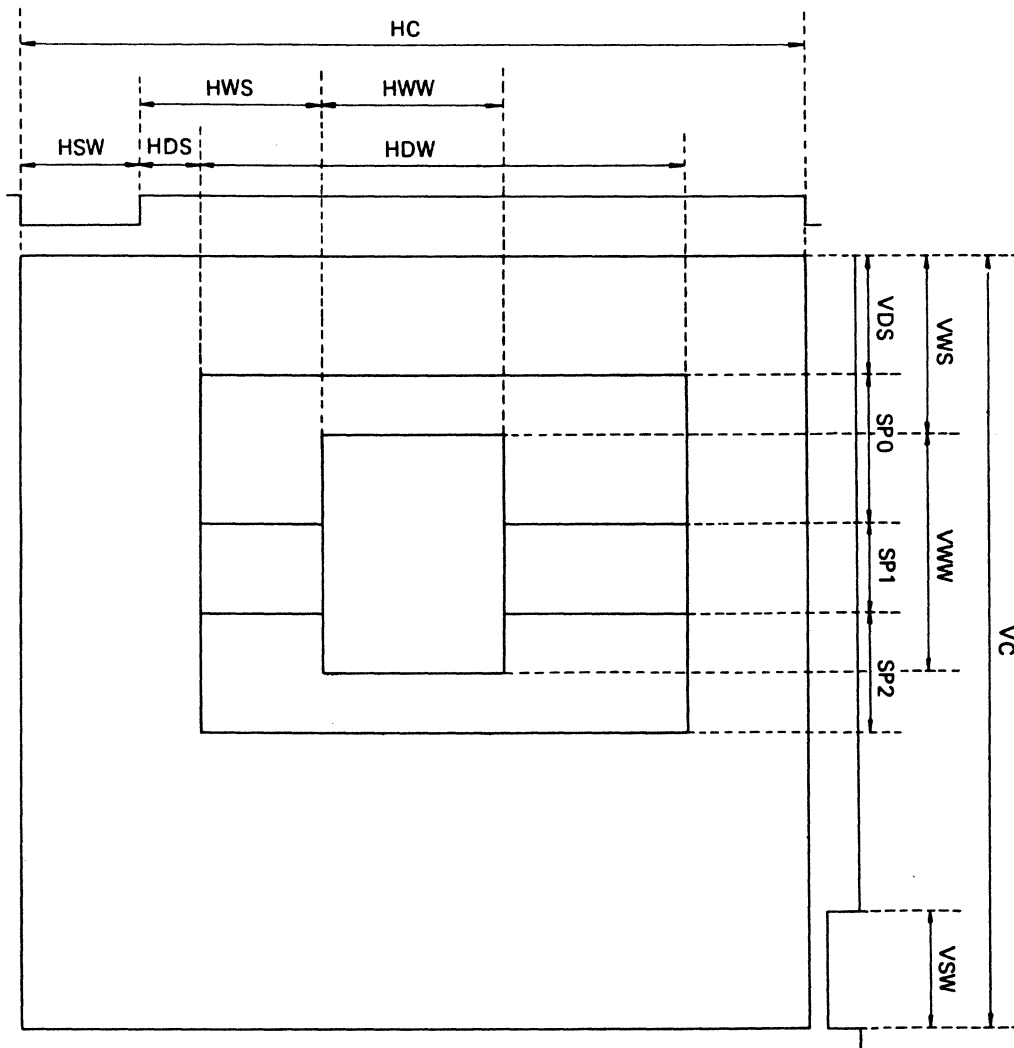
To recap: SYSTEM SPEC

Scan standard	1333 lines interlaced
Scan rate	40 KHz
Field rate	60 Hz
Frame rate	30 Hz
Horizontal resolution	1280 pixels
Vertical resolution	1239 lines
Displayed vertical resolution	1024 lines
Video memory capacity	2M Byte (64 of 64kx4 DRAM)
Video memory cycle period	250 ns
ACRTC clock frequency	8 MHz
Pixel rate	64 Mhz

Monitor parameters :

Line period	25.0 usec
Horizontal Sync width	2.0 usec
Horizontal back porch	2.0 usec
Horizontal front porch	1.0 usec
Vertical Sync width	min. 200 usec
Vertical front porch	990 usec
Vertical back porch	1.375 msec

Figure 7-1: Display Screen Specification



## 7.1 Horizontal Timing

	SET VIA :
HSYNC = 2.0 usec = 8.0 %	HC
BACK PORCH = 2.0 usec = 8.0 %	HSW
FRONT PORCH = 1.0 usec = 4.0 %	HDS
ACTIVE DISPLAY PERIOD = 20.0 usec = 80.0 %	HDW
LINE PERIOD = 25.0 usec = 100 %	HC

The video memory cycle time is 250 nsec

### (i) Line Period:

This is set via HC, the Horizontal Cycle Field:

$$25.0 \text{ usec required: } \frac{25.0 \text{ usec}}{250 \text{ nsec}} = 100 \text{ cycles}$$

note:

As this is an even number, it is suitable for interlaced operation. HC is loaded with one less than this value, thus:

$$100 - 1 = 99 = \$63$$

$$\text{=====> } \underline{\underline{HC = \$63}}$$

### (ii) Horizontal Sync Period:

This is set via HSW, the Horizontal Sync Field,

$$2.0 \text{ usec required: } \frac{2.0 \text{ usec}}{250 \text{ nsec}} = 8 \text{ cycles}$$

$$\text{=====> } \underline{\underline{HSW = \$08}}$$

Note:

- This is greater than 3, so it will allow the RCR to be read.

(iii) Horizontal Back Porch:

Set via HDS, the Horizontal Display Start Field.

$$2.0 \text{ usec required: } \frac{4.0 \text{ usec}}{250 \text{ nsec}} = 8 \text{ cycles}$$

HDS is loaded with one less than this value, thus:

$$\text{=====> } 8 - 1 = 7 = \$07 \quad \underline{\underline{\text{HDS} = \$07}}$$

(iv) Active Display Period:

Set via HDW, the Horizontal Display Width Field.

$$20 \text{ usec required: } \frac{20.0 \text{ usec}}{250 \text{ nsec}} = 80 \text{ cycles}$$

HDW is loaded with one less than this value, thus:

$$\text{=====> } 80 - 1 = 79 = \$4F \quad \underline{\underline{\text{HDW} = \$4F}}$$

(v) Front Porch

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

$$\text{Front Porch} = \text{HC} - (\text{HSW} + \text{HDS} + \text{HDW})$$

Note:

This equation uses the values calculated, not the ones loaded because sometimes they are one less.

$$\text{Front Porch} = 25.0 \text{ usec} - (2.0 \text{ usec} + 2.0 \text{ usec} + 20.0 \text{ usec})$$

$$\text{=====> } \underline{\underline{\text{Front Porch} = 1.0 \text{ usec}}}$$

Summary, the four values are therefore:

1. HC = \$63
2. HSW = \$08
3. HDS = \$07
4. HDW = \$4F

HC and HSW are combined in r82 (HSR).

HDS and HDW are combined in r84 (HDR).

This results in the two registers having the following values:

r82 = \$6308

r84 = \$074F

## 7.2 Vertical Timing

Normally, the vertical flyback period occupies about 7% of the vertical period. In the case of 1333 lines system example, 94 lines are lost per frame, that is 47 lines per field. So only 1239 lines can be used for displaying. As the resolution will have 1024 lines, then the front and the back porches will have 100-115 lines. This should place the displayed lines almost centrally on the display monitor tube face.

### (i) Frame Rasters:

Set by VC, the Vertical Cycle Field;

1333 lines required:  $V = 1333$  cycles per frame

Note:

$V = VC$  for interlaced sync and video mode.

=====>  $\frac{VC}{-----} = \frac{1333}{-----} = \frac{\$535}{-----}$

### (ii) Vertical Back Porch:

Set by VDS, the Vertical Display Start Field;

100 lines required :  $100 = \$64$

Note:

VDS should be loaded with one less than the value required:

=====>  $100 - 1 = 99 = \$63$        $\frac{VDS}{-----} = \frac{\$63}{-----}$   
(using interlace sync & video mode)

### (iii) Vertical Sync Period:

Set by VSW, the Vertical Sync Width Field:

47 lines required per display field (i.e.: 94 lines per display frame).

=====>  $\frac{VSW}{-----} = \frac{47}{-----} = \frac{\$2F}{-----}$

(iv) Display Period:

Set by SP0, the Split Screen Width Field:

$$1024 \text{ lines required: } \quad \underline{\underline{SP0 = 1024 = \$400}}$$

Note: Only the Base Screen is in use.

(v) Front Porch:

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

$$\text{Front Porch} = VC - (VDS + SP0 + 2 \times VSW)$$

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

$$\text{Front Porch} = 1333 - (100 + 1024 + 94)$$

$$\text{Front Porch} = 115 \text{ lines per frame}$$

As the split screens are not in use, SP1 and SP0 does not have to be defined. As the address register auto-increments, it is easier to load these registers r8C and r8E with say, \$0000 than to specifically skip them.

This also applies to the blink control, window display and graphic cursor control registers (see Chapter 4). However, if most of these functions are not to be used, then it is better to skip a continuous group of registers by reloading the address register. The unused functions can be left undefined. The registers so far initialised, (r80 - r8A) are the minimum necessary for the timing control RAM to produce a stable raster timing. The result of the vertical timing calculations are:

$$VC = \$535 \quad VSW = \$2F$$

$$VDS = \$63 \quad SP0 = \$400$$

VDS and VSW are combined in register r88 (VDR). The result is, that the three register have the following values.

$$r86 = \$0535 \quad r88 = \$632F$$

$$r8A = \$0400$$

This completes the programming of the Timing Control RAM.



### 7.3 Display Control RAM (rC0 - rEB)

The display format is specified through the use of this registers. In the example that follows, only the Base Screen will be used for simplicity. The other screens do not need to be defined, even it is not enabled. Configuration is further simplified as the character mode (only with the lX-option) is not used and neither are the cursors.

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

rC0 - rC6	Upper screen - not defined
rC8 - rCe	Base screen - to be defined
rD0 - rD6	Lower screen - not defined
rD8 - rDE	Window screen - not defined
rE0 - rE8	Cursor - not defined
rEA	Zoom factor - to be defined
rEB	Light pen - read only

#### Base Screen Definition:

rCA - Memory Width of the Base Screen

The hardware supports 2 Mbyte of video memory. We only need to consider the Base screen.

The display is 1280 x 1024 pixels, each of 4 bits. As the base screen occupies the whole of this, it represents 640 Kbyte of data, nearly a third of the video memory capacity. If no other screen is defined, there are many possibilities for configuring the base screen in relation to the video memory.

The horizontal display width is  $1280 \times 4 \text{ bits} = 320 \text{ words}$

Recall that HDW was set to 80 cycles and using a GAI = +8 and Dual Access Mode 0

$$\text{HDW} = \frac{80 \times 8}{2} = 320 \text{ words}$$

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

- (i) If the memory width is made equal to the display width  
 MW = 320 words

As the video memory capacity = 1 Mwords it will support:

$$\frac{1 \text{ Mword}}{320} = 3.2 \text{ k or } 3277 \text{ rasters}$$

This will allow vertical scrolling, but no horizontal scrolling.

- (ii) If the memory width is made twice that of the display width,  
 MW = 2 x 320 = 640 words

Frame buffer capacity = 1 Mwords, so it will support:

$$\frac{1 \text{ Mword}}{640} = 1.6 \text{ k or } 1638 \text{ rasters}$$

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

The offset has both horizontal (X) and vertical (Y) components.

(X)

The memory width is 640 words, the display width is 320 words.

$$640 - 320 = 320 \text{ words total margin}$$

Equally divided between left and right margins gives:

$$X = \frac{320}{2} = 160 \text{ words, the horizontal offset}$$

Likewise, the memory supports 1638 rasters, the display uses 1024 rasters.

$$1638 - 1024 = 614 \text{ rasters total margin}$$

Equally divided between top and bottom margins give:

$$Y = \frac{614}{2} = 307 \text{ rasters, the vertical offset}$$

Choosing the latter result, means that the memory width of the base screen must be set to 640 words:

$$\text{MWB} = \$280$$

$$\text{Hence } \underline{\text{rCA}} = \underline{\$0280}$$

Start Address:

If the display screen is to be positioned about central to the video memory, the screen start address must be offset from that of the video memory.

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

The start of the 308th raster will be  $640 \text{ words} \times 307 = 196480$  words from the start of memory, adding a horizontal offset of 160 words.

$196480 + 160 = 196640$  words from the start of memory, because the memory starts at address \$0.

Base Screen Start Address = \$30020

This value is split between registers rCC and rCE, because no smooth horizontal scrolling can be applied, the start dot address (SDA) is \$0 because it has a 20 bit range.

Therefore rCC = \$0003

rCE = \$0020

Zoom Factor = rEA

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

rEA = \$0000

This completes the programming of the Display Control RAM.

## 7.4 Control Registers

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

The preferred order of initialization is:

- (1) Command Control Register (CCR: r02)
- (2) Display Control Register (DCR: r06)
- (3) Operation Mode Register (OMR: r04)

Together these registers hold 30 fields of control bits and each must carefully be considered in relation to the application. Chapter 4.9 gives detailed explanation on the function of each field. The following example applies to the example system and represents a simple application for clarity.

### (1) Command Control Register (r02)

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

Bits 7 - 0 :

Enable/disable the interrupt sources. This example uses polled status to control transfers and so all this can be disabled.

Bits 8 - 10 :

Graphic Bit Mode, this sets the number of bits per pixel. This example uses 4 bits per pixel, and so the mode is '010' i.e. \$3.

Bits 11 - 13 :

The DMA control bits, as DMA is not used these are all 0.

Bit 14 PAUSE :

This bit halts the command execution, it must be 0 in order to permit commands to be processed later.

Bit 15 ABORT :

Reset left this bit set, it must now be cleared to enable command execution later.

The above values can now be written into the CCR.

The resulting value is thus : CCR = \$0200

(2) Display Control Register (r06)

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

Bits 0 - 3 Colour Look-up Table Control :

In this example LUT 0 is used for display, so this bits must be set to \$0.

Bits 4 - 6 Display Mode Control :

Since 4 bits per pixel are displayed, the value of these bits is set to `001' i.e. \$1.

Bits 8 - 13 Split Enables :

As these screens are not used, therefore they not defined, they are all cleared to disable these screens.

Bit 14 Split Enable 1 (Base) :

This bit enables the base screen. As this screen is in use, it must be set = \$1.

Bit 15 DISP Control :

The DISP signals, together with the HSYNC and VSYNC signals, allow blanking of the video signals and generation of front and back porches. These can be used for driving display monitors.

DISP 1 provides a combined horizontal and vertical blanking signal for both background and window screens when this bit is set. In fact, this example only uses the base screen, the bit could also be cleared, when DISP1 only applies to the background screen(s) and DISP2 to the window screen.

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

Therefore the value of the DCR is :

$$\underline{\text{DCR}} = \underline{\text{\$C010}}$$

(3) Operation Mode Register (r04)

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

Bits 0 - 1 Raster Scan Mode :

In order to operate in interlace sync & video mode, these must both set i.e.: \$3.

Bits 2 - 3 Access Mode :

For improved drawing speed the SYS68K/AGC-1 uses interleaved access mode (DA0), hence these have the value \$2.

Bits 4 - 6 Graphics Increment Mode :

Because of the hardware structure of the SYS68K/AGC-1 128 bits are obtained from the video memory per display access, hence the addressing must increment by 8 words, so set GAI = \$3.

Bits 7 - 12 :

These bits are all set to \$0 because of the hardware design of the SYS68K/AGC-1.

Bit 13 Access Priority :

To avoid disruption of the displayed image due to drawing operations, the display process will be given priority over drawing, hence this bit will be \$0.

Bit 14 START :

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

Bit 15 Master/Slave :

As this example system will not be synchronized from an external source, this bit will be set so as the ACRTC acts in master mode.

Therefore the resulting value for the Operation Mode Register is:

$$\underline{\text{OMR}} = \underline{\text{\$C03B}}$$

The initialisation of the SYS68K/AGC-1 is now completed.

**APPENDIX TO THE**  
**HARDWARE**  
**USER'S MANUAL**

FORCE COMPUTERS Inc./GmbH  
All Rights Reserved

This document shall not be duplicated, nor its contents used  
for any purpose, unless express permission has been granted.

Copyright by FORCE Computers®





A. SPECIFICATION OF THE SYS68K/AGC-1.....A-1

B. MEMORY MAP OF THE SYS68K/AGC-1.....B-1

    B.1. Standard Memory Access.....B-1

    B.2. Short I/O Access.....B-2

C. ADDRESS ASSIGNMENTS AND MEMORY LAYOUT OF THE DEVICES.....C-1

D. CIRCUIT SCHEMATICS OF THE SYS68K/AGC-1.....D-1

E. CONNECTOR PIN ASSIGNMENTS OF THE SYS68K/AGC-1 BOARDS.....E-1

    E.1. Master Board P1.....E-1

    E.2. Master Board P2.....E-2

    E.3. Master Board P3.....E-3

    E.4. Master Board P4.....E-4

    E.5. Master Board P6.....E-5

    E.6. Slave Board P1.....E-6

    E.7. Slave Board P2.....E-7

    E.8. Slave Board P3.....E-8

F. COMPONENT PART LIST OF THE SYS68K/AGC-1 BOARDS.....F-1

G. LITERATURE REFERENCE.....G-1

H. DEFAULT JUMPER SETTINGS.....H-1



## APPENDIX A

### SPECIFICATION OF THE SYS68K/AGC-1

- 63484 ACRTC with 8 MHz Clock Frequency
- 2 Mbyte dynamic Video-RAM with 120ns access time for up to 2048 \* 2048 pixel (in 4 bit/pixel mode)
- Direct video memory access via VMEbus during display time
- Zoom logic for magnifications in horizontal and vertical direction up to 16
- Smooth scroll logic for vertical and horizontal direction
- Video interface bus for upgrading with character overlay
- 3 Graphic Color Palette AM8151 with 256 entries for each color and 64 MHz Pixel clock
- Bus Interrupt Modul for all local Interrupt sources
- Interrupt handling via programmable interrupt vectors
- Each VMEbus IRQ level can be enabled/disabled via software
- Fully decoding of the address modifiers
- Jumper selectable access for short I/O or standard memory access
- RUN/LOCAL switch for complete isolation
- Fully VMEbus and IEEE P1014 compatible
  
- Power Requirements per slot
  - + 5V (max) 4.9A
  - 12V (max) 1.0A
  
- Operating Temperature (degree C) 0 to 50
- Storage Temperature (degree C) -40 to 85
- Relative Humidity (Non-condensing) 0 to 95%
  
- Board Dimensions 234x160mm (9.2x6.3")
- No. of Slots used 2
- Thickness 38mm (1.39")



APPENDIX B

MEMORY MAP OF THE SYS68K/AGC-1

STANDARD MEMORY ACCESS

The Board Base Address (BBA) is jumper selectable in 256K steps

Start Address	End Address	Memory Area
C00000	C35FFF	BIM 68153 13.824 times
C36000	C37FFF	GCP 1 red 16 times
C38000	C39FFF	GCP 2 green 16 times
C3A000	C3BFFF	GCP 3 blue 16 times
C3C000	C3FFFF	ACRTC 63484 4096 times
C40000	E3FFFF	Video-RAM 1 time

APPENDIX B

SHORT I/O MEMORY ACCESS

The Board Base Address (BBA) is jumperselectable in 4K steps

Start Address	End Address	Memory Area
C00000	C005FF	BIM 68153 96 times
C00600	C007FF	GCP 1 red 1 times
C00800	C009FF	GCP 2 green 1 times
C00A00	C00BFF	GCP 3 blue 1 times
CO0COO	COODFF	ACRTC 63484 128 times
only accessible via ACRTC 63484		Video-RAM

APPENDIX C

ADDRESS ASSIGNMENTS AND REGISTER LAYOUT OF THE DEVICES FOR STANDARD MEMORY ACCESS

The BIM Register Layout

Default Address	Register Name	Function	Reset Value
C00000	CONTROL REGISTER 1	reserved	\$00
C00002	CONTROL REGISTER 2	ACRTC Interrupt	\$00
C00004	CONTROL REGISTER 3	VSUNC Interrupt	\$00
C00006	CONTROL REGISTER 4	reserved	\$00
C00008	VECTOR REGISTER 1	reserved	\$0F
C0000A	VECTOR REGISTER 2	ACRTC	\$0F
C0000C	VECTOR REGISTER 3	VSUNC	\$0F
C0000E	VECTOR REGISTER 4	reserved	\$0F

The ACRTC Register Layout

Default Address	Register Name	Reset Value
C3C000	ADDRESS REGISTER 1	\$FF23
C3C002	FIFO ENTRY	\$0000



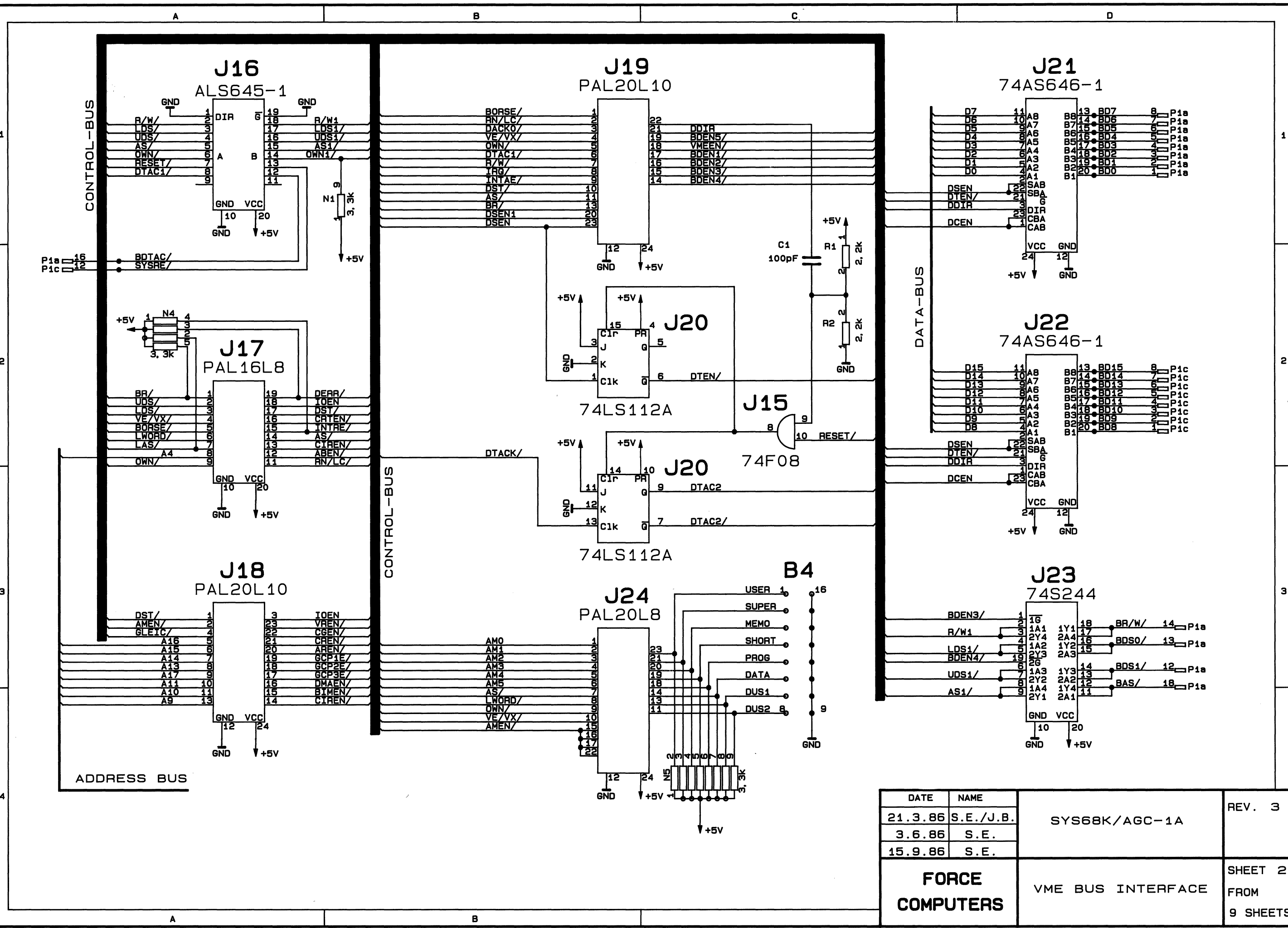


APPENDIX D  
CIRCUIT SCHEMATICS OF THE SYS68K/AGC-1 BOARDS





Date : 22.09.1986 Time : 17:58:07  
 2sh2



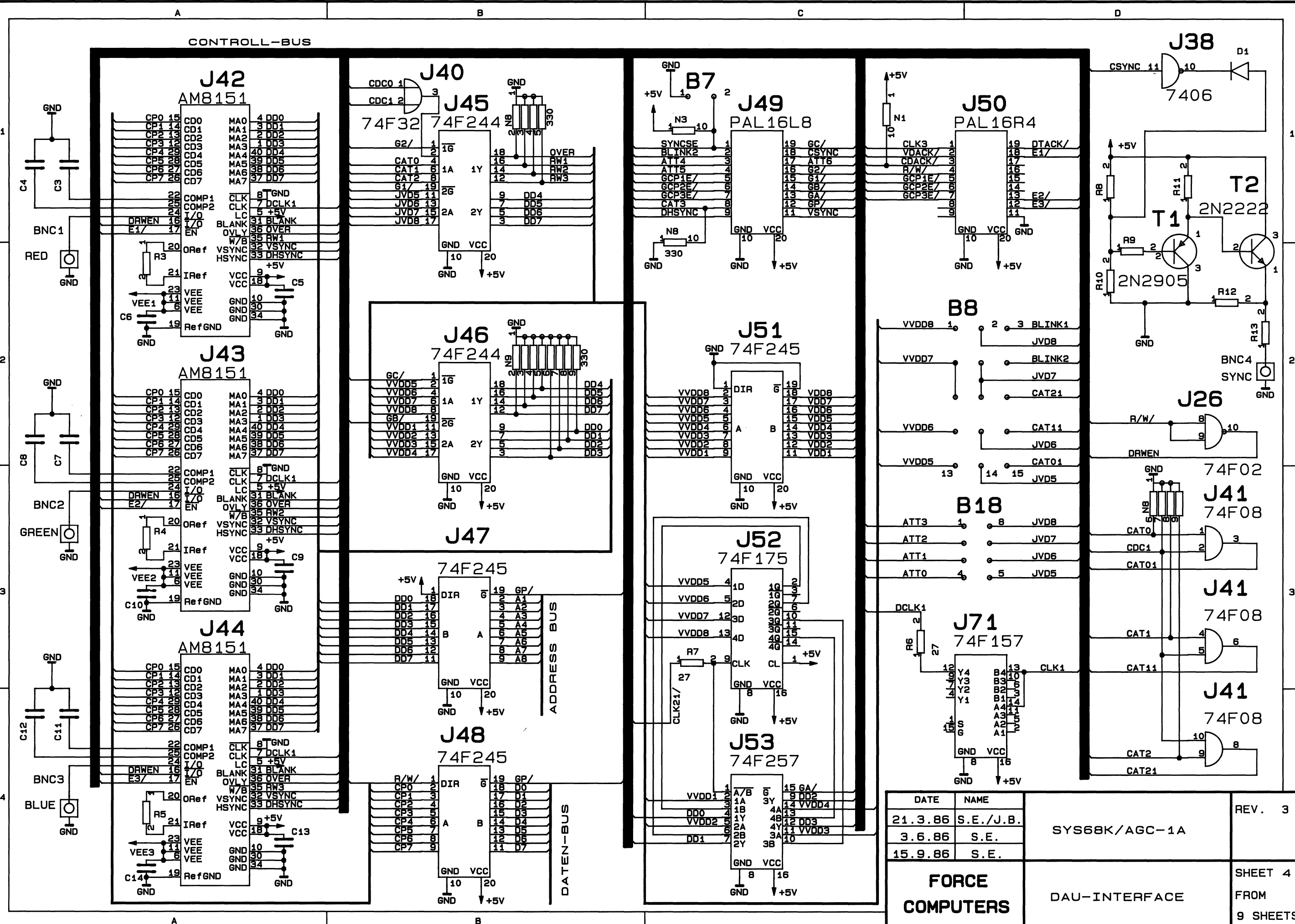
DATE	NAME	SYS68K/AGC-1A	REV. 3
21.3.86	S.E./J.B.		
3.6.86	S.E.		
15.9.86	S.E.		
<b>FORCE COMPUTERS</b>		VME BUS INTERFACE	SHEET 2 FROM 9 SHEETS



Time : 18:47:55

Date : 22.09.1986

062sh4

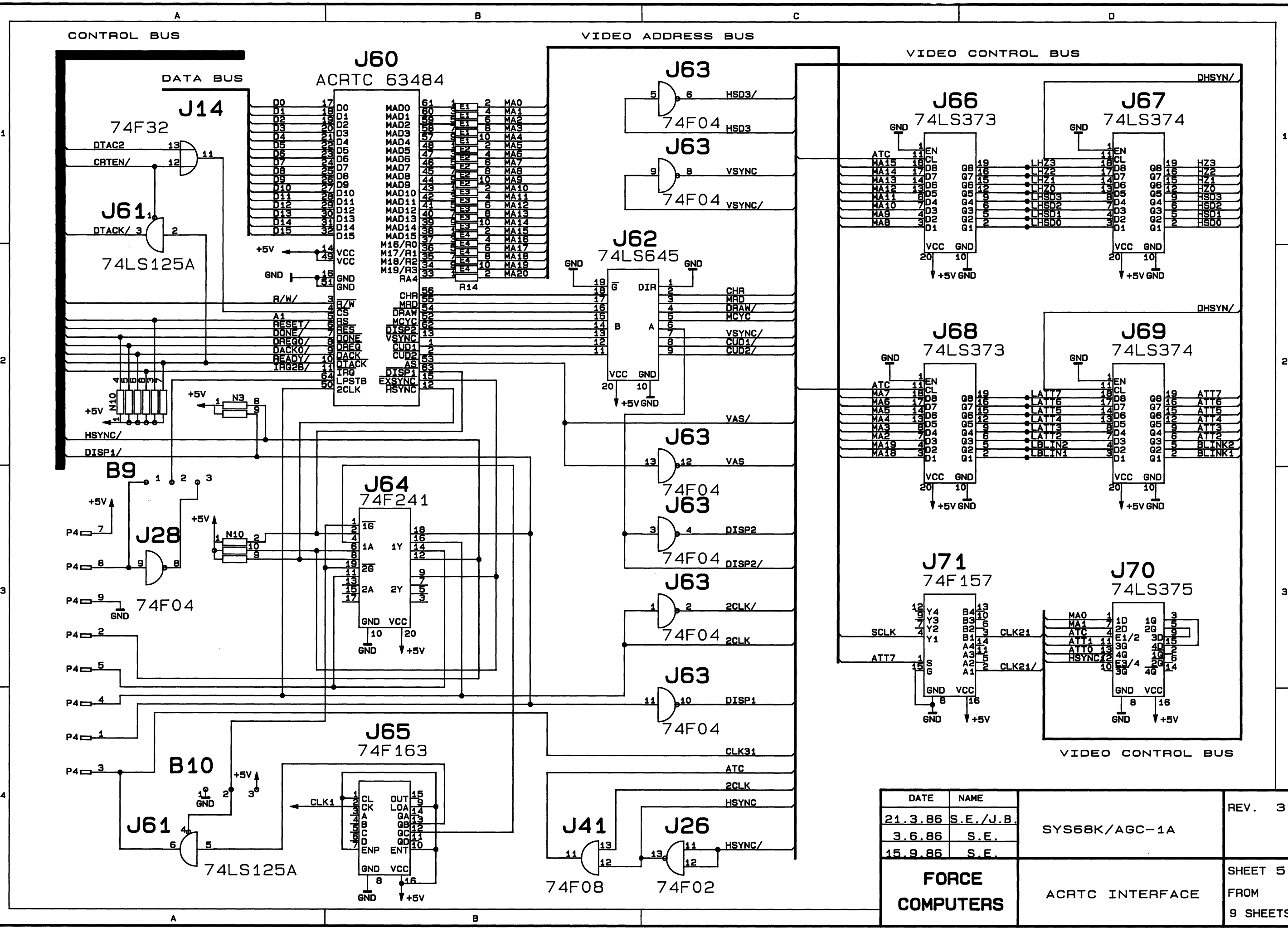


DATE	NAME	REV.
21.3.86	S.E./J.B.	3
3.6.86	S.E.	
15.9.86	S.E.	

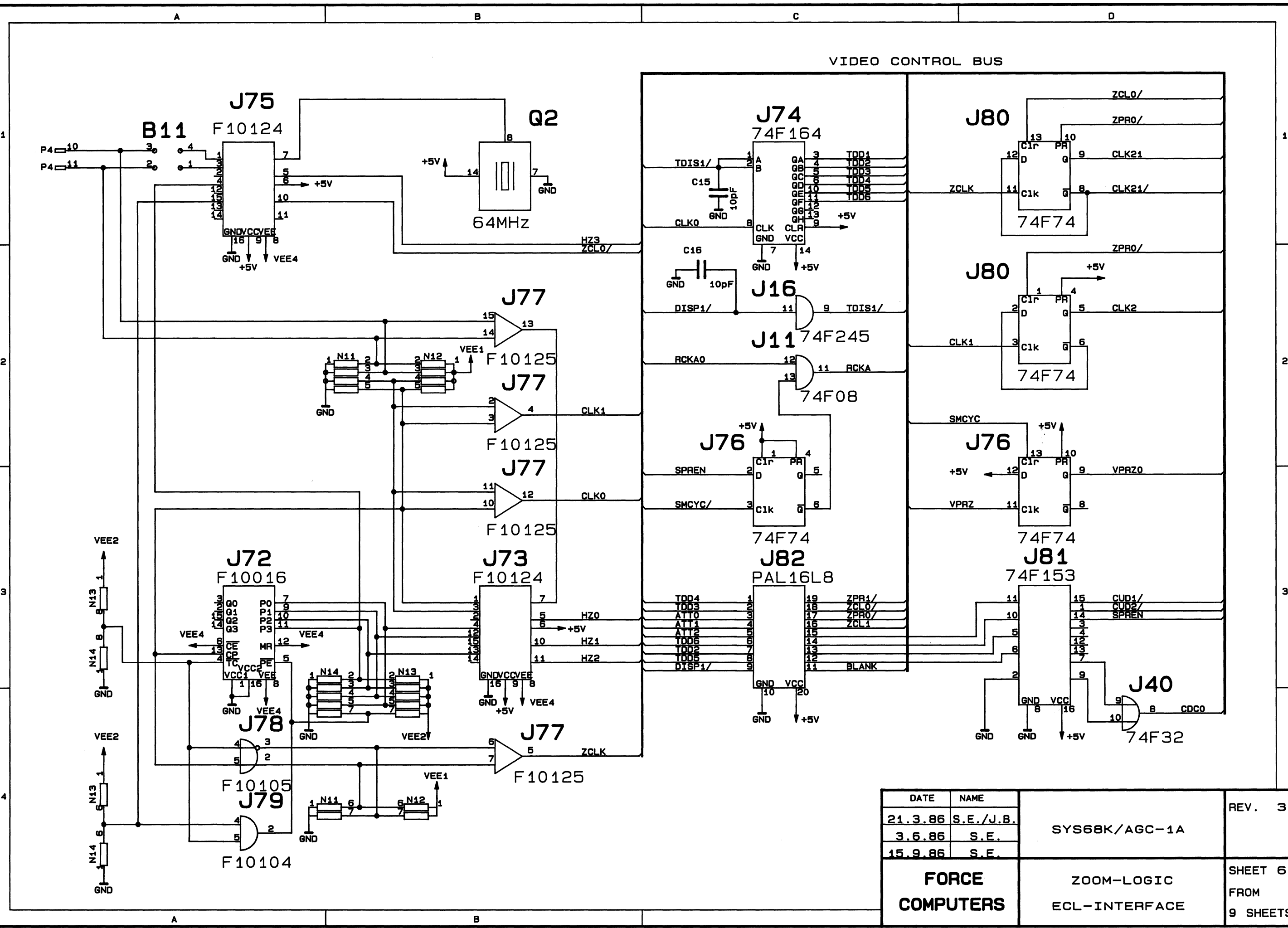
<b>FORCE COMPUTERS</b>	SYS68K/AGC-1A	SHEET 4
	DAU-INTERFACE	FROM 9 SHEETS

Date : 22.09.1986  
 Time : 19:15:37  
 062sn5



DATE	NAME	SYS68K/AGC-1A	REV.
21.3.86	S.E./J.B.		3
3.6.86	S.E.		
15.9.86	S.E.		
<b>FORCE COMPUTERS</b>		ACRTC INTERFACE	SHEET 5 FROM 9 SHEETS

Date : 22.09.1986 Time : 19:50:33  
062sh6



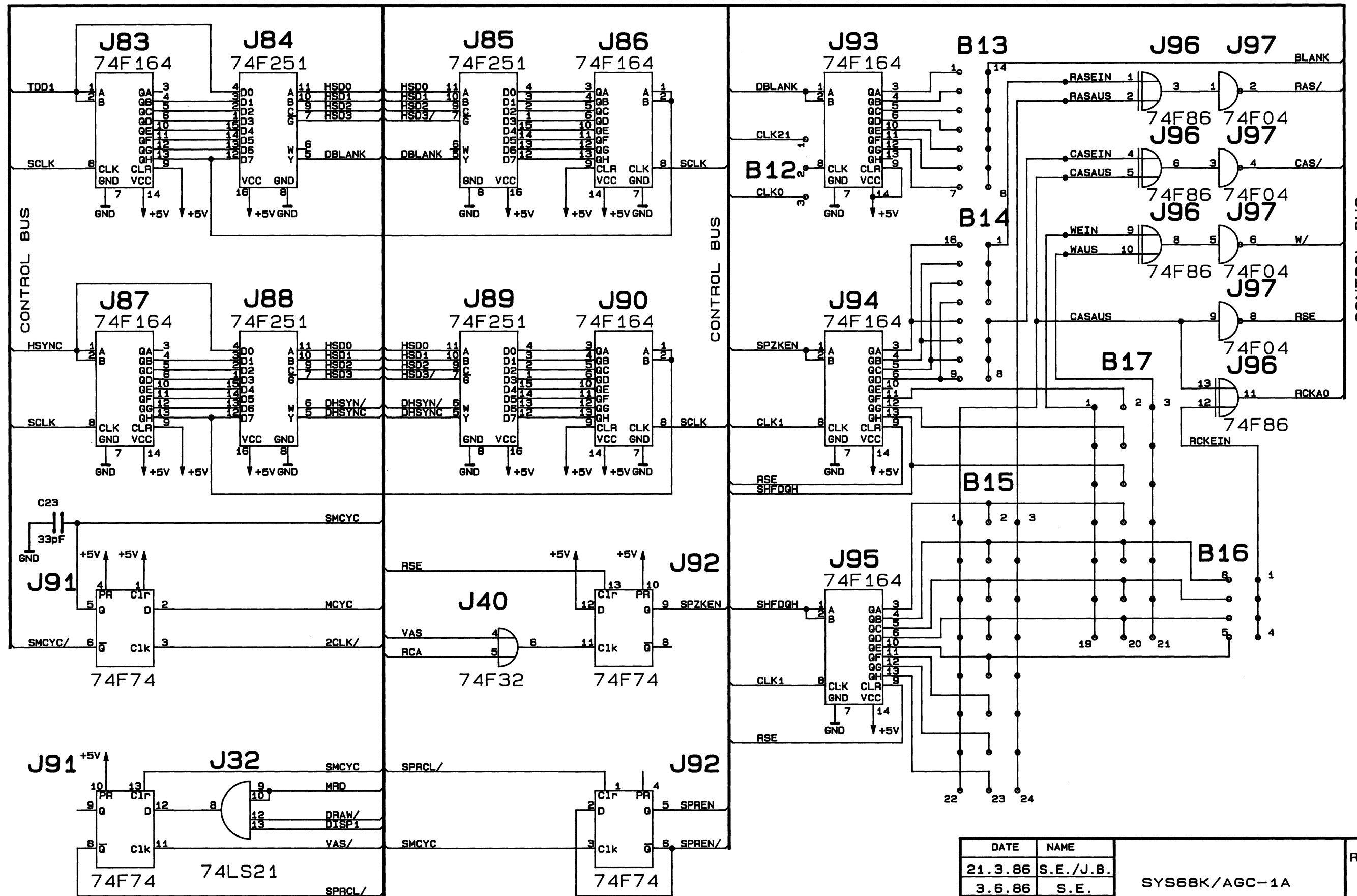
DATE	NAME		REV. 3
21.3.86	S.E./J.B.	SYS68K/AGC-1A	
3.6.86	S.E.		
15.9.86	S.E.		
<b>FORCE COMPUTERS</b>		ZOOM-LOGIC ECL-INTERFACE	SHEET 6 FROM 9 SHEETS



Time : 19:35:46

Date : 22.09.1986

62sh7



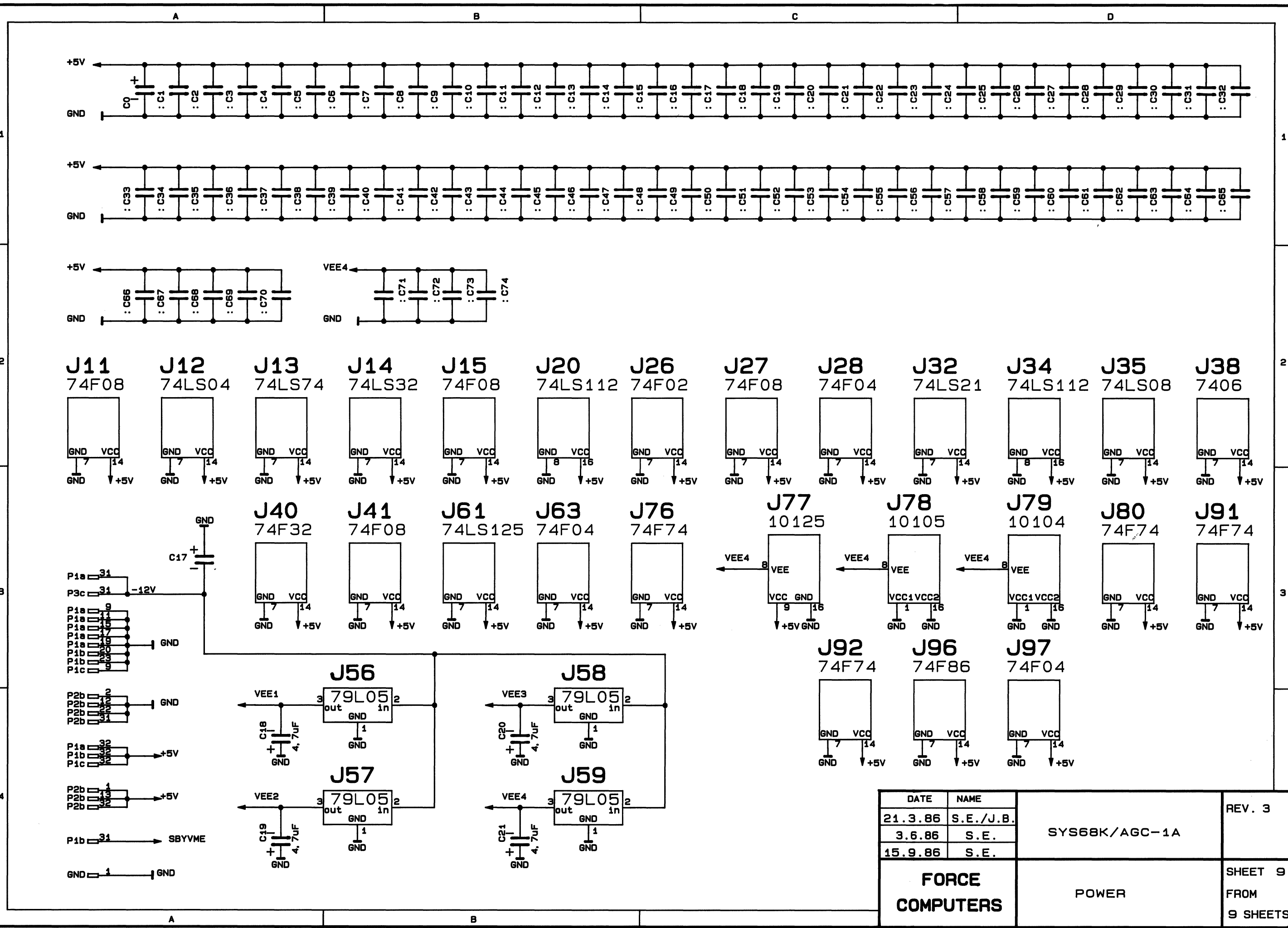
DATE	NAME	SYS68K/AGC-1A	REV. 3
21.3.86	S.E./J.B.		
3.6.86	S.E.		
15.9.86	S.E.		
FORCE COMPUTERS			SHEET 7 FROM 9 SHEETS

Date : 22.09.1986  
 Time : 20:20:55  
 062sh8

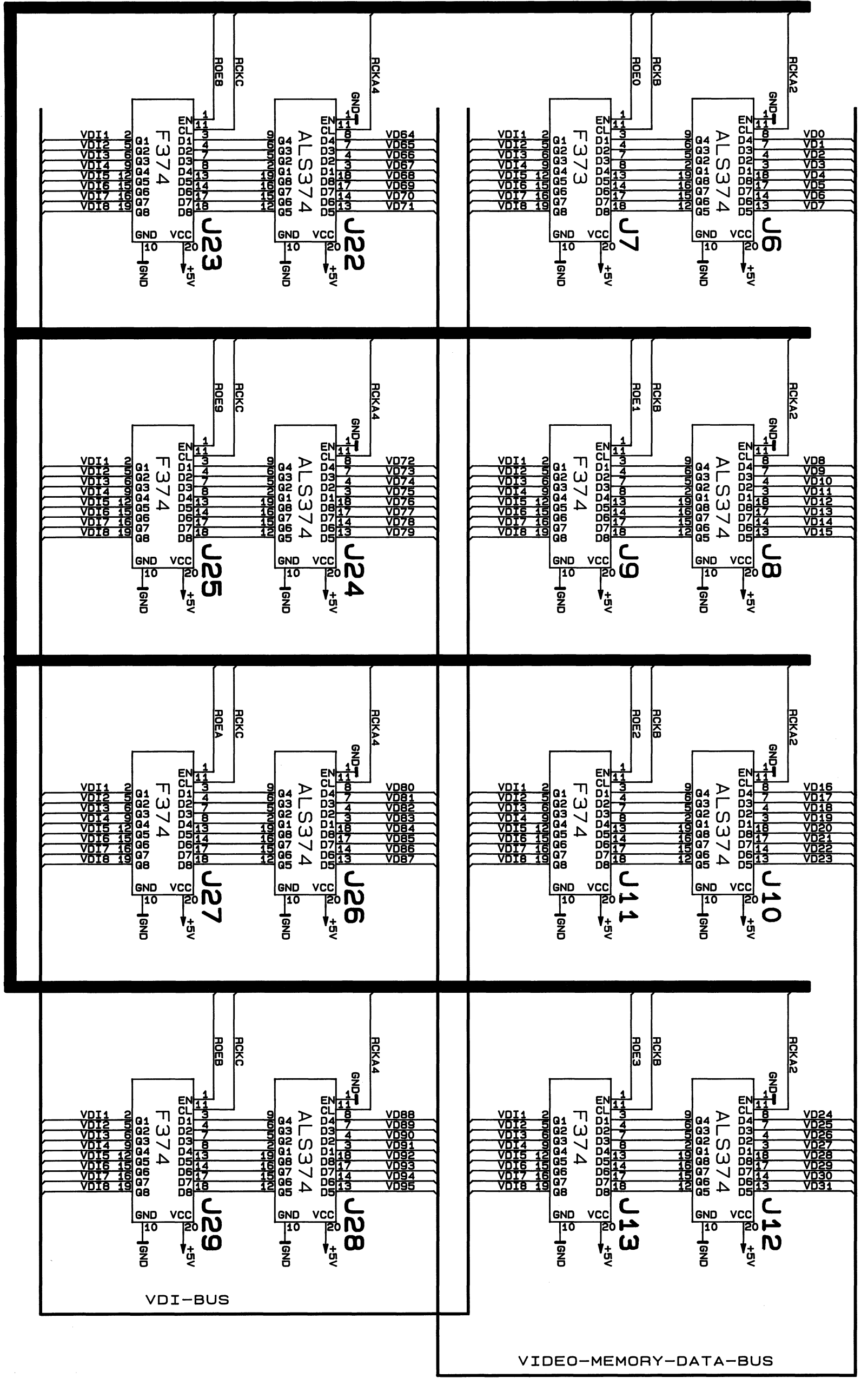
A	B	C	D
D15 1 P3a	D14 1 P3b	D13 1 P3c	IAC1B/ 1 P6a
LDS/ 2 P3a	DBCLK 2 P3b	D12 2 P3c	CGEN/ 2 P6a
R/W/ 3 P3a	MA19 3 P3b	D11 3 P3c	CREN/ 3 P6a
VPRZ 4 P3a	MA18 4 P3b	D10 4 P3c	ABEN/ 4 P6a
DGTE0/ 5 P3a	MA17 5 P3b	D9 5 P3c	AREN/ 5 P6a
ZCL1 6 P3a	MA16 6 P3b	D8 6 P3c	DMAEN/ 6 P6a
SPREN 7 P3a	GBPEN/ 7 P3b	D7 7 P3c	IRQ1/ 7 P6a
GND 8 P3a	DGTE1/ 8 P3b	D6 8 P3c	BDEN5/ 8 P6a
MRO 9 P3a	DRAW/ 9 P3b	D5 9 P3c	IRQ1B/ 9 P6a
CHR 10 P3a	VDD8 10 P3b	D4 10 P3c	DACK0/ 10 P6a
ZPR1/ 11 P3a	MA13 11 P3b	D3 11 P3c	DDIR 11 P6a
GND 12 P3a	MA12 12 P3b	D2 12 P3c	LAS/ 12 P6a
SMCYC 13 P3a	MA11 13 P3b	D1 13 P3c	CLK31 13 P6a
GBDEN 14 P3a	MA10 14 P3b	D0 14 P3c	UDS1/ 14 P6a
BPAEN/ 15 P3a	MA9 15 P3b	VAS 15 P3c	15 P6a
BPGEN1 16 P3a	MA8 16 P3b	RCKA 16 P3c	LDS1/ 16 P6a
BPGEN0 17 P3a	MA14 17 P3b	W/ 17 P3c	CDC0 17 P6a
VTDIR 18 P3a	MA15 18 P3b	CAS/ 18 P3c	CDC1 18 P6a
GND 19 P3a	VDD7 19 P3b	RAS/ 19 P3c	19 P6a
20 P3a	VDD6 20 P3b	UDS/ 20 P3c	20 P6a
A18 21 P3a	A17 21 P3b	MA5 21 P3c	DSEN1 21 P6a
A20 22 P3a	A19 22 P3b	MA4 22 P3c	DISP2 22 P6a
A10 23 P3a	A9 23 P3b	MA3 23 P3c	23 P6a
A12 24 P3a	A11 24 P3b	MA2 24 P3c	24 P6a
A14 25 P3a	A13 25 P3b	MA1 25 P3c	25 P6a
A16 26 P3a	A15 26 P3b	MA0 26 P3c	DTAC2/ 26 P6a
A2 27 P3a	A1 27 P3b	MA6 27 P3c	27 P6a
A4 28 P3a	A3 28 P3b	MA7 28 P3c	28 P6a
A6 29 P3a	A5 29 P3b	CLK21 29 P3c	29 P6a
A8 30 P3a	A7 30 P3b	HSYNC 30 P3c	30 P6a
VDD4 31 P3a	VDD2 31 P3b	-12V 31 P3c	31 P6a
VDD5 32 P3a	VDD3 32 P3b	VDD1 32 P3c	32 P6a
BG0IN/ 4 P1b			CDACK/ 1 P6b
BG0OU/ 5 P1b			AM0 2 P6b
BG1IN/ 6 P1b			AM1 3 P6b
BG1OU/ 7 P1b			AM2 4 P6b
BG2IN/ 8 P1b			OWN1/ 5 P6b
BG2OU/ 9 P1b			RESET/ 6 P6b
BG3IN/ 10 P1b			LWORD/ 7 P6b
BG3OU/ 11 P1b			DERR/ 8 P6b
			CLK3 9 P6b
			DCEN 10 P6b
			BR/ 11 P6b
			CLK2 12 P6b
			13 P6b
			ATT7 14 P6b
			LD4-2 15 P6b
			LD4-1 16 P6b
			17 P6b
			CAT0 18 P6b
			CAT1 19 P6b
			CAT2 20 P6b
			21 P6b
			22 P6b
			23 P6b
			CLK1 24 P6b
			25 P6b
			26 P6b
			27 P6b
			28 P6b
			29 P6b
			30 P6b
			ISIO/ 31 P6b
			IASIO/ 32 P6b
			CAT3 1 P6c
			AM4 2 P6c
			R/W1 3 P6c
			RUN/ 4 P6c
			DREQ0/ 5 P6c
			DTAC1/ 6 P6c
			DONE/ 7 P6c
			READY/ 8 P6c
			ATT3 9 P6c
			AS1/ 10 P6c
			DTACK/ 11 P6c
			MA20 12 P6c
			13 P6c
			14 P6c
			15 P6c
			16 P6c
			17 P6c
			18 P6c
			19 P6c
			20 P6c
			21 P6c
			22 P6c
			HSYNC/ 23 P6c
			24 P6c
			25 P6c
			26 P6c
			27 P6c
			28 P6c
			29 P6c
			30 P6c
			31 P6c
			32 P6c

DATE	NAME	SYS68K/AGC-1A	REV.
21.3.86	S.E./J.B.		3
3.6.86	S.E.		
15.9.86	S.E.		
<b>FORCE COMPUTERS</b>			SHEET 8 FROM 9 SHEETS

Date : 22.09.1986 Time : 20:58:59 062sh9



DATE	NAME	SYS68K/AGC-1A	REV. 3	
21.3.86	S.E./J.B.		POWER	SHEET 9 FROM 9 SHEETS
3.6.86	S.E.			
15.9.86	S.E.	FORCE COMPUTERS		



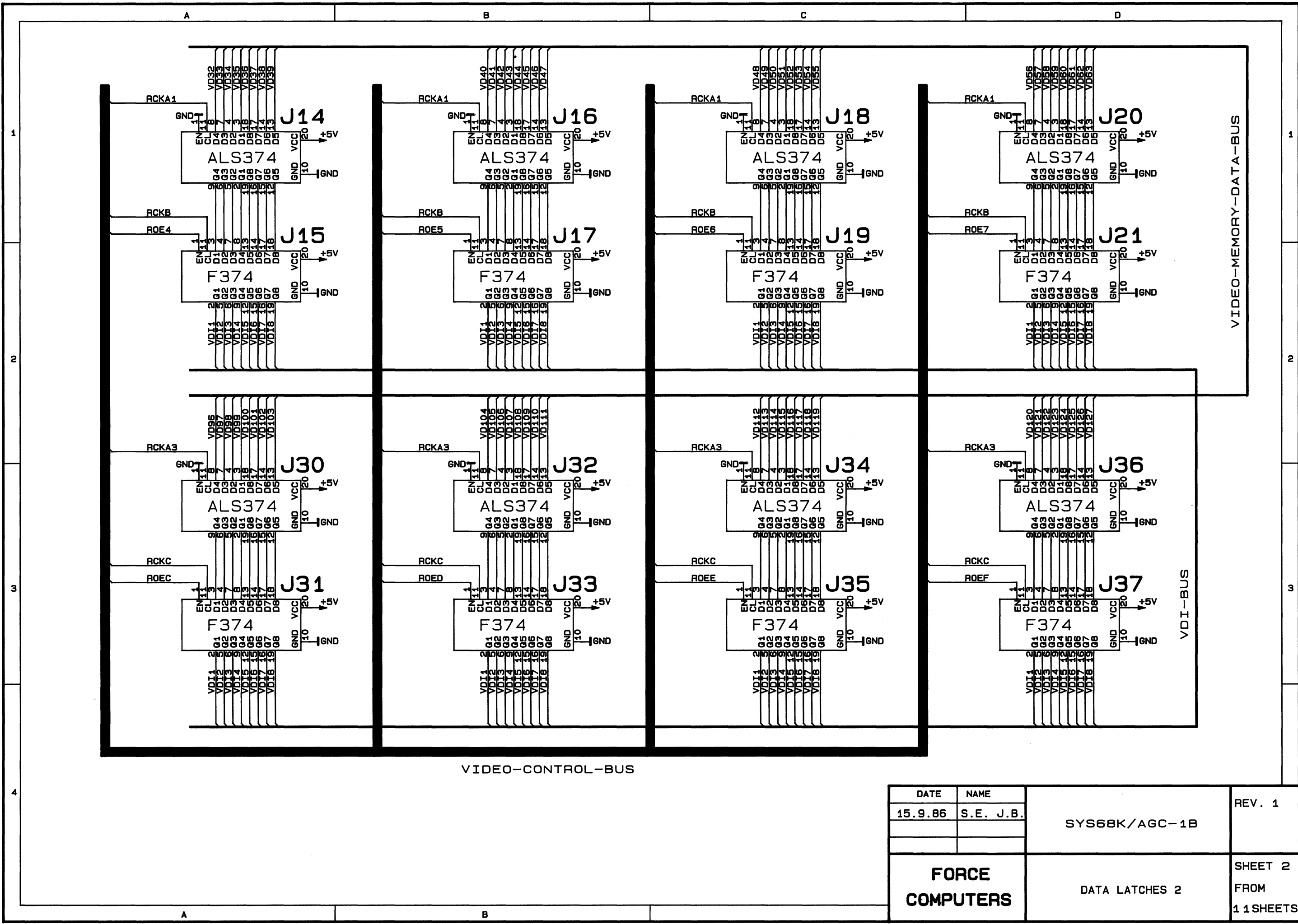
VIDEO-CONTROL-BUS

VIDEO-MEMORY-DATA-BUS

DATE	NAME		REV. 1
15.9.86	S.E. J.B.	SYS6BK/AGC-1B	
<b>FORCE COMPUTERS</b>		DATA LATCHES 1	SHEET 1 FROM 11 SHEETS

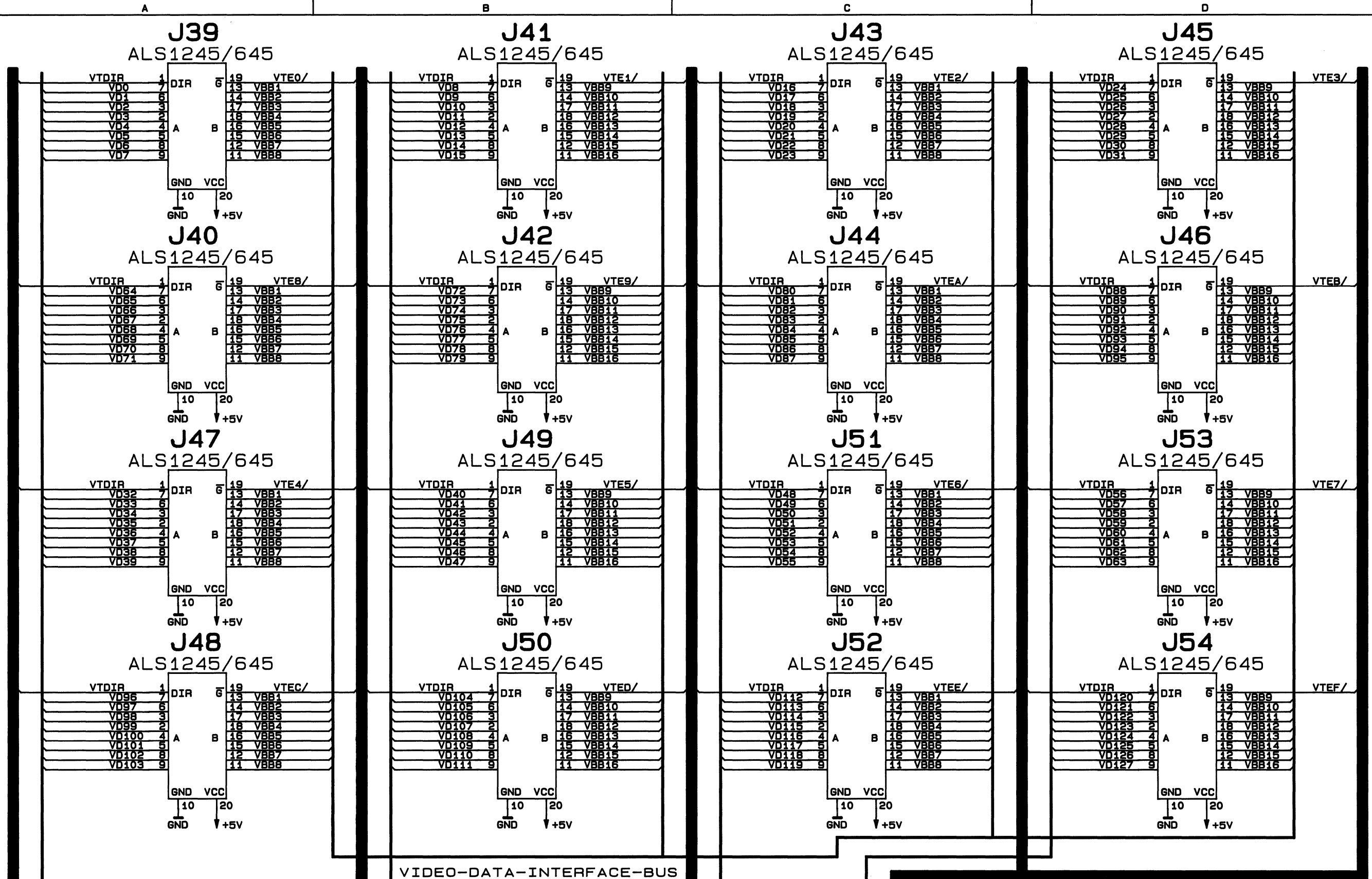
Date : 21.09.1986 Time : 11:31:22

014sh2



DATE	NAME	SYS68K/AGC-1B	REV. 1
15.9.86	S.E. J.B.		
FORCE COMPUTERS		DATA LATCHES 2	SHEET 2 FROM 11 SHEETS

Date : 20.09.1986 Time : 16:38:28  
 014sh3



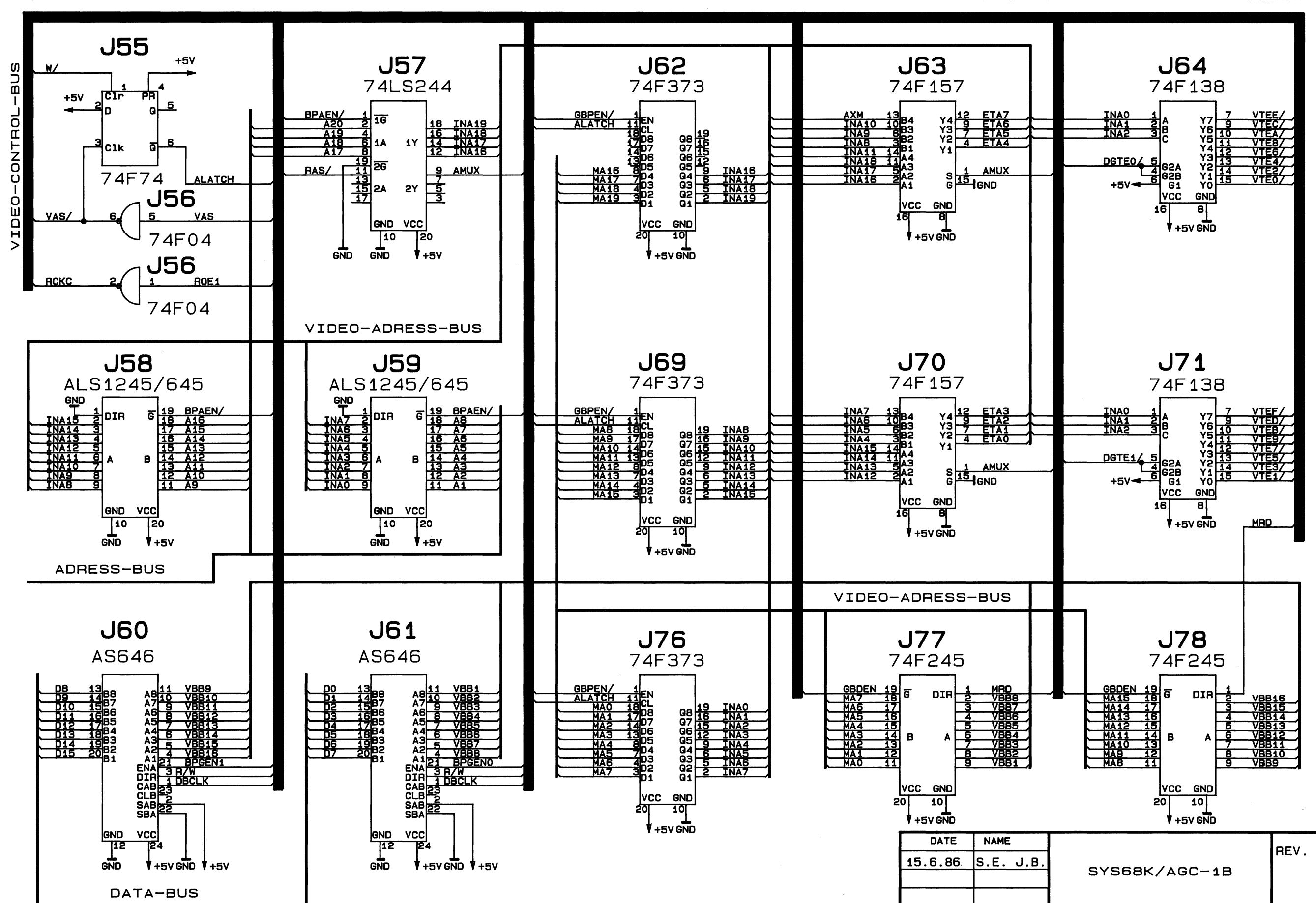
VIDEO-MEMORY-DATA-BUS

VIDEO-DATA-INTERFACE-BUS

VIDEO-CONTROL-BUS

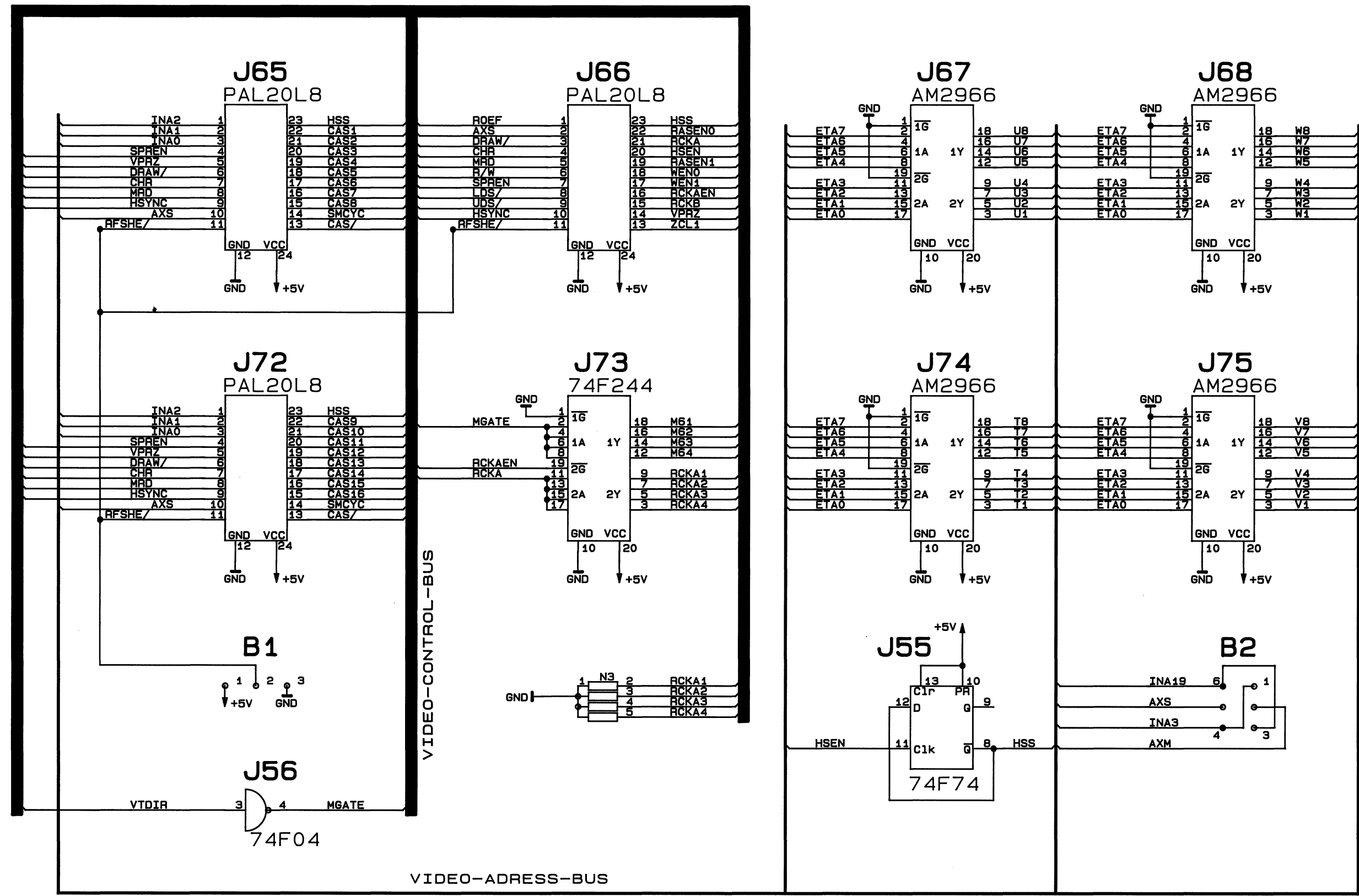
DATE	NAME	REV.
15.9.86	S.E. J.B.	1
SYS68K/AGC-1B		
FORCE COMPUTERS		SHEET 3
DRIVER		FROM
		11 SHEETS

Date : 21.09.1986 Time : 11:49:11



DATE	NAME	SYS68K/AGC-1B	REV. 1
15.6.86	S.E. J.B.		
<b>FORCE COMPUTERS</b>		GRAPHIC-VIDEO-RAM INTERFACE	SHEET 4 FROM 11 SHEETS

Date : 21.09.1986 Time : 12:25:09

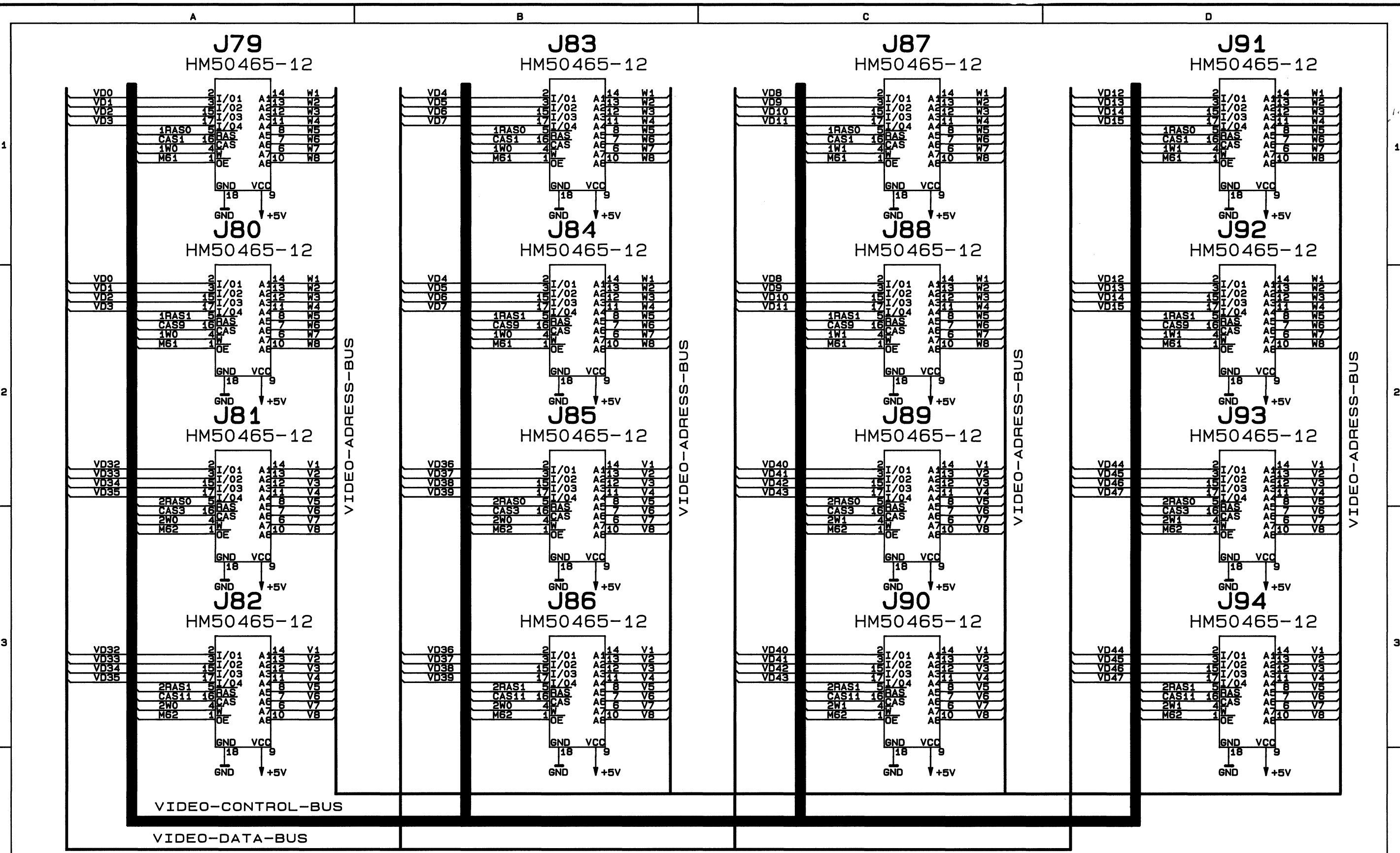


DATE	NAME	SYS68K/AGC-1B	REV. 1
15.9.86	S.E. J.B.		
<b>FORCE COMPUTERS</b>		GRAPHIC-VIDEO-RAM INTERFACE	SHEET 5 FROM 11 SHEETS



Date : 21.09.1986 Time : 12:46:06

014sh6

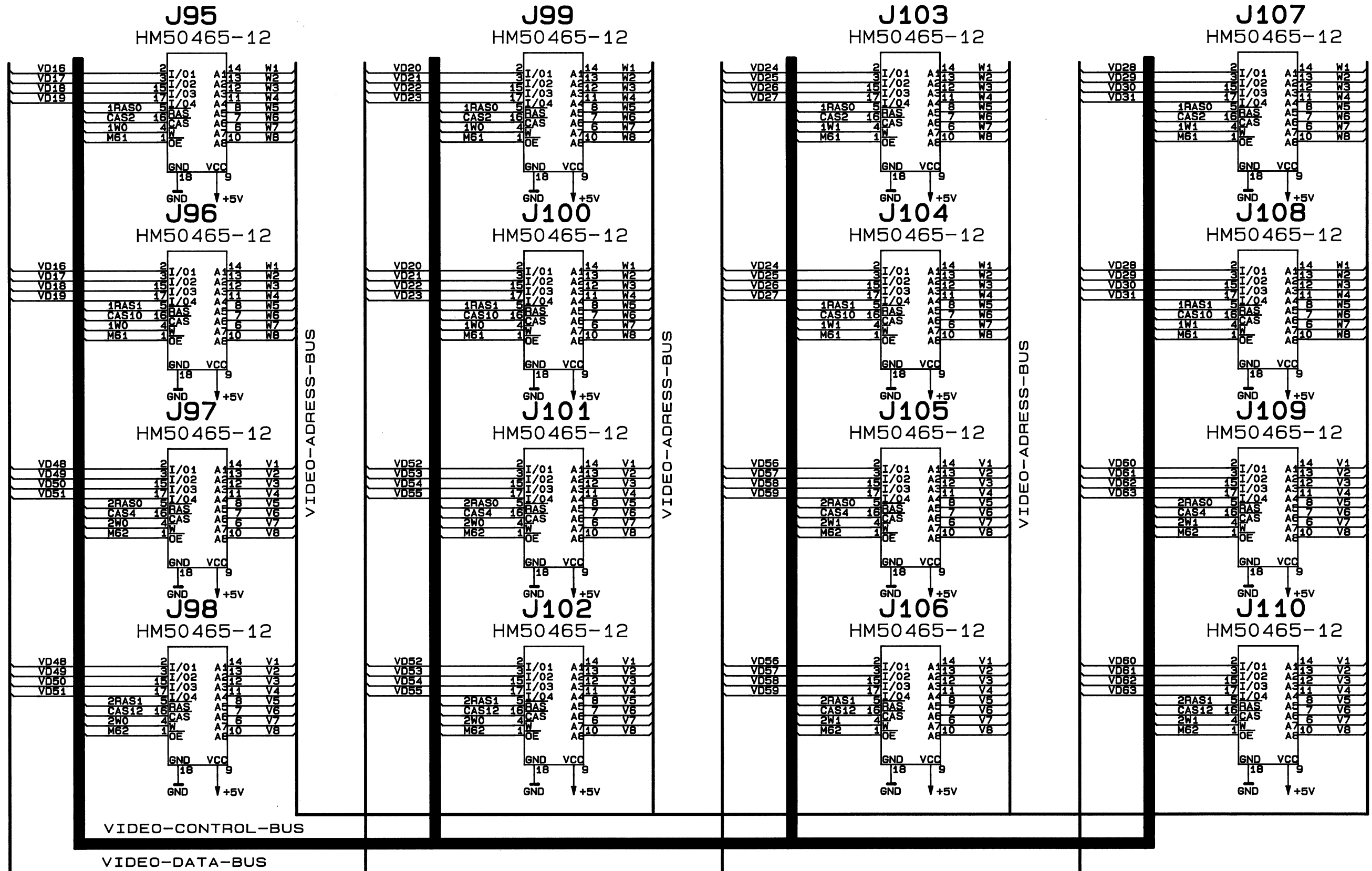


DATE	NAME	SYS68K/AGC-1B	REV. 1
15.9.86	S.E. J.B.		
FORCE COMPUTERS		GRAPHIC-VIDEO-RAM	SHEET 6
			FROM 11 SHEETS

Time : 13:06:33

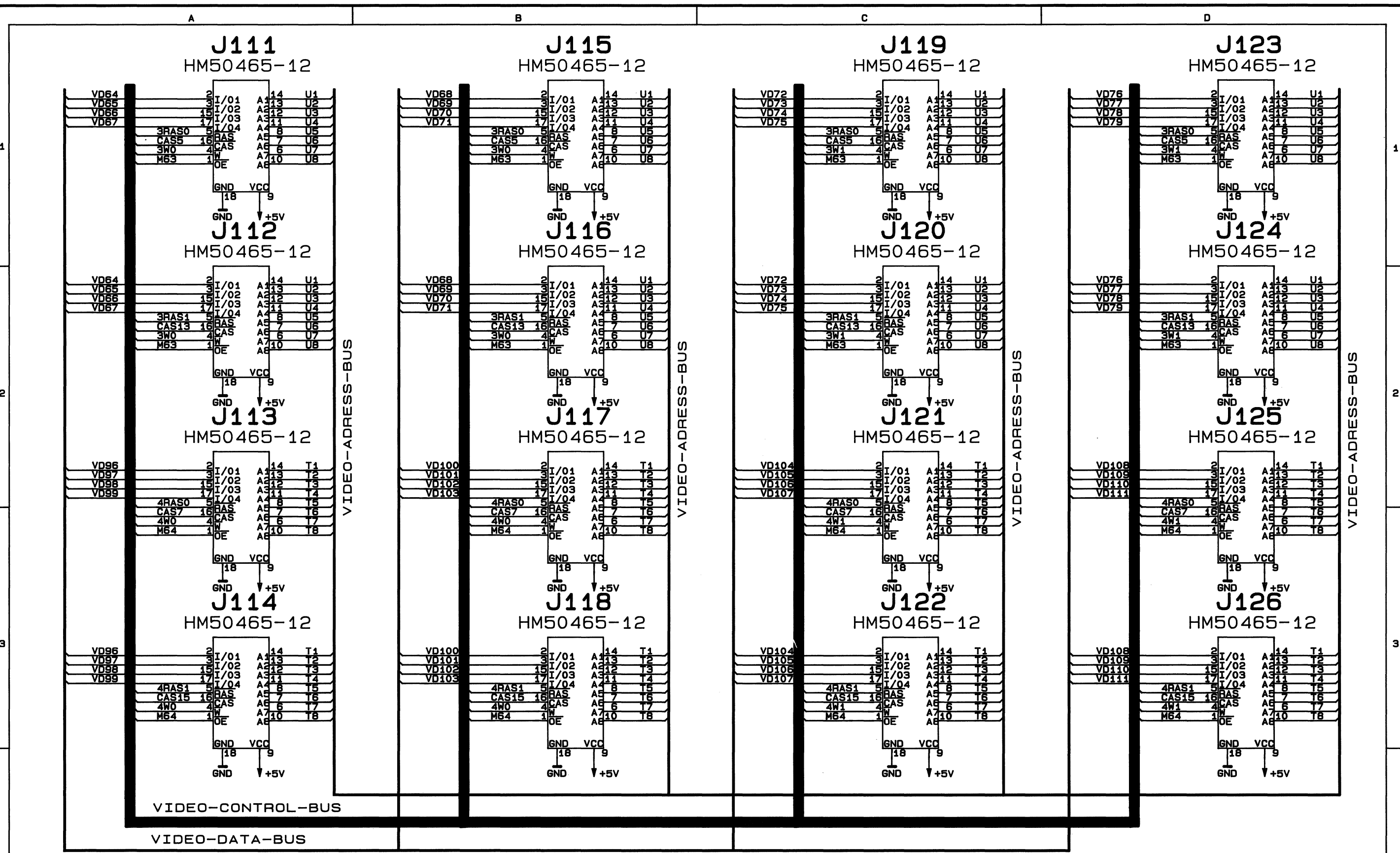
Date : 21.09.1986

014sh7



DATE	NAME		REV. 1
15.9.86	S.E. J.B.	SYS68K/AGC-1B	
FORCE COMPUTERS		GRAPHIC-VIDEO-RAM	SHEET 7
			FROM 11 SHEETS

Date : 21.09.1986 Time : 13:46:15 014sh8

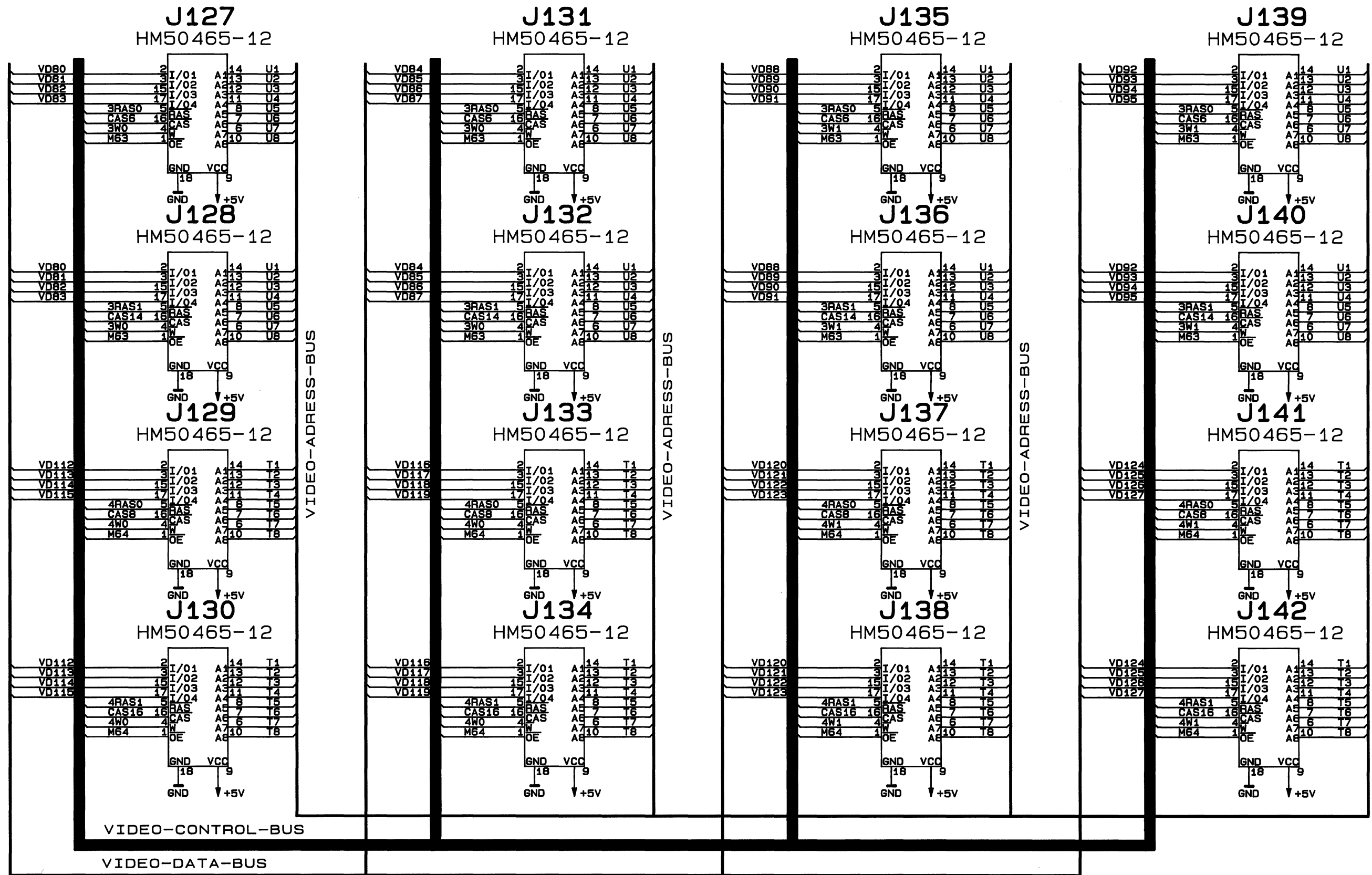


DATE	NAME	SYS68K/AGC-1B	REV. 1
15.9.86	S.E. J.B.		
FORCE COMPUTERS		GRAPHIC-VIDEO-RAM	SHEET 8
			FROM 11 SHEETS

Time : 14: 04: 41

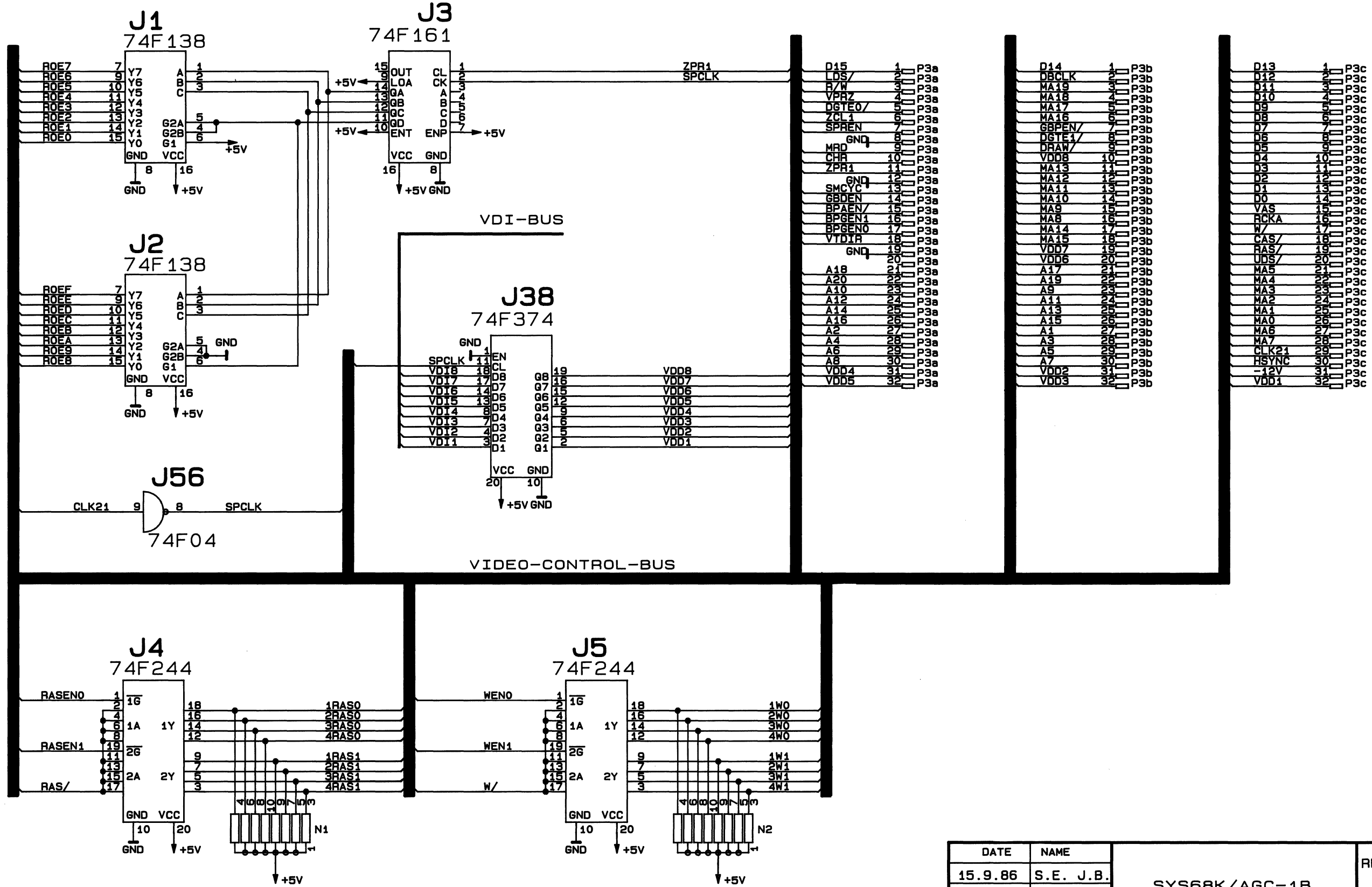
Date : 21.09.1986

014sh9



DATE	NAME	SYS68K/AGC-1B	REV. 1
15.9.86	S.E. J.B.		
FORCE COMPUTERS		GRAPHIC-VIDEO-RAM	SHEET 9
			FROM 11 SHEETS

: 21.09.1986 Time : 14:23:36

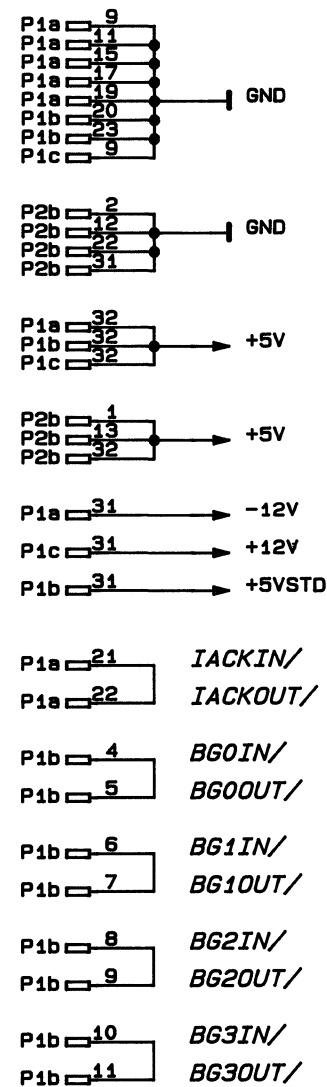
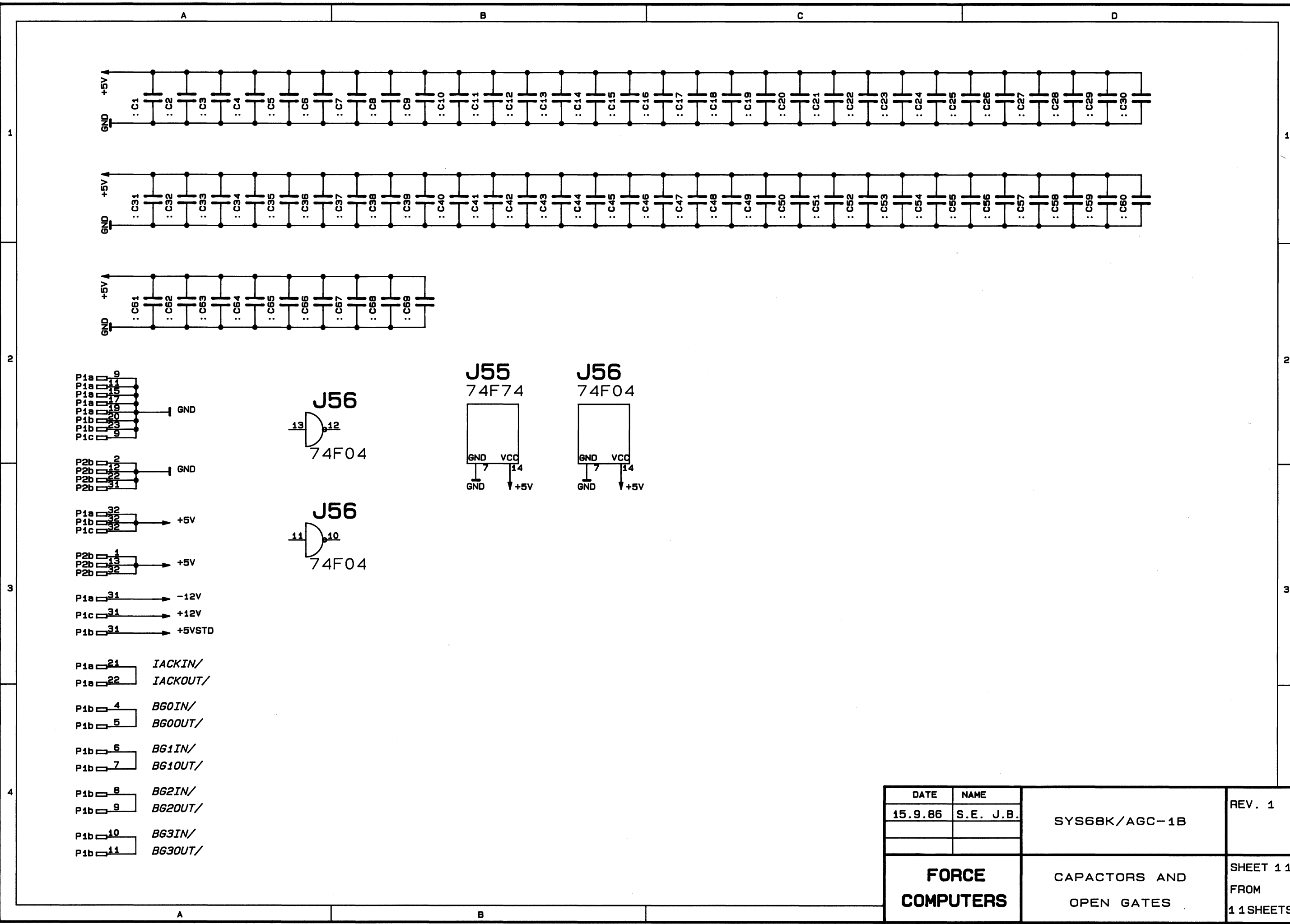


DATE	NAME	SYS68K/AGC-1B	REV. 1
15.9.86	S.E. J.B.		
<b>FORCE COMPUTERS</b>		ACRTC INTERFACE	SHEET 10 FROM 11 SHEETS

Time : 14:38:56

Date : 21.09.1986

014sh11



DATE	NAME	SYS68K/AGC-1B	REV. 1
15.9.86	S.E. J.B.		
<b>FORCE COMPUTERS</b>		CAPACITORS AND OPEN GATES	SHEET 11 FROM 11 SHEETS

APPENDIX E

CONNECTOR PIN ASSIGNMENTS OF THE SYS68K/AGC-1

MASTER BOARD P1

PIN NUMBER	ROW A SIGNAL MNEMONIC	ROW B SIGNAL MNEMONIC	ROW C SIGNAL MNEMONIC
1	D00		D08
2	D01		D09
3	D02		D10
4	D03	/BG0IN	D11
5	D04	/BG0OUT	D12
6	D05	/BG1IN	D13
7	D06	/BG1OUT	D14
8	D07	/BG2IN	D15
9	GND	/BG3IN	GND
10		/BG3OUT	
11	GND		
12	/DS1		/SYSRESET
13	/DS0		/LWORD
14	/WRITE		AM5
15	GND		A23
16	/DTACK	AM0	A22
17	GND	AM1	A21
18	/AS	AM2	A20
19	GND	AM3	A19
20	/IACK	GND	A18
21	/IACKIN		A17
22	/IACKOUT		A16
23	AM4	GND	A15
24	A07	/IRQ7	A14
25	A06	/IRQ6	A13
26	A05	/IRQ5	A12
27	A04	/IRQ4	A11
28	A03	/IRQ3	A10
29	A02	/IRQ2	A09
30	A01	/IRQ1	A08
31	-12V	+5VSTDBY	
32	+5V	+5V	+5V

APPENDIX E

CONNECTOR ASSIGNMENTS OF THE AGC-1

MASTER BOARD P2

PIN NUMBER	ROW A SIGNAL MNEMONIC	ROW B SIGNAL MNEMONIC	ROW C SIGNAL MNEMONIC
1		+5V	
2		GND	
3			
4			
5			
6			
7			
8			
9			
10			
11			
12		GND	
13		+5V	
14			
15			
16			
17			
18			
19			
20			
21			
22		GND	
23			
24			
25			
26			
27			
28			
29			
30			
31		GND	
32		+5V	



APPENDIX E

CONNECTOR ASSIGMENTS OF THE AGC-1

MASTER BOARD P3

PIN NUMBER	ROW A SIGNAL MNEMONIC	ROW B SIGNAL MNEMONIC	ROW C SIGNAL MNEMONIC
1	D15	D14	D13
2	/LDS	DBCLK	D12
3	R/W/	MA19	D11
4	VPRZ	MA18	D10
5	DTGE0/	MA17	D09
6	ZCL1	MA16	D08
7	SPREN	GBPEN/	D07
8	GND	DGTEL/	D06
9	MRD	DRAW/	D05
10	CHR	VDD8	D04
11	ZPR1/	MA13	D03
12	GND	MA12	D02
13	SMCYC	MA11	D01
14	GBDEN	MA10	D00
15	BPAEN/	MA09	VAS
16	BPGEN1	MA08	RCKA
17	BPGEN0	MA14	W/
18	VTDIR	MA15	CAS/
19	GND	VDD7	RAS/
20		VDD6	UDS/
21	A18	A17	MA5
22	A20	A19	MA4
23	A10	A09	MA3
24	A12	A11	MA2
25	A14	A13	MA1
26	A16	A15	MA0
27	A02	A01	MA6
28	A04	A03	MA7
29	A06	A05	CLK 21
30	A08	A07	HSYNC
31	VDD4	VDD2	-12V
32	VDD5	VDD3	VDD1

APPENDIX E

CONNECTOR ASSIGNMENTS OF THE AGC-1

MASTER BOARD P4

PIN NUMBER	SIGNAL MNEMONIC
1	DISP1
2	/HSYNC
3	CLK31
4	2CLK
5	/EXSYNC
6	
7	+5V
8	LIGHT PEN
9	GND
10	64M
11	/64M
12	
13	
14	
15	

APPENDIX E

CONNECTOR ASSIGNMENTS OF THE AGC-1

MASTER BOARD P6

PIN NUMBER	ROW A SIGNAL MNEMONIC	ROW B SIGNAL MNEMONIC	ROW C SIGNAL MNEMONIC
1	IAC1B/	CDACK/	CAT3
2	CGEN/	AM0	AM4
3	CREN/	AM1	R/W1
4	ABEN/	AM2	RUN/
5	AREN/	OWN1/	DREQ0/
6	DMAEN/	RESET/	DTAC1/
7	IRQ1/	LWORD/	DONE/
8	BDEN5/	DERR/	READY/
9	IRQ1B/	CLK3	ATT3
10	DACK0/	DCEN	AS1/
11	DDIR	BR/	DTACK/
12	LAS/	CLK2	MA20
13	CLK31		
14	UDS1/	ATT7	
15		LD4-2	
16	LDS1/	LD4-1	
17	CDC0		
18	CDC1	CAT0	
19		CAT1	
20		CAT2	
21	DSEN1		
22	DISP2		
23			HSYNC/
24		CLK1	
25			
26	/DTAC2		
27			
28			
29			
30			
31		ISIO/	
32		IASIO/	

APPENDIX E

CONNECTOR ASSIGNMENTS OF THE AGC-1

SLAVE BOARD P1

PIN NUMBER	ROW A SIGNAL MNEMONIC	ROW B SIGNAL MNEMONIC	ROW C SIGNAL MNEMONIC
1			
2			
3			
4		BG0IN	
5		BG0OUT	
6		BG1IN	
7		BG1OUT	
8		BG2IN	
9	GND	BG2OUT	GND
10		BG3IN	
11	GND	BG3OUT	
12			
13			
14			
15	GND		
16			
17	GND		
18			
19	GND		
20		GND	
21	IACKIN		
22	IACKOUT		
23		GND	
24			
25			
26			
27			
28			
29			
30			
31	-12V	+5VSTDBY	
32	+5V	+5V	+5V

APPENDIX E

---

CONNECTOR ASSIGNMENTS OF THE AGC-1

SLAVE BOARD P2

PIN NUMBER	ROW A SIGNAL MNEMONIC	ROW B SIGNAL MNEMONIC	ROW C SIGNAL MNEMONIC
1		+5V	
2		GND	
3			
4			
5			
6			
7			
8			
9			
10			
11			
12		GND	
13		+5V	
14			
15			
16			
17			
18			
19			
20			
21			
22		GND	
23			
24			
25			
26			
27			
28			
29			
30			
31		GND	
32		+5V	

APPENDIX E

CONNECTOR ASSIGNMENTS OF THE AGC-1

SLAVE BOARD P3

PIN NUMBER	ROW A SIGNAL MNEMONIC	ROW B SIGNAL MNEMONIC	ROW C SIGNAL MNEMONIC
1	D15	D14	D13
2	/LDS	DBCLK	D12
3	R/W/	MA19	D11
4	VPRZ	MA18	D10
5	DTGE0/	MA17	D09
6	ZCL1	MA16	D08
7	SPREN	GBPEN/	D07
8	GND	DGTEL/	D06
9	MRD	DRAW/	D05
10	CHR	VDD8	D04
11	ZPR1/	MA13	D03
12	GND	MA12	D02
13	SMCYC	MA11	D01
14	GBDEN	MA10	D00
15	BPAEN/	MA09	VAS
16	BPGEN1	MA08	RCKA
17	BPGEN0	MA14	W/
18	VTDIR	MA15	CAS/
19	GND	VDD7	RAS/
20		VDD6	UDS/
21	A18	A17	MA5
22	A20	A19	MA4
23	A10	A09	MA3
24	A12	A11	MA2
25	A14	A13	MA1
26	A16	A15	MA0
27	A02	A01	MA6
28	A04	A03	MA7
29	A06	A05	CLK21
30	A08	A07	HSYNC
31	VDD4	VDD2	-12V
32	VDD5	VDD3	VDD1

APPENDIX F

COMPONENT PART LIST SYS68K/AGC-1A

ICs

Location	Type	Manufacturer
J1	74ALS645-1	TI
J2	74ALS645-1	TI
J3	74ALS645-1	TI
J4	74ALS645-1	TI
J5	74ALS641-1	TI
J6	74LS682	TI, MOT
J7	74LS283	TI, MOT
J8	74LS283	TI, MOT
J9	74LS682	TI, MOT
J10	74LS125A	TI, MOT
J11	74F08	MOT, FAIR, VALVO
J12	74LS04	TI, MOT
J13	74LS74	TI, MOT
J14	74LS32	TI, MOT
J15	74F08	MOT, FAIR, VALVO
J16	74ALS645-1	TI
J17	PAL 16L8A	NAT, MMI
J18	PAL 20L10CN	NAT, MMI
J19	PAL 20L10CN	NAT, MMI
J20	74LS112A	TI, MOT
J21	74AS646	TI
J22	74AS646	TI
J23	74S244	TI, MOT

COMPONENT PART LIST SYS68K/AGC-1A

ICs

Location	Type	Manufacturer
J24	PAL 20L8A	NAT, MMI
J25	74F74	MOT, FAIR, VALVO
J26	74F02	MOT, FAIR, VALVO
J27	74LS08	MOT, TI
J28	74F04	MOT, FAIR, VALVO
J29	PAL 20L8A	NAT, MMI
J30	74F164	MOT, FAIR, VALVO
J31	68153 BIM	MOT
J32	74LS21	TI, MOT
J33	74LS139	TI, MOT
J34	74LS112A	TI, MOT
J35	74LS08	TI, MOT
J36	74LS85	TI, MOT
J37	74LS74	TI, MOT
J38	7406	TI, MOT
J39	74LS195A	TI, MOT
J40	74F32	MOT, FAIR, VALVO
J41	74F08	MOT, FAIR, VALVO
J42	AM 8151 GCP	AMD
J43	AM 8151 GCP	AMD
J44	AM 8151 GCP	AMD
J45	74F244	MOT, FAIR, VALVO



COMPONENT PART LIST SYS68K/AGC-1A

ICs

Location	Type	Manufacturer
J46	74F244	MOT, FAIR, VALVO
J47	74F245	MOT, FAIR, VALVO
J48	74F245	MOT, FAIR, VALVO
J49	PAL 16L8A	NAT, MMI
J50	PAL 16R4A	NAT, MMI
J51	74F245	MOT, FAIR, VALVO
J52	74F175	MOT, FAIR, VALVO
J53	74F257	MOT, FAIR, VALVO
J56	MC 7905	VARIOUS
J57	MC 7905	VARIOUS
J58	MC 7905	VARIOUS
J59	MC 7905	VARIOUS
J60	HD63484 ACRTC	HITACHI
J61	74LS125A	TI, MOT
J62	74ALS645-1	TI
J63	74F04	MOT, FAIR, VALVO
J64	74F241	MOT, FAIR, VALVO
J65	74F163	MOT, FAIR, VALVO
J66	74LS373	TI, MOT
J67	74ALS374	TI
J68	74LS373	TI, MOT
J69	74ALS374	TI

COMPONENT PART LIST SYS68K/AGC-1A

ICs

Location	Type	Manufacturer
J70	74LS375	TI, MOT
J71	74F157	MOT, FAIR, VALVO
J72	F10016	FAIR
J73	F10124	FAIR
J74	74F164	MOT, FAIR, VALVO
J75	F10124	FAIR
J76	74F74	MOT, FAIR, VALVO
J77	MC10125	FAIR, MOT
J78	MC10105	FAIR, MOT
J79	MC10104	FAIR, MOT
J80	74F74	MOT, FAIR, VALVO
J81	74F153	MOT, FAIR, VALVO
J82	PAL 16L8A	NAT, MMI
J83	74F164	MOT, FAIR, VALVO
J84	74F251	MOT, FAIR, VALVO
J85	74F251	MOT, FAIR, VALVO
J86	74F164	MOT, FAIR, VALVO
J87	74F164	MOT, FAIR, VALVO
J88	74F251	MOT, FAIR, VALVO
J89	74F251	MOT, FAIR, VALVO
J90	74F164	MOT, FAIR, VALVO
J91	74F74	MOT, FAIR, VALVO

COMPONENT PART LIST SYS68K/AGC-1A

ICs

Location	Type	Manufacturer
J92	74F74	MOT, FAIR, VALVO
J93	74F164	MOT, FAIR, VALVO
J94	74F164	MOT, FAIR, VALVO
J95	74F164	MOT, FAIR, VALVO
J96	74F86	MOT, FAIR, VALVO
J97	74F04	MOT, FAIR, VALVO

COMPONENT PART LIST SYS68K/AGC-1A

RESISTOR NETWORKS

Location	Type	Manufacturer
N1	9 * 3.3K OHM	VARIOUS
N2	9 * 3.3K OHM	VARIOUS
N3	9 * 3.3K OHM	VARIOUS
N4	9 * 3.3K OHM	VARIOUS
N5	9 * 3.3K OHM	VARIOUS
N6	9 * 3.3K OHM	VARIOUS
N7	9 * 100 OHM	VARIOUS
N8	9 * 330 OHM	VARIOUS
N9	9 * 150 OHM	VARIOUS
N10	9 * 3.3K OHM	VARIOUS
N11	9 * 150 OHM	VARIOUS
N12	9 * 270 OHM	VARIOUS
N13	9 * 270 OHM	VARIOUS
N14	9 * 150 OHM	VARIOUS
E1	5 * 27 OHM	10 PIN SLIM LINE
E2	5 * 27 OHM	10 PIN SLIM LINE
E3	5 * 27 OHM	10 PIN SLIM LINE
E4	5 * 27 OHM	10 PIN SLIM LINE

COMPONENT PART LIST SYS68K/AGC-1A

RESISTORS

Location	Type	Manufacturer
R1	2.2K OHM	VARIOUS
R2	2.2K OHM	VARIOUS
R3	2.0K OHM 1%	VARIOUS
R4	2.0K OHM 1%	VARIOUS
R5	2.0K OHM 1%	VARIOUS
R6	27 OHM	VARIOUS
R7	27 OHM	VARIOUS
R8	330 OHM	VARIOUS
R9	220 OHM	VARIOUS
R10	680 OHM	VARIOUS
R11	1K OHM	VARIOUS
R12	1K OHM	VARIOUS
R13	68 OHM	VARIOUS

Note: All resistors are type RGU mini 2.54 RM.

COMPONENT PART LIST SYS68K/AGC-1A

CAPACITORS

Location	Type	Manufacturer
:C1 - :C76	100 nF KER	VARIOUS
C0	ELKO 220uF/6V 5.08mm RM.	VARIOUS
C1	47 pF	VARIOUS
C2	470 pF + 10%	VARIOUS
C3 - C14	100 nF	VARIOUS
C15 - C16	10 pF + 10%	VARIOUS
C17	ELKO 100uF/>12V 5.08mm RM.	VARIOUS
C18 - C21	10 uF 6.3 V TAN	VARIOUS
C22	220 pF + 10%	VARIOUS
C23	470 pF + 10%	VARIOUS
C24	1nF	VARIOUS

CRYSTALS

Location	Type	Manufacturer
Q1	16.000 MHz	SE, JAUCH
Q2	64.000 MHz	SE, JAUCH

COMPONENT PART LIST SYS68K/AGC-1A

DIODES

Location	Type	Manufacturer
LD1	550 - 2206	LED GREEN
LD2	550 - 2406	LED RED
LD3	550 - 2306	LED YELLOW
D1	1N 4148	DIODE
T1	2N 2905	TRANSISTOR
T2	2N 2222	TRANSISTOR

MECHANICAL PARTS

Location	Type	Manufacturer
J17	20 PIN SOCKET	VARIOUS
J18 - J19	24 PIN SLIM SOCKET	VARIOUS
J24	24 PIN SLIM SOCKET	VARIOUS
J29	24 PIN SLIM SOCKET	VARIOUS
J31	40 PIN SLIM SOCKET	VARIOUS
J42 - J44	40 PIN SLIM SOCKET	VARIOUS
J49 - J50	20 PIN SOCKET	VARIOUS
J60	64 PIN SLIM SOCKET	VARIOUS
J82	20 PIN SOCKET	VARIOUS

COMPONENT PART LIST SYS68K/AGC-1A

MECHANICAL PARTS

Location	Type	Manufacturer
P1	VG MALE CONNECTOR 96 PIN 90 DEGREE 2 x SCREW 2.5/10 2 x MOTHER 2.5	VARIOUS
P2	VG MALE CONNECTOR 96 PIN 90 DEGREE 2 x SCREW 2.5/10 2 x MOTHER 2.5	VARIOUS
P3	FORCE 983040	VARIOUS
P3	FORCE 983041	VARIOUS
P4	D FEMALE CONNECTOR 15 PIN 90 DEGREE 2 PLATED SCREW 2.9/9.5mm	VARIOUS
P6	VG FEMALE CONNECTOR 96 PIN Nr 983064 2 x SCREW 2.5/10 2 x MOTHER 2.5	VARIOUS (without wire wrap)
B1	DW 8	VARIOUS
B2	DW 8	VARIOUS
B3	DW 12	VARIOUS
B4	DW 16	VARIOUS
B5	DW 10	VARIOUS
B6	DW 10	VARIOUS
B7	EW 2	VARIOUS



COMPONENT PART LIST SYS68K/AGC-1A

MECHANICAL PARTS

Location	Type	Manufacturer
B8	TW 15	VARIOUS
B9	EW 3	VARIOUS
B10	EW 3	VARIOUS
B11	DW 4	VARIOUS
B12	EW 3	VARIOUS
B13	DW 14	VARIOUS
B14	DW 16	VARIOUS
B15	TW 24	VARIOUS
B16	DW 8	VARIOUS
B17	TW 21	VARIOUS
B18	DW 8	VARIOUS
BNC1	BNC- CONNECTOR	WITH ISOLATED MOUNTING
BNC2	BNC- CONNECTOR	WITH ISOLATED MOUNTING
BNC3	BNC- CONNECTOR	WITH ISOLATED MOUNTING
BNC4	BNC- CONNECTOR	WITH ISOLATED MOUNTING
Frontpanel	FRBL AGC-1A	VARIOUS
	PC-Board AGC-1A	VARIOUS
Cooling Panel	Specially designed	VARIOUS
SW1	Switch ATE-1D-RA	KNITTER
SW2	Switch ATE-1D-RA	KNITTER
Bx	JUMPER (41)	VARIOUS

COMPONENT PART LIST SYS68K/AGC-1B

ICs

Location	Type	Manufacturer
J1	74F138	MOT, FAIR, VALVO
J2	74F138	MOT, FAIR, VALVO
J3	74F161	MOT, FAIR, VALVO
J4	74F244	MOT, FAIR, VALVO
J5	74F244	MOT, FAIR, VALVO
J6	74LS374	TI, MOT
J7	74F373	MOT, FAIR, VALVO
J8	74LS374	TI, MOT
J9	74F374	MOT, FAIR, VALVO
J10	74LS374	MOT, FAIR, VALVO
J11	74F374	MOT, FAIR, VALVO
J12	74LS374	TI, MOT
J13	74F374	MOT, FAIR, VALVO
J14	74LS374	MOT, FAIR, VALVO
J15	74F374	MOT, FAIR, VALVO
J16	74LS374	MOT, FAIR, VALVO
J17	74F374	MOT, FAIR, VALVO
J18	74LS374	MOT, FAIR, VALVO
J19	74F374	MOT, FAIR, VALVO
J20	74LS374	MOT, FAIR, VALVO
J21	74F374	MOT, FAIR, VALVO
J22	74LS374	MOT, FAIR, VALVO
J23	74F374	MOT, FAIR, VALVO

COMPONENT PART LIST SYS68K/AGC-1B

ICs

Location	Type	Manufacturer
J24	74LS374	TI, MOT
J25	74F374	MOT, FAIR, VALVO
J26	74LS374	MOT, FAIR, VALVO
J27	74F374	MOT, FAIR, VALVO
J28	74LS374	MOT, FAIR, VALVO
J29	74F374	MOT, FAIR, VALVO
J30	74LS374	MOT, FAIR, VALVO
J31	74F374	MOT, FAIR, VALVO
J32	74LS374	MOT, FAIR, VALVO
J33	74F374	MOT, FAIR, VALVO
J34	74LS374	MOT, FAIR, VALVO
J35	74F374	MOT, FAIR, VALVO
J36	74LS374	MOT, FAIR, VALVO
J37	74F374	MOT, FAIR, VALVO
J38	74F374	MOT, FAIR, VALVO
J39	74ALS645-1	TI
J40	74ALS645-1	TI
J41	74ALS645-1	TI
J42	74ALS645-1	TI
J43	74ALS645-1	TI
J45	74ALS645-1	TI
J44	74ALS645-1	TI
J46	74ALS645-1	TI
J47	74ALS645-1	TI

COMPONENT PART LIST SYS68K/AGC-1B

ICs

Location	Type	Manufacturer
J48	74ALS645-1	TI
J49	74ALS645-1	TI
J50	74ALS645-1	TI
J51	74ALS645-1	TI
J52	74ALS645-1	TI
J53	74ALS645-1	TI
J54	74ALS645-1	TI
J55	74F74	MOT, FAIR, VALVO
J56	74F04	MOT, FAIR, VALVO
J57	74LS244	TI
J58	74F245	MOT, FAIR, VALVO
J59	74F245	MOT, FAIR, VALVO
J60	74AS646	TI
J61	74AS646	TI
J62	74F373	MOT, FAIR, VALVO
J63	74F157	MOT, FAIR, VALVO
J64	74F138	MOT, FAIR, VALVO
J65	PAL 20L8A	MMI, NS
J66	PAL 20L8A	MMI, NS
J67	AM2966	AMD
J68	AM2966	AMD
J69	74F373	MOT, FAIR, VALVO

COMPONENT PART LIST SYS68K/AGC-1B

ICs

Location	Type	Manufacturer
J70	74F157	MOT, FAIR, VALVO
J71	74F138	MOT, FAIR, VALVO
J72	PAL 20L8A	MMI, NS
J73	74F244	MOT, FAIR, VALVO
J74	AM 2966	AMD
J75	AM 2966	AMD
J76	74F373	MOT, FAIR, VALVO
J77	74F245	MOT, FAIR, VALVO
J78	74F245	MOT, FAIR, VALVO
J79	41464P-12	HIT
J80	41464P-12	HIT, NEC
J81	41464P-12	HIT, NEC
J82	41464P-12	HIT, NEC
J83	41464P-12	HIT, NEC
J84	41464P-12	HIT, NEC
J85	41464P-12	HIT, NEC
J86	41464P-12	HIT, NEC
J87	41464P-12	HIT, NEC
J88	41464P-12	HIT, NEC
J89	41464P-12	HIT, NEC
J90	41464P-12	HIT, NEC
J91	41464P-12	HIT, NEC

COMPONENT PART LIST SYS68K/AGC-1B

ICs

Location	Type	Manufacturer
J92	41464P-12	HIT, NEC
J93	41464P-12	HIT, NEC
J94	41464P-12	HIT, NEC
J95	41464P-12	HIT, NEC
J96	41464P-12	HIT, NEC
J97	41464P-12	HIT, NEC
J98	41464P-12	HIT, NEC
J98	41464P-12	HIT, NEC
J99	41464P-12	HIT, NEC
J100	41464P-12	HIT, NEC
J101	41464P-12	HIT, NEC
J102	41464P-12	HIT, NEC
J103	41464P-12	HIT, NEC
J104	41464P-12	HIT, NEC
J105	41464P-12	HIT, NEC
J106	41464P-12	HIT, NEC
J107	41464P-12	HIT, NEC
J108	41464P-12	HIT, NEC
J109	41464P-12	HIT, NEC
J110	41464P-12	HIT, NEC
J111	41464P-12	HIT, NEC
J112	41464P-12	HIT, NEC

COMPONENT PART LIST SYS68K/AGC-1B

ICs

Location	Type	Manufacturer
J113	41464P-12	HIT, NEC
J114	41464P-12	HIT, NEC
J115	41464P-12	HIT, NEC
J116	41464P-12	HIT, NEC
J117	41464P-12	HIT, NEC
J118	41464P-12	HIT, NEC
J119	41464P-12	HIT, NEC
J120	41464P-12	HIT, NEC
J121	41464P-12	HIT, NEC
J122	41464P-12	HIT, NEC
J123	41464P-12	HIT, NEC
J124	41464P-12	HIT, NEC
J125	41464P-12	HIT, NEC
J126	41464P-12	HIT, NEC
J127	41464P-12	HIT, NEC
J128	41464P-12	HIT, NEC
J129	41464P-12	HIT, NEC
J130	41464P-12	HIT, NEC
J131	41464P-12	HIT, NEC
J132	41464P-12	HIT, NEC
J133	41464P-12	HIT, NEC
J134	41464P-12	HIT, NEC

COMPONENT PART LIST SYS68K/AGC-1B

ICs

Location	Type	Manufacturer
J135	41464P-12	HIT, NEC
J136	41464P-12	HIT, NEC
J137	41464P-12	HIT, NEC
J138	41464P-12	HIT, NEC
J139	41464P-12	HIT, NEC
J140	41464P-12	HIT, NEC
J141	41464P-12	HIT, NEC
J142	41464P-12	HIT, NEC



COMPONENT PART LIST SYS68K/AGC-1B

RESISTOR NETWORKS

Location	Type	Manufacturer
N1	9 * 3.3K OHM	VARIOUS
N2	9 * 3.3K OHM	VARIOUS
N3	9 * 470 OHM	VARIOUS

CAPACITORS

Location	Type	Manufacturer
C1 - C69	100 nF KER	VARIOUS

COMPONENT PART LIST SYS68K/AGC-1B

MECHANICAL PARTS

Location	Type	Manufacturer
J1 - J3	16 PIN SOCKET	VARIOUS
J4 - J37	20 PIN STACKED DIP SOCKET	T & B
J38	20 PIN SOCKET	VARIOUS
J39 - J54	20 PIN STACKED DIP SOCKET	T & B
J55 - J56	14 PIN SOCKET	VARIOUS
J57 - J59	20 PIN STACKED DIP SOCKET	T & B
J60 - J61	24 PIN SLIM SOCKET	VARIOUS
J62 - J63	20 PIN STACKED DIP SOCKET	T & B
J64	16 PIN SOCKET	VARIOUS
J65 - J66	24 PIN SLIM SOCKET	VARIOUS
J67 - J69	20 PIN STACKED DIP SOCKET	T & B
J70 - J71	16 PIN SOCKET	VARIOUS
J72	24 PIN SLIM SOCKET	VARIOUS
J73 - J142	20 PIN STACKED DIP SOCKET	T & B
20 PIN STACKED DIP SOCKETS TOTAL OF 64 PIECES		

COMPONENT PART LIST SYS68K/AGC-1B

MECHANICAL PARTS

Location	Type	Manufacturer
P1	VG MALE CONNECTOR 2 x SCREW 2.5/10 2 x MOTHER 2.5	FORCE 983039
P2	VG MALE CONNECTOR 2 x SCREW 2.5/10 2 x MOTHER 2.5	FORCE 983038
P3	VG FEMALE CONNECTOR 96 PIN Nr 983064 2 x SCREW 2.5/10 2 x MOTHER 2.5	VARIOUS  (without wire wrap)
B1 - B2	TW 9	VARIOUS
Frontpanel	FRBL AGC-1B  PC-Board AGC-1B  JUMPER 3x	VARIOUS  VARIOUS



APPENDIX G

LITERATURE REFERENCE

Please refer to the following books for further more detailed information.

- 1) User`s manual of the 63484 ACRTC, including description of the instructions - M-01-85--1.ACRTC(680-1-31) HITACHI
- 3) HD63484 ACRTC APPLICATION NOTE I 680-3-08 HITACHI
- 4) HD63484 ACRTC APPLICATION NOTE II 680-3-07 HITACHI
- 5) VMEbus specifications - 2618 S Shannon  
Tempe Arizona 85282  
(602) 966-5936



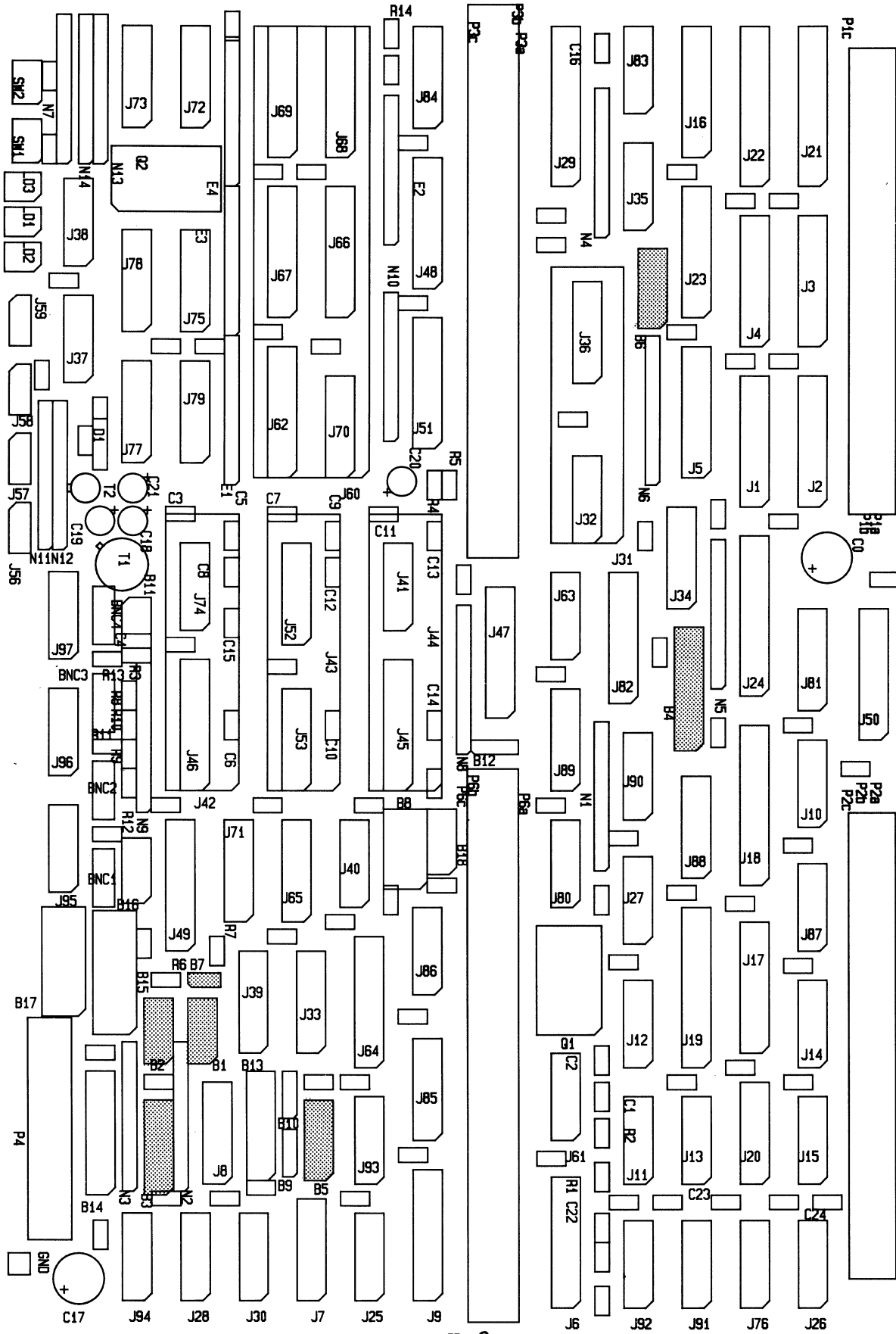
APPENDIX H

DEFAULT JUMPER SETTINGS ON THE SYS68K/AGC-1 BOARDS

Description	Jumpers	Default	Schematics	See Page
Board Size Selection	B1	1-8 2-7 3 6 4-5	1-C2	1-19
		B2		
Base Address Selection	B3	1-12 2-11 3-10 4-9 5 8 6 7	1-D1	1-9
Address Modifier Selection	B4	1-16 2-15 3-14 4 13 5 12 6-11 7 10 8-9	2-C3	1-26
Access-Synchronisation Setting	B5	1-10 2 9 3 8 4 7 5 6	3-A3	--
Short I/O-Access Selection	B6	1-10 2-9 3-8 4-7 5 6	3-C3	1-15
CSYNC Polarity	B7	1-2	4-C1	1-127

APPENDIX H

Jumper Location Diagram of Jumpers B1 - B7





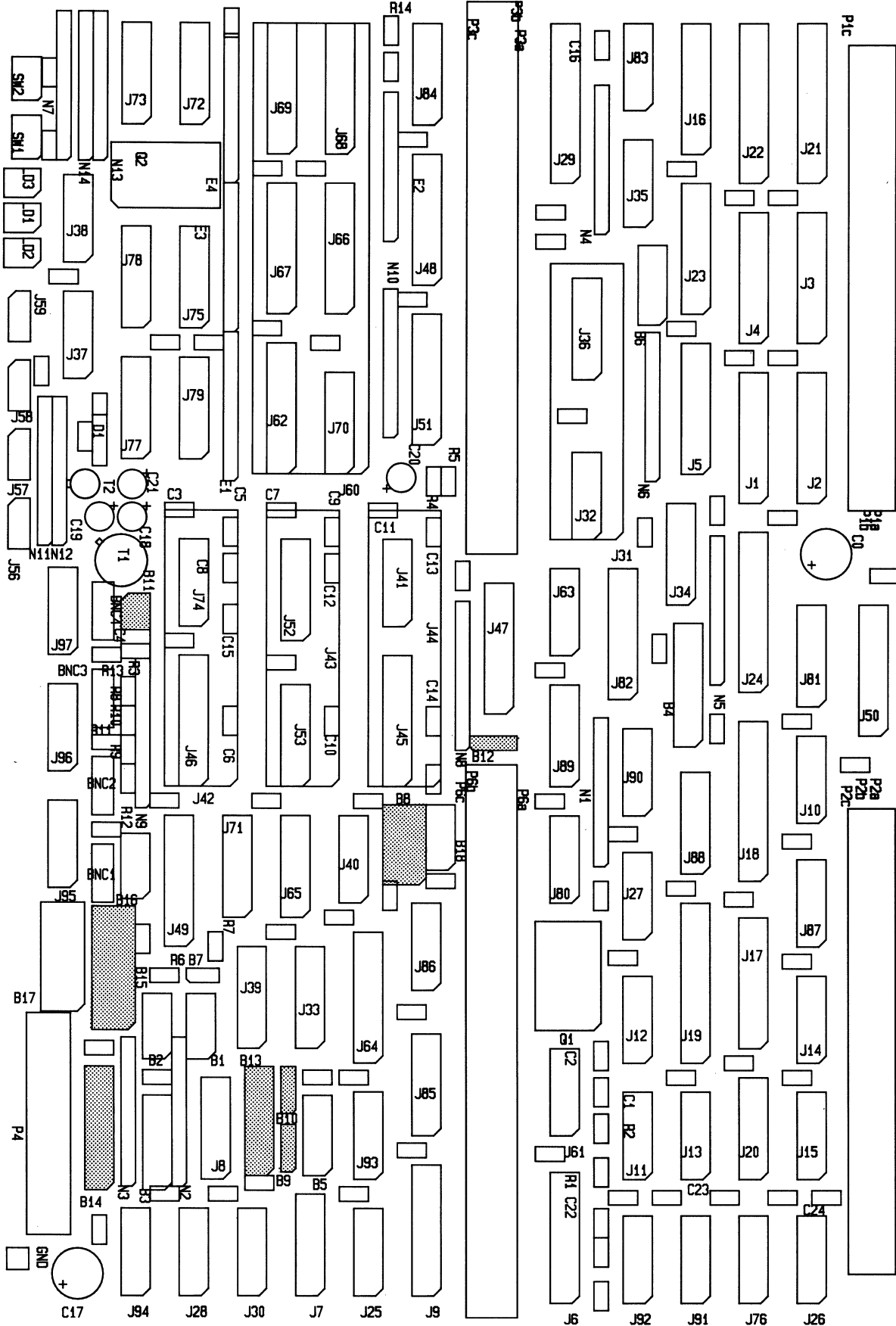
APPENDIX H

DEFAULT JUMPER SETTINGS ON THE SYS68K/AGC-1

Description	Jumpers	Default	Schematics	See Page
Blink Switch Mode	B8	1 2-3 4 5-6 6 7 8 9 10 11	4-D2	1-39
Light Pen Strobe Polarity	B9	1-2 3	5-A3	1-123
Master/Slave Mode	B10	1-2 3	5-A4	1-126
	B11	1-2 3-4	6-A1	1-126
Scroll Clock	B12	1-2 3	7-C1	1-120
BLANK Delay	B13	1 14 2 13 3 12 4 11 5-10 6 9 7 8	7-C1	--
RAS - Timing	B14	1 16 2 15 3 14 4 13 5 12 6 11 7 10 8 9	7-C2	--
CAS - Timing	B15	1 2 3 4 5 6 7 8 9 10 11-12 13-14 15 16 17 18 19 20 21 22 23 24	7-C3	--

APPENDIX H

Jumper Location Diagram of Jumpers B8 - B15

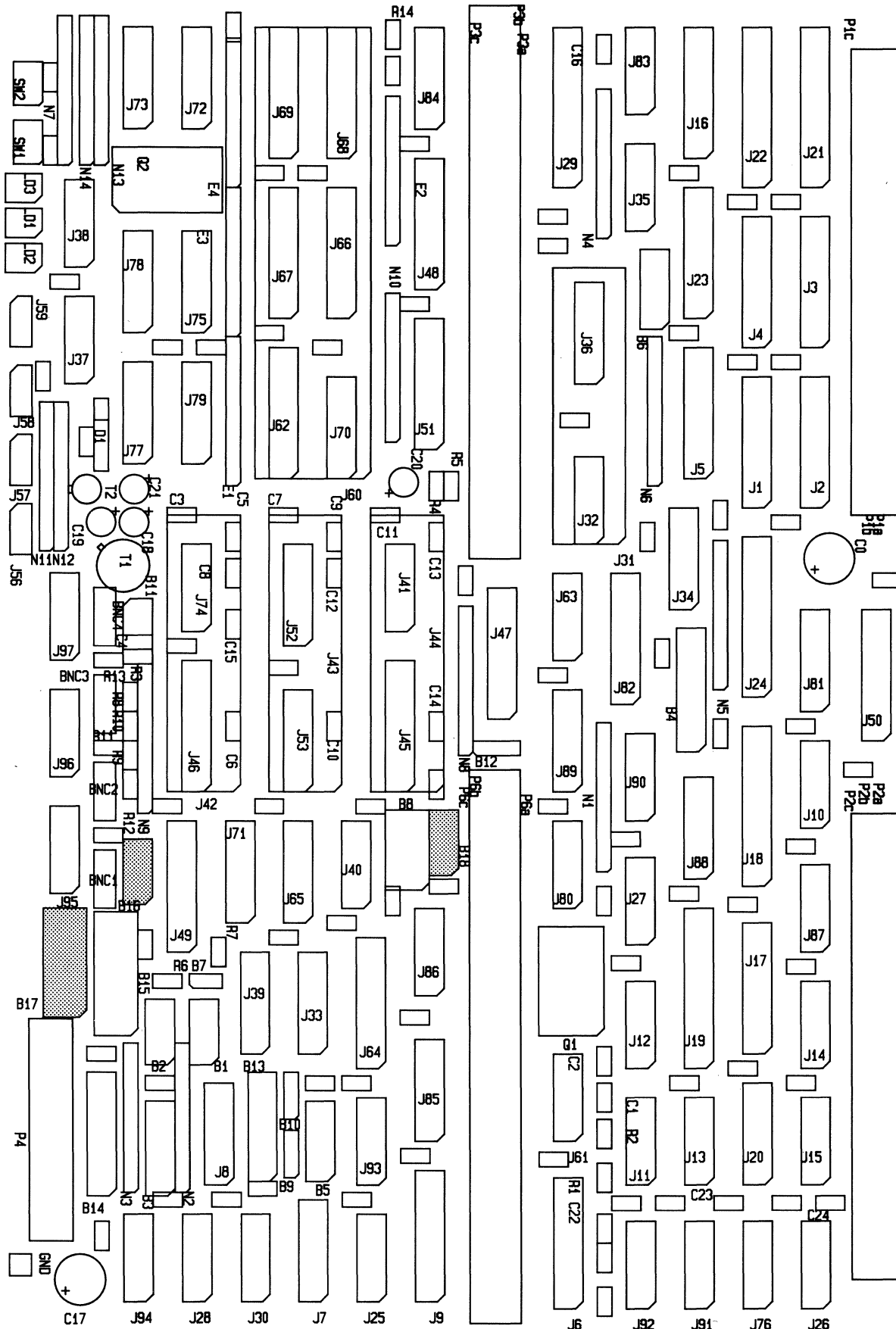


APPENDIX H  
DEFAULT JUMPER SETTINGS ON THE SYS68K/AGC-1

Description	Jumpers	Default	Schematics	See Page
Register Timing	B16	1-8 2 7 3 6 4 5	7-D3	--
WRITE - Timing	B17	1 2 3 4 5 6 7 8 9 10-11 12 13 14 15 16 17 18 19 20-21	7-D3	--
Switch Colour Mode	B18	1 8 2 7 3-6 4-5	4-C3	1-36

APPENDIX H

Jumper Location Diagram of Jumpers B16 - B18



# THE GRAPHIC COMMAND SUMMARY

FORCE COMPUTERS Inc./GmbH  
All Rights Reserved

This document shall not be duplicated, nor its contents used  
for any purpose, unless express permission has been granted.

Copyright by FORCE Computers®



# T A B L E O F C O N T E N T S

## GRAPHIC COMMANDS SUMMARY

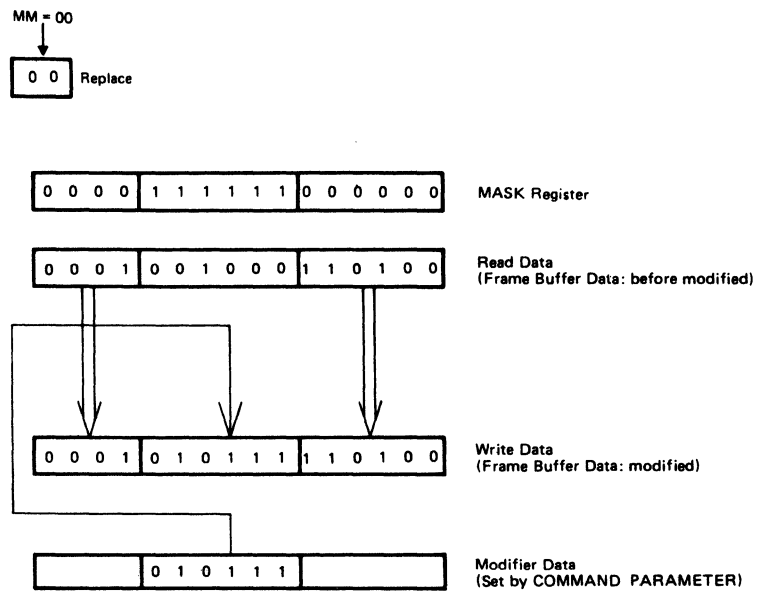
1.	Modify Mode Examples.....	1-1
2.	Operation Mode Examples.....	2-1
3.	Colour Mode Examples.....	3-1
4.	Area Mode Examples.....	4-1
5.	Graphic Commands.....	5-1





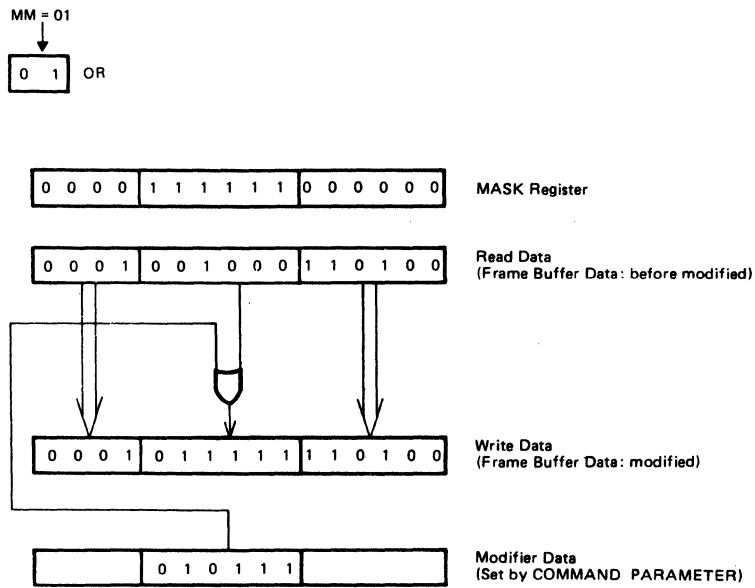
# GRAPHIC COMMANDS SUMMARY

## 1. Modify Mode



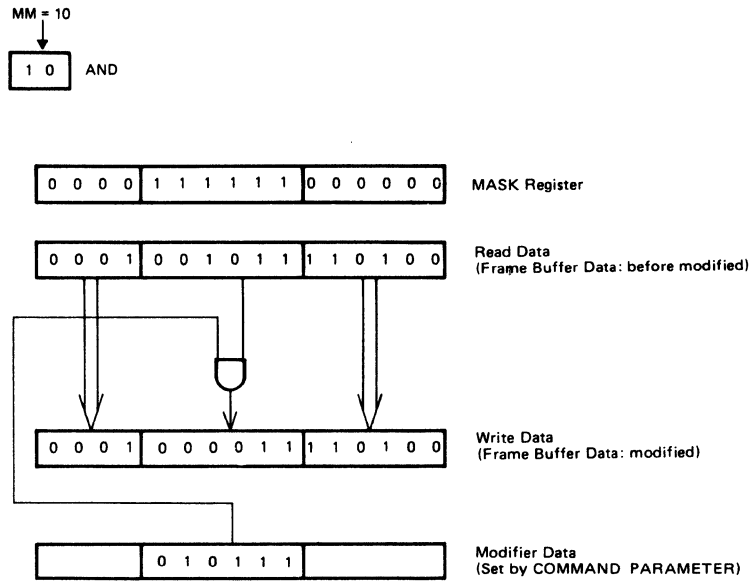
The read data bit positions for which the MASK register contains '1' is REPLACED with the command parameter modifier data. The result is rewritten to the read data location in the frame buffer.

Figure 6.5(a) REPLACE Modify Mode



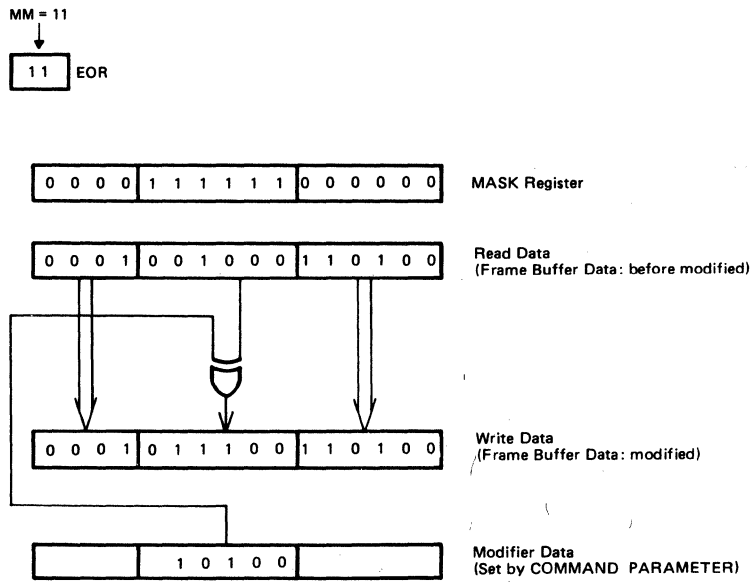
The read data bit positions for which the MASK register contains '1' is ORed with the command parameter modifier data. The result is rewritten to the read data location in the frame buffer.

**Figure 6.5(b) OR Modify Mode**



The read data bit positions for which the MASK register contains '1' is ANDed with the command parameter modifier data. The result is rewritten to the read data location in the frame buffer.

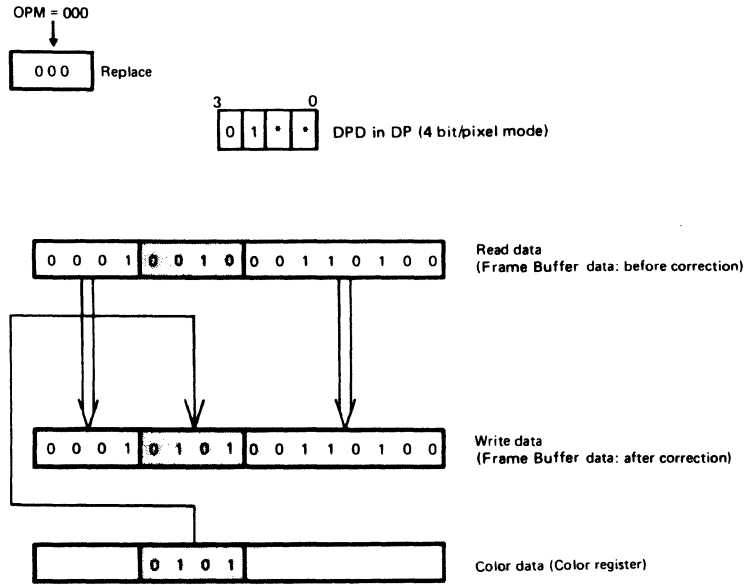
**Figure 6.5(c) AND Modify Mode**



The read data bit positions for which the MASK register contains '1' is EORed with the command parameter modifier data. The result is rewritten to the read data location in the frame buffer.

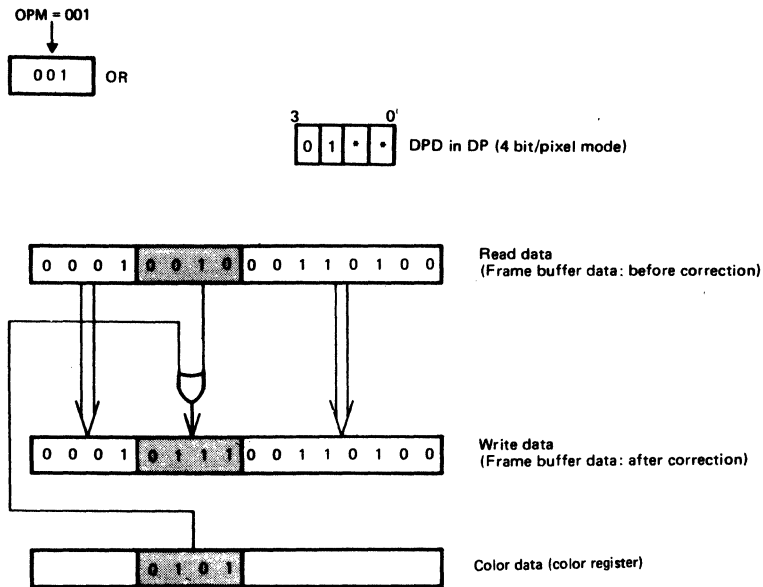
**Figure 6.5(d) EOR Modify Mode**

## 2. Operation Mode Examples



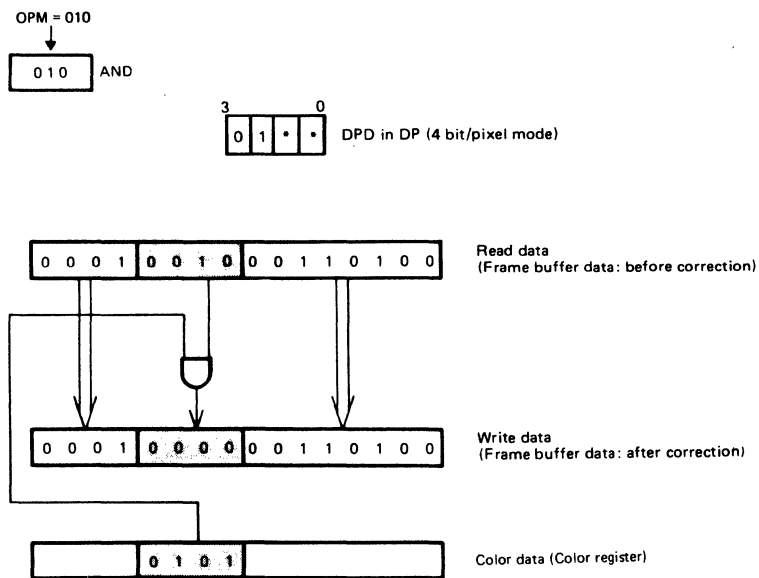
One pixel of the frame buffer read data is REPLACED with the corresponding color register data and the result is rewritten to the frame buffer read data location. The dot pointer serves to extract the pixel from the frame buffer word — in this example, 4 bits/pixel.

**Figure 6.9(a) REPLACE Operation Mode**



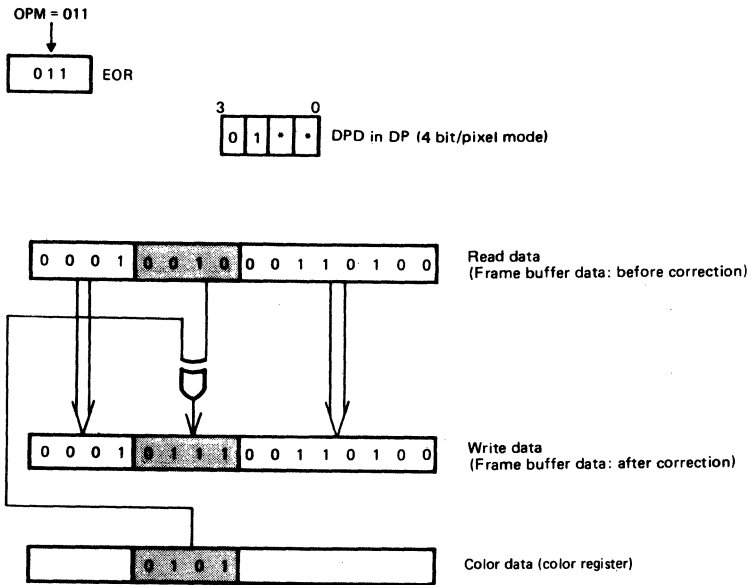
One pixel of the frame buffer read data is ORed with the corresponding color register data and the result is rewritten to the frame buffer read data location. The dot pointer serves to extract the pixel from the frame buffer word — in this example, 4 bits/pixel.

**Figure 6.9(b) OR Operation Mode**



One pixel of the frame buffer read data is ANDed with the corresponding color register data and the result is rewritten to the frame buffer read data location. The dot pointer serves to extract the pixel from the frame buffer word – in this example, 4 bits/pixel.

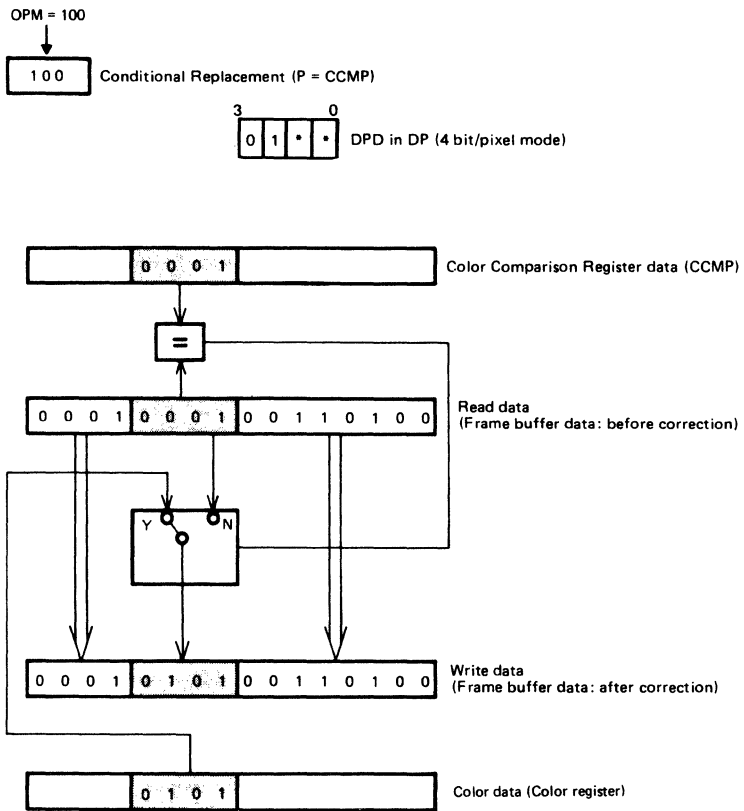
**Figure 6.9(c) AND Operation Mode**



One pixel of the frame buffer read data is EORed with the corresponding color register data and the result is rewritten to the frame buffer read data location. The dot pointer serves to extract the pixel from the frame buffer word — in this example, 4 bits/pixel.

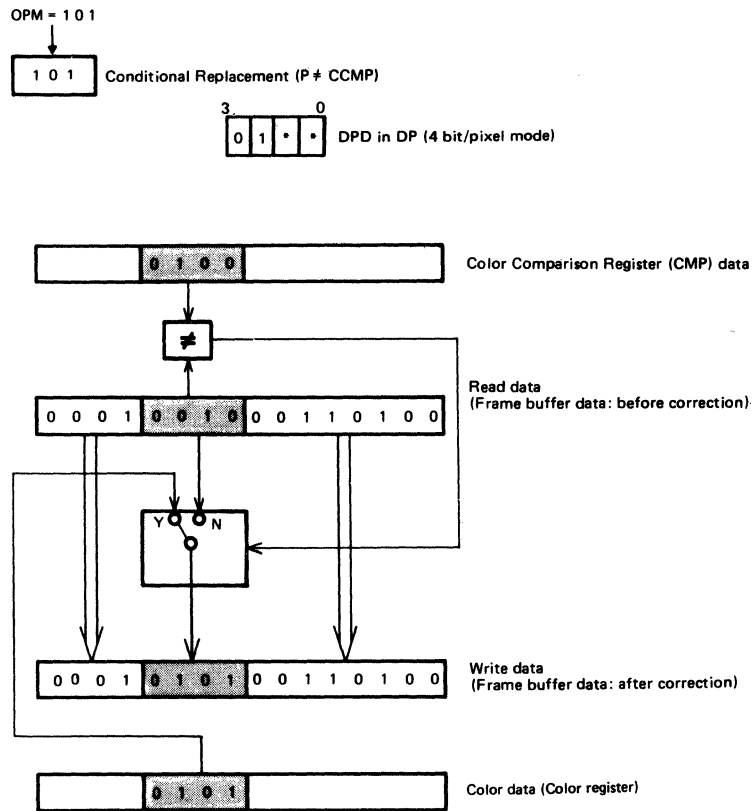
**Figure 6.9(d) EOR Operation Mode**





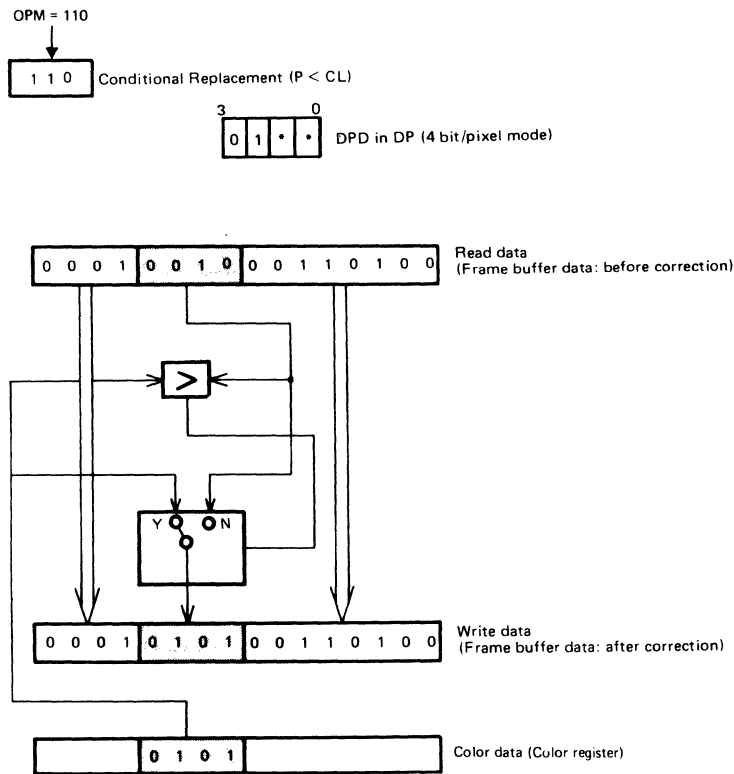
One pixel of the frame buffer read data is compared with the corresponding one pixel contents of the Color Comparison Register (CCMP). If equal, the read data is replaced with the color data and the result is rewritten to the read data location in the frame buffer. If not equal, the read data (unmodified) is rewritten to the read data location in the frame buffer. The dot pointer serves to extract the pixel from the frame buffer word – in this example, 4 bits/pixel.

**Figure 6.9(e) P=CCMP Operation Mode**



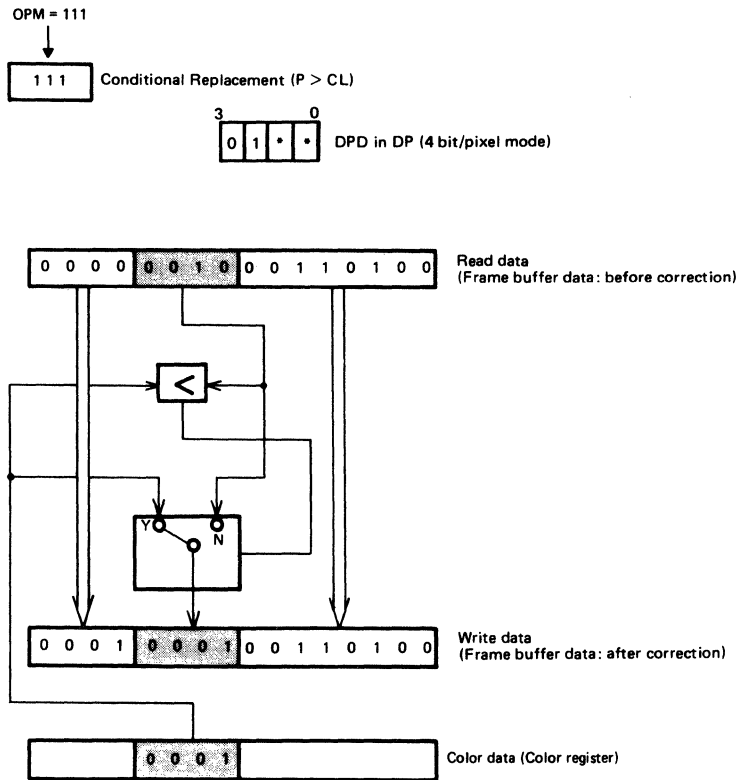
One pixel of the frame buffer read data is compared with the corresponding one pixel contents of the Color Comparison Register (CCMP). If not equal, the read data is replaced with the color data and the result is rewritten to the read data location in the frame buffer. If equal, the read data (unmodified) is rewritten to the read data location in the frame buffer. The dot pointer serves to extract the pixel from the frame buffer word – in this example, 4 bits/pixel.

**Figure 6.9(f) P ≠ CCMP Operation Mode**



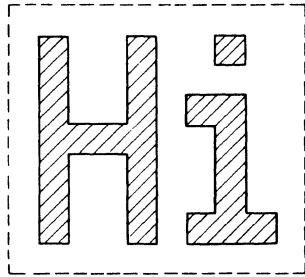
One pixel of the frame buffer read data is compared with the corresponding one pixel contents of the color data (CL). If the read data is LESS than the color data, the read data is replaced with the color data and the result is rewritten to the read data location in the frame buffer. If the read data is GREATER than or EQUAL to the color data, the read data (unmodified) is rewritten to the read data location in the frame buffer. The dot pointer serves to extract the pixel from the frame buffer word – in this example, 4 bits/pixel.

**Figure 6.9(g) P < CL Operation Mode**

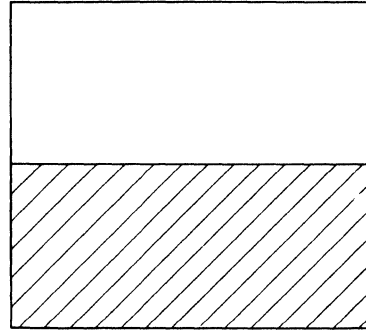


One pixel of the frame buffer read data is compared with the corresponding one pixel contents of the color data (CL). If the read data is GREATER than the color data, the read data is replaced with the color data and the result is rewritten to the read data location in the frame buffer. If the read data is LESS than or EQUAL to the color data, the read data (unmodified) is rewritten to the read data location in the frame buffer. The dot pointer serves to extract the pixel from the frame buffer word – in this example, 4 bits/pixel.

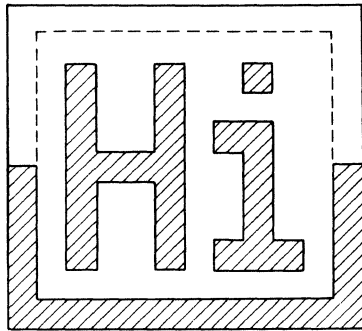
**Figure 6.9(h) P > CL Operation Mode**



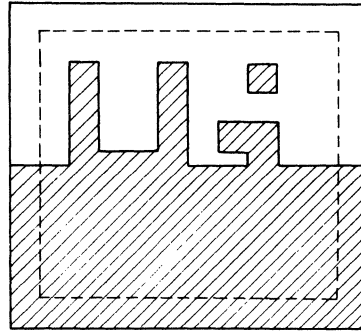
Drawing Pattern



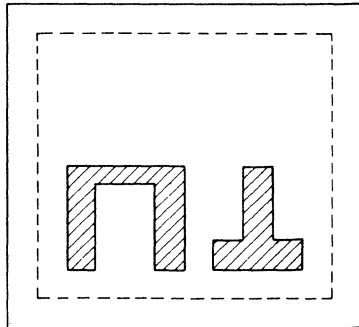
Picture Memory before Drawing



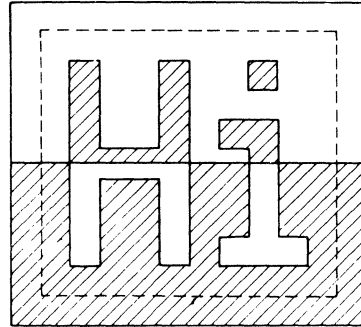
Replacement



OR



AND

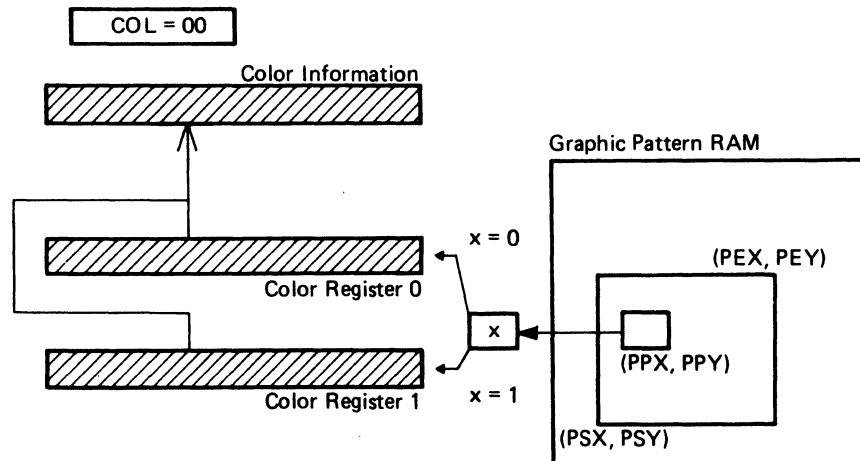


EOR

Figure 6.10 Operation Mode Example

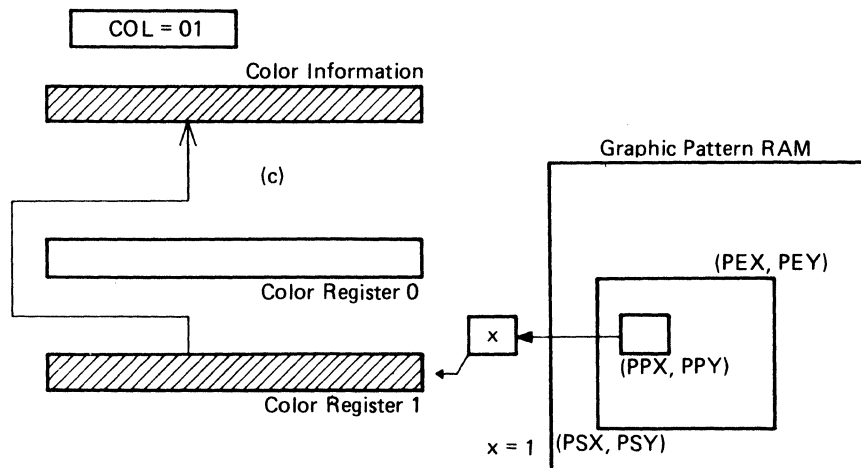


### 3. Colour Mode Examples



If the scanned Pattern RAM bit is equal '0', Color Register 0 (CL0) determines the color information. If the scanned Pattern RAM bit is equal '1', Color Register 1 (CL1) determines the color information.

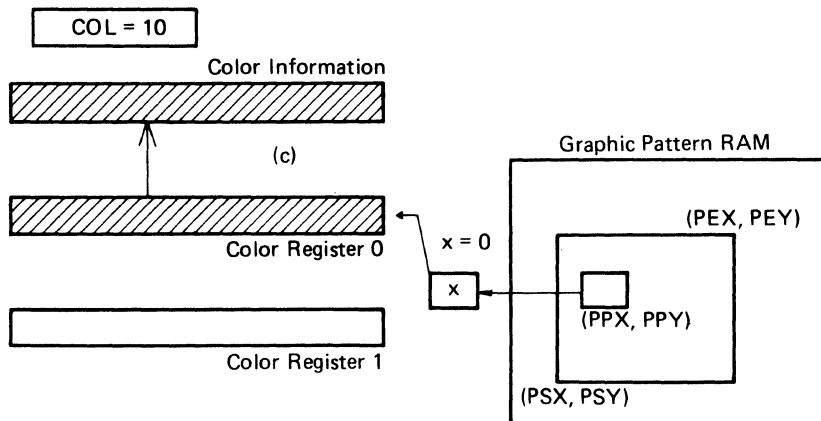
**Figure 6.11(a) Color Mode = 00**



If the scanned Pattern RAM bit is equal '0', the drawing operation is suppressed and the frame buffer is not changed. If the scanned Pattern RAM bit is equal '1', Color Register 1 (CL1) determines the color information.

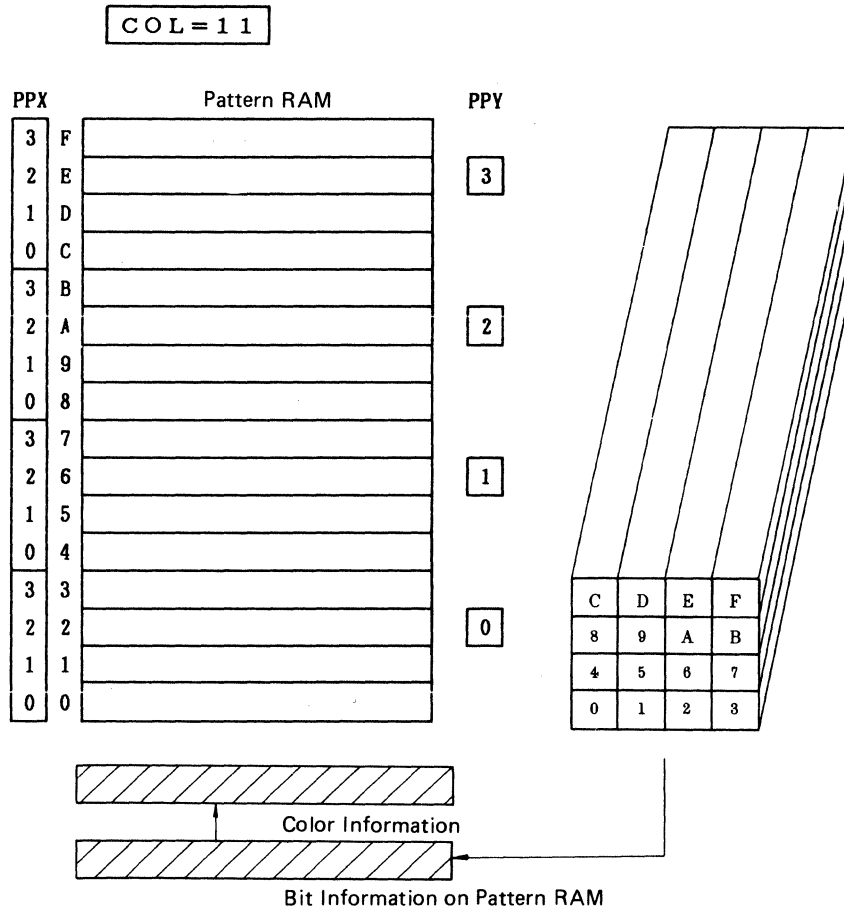
**Figure 6.11(b) Color Mode = 01**





If the scanned Pattern RAM bit is equal '1', the drawing operation is suppressed and the frame buffer is not changed. If the scanned Pattern RAM bit is equal '0', Color Register 0 (CL0) determines the color information.

**Figure 6.11(c) Color Mode = 10**

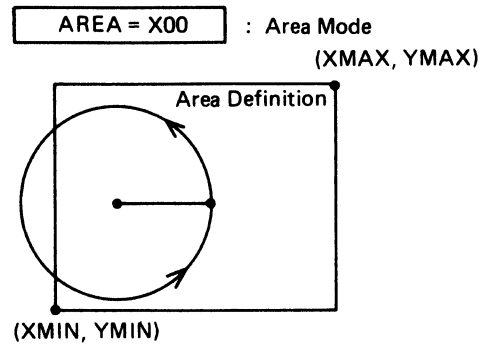


**Figure 6.11(d) Color Mode = 11**

In the former three color modes (Pattern RAM indirect), the actual color information is stored in the color registers (CL0, CL1) and selection is based on the 0 or 1 bit value during Pattern RAM scanning.

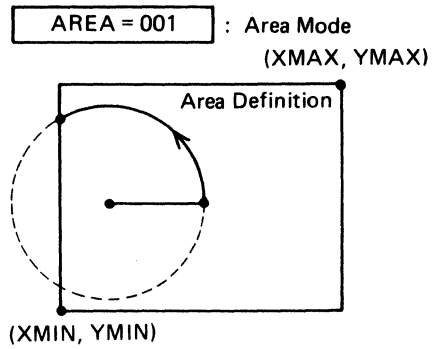
In color mode = 11 (Pattern RAM direct), the Pattern RAM contents are directly used to generate color information. This is accomplished by remapping of the Pattern RAM so that it is interpreted as containing up to 4 by 4 logical pixel color patterns, each of which contains 16 bits of color information.

#### 4. Area Mode Examples



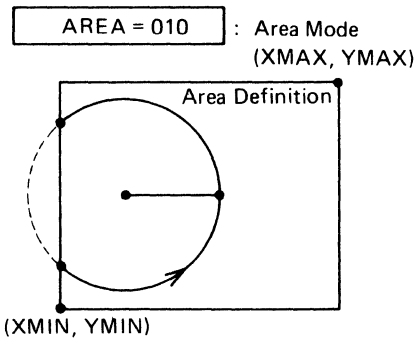
Drawing is executed without area checking.

**Figure 6.12(a) Area Mode = X00**



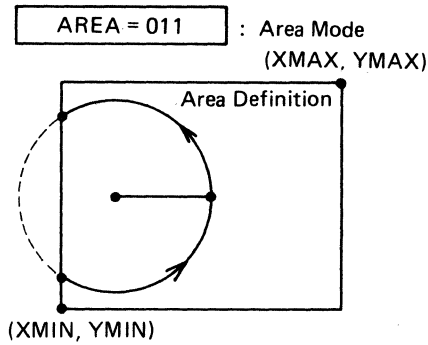
Drawing is executed as long as the CP (Current Pointer) resides in the defined area. When the drawing operation causes the CP to go outside the defined area, the drawing instruction is terminated and the ARD (Area Detect) and CED (Command End) flags in the Status Register (SR) are set to '1'.

**Figure 6.12(b) Area Mode = 001**



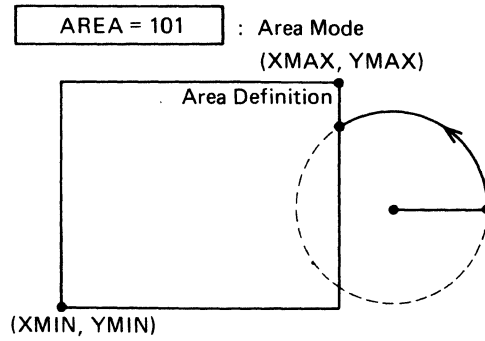
When the CP (Current Pointer) is outside the defined area, drawing is suppressed but the drawing operation continues. When CP is inside the defined area, drawing operation is enabled. When the drawing instruction execution is completed, the CED (Command End) bit in the Status Register (SR) is set to '1'. The ARD bit (Area Detect) bit in the Status Register is not set to '1' at any time during the drawing instruction execution regardless of whether CP goes inside or outside the defined area.

**Figure 6.12(c) Area Mode = 010**



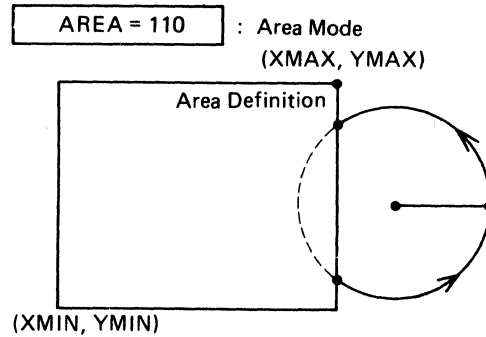
This mode is the same as AREA MODE = 010 in that drawing is enabled when CP (Current Pointer) is inside the defined area and suppressed when CP is outside the defined area. However, if at any time during the drawing instruction execution, CP goes outside the defined area, the ARD (Area Detect) bit in the Status Register (SR) will be set to '1'. The ARD bit can be monitored to determine when the CP goes outside the defined area.

**Figure 6.12(d) Area Mode = 011**



Drawing is executed as long as the CP (Current Pointer) resides outside the defined area. When the drawing operation causes the CP to go inside the defined area, the drawing instruction is terminated and the ARD (Area Detect) and CED (Command End) flags in the Status Register (SR) are set to '1'.

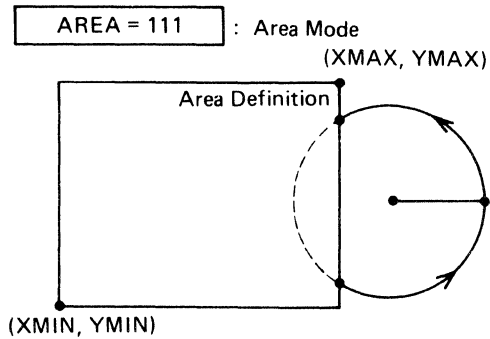
**Figure 6.12(e) Area Mode = 101**



When the CP (Current Pointer) is inside the defined area, drawing is suppressed but the drawing operation continues. When CP is outside the defined area, drawing operation is enabled. When the drawing instruction execution is completed, the CED (Command End) bit in the Status Register (SR) is set to '1'. The ARD bit (Area Detect) bit in the Status Register is not set to '1' at any time during the drawing instruction execution regardless of whether CP goes inside or outside the defined area.

**Figure 6.12(f) Area Mode = 110**





This mode is the same as AREA MODE = 110 in that drawing is enabled when CP (Current Pointer) is outside the defined area and suppressed when CP is inside the defined area. However, if at any time during the drawing instruction execution, CP goes inside the defined area, the ARD (Area Detect) bit in the Status Register (SR) will be set to '1'. The ARD bit can be monitored to determine when the CP goes inside the defined area.

**Figure 6.12(g) Area Mode = 111**



# 5. COMMANDS

TYPE	MNEMONIC	COMMAND NAME	OPERATION CODE	PARAMETER	# (words)	~ (cycles)
Register Access Command	ORG	Origin	0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	DPH DPL	3	8
	WPR	Write Parameter Register	0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	RN D	2	6
	RPR	Read Parameter Register	0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	RN	1	6
	WPTN	Write Pattern RAM	0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	PRA n D <sub>1</sub> , ..., D <sub>n</sub>	n+2	4n+8
	RPTN	Read Pattern RAM	0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0	PRA n	2	4n+10
Data Transfer Command	DRD	DMA Read	0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	AX AY	3	(4x+8)y+12[x·y/8↑]+(62~68)
	DWT	DMA Write	0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	AX AY	3	(4x+8)y+16[x·y/8↑]+34
	DMOD	DMA Modify	0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	MM AX AY	3	(4x+8)y+16[x·y/8↑]+34
	RD	Read	0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0		1	12
	WT	Write	0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	D	2	8
	MOD	Modify	0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	MM D	2	8
	CLR	Clear	0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	D AX AY	4	(2x+8)y+12
	SCLR	Selective Clear	0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	MM D AX AY	4	(4x+6)y+12
	CPY	Copy	0 1 1 0 S DSD 0 0 0 0 0 0 0 0 0 0 0 0 0 0	SAH SAL AX AY	5	(6x+10)y+12
	SCPY	Selective Copy	0 1 1 1 S DSD 0 0 0 0 0 0 0 0 0 0 0 0 0 0	SAH SAL AX AY	5	(6x+10)y+12
Graphic Command	AMOVE	Absolute Move	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	X Y	3	56
	RMOVE	Relative Move	1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	dX dY	3	56
	ALINE	Absolute Line	1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	AREA COL OPM X Y	3	P·L+18
	RLINE	Relative Line	1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	AREA COL OPM dX dY	3	P·L+18
	ARCT	Absolute Rectangle	1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	AREA COL OPM X Y	3	2P(A+B)+54
	RRCT	Relative Rectangle	1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	AREA COL OPM dX dY	3	2P(A+B)+54
	APLL	Absolute Polyline	1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	AREA COL OPM n X <sub>1</sub> , Y <sub>1</sub> , ..., X <sub>n</sub> , Y <sub>n</sub>	2n+2	Σ[P·L+16]+8
	RPLL	Relative Polyline	1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	AREA COL OPM n dX <sub>1</sub> , dY <sub>1</sub> , ..., dX <sub>n</sub> , dY <sub>n</sub>	2n+2	Σ[P·L+16]+8
	APLG	Absolute Polygon	1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	AREA COL OPM n X <sub>1</sub> , Y <sub>1</sub> , ..., X <sub>n</sub> , Y <sub>n</sub>	2n+2	Σ[P·L+16]+P·Lo+20
	RPLC	Relative Polygon	1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	AREA COL OPM n dX <sub>1</sub> , dY <sub>1</sub> , ..., dX <sub>n</sub> , dY <sub>n</sub>	2n+2	Σ[P·L+16]+P·Lo+20
	CRCL	Circle	1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	AREA COL OPM r	2	8d+66
	ELPS	Ellipse	1 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	AREA COL OPM a b dX	4	10d+90
	AARC	Absolute Arc	1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	AREA COL OPM Xc Yc Xc Ye	5	8d+18
	RARC	Relative Arc	1 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	AREA COL OPM dXc dYc dXc dYe	5	8d+18
	AEARC	Absolute Ellipse Arc	1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	AREA COL OPM a b Xc Yc Xc Ye	7	10d+96
	REARC	Relative Ellipse Arc	1 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	AREA COL OPM a b dXc dYc dXc dYe	7	10d+96
	AFRCT	Absolute Filled Rectangle	1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	AREA COL OPM X Y	3	(P·A+B)B+18
	RFRICT	Relative Filled Rectangle	1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	AREA COL OPM dX dY	3	(P·A+B)B+18
	PAINT	Paint	1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	AREA COL OPM E	1	(18A+102)B-58 *1)
	DOT	Dot	1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	AREA COL OPM	1	8
	PTN	Pattern	1 1 0 1 SL SD	AREA COL OPM SZ	*2) 2	(P·A+10)B+20
	AGCPY	Absolute Graphic Copy	1 1 1 0 S DSD	AREA 0 0 OPM Xs Ys DX DY	5	((P+2)A+10)B+70
	RGCPY	Relative Graphic Copy	1 1 1 1 S DSD	AREA 0 0 OPM dXs dYs DX DY	5	((P+2)A+10)B+70

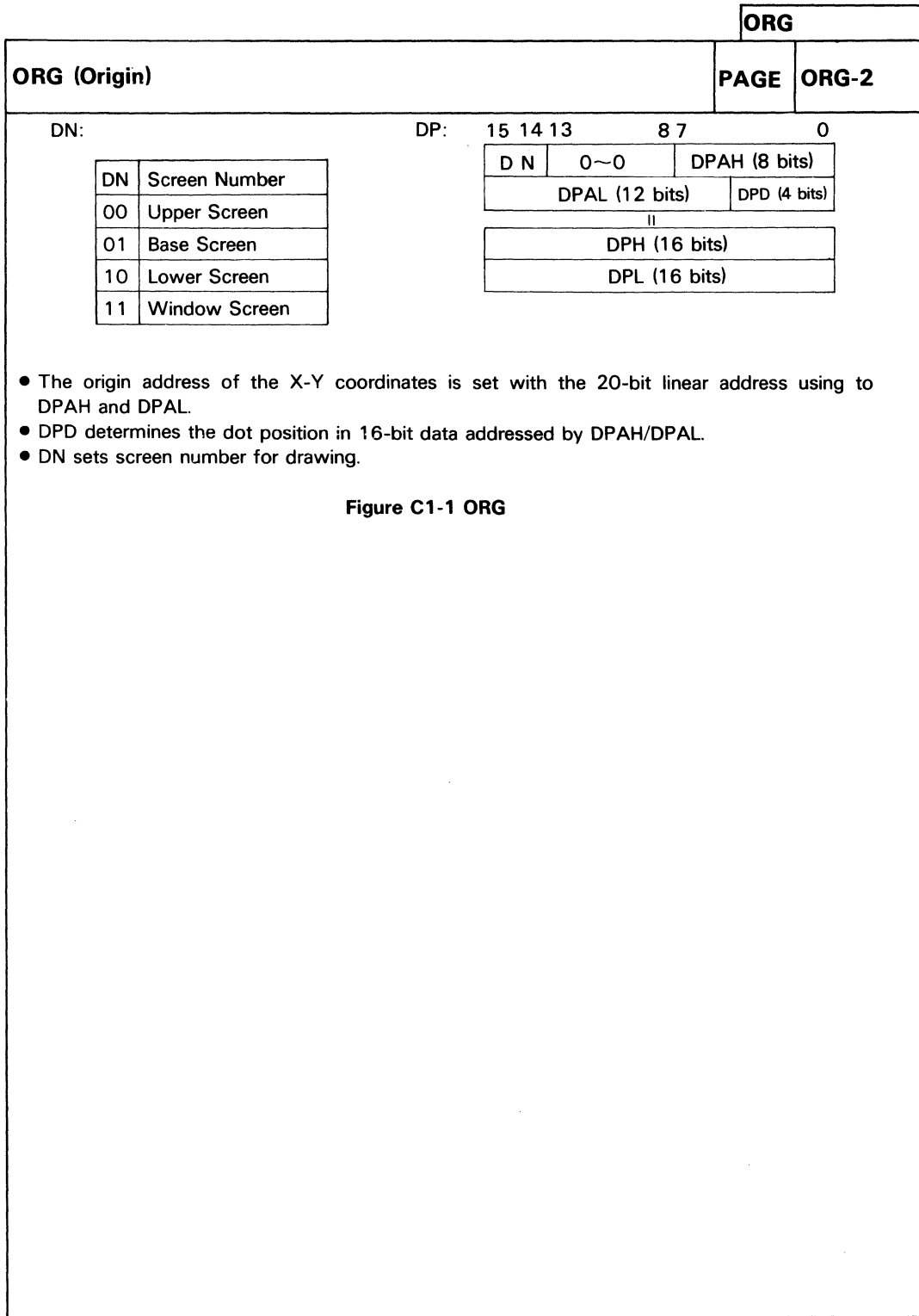
\*1) In case of rectangular filling

\*2) SZ:  $\begin{matrix} 15 & 87 & 0 \\ \boxed{SZy} & \boxed{SZx} & \end{matrix}$  SZy, SZx: Pattern Size

n: number of repetition x/y: drawing words of x direction/y-direction L/Lo/d: sum of drawing dots A/B: drawing dots of main/sub direction E: [E=0 (stop at Edge color), E=1 (stop at excepting Edge color)] C: [C=1 (clock wise), C=0 (reverse)] [↑]: rounding up

P= [4: OPM-000 ~ 011  
6: OPM-100 ~ 111

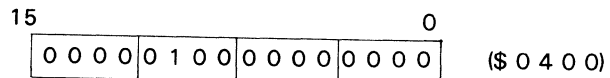
		<b>ORG</b>	
<b>[1] ORG (Origin)</b>	<b>PAGE</b>	<b>ORG-1</b>	
<p>&lt; <b>FUNCTION</b> &gt;  Associates a logical X-Y screen origin with a physical frame buffer address.</p> <p>&lt; <b>MNEMONIC</b> &gt;  ORG DPH,DPL</p>		TYPE	Register Access Command
<p>&lt; <b>FORMAT</b> &gt;</p> <p>COMMAND CODE <span style="float: right;">hexadecimal notation</span></p> <div style="display: flex; align-items: center; margin-left: 20px;"> <div style="text-align: right; margin-right: 5px;">15</div> <div style="border: 1px solid black; padding: 2px; display: flex; align-items: center; gap: 5px;"> <span style="font-size: small;">0 0 0 0</span> <span style="font-size: small;">0 1 0 0</span> <span style="font-size: small;">0 0 0 0</span> <span style="font-size: small;">0 0 0 0</span> </div> <div style="margin-left: 10px; text-align: right;">0</div> </div> <p style="margin-left: 20px;">(\$ 0 4 0 0)</p> <p>COMMAND PARAMETERS</p> <div style="margin-left: 20px;"> <div style="display: flex; align-items: center; margin-bottom: 10px;"> <div style="text-align: right; margin-right: 5px;">15</div> <div style="border: 1px solid black; padding: 2px; flex-grow: 1; text-align: center; margin: 0 auto;">DPH</div> <div style="margin-left: 10px; text-align: right;">0</div> </div> <div style="display: flex; align-items: center;"> <div style="text-align: right; margin-right: 5px;">15</div> <div style="border: 1px solid black; padding: 2px; flex-grow: 1; text-align: center; margin: 0 auto;">DPL</div> <div style="margin-left: 10px; text-align: right;">0</div> </div> </div>		<p>WORD NUMBER W<sub>n</sub>=3</p> <p>EXECUTION CYCLES C<sub>n</sub>=8</p>	
<p>&lt; <b>DESCRIPTION</b> &gt;</p> <p>The ORG command must be issued to the ACRTC prior to graphic drawing. ORG defines the logical X-Y coordinate origin upon which all graphic drawing addresses are based and sets the screen number in which to draw.</p> <p>The DPH and DPL (Drawing Pointer High, Low) parameters establish the physical address in the frame buffer at which the origin is set. This physical address is composed of the following three components – DN (Screen Number) is a screen designator, DPAH, DPAL (Drawing Pointer Address High, Low) is a 20 bit address selecting one of 1 megawords in the frame buffer and DPD (Drawing Pointer Dot) specifies the bit field associated with the addressed logical pixel.</p> <p>The ORG command initializes the Drawing Pointer (DP) to the origin and clears the Current Pointer (CP).</p>			



< EXAMPLE >

The origin for the Upper screen (screen number 0) is set to bit position 4-7 at frame buffer word address \$25. 4 bits per logical pixel and Memory Width (MW) = \$10 are assumed.

COMMAND CODE



COMMAND PARAMETERS

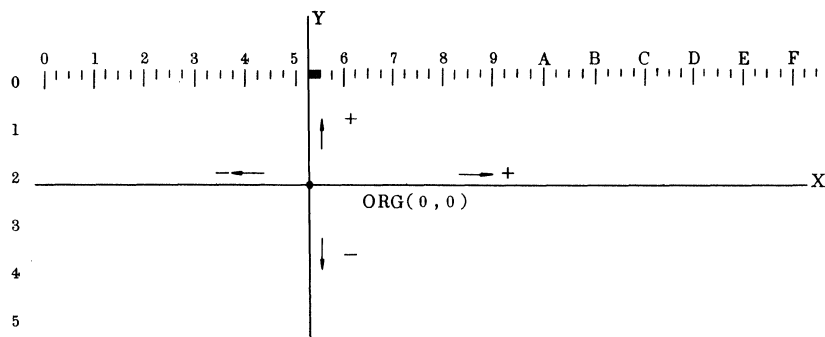
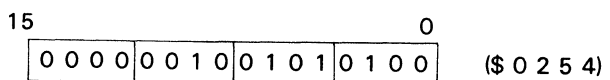
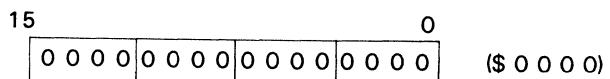


Figure C1-2 ORG Execution Example

		WPR						
[2] WPR (Write Parameter Register)		PAGE	WPR-1					
<p>&lt; FUNCTION &gt; Write the contents of the Drawing Parameter Registers.</p> <p>&lt; MNEMONIC &gt; WPR (RN) D</p>		TYPE	Register Access Command					
<p>&lt; FORMAT &gt;</p> <p>COMMAND CODE</p> <div style="display: flex; align-items: center;"> <div style="margin-right: 20px;"> <p>15</p> <table border="1" style="border-collapse: collapse; text-align: center;"> <tr> <td style="width: 40px;">0 0 0 0</td> <td style="width: 40px;">1 0 0 0</td> <td style="width: 40px;">0 0 0 0</td> <td style="width: 40px;">RN</td> </tr> </table> <p style="text-align: center;">5 bits</p> </div> <div> <p>hexadecimal notation</p> <p>(\$ 0 8 0 X)</p> </div> </div> <p style="margin-top: 20px;">COMMAND PARAMETERS</p> <div style="display: flex; align-items: center;"> <div style="margin-right: 20px;"> <p>15</p> <table border="1" style="border-collapse: collapse; text-align: center;"> <tr> <td style="width: 100px;">D (Data)</td> </tr> </table> </div> <div> <p>0</p> </div> </div>		0 0 0 0	1 0 0 0	0 0 0 0	RN	D (Data)	<p>WORD NUMBER W<sub>n</sub>=2</p> <p>EXECUTION CYCLES C<sub>n</sub>=6</p>	
0 0 0 0	1 0 0 0	0 0 0 0	RN					
D (Data)								
<p>&lt; DESCRIPTION &gt;</p> <p>The Drawing Parameter Register number to be written is specified in the RN (Register Number) field of the op-code. The contents of the parameter (D) is written to the selected register.</p>								

## &lt; EXAMPLE &gt;

The value \$1111 is written to the CL1 (Color 1) of the drawing parameter register.

## COMMAND CODE

15	0
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1	(\$ 0 8 0 1)

## COMMAND PARAMETERS

15	0
0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1	(\$ 1 1 1 1)

< Color Register >      RN=01

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

Figure C2-1 WPR Execution Example

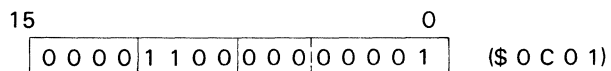


		RPR	
[3] RPR (Read Parameter Register)		PAGE	RPR-1
<p>&lt; FUNCTION &gt; Read the contents of the Drawing Parameter Registers.</p> <p>&lt; MNEMONIC &gt; RPR (RN)</p>		TYPE	Register Access Command
<p>&lt; FORMAT &gt;</p> <p>COMMAND CODE</p> <div style="display: flex; align-items: center; justify-content: center;"> <div style="text-align: right; margin-right: 5px;">15</div> <div style="border: 1px solid black; padding: 2px; display: flex; align-items: center; gap: 2px;"> <span>0</span><span>0</span><span>0</span><span>0</span><span>1</span><span>1</span><span>0</span><span>0</span><span>0</span><span>0</span><span>0</span> </div> <div style="margin-left: 5px; text-align: left;">0</div> </div> <p style="text-align: center; margin-left: 100px;">5 bits</p> <p style="margin-left: 100px;">hexadecimal notation (\$ 0 C 0 X)</p> <p>COMMAND PARAMETERS - NON -</p>		<p>WORD NUMBER Wn= 1</p> <p>EXECUTION CYCLES Cn= 6</p>	
<p>&lt; DESCRIPTION &gt;</p> <p>The Drawing Parameter Register number to be read is specified in the RN (Register Number) field of the command code. After execution, the contents of the specified Drawing Parameter Register is loaded into the Read FIFO.</p>			

< EXAMPLE >

The value \$1111 in the Drawing Parameter Register (Color Register 1: CL1) is loaded into the Read FIFO.

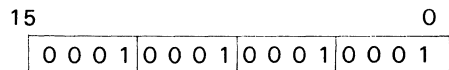
COMMAND CODE



COMMAND PARAMETER

- NON -

< Color Register 1 >



< Read FIFO >

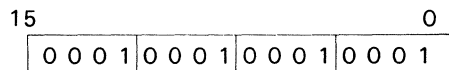


Figure C3-1 RPR Execution Example

		WPTN														
[4] WPTN (Write Pattern RAM)		PAGE	WPTN-1													
<p>&lt; FUNCTION &gt; Write data to the Pattern RAM.</p> <p>&lt; MNEMONIC &gt; WPTN (PRA) n, D<sub>1</sub>, D<sub>2</sub>, ... D<sub>n</sub></p>		TYPE	Register Access Command													
<p>&lt; FORMAT &gt;</p> <p>COMMAND CODE <span style="float: right;">hexadecimal notation</span></p> <p style="text-align: center;">15 <span style="float: right;">0</span></p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="padding: 2px 5px;">0</td> <td style="padding: 2px 5px;">0</td> <td style="padding: 2px 5px;">0</td> <td style="padding: 2px 5px;">1</td> <td style="padding: 2px 5px;">1</td> <td style="padding: 2px 5px;">0</td> <td style="padding: 2px 5px;">0</td> <td style="padding: 2px 5px;">0</td> <td style="padding: 2px 5px;">0</td> <td style="padding: 2px 5px;">PRA</td> </tr> </table> <p style="text-align: center;">(\$ 1 8 0 X)</p> <p>COMMAND PARAMETERS</p> <p style="text-align: center;">15 <span style="float: right;">0</span></p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="padding: 2px 5px; text-align: center;">n (Number of Words)</td> </tr> </table> <p style="text-align: center;">15 <span style="float: right;">0</span></p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="padding: 2px 5px; text-align: center;">D1 (Pattern Data)</td> </tr> </table> <p style="text-align: center;">⋮</p> <p style="text-align: center;">15 <span style="float: right;">0</span></p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="padding: 2px 5px; text-align: center;">Dn (Pattern Data)</td> </tr> </table>		0	0	0	1	1	0	0	0	0	PRA	n (Number of Words)	D1 (Pattern Data)	Dn (Pattern Data)	<p>WORD NUMBER W<sub>n</sub>=n+2</p> <p>EXECUTION CYCLES C<sub>n</sub>=4n+8</p>	
0	0	0	1	1	0	0	0	0	PRA							
n (Number of Words)																
D1 (Pattern Data)																
Dn (Pattern Data)																
<p>&lt; DESCRIPTION &gt;</p> <p>WPTN command is used to write data into the Pattern RAM.</p> <p>Pattern RAM Address (PRA) of \$0~\$F is allocated to the Pattern RAM and each PRA represents 1 word (16 bits) of pattern RAM.</p> <p>The PRA (Pattern RAM Address) field of the command code selects the Pattern RAM word-address at which writing starts. The first parameter is n, the number of words to be written. This is followed by n data words (D1-Dn).</p> <p>For the 8-bit interface, 1 word is divided into high and low bytes. The pattern data is sent in the order of the high byte, then the low byte. The first parameter n must be set to (the number of words) × 2. (In this case writing in unit of byte is not allowed.)</p>																

WPTN (Write Pattern RAM)

<EXAMPLE>

Two words of data, \$2314 and \$5713, are written to the Pattern RAM beginning at address \$B.

COMMAND CODE

15								0	
	0	0	0	1	1	0	0	0	0
									1 0 1 1

(\$ 1 8 0 B)

COMMAND PARAMETERS

15								0	
	0	0	0	0	0	0	0	0	0
									0 0 1 0

(\$ 0 0 0 2)

15								0	
	0	0	1	0	0	0	1	1	0
									0 1 0 0

(\$ 2 3 1 4)

15								0	
	0	1	0	1	0	1	1	0	0
									0 0 1 1

(\$ 5 7 1 3)

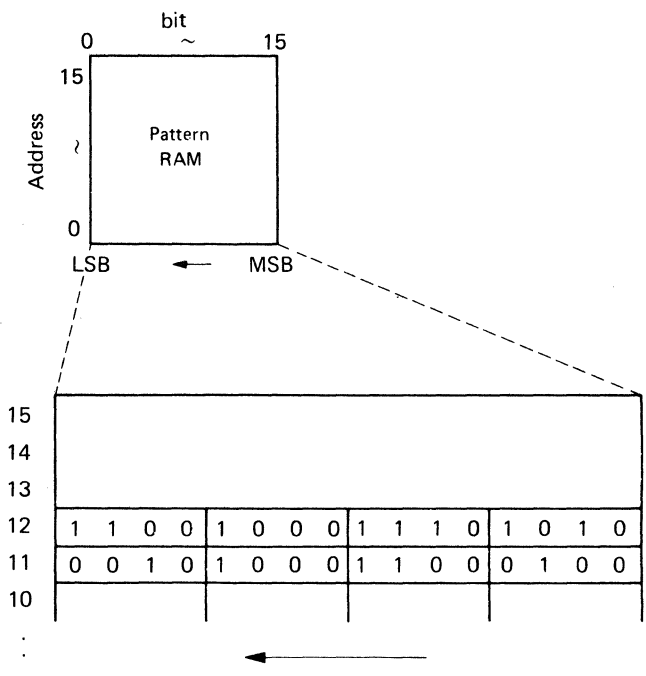


Figure C4-1 WPTN Execution Example

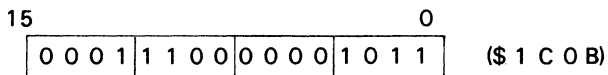
		RPTN														
[5] RPTN (Read Pattern RAM)	PAGE	RPTN-1														
<p>&lt; FUNCTION &gt; Read Data from the Pattern RAM.</p> <p>&lt; MNEMONIC &gt; RPTN (PRA) n</p>		TYPE	Register Access Command													
<p>&lt; FORMAT &gt;</p> <p>COMMAND CODE <span style="float: right;">hexadecimal notation</span></p> <p style="text-align: center;">15 <span style="float: right;">0</span></p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td style="text-align: center;">PRA</td> </tr> </table> <p style="text-align: center;">(\$ 1 C 0 X)</p> <p>COMMAND PARAMETERS</p> <p style="text-align: center;">15 <span style="float: right;">0</span></p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="text-align: center;">n (Number of word)</td> </tr> </table>		0	0	0	1	1	1	0	0	0	0	0	PRA	n (Number of word)	<p>WORD NUMBER <math>W_n = 2</math></p> <p>EXECUTION CYCLES <math>C_n = 4n + 10</math></p>	
0	0	0	1	1	1	0	0	0	0	0	PRA					
n (Number of word)																
<p>&lt; DESCRIPTION &gt;</p> <p>RPTN command is used to read the data in the Pattern RAM.</p> <p>Pattern RAM address (PRA) of \$0~\$F is allocated to the Pattern RAM and each PRA represents 1 word (16 bits) of Pattern RAM.</p> <p>The PRA (Pattern RAM Address) field of the command code select the Pattern RAM word address at which reading starts. The parameter n specifies the number of words to be read. The specified Pattern RAM contents are loaded into the Read FIFO.</p> <p>For the 8 bit interface, 1 word of the pattern RAM is divided into high and the low bytes. The pattern data is put into the Read FIFO in the order of the high byte, the low byte.</p>																

RPTN (Read Pattern RAM)

< EXAMPLE >

Two words of data, \$2314 and \$5713 from the Pattern RAM beginning from address \$B is placed in the Read FIFO.

COMMAND CODE



COMMAND PARAMETERS

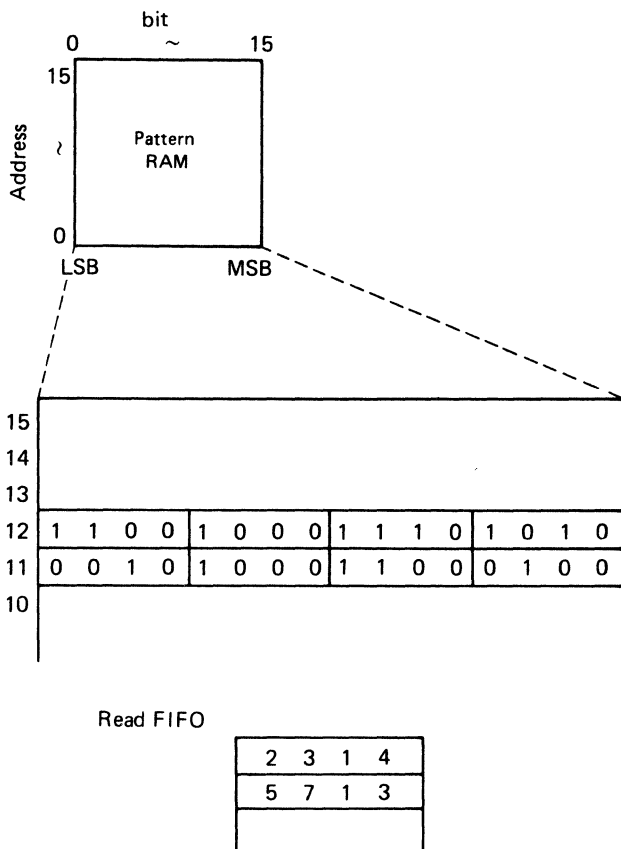
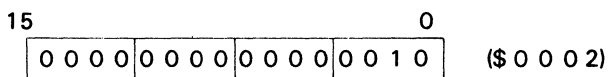
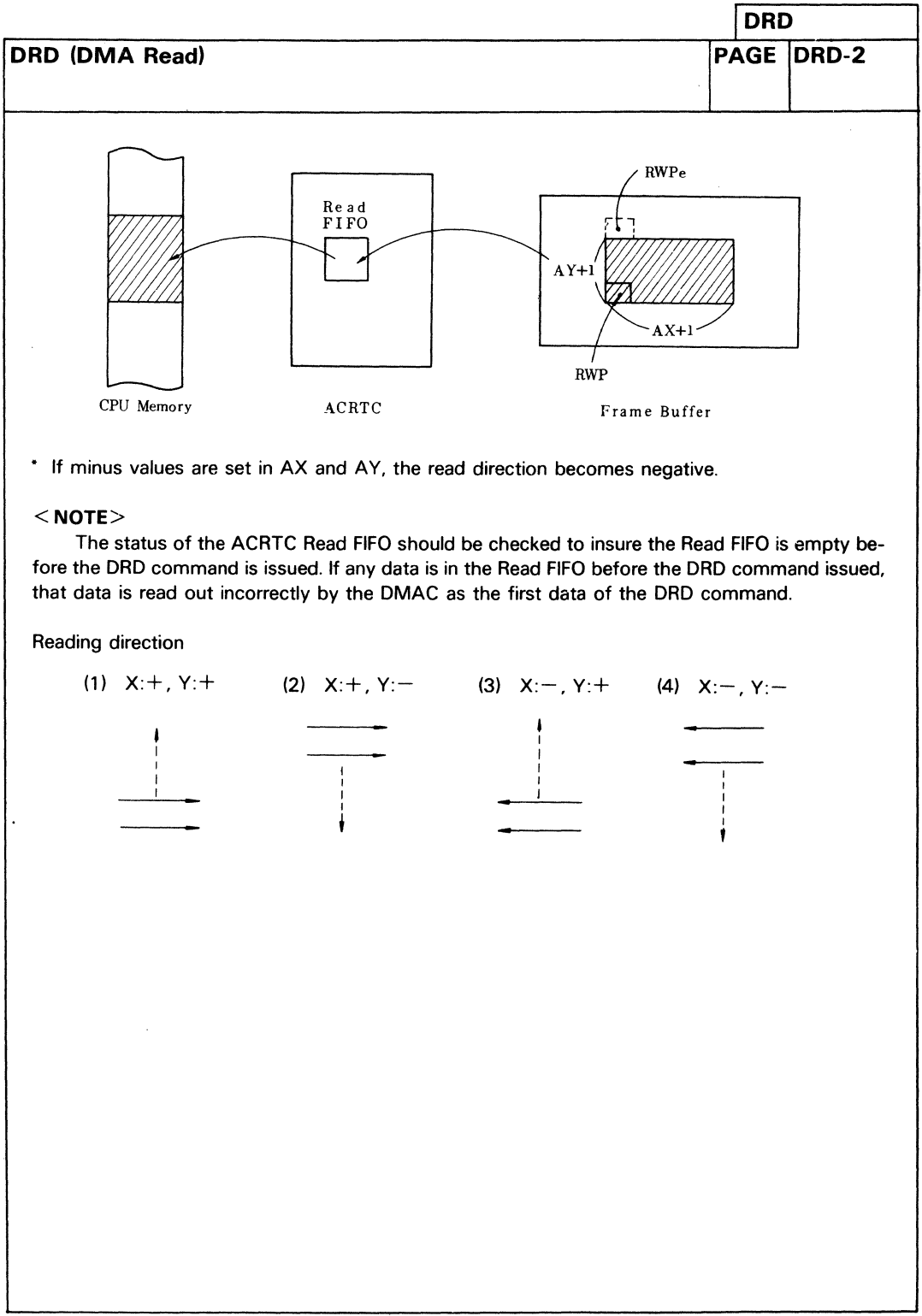


Figure C5-1 RPTN Execution Example

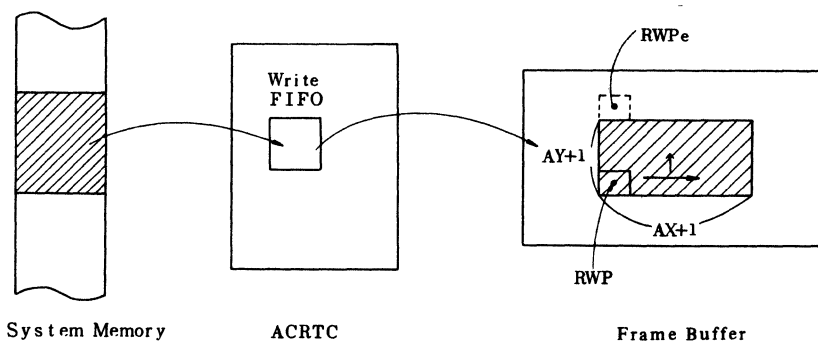
		DRD																			
[6] DRD (DMA Read)		PAGE	DRD-1																		
<p>&lt; FUNCTION &gt; Transfer data from the frame buffer to the MPU system memory.</p> <p>&lt; MNEMONIC &gt; DRD AX, AY</p>		TYPE	Data Transfer Command																		
<p>&lt; FORMAT &gt;</p> <p>COMMAND CODE <span style="float: right;">hexadecimal notation</span></p> <p>15 <span style="float: right;">0</span></p> <table border="1" style="margin-left: 40px;"> <tr> <td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td> </tr> </table> <p style="margin-left: 100px;">(\$ 2 4 0 0)</p> <p>COMMAND PARAMETERS</p> <p>15 <span style="float: right;">0</span></p> <table border="1" style="margin-left: 40px; width: 100%;"> <tr> <td style="text-align: center;">AX</td> </tr> </table> <p>15 <span style="float: right;">0</span></p> <table border="1" style="margin-left: 40px; width: 100%;"> <tr> <td style="text-align: center;">AY</td> </tr> </table>		0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	AX	AY	<p>WORD NUMBER <math>W_n=3</math></p> <p>EXECUTION CYCLES <math>C_n=(4x+8)y+12 \left\lceil \frac{x \cdot y}{8} \right\rceil + (62 \sim 68)</math></p> <p><math>x =  AX  + 1</math> <math>y =  AY  + 1</math></p>	
0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0						
AX																					
AY																					
<p>&lt; DESCRIPTION &gt;</p> <p>DRD command causes the ACRTC to enter DMA Data Transfer Mode in which the ACRTC will control the external DMAC to transfer data (in unit of words) from the rectangular area in the frame buffer to the MPU memory. The frame buffer data origin must be predefined in the Read Write Pointer (RWP). The parameters of the command define the frame buffer area to be read in units of physical frame buffer words. At the end of DRD command execution, RWP will be set to RWPe.</p>																					





**DWT**

<b>[7] DWT (DMA Write)</b>	<b>PAGE</b>	<b>DWT-1</b>																		
<p><b>&lt; FUNCTION &gt;</b> Transfer data from the MPU system memory to the frame buffer.</p> <p><b>&lt; MNEMONIC &gt;</b> DWT AX, AY</p>	TYPE	Data Transfer Command																		
<p><b>&lt; FORMAT &gt;</b></p> <p>COMMAND CODE <span style="float: right;">hexadecimal notation</span></p> <div style="display: flex; align-items: center; justify-content: space-between;"> <div style="text-align: center;"> <p>15 <span style="margin-left: 100px;">0</span></p> <table border="1" style="border-collapse: collapse; text-align: center;"> <tr> <td style="width: 25px;">0</td><td style="width: 25px;">0</td><td style="width: 25px;">1</td><td style="width: 25px;">0</td><td style="width: 25px;">0</td><td style="width: 25px;">0</td><td style="width: 25px;">0</td><td style="width: 25px;">0</td><td style="width: 25px;">0</td><td style="width: 25px;">0</td><td style="width: 25px;">0</td><td style="width: 25px;">0</td><td style="width: 25px;">0</td><td style="width: 25px;">0</td><td style="width: 25px;">0</td><td style="width: 25px;">0</td><td style="width: 25px;">0</td> </tr> </table> </div> <div style="margin-left: 20px;">(\$ 2 8 0 0)</div> </div> <p>COMMAND PARAMETERS</p> <div style="margin-bottom: 10px;"> <p>15 <span style="margin-left: 100px;">0</span></p> <table border="1" style="border-collapse: collapse; width: 100%; text-align: center;"> <tr><td style="width: 100px; height: 20px;">AX</td></tr> </table> </div> <div> <p>15 <span style="margin-left: 100px;">0</span></p> <table border="1" style="border-collapse: collapse; width: 100%; text-align: center;"> <tr><td style="width: 100px; height: 20px;">AY</td></tr> </table> </div>	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	AX	AY	<p>WORD NUMBER W<sub>n</sub>=3</p> <p>EXECUTION CYCLES C<sub>n</sub>=(4x+8)y+16 <math>\left\lceil \frac{xy}{8} \right\rceil</math> +34  <math display="block">\begin{cases} x =  AX  + 1 \\ y =  AY  + 1 \end{cases}</math></p>
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
AX																				
AY																				
<p><b>&lt; DESCRIPTION &gt;</b></p> <p>DWT command causes the ACRTC to enter DMA Data Transfer Mode in which the ACRTC will control the external DMAC to transfer data (in unit of words) from the MPU memory to the rectangular area in the frame buffer. The frame buffer data origin must be predefined in the Read Write Pointer (RWP). The parameters of the command (AX, AY) define the frame buffer area to be written in units of physical frame buffer words. At the end of DWT command execution, RWP will be set to RWPe.</p>																				



\* For AX and AY, negative value can also be set.

< NOTE >

After DWT is issued, no further commands should be issued until the DMA data is transferred and the DWT command terminates.

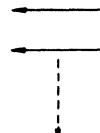
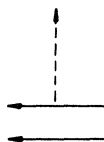
Writing direction

(1) X:+, Y:+

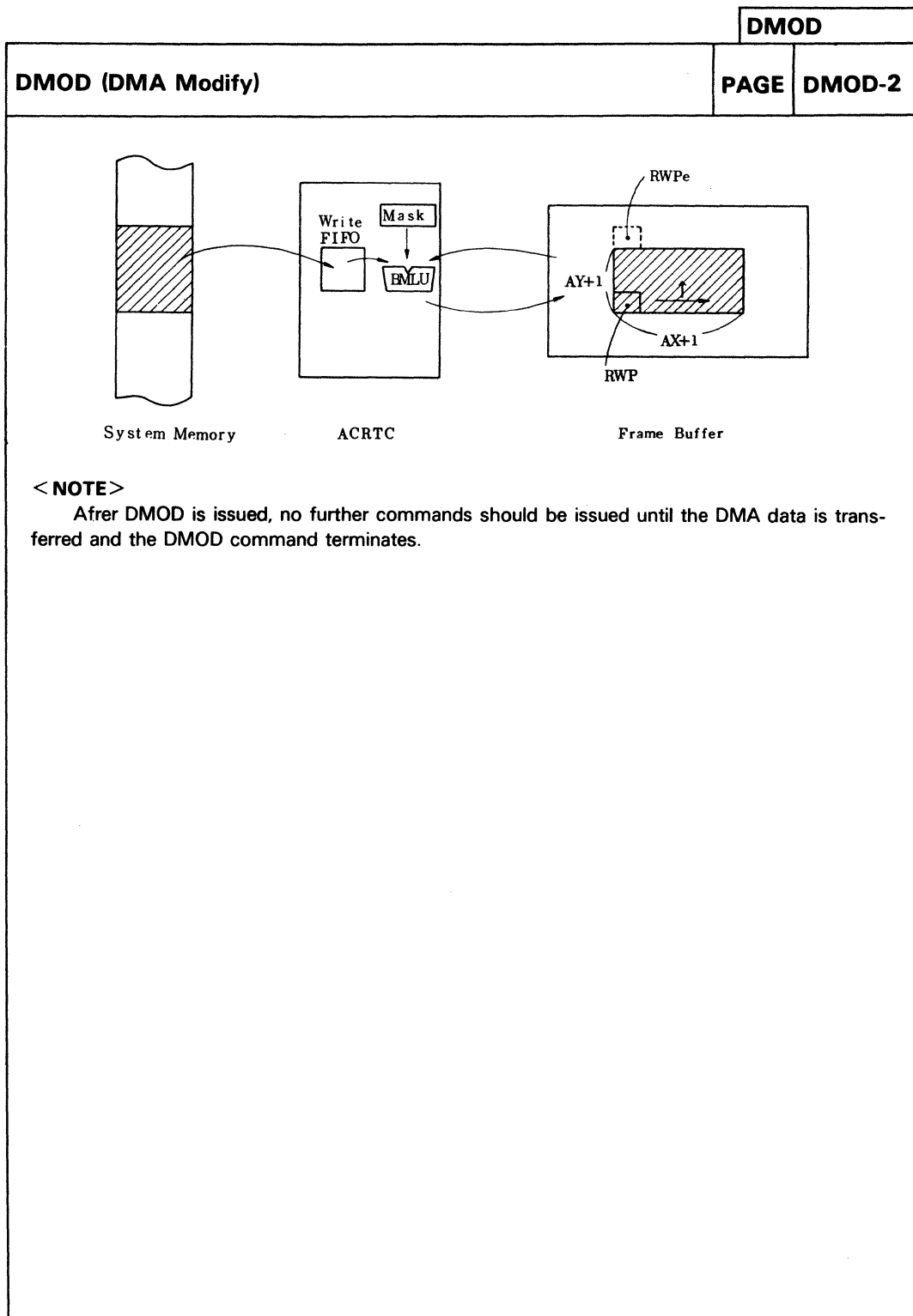
(2) X:+, Y:-

(3) X:-, Y:+

(4) X:-, Y:-



		DMOD																	
[8] DMOD (DMA Modify)		PAGE	DMOD-1																
<p>&lt; <b>FUNCTION</b> &gt; Transfer data from the MPU system memory to the frame buffer subject to logical modification.</p> <p>&lt; <b>MNEMONIC</b> &gt; DMOD (MM) AX, AY</p>		TYPE	Data Transfer Command																
<p>&lt; <b>FORMAT</b> &gt;</p> <p>COMMAND CODE</p> <div style="display: flex; justify-content: space-between; align-items: center;"> <span>15</span> <span>0</span> </div> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="width: 20px;">0</td><td style="width: 20px;">0</td><td style="width: 20px;">1</td><td style="width: 20px;">0</td><td style="width: 20px;">1</td><td style="width: 20px;">1</td><td style="width: 20px;">0</td><td style="width: 20px;">0</td><td style="width: 20px;">0</td><td style="width: 20px;">0</td><td style="width: 20px;">0</td><td style="width: 20px;">0</td><td style="width: 20px;">0</td><td style="width: 20px;">MM</td> </tr> </table> <p style="text-align: right; margin-right: 20px;">hexadecimal notation (\$ 2 C 0 X)</p> <p>COMMAND PARAMETERS</p> <div style="display: flex; justify-content: space-between; align-items: center; margin-bottom: 10px;"> <span>15</span> <span>0</span> </div> <table border="1" style="margin-left: auto; margin-right: auto; width: 80%;"> <tr> <td style="text-align: center;">AX</td> </tr> </table> <div style="display: flex; justify-content: space-between; align-items: center; margin-bottom: 10px;"> <span>15</span> <span>0</span> </div> <table border="1" style="margin-left: auto; margin-right: auto; width: 80%;"> <tr> <td style="text-align: center;">AY</td> </tr> </table>		0	0	1	0	1	1	0	0	0	0	0	0	0	MM	AX	AY	<p>WORD NUMBER Wn=3</p> <p>EXECUTION CYCLES Cn= (4x+8)y+16 <math>\left\lceil \frac{xy}{8} \right\rceil</math> +34  <math display="block">\begin{cases} x =  AX  + 1 \\ y =  AY  + 1 \end{cases}</math></p>	
0	0	1	0	1	1	0	0	0	0	0	0	0	MM						
AX																			
AY																			
<p>&lt; <b>DESCRIPTION</b> &gt;</p> <p>DMOD causes the ACRTC to enter DMA Data Transfer Mode in which the ACRTC will control the external DMAC to modify data in the rectangular area in the frame buffer using data in the MPU memory (in unit of words). The frame buffer data origin must be predefined in the Read Write Pointer (RWP). The parameters of the command (AX, AY) define the frame buffer area to be written in units of physical frame buffer words. At the end of DMOD command execution, RWP will be set to RWPe.</p> <p>The MM (Modify Mode) field of the command code specifies the DMA data transfer modify mode. Each pixel transferred from MPU system memory is logically operated on the corresponding pixel from the frame buffer, and the result is rewritten to the frame buffer. Logic operation can be enabled and disabled on a bit by bit basis based on the contents of the MASK register.</p>																			



		RD																	
[9] RD (Read)	PAGE	RD-1																	
<p>&lt; FUNCTION &gt; Read one word of data from the frame buffer and load the word into Read FIFO.</p> <p>&lt; MNEMONIC &gt; RD</p>	TYPE	Data Transfer Command																	
<p>&lt; FORMAT &gt;</p> <p>COMMAND CODE <span style="float: right;">hexadecimal notation</span></p> <p>15 <span style="float: right;">0</span></p> <table border="1" style="margin-left: 40px;"> <tr> <td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td> </tr> </table> <p style="margin-left: 100px;">(\$ 4 4 0 0)</p> <p>COMMAND PARAMETER - NON -</p>	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0		WORD NUMBER Wn=1	EXECUTION CYCLES Cn=12
0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0				
<p>&lt; DESCRIPTION &gt;</p> <p>RD reads one word (16 bits) of data from the frame buffer. The frame buffer address to be read must be predefined in the Read Write Pointer (RWP) before the RD command is issued. The results are loaded into the Read FIFO.</p> <p>The result may be read from the Read FIFO by the MPU anytime after the RD command is issued. If the Read FIFO is full when the command is executed, the ACRTC will enter a wait state until space becomes available in the Read FIFO.</p> <p>At the end of the RD command execution, the ACRTC increments RWP by one.</p>																			



Screen: 00

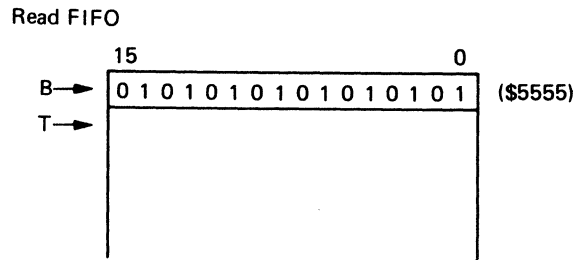
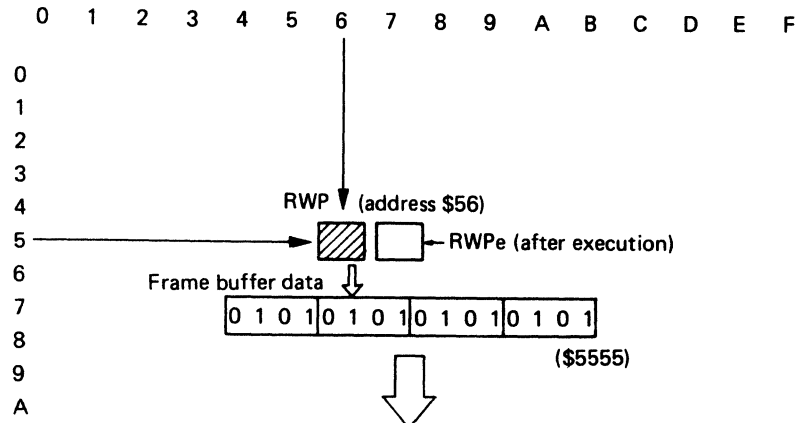


Figure C9-3 RD Execution Example

		WT						
[10] WT (Write)		PAGE	WT-1					
<p>&lt; FUNCTION &gt; Write one word of data to the frame buffer.</p> <p>&lt; MNEMONIC &gt; WT D</p>		TYPE	Data Transfer Command					
<p>&lt; FORMAT &gt;</p> <p>COMMAND CODE <span style="float: right;">hexadecimal notation</span></p> <p>15 <span style="float: right;">0</span></p> <table border="1" style="margin-left: 20px;"> <tr> <td style="width: 40px;">0 1 0 0</td> <td style="width: 40px;">1 0 0 0</td> <td style="width: 40px;">0 0 0 0</td> <td style="width: 40px;">0 0 0 0</td> </tr> </table> <p style="margin-left: 100px;">(\$ 4 8 0 0)</p> <p>COMMAND PARAMETERS</p> <p>15 <span style="float: right;">0</span></p> <table border="1" style="margin-left: 20px;"> <tr> <td style="width: 100px; text-align: center;">D (16 bits)</td> </tr> </table>		0 1 0 0	1 0 0 0	0 0 0 0	0 0 0 0	D (16 bits)	WORD NUMBER Wn=2	EXECUTION CYCLES Cn=8
0 1 0 0	1 0 0 0	0 0 0 0	0 0 0 0					
D (16 bits)								
<p>&lt; DESCRIPTION &gt;</p> <p>WT writes one word (16 bits) of data to the frame buffer. The frame buffer address to be written must be predefined in the Read Write Pointer (RWP) before the WT command is issued. The command parameter (D) is the data to be written.</p> <p>At the end of the WT command execution, the ACRTC increments the RWP by one.</p>								



WT

WT (Write)

PAGE

WT-2

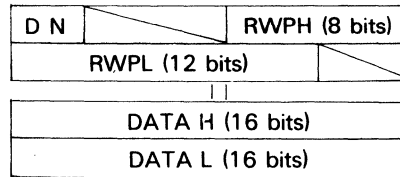
DN:

RWP

15

0

DN	Screen Number
00	Upper Screen
01	Base Screen
10	Lower Screen
11	Window Screen



- The frame memory is a 20-bit linear address separated into highorder RWPH (8 bits) and loworder RWPL (12 bits).
- Specify the Screen No. where drawing is executed.

Figure C10-1 RWP Set

< EXAMPLE >

Write the 16-bit data word \$5555 to frame buffer address \$56 on screen 0 (upper screen). For this example, Memory Width (MW) is assumed to be \$10.

RWP

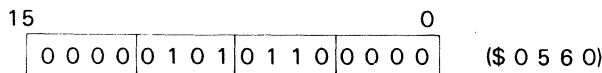
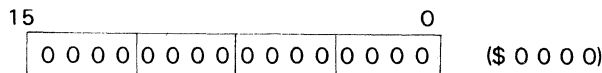
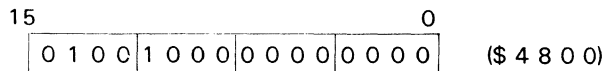
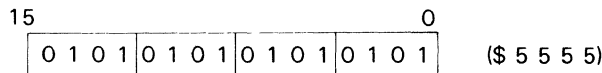


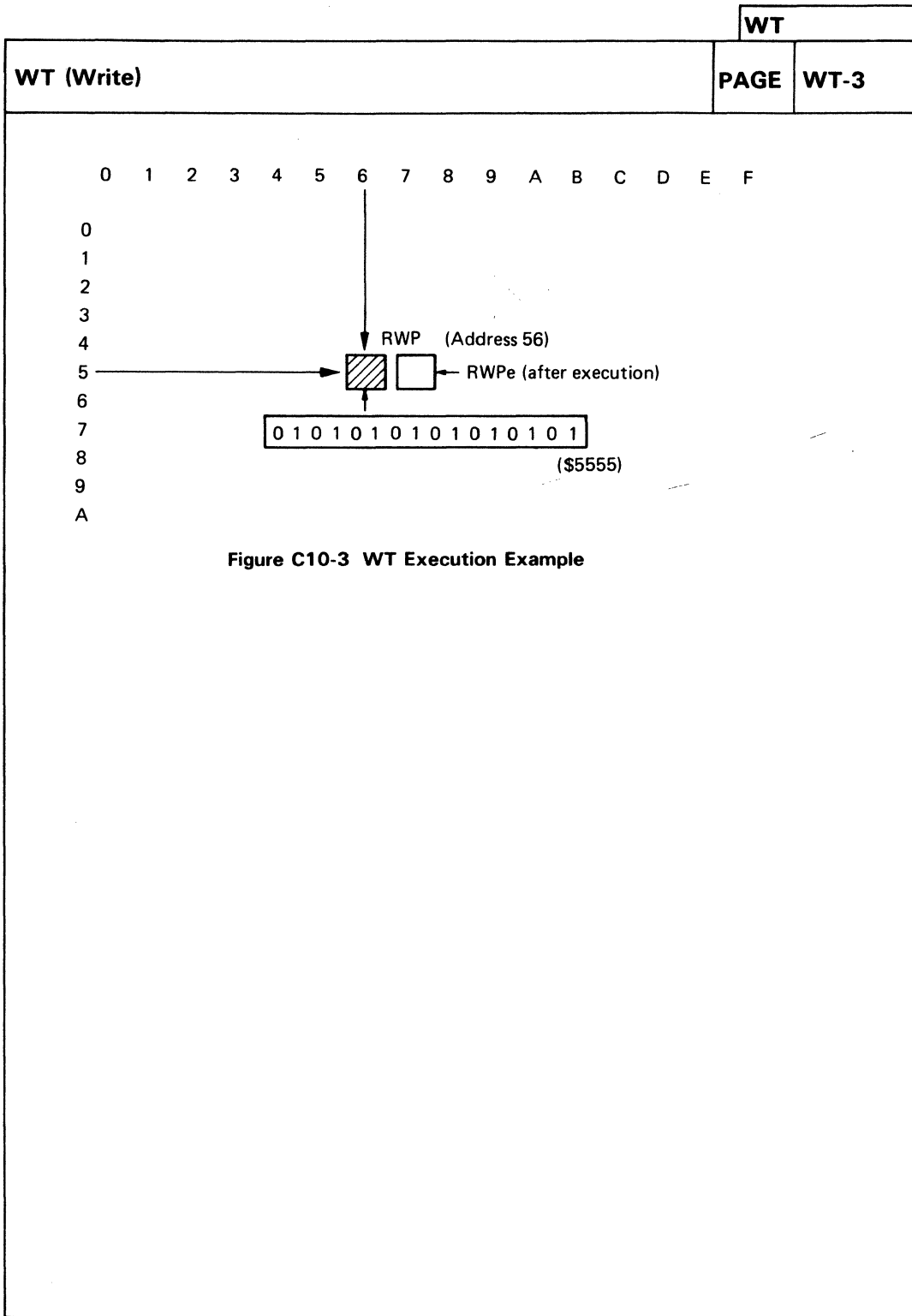
Figure C10-2 Example of RWP Setting

COMMAND CODE



COMMAND PARAMETERS





**Figure C10-3 WT Execution Example**

		MOD																	
[11] MOD (Modify)		PAGE	MOD-1																
<p>&lt; <b>FUNCTION</b> &gt; Perform logical operation on one word in the frame buffer.</p> <p>&lt; <b>MNEMONIC</b> &gt; MOD (MM) D</p>		TYPE	Data Transfer Command																
<p>&lt; <b>FORMAT</b> &gt;</p> <p>COMMAND CODE <span style="float: right;">hexadecimal notation</span></p> <p style="text-align: center;">15 <span style="margin-left: 150px;">2 1 0</span></p> <table border="1" style="margin-left: 20px;"> <tr> <td style="width: 20px;">0</td><td style="width: 20px;">1</td><td style="width: 20px;">0</td><td style="width: 20px;">0</td><td style="width: 20px;">1</td><td style="width: 20px;">1</td><td style="width: 20px;">0</td><td style="width: 20px;">0</td><td style="width: 20px;">0</td><td style="width: 20px;">0</td><td style="width: 20px;">0</td><td style="width: 20px;">0</td><td style="width: 20px;">0</td><td style="width: 20px;">0</td><td style="width: 20px;">MM</td> </tr> </table> <p style="margin-left: 100px;">(\$ 4 C O X)</p> <p>COMMAND PARAMETER</p> <p style="text-align: center;">15 <span style="margin-left: 150px;">0</span></p> <table border="1" style="margin-left: 20px;"> <tr> <td style="width: 150px; text-align: center;">D (16 bits)</td> </tr> </table>		0	1	0	0	1	1	0	0	0	0	0	0	0	0	MM	D (16 bits)	<p>WORD NUMBER W<sub>n</sub>=2</p> <p>EXECUTION CYCLES C<sub>n</sub>=8</p>	
0	1	0	0	1	1	0	0	0	0	0	0	0	0	MM					
D (16 bits)																			
<p>&lt; <b>DESCRIPTION</b> &gt;</p> <p>The MM (Modify Mode) field of the command code specifies the data transfer modify mode. This command performs logical operation on one word in the frame buffer with the data given the parameter and writes the result back in the frame buffer. The frame buffer word address to be modified must be predefined in the Read Write Pointer (RWP).</p> <p>The word is read from the frame buffer, then the logical operation defined by MM is performed between the data read from the frame buffer and the command parameter (D) for those bits not masked in the MASK register, and the result is rewritten to the frame buffer.</p> <p>At the end of the MOD command execution, the ACRTC increments the RWP by one.</p>																			



## &lt; EXECUTION EXAMPLE &gt;

RWP

15								0	
	0	0	0	0	0	0	0	0	0

 (\$ 0 0 0 0)

15								0	
	0	0	0	0	1	0	1	0	0

 (\$ 0 5 6 0)

MASK

15								0	
	1	1	1	1	1	1	1	1	1

 (\$ F F F F)

Figure C11-2 Examples of RWP and MASK Setting

COMMAND CODE

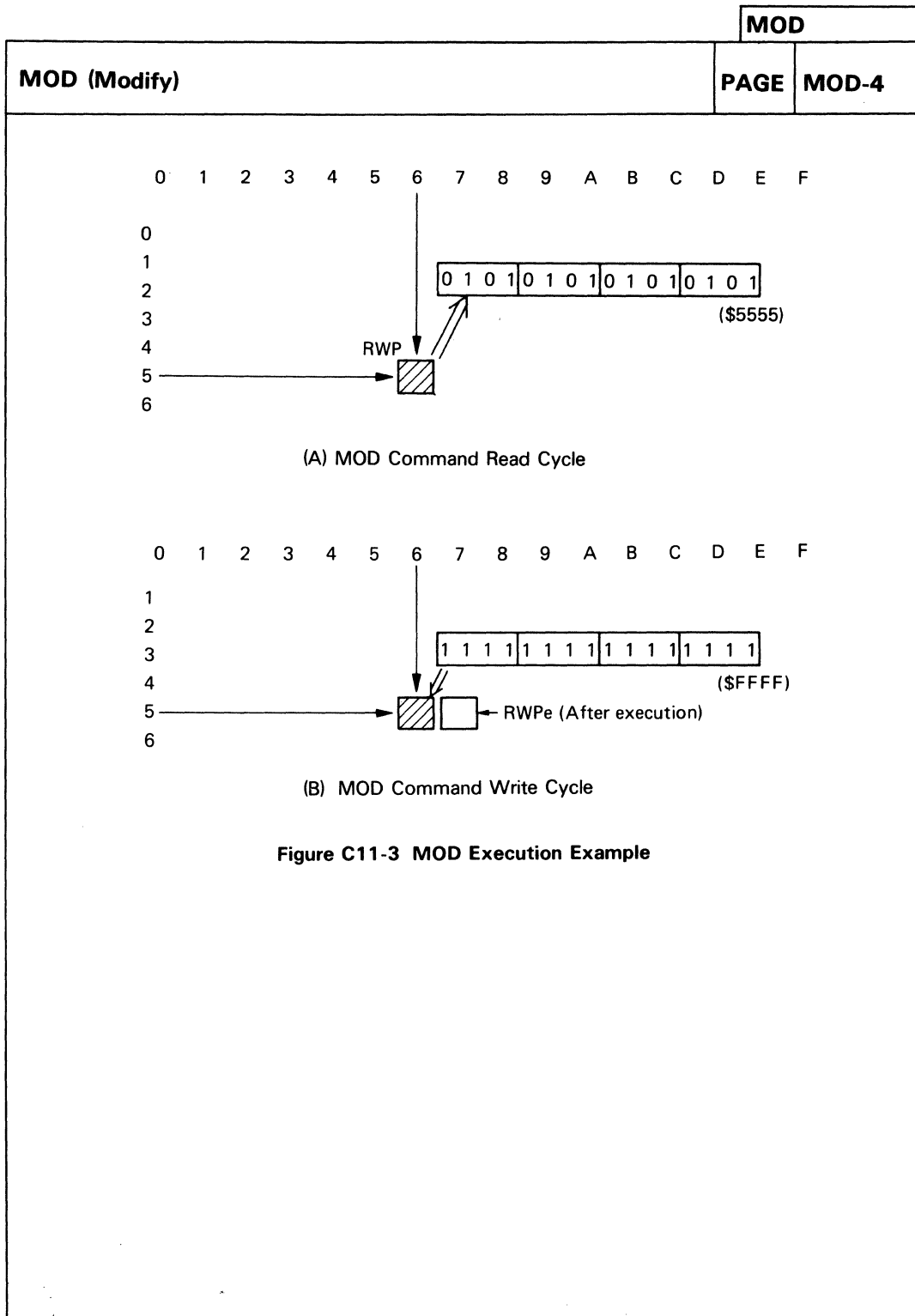
15									0
	0	1	0	0	1	1	0	0	0

 (\$ 4 C 0 1)

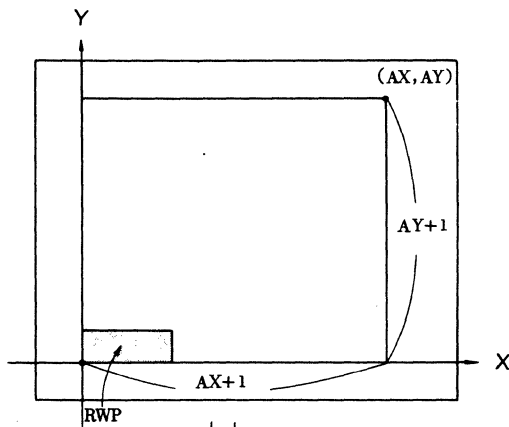
COMMAND PARAMETER

15								0	
	1	0	1	0	1	0	1	0	1

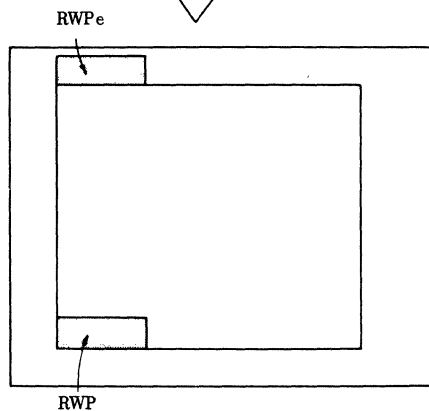
 (\$ A A A A)



		CLR								
[12] CLR (Clear)	PAGE	CLR-1								
<p>&lt; FUNCTION &gt; Initialize a frame buffer area with a data in the command parameter.</p> <p>&lt; MNEMONIC &gt; CLR D, AX, AY</p>	TYPE	Data Transfer Command								
<p>&lt; FORMAT &gt;</p> <p>COMMAND CODE <span style="float: right;">hexadecimal notation</span></p> <p>15 <span style="float: right;">0</span></p> <table border="1" style="margin-left: 40px;"> <tr> <td style="width: 40px;">0 1 0 1</td> <td style="width: 40px;">1 0 0 0</td> <td style="width: 40px;">0 0 0 0</td> <td style="width: 40px;">0 0 0 0</td> </tr> </table> <p style="margin-left: 100px;">(\$ 5 8 0 0)</p> <p>COMMAND PARAMETERS</p> <p>15 <span style="float: right;">0</span></p> <table border="1" style="margin-left: 40px;"> <tr> <td style="width: 100px; text-align: center;">D (16 bits)</td> </tr> </table> <p>15 <span style="float: right;">0</span></p> <table border="1" style="margin-left: 40px;"> <tr> <td style="width: 100px; text-align: center;">AX (16 bits)</td> </tr> </table> <p>15 <span style="float: right;">0</span></p> <table border="1" style="margin-left: 40px;"> <tr> <td style="width: 100px; text-align: center;">AY (16 bits)</td> </tr> </table>	0 1 0 1	1 0 0 0	0 0 0 0	0 0 0 0	D (16 bits)	AX (16 bits)	AY (16 bits)	WORD NUMBER W <sub>n</sub> =4	EXECUTION CYCLES C <sub>n</sub> = (2x + 8)y + 12 $\begin{cases} x =  AX  + 1 \\ y =  AY  + 1 \end{cases}$	
0 1 0 1	1 0 0 0	0 0 0 0	0 0 0 0							
D (16 bits)										
AX (16 bits)										
AY (16 bits)										
<p>&lt; DESCRIPTION &gt;</p> <p>The frame buffer area defined by the physical origin (RWP) and physical frame buffer word address (AX and AY) parameters is filled with the data parameter (D).</p> <p>Since the ACRTC performs the clear using 16 bit words, multiple logical pixels (if 4 bits/pixel then 4 pixels) are cleared in one access. D is normally specified to contain multiple copies (if 4 bits/pixel then 4 copies) of the color information for a single color clear.</p> <p>At the end of CLR command execution, RWP will be set to RWPe.</p>										



AX: 2nd parameter  
AY: 3rd parameter  
(4-bits/pixel)



RWP is set with a 2-word  
(32-bit) data, as shown in  
Fig. C12-1.

The RWP needs to be specified in advance as follows.

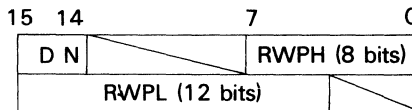


**CLR (Clear)**

DN:

DN	Screen Number
00	Upper Screen
01	Base Screen
10	Lower Screen
11	Window Screen

RWP:



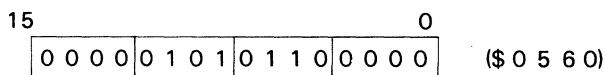
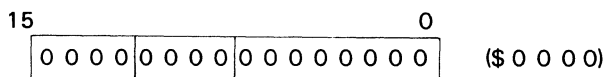
- The frame buffer 20-bit linear address is separated into high order RWPH (8 bits) and low order RWPL (12 bits).
- Specify the Screen No. where drawing is executed.

**Figure C12-1 RWP Set**

**< EXAMPLE >**

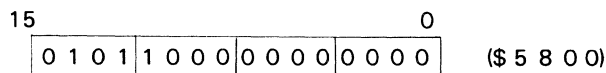
For this example 4 bits per logical pixel is used, the Memory Width (MW) is \$10 and the clear operation is to start at address \$56 on screen 0.

RWP

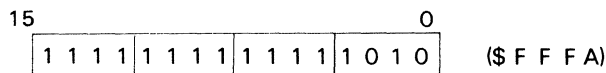
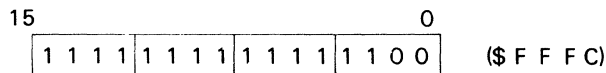
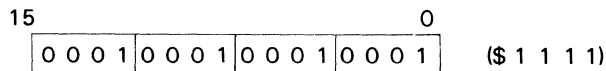


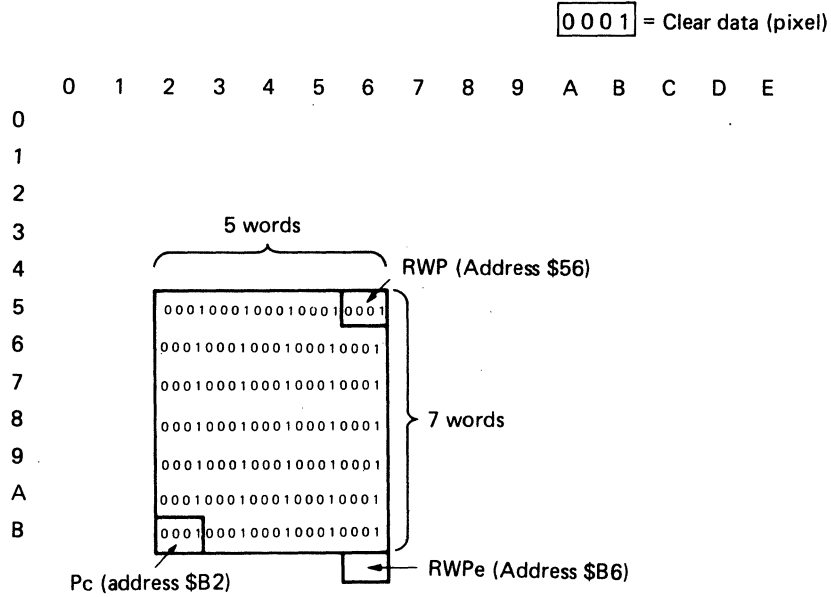
**Figure C12-2 Example of RWP Setting**

COMMAND CODE



COMMAND PARAMETERS





**Figure C12-3 CLR Execution Example**

**SCLR**

[13] SCLR (Selective Clear)	PAGE	SCLR-1																										
<p>&lt; <b>FUNCTION</b> &gt;            Initialize a frame buffer area with a constant value subject to logical modification.</p> <p>&lt; <b>MNEMONIC</b> &gt;            SCLR (MM) D, AX, AY</p>	TYPE	Data Transfer Command																										
<p>&lt; <b>FORMAT</b> &gt;</p> <p>COMMAND CODE <span style="float: right;">hexadecimal notation</span></p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: center; width: 15%;">15</td> <td style="width: 80%;"></td> <td style="text-align: center; width: 5%;">0</td> </tr> <tr> <td></td> <td style="text-align: center; border: 1px solid black;">0 1 0 1 1 1 0 0 0 0 0 0 0 0 MM</td> <td></td> </tr> <tr> <td></td> <td style="text-align: center;">(\$ 5 C 0 X)</td> <td></td> </tr> </table> <p>COMMAND PARAMETERS</p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: center; width: 15%;">15</td> <td style="width: 80%;"></td> <td style="text-align: center; width: 5%;">0</td> </tr> <tr> <td></td> <td style="text-align: center; border: 1px solid black;">D (16 bits)</td> <td></td> </tr> <tr> <td style="text-align: center;">15</td> <td style="width: 80%;"></td> <td style="text-align: center;">0</td> </tr> <tr> <td></td> <td style="text-align: center; border: 1px solid black;">AX (16 bits)</td> <td></td> </tr> <tr> <td style="text-align: center;">15</td> <td style="width: 80%;"></td> <td style="text-align: center;">0</td> </tr> <tr> <td></td> <td style="text-align: center; border: 1px solid black;">AY (16 bits)</td> <td></td> </tr> </table>	15		0		0 1 0 1 1 1 0 0 0 0 0 0 0 0 MM			(\$ 5 C 0 X)		15		0		D (16 bits)		15		0		AX (16 bits)		15		0		AY (16 bits)		<p>WORD NUMBER  <math>W_n = 4</math></p> <p>EXECUTION CYCLES  <math>C_n = (4x + 6)y + 12</math></p> $\begin{cases} x =  AX  + 1 \\ y =  AY  + 1 \end{cases}$
15		0																										
	0 1 0 1 1 1 0 0 0 0 0 0 0 0 MM																											
	(\$ 5 C 0 X)																											
15		0																										
	D (16 bits)																											
15		0																										
	AX (16 bits)																											
15		0																										
	AY (16 bits)																											
<p>&lt; <b>DESCRIPTION</b> &gt;</p> <p>The MM (Modify Mode) field of the command code specifies the data transfer modify mode.</p> <p>The frame buffer area defined by the RWP origin and the physical frame buffer word address (AX and AY) parameters is selectively cleared. The contents of the frame buffer are read, and that data is logically operated on with the D parameter (excepts bits masked in the MASK register) using the logical operation defined by MM. The result is rewritten to the frame buffer.</p> <p>Since the ACRTC performs the selective clear using 16-bit words, multiple logical pixels (if 4 bits/pixel then 4 pixels) are cleared in one access. D is normally specified to contain multiple copies (if 4 bits/pixel then 4 copies) of the color information for a single color selective clear.</p> <p>At the end of SCLR command execution, RWP will be set to RWPe.</p>																												

< DESCRIPTION >

- : Modifier information
  - : 1st parameter
  - 2 : 2nd parameter
  - 4 : 3rd parameter
- } in units of words

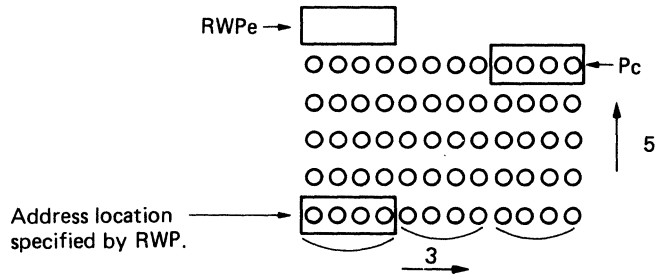


Figure C13-1 Command Parameter Set

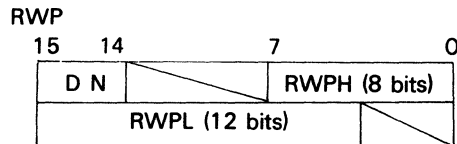
The operation is specified by the above operation mode, and is set with bits 1, 0 in the command code.

This command can be utilized for clearing the character code, the specific attribute bits, and the specific color plane in the graphic display.

The RWP needs to be specified in advance as follows.

DN:

DN	Screen Number
00	Upper Screen
01	Base Screen
10	Lower Screen
11	Window Screen



- The frame memory is a 20-bit linear address separated into high order RWP (8 bits) and low order RWPL (12 bits).
- Specify the Screen No. where drawing is executed.

Figure C13-2 RWP Set

## &lt; EXAMPLE &gt;

For this example 4 bits per logical pixel is used, the Memory Width (MW) is \$10, the MASK register contains \$FOFO and the selective clear operation is to start at address \$56 on screen 0.

Based on MM, a logical operation (REPLACE, OR, AND or EOR) is defined and SCLR is executed as shown.

## RWP

15	0	
0	0	(\$ 0 0 0 0)
0	0	(\$ 0 5 6 0)

## MASK

15	0	
1	1	(\$ F 0 F 0)

Figure C13-3 Examples of RWP and MASK Setting

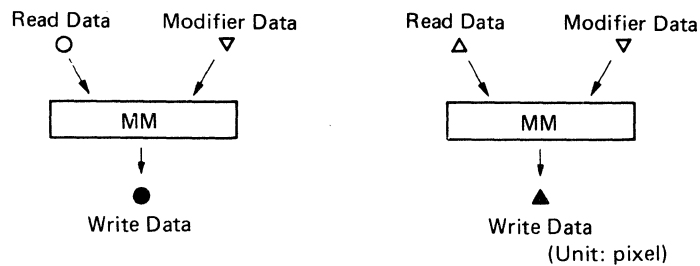


Figure C13-4 Notation of Data

< EXECUTION EXAMPLE >

COMMAND CODE

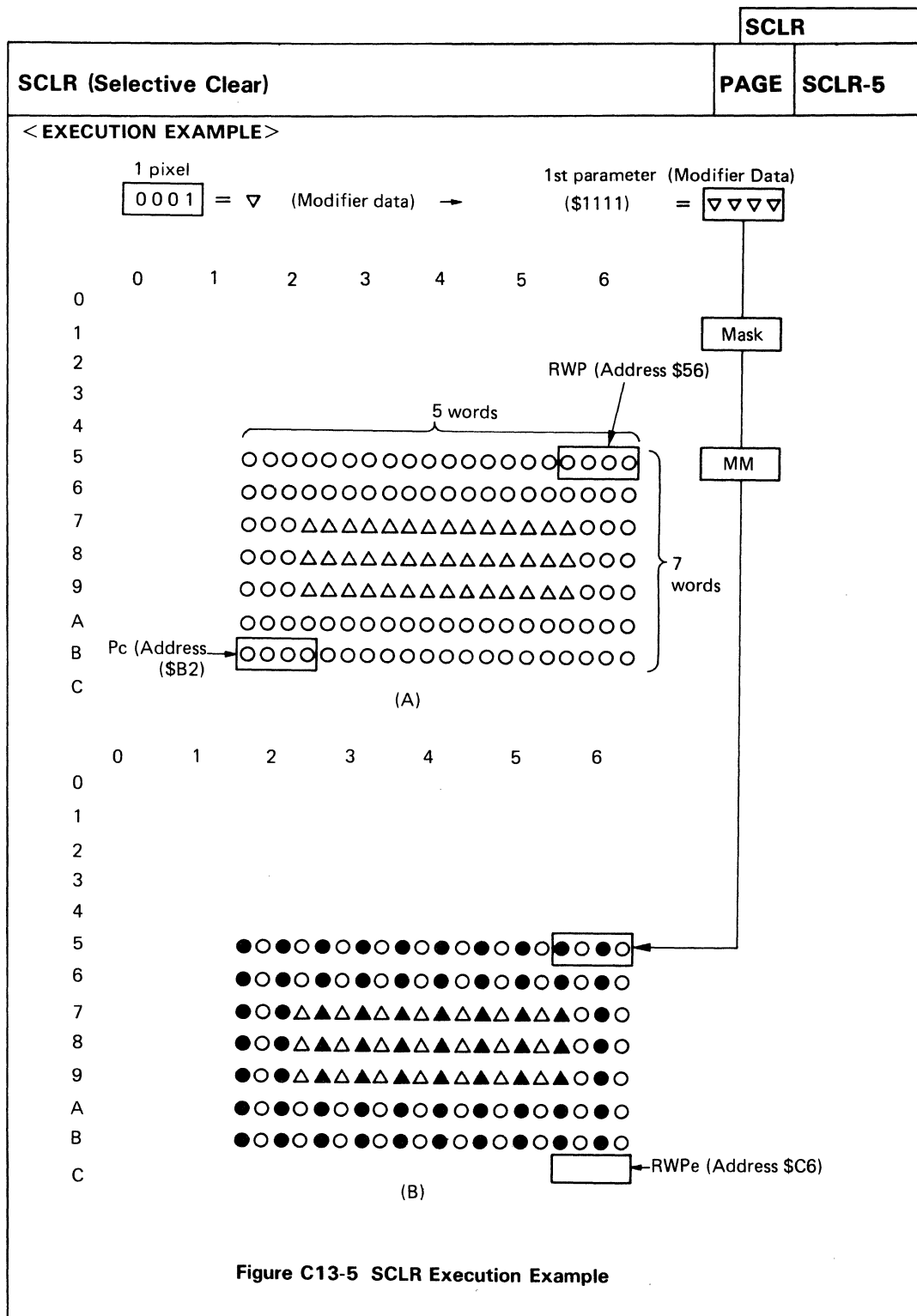
15 0  
0 1 0 1 1 1 0 0 0 0 0 0 0 0 MM (\$ 5 C 0 X)

COMMAND PARAMETERS

15 0  
0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 (\$ 1 1 1 1)

15 0  
1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 (\$ F F F C)

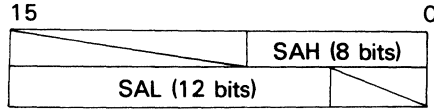
15 0  
1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 (\$ F F F A)



		CPY																			
[14] CPY (Copy)	PAGE	CPY-1																			
<p>&lt; FUNCTION &gt; Copy frame buffer data from one area (source area) to another area (destination area).</p> <p>&lt; MNEMONIC &gt; CPY (S, DSD) SAH, SAL, AX, AY</p>		TYPE	Data Transfer Command																		
<p>&lt; FORMAT &gt;</p> <p>COMMAND CODE <span style="float:right">hexadecimal notation</span></p> <p>15            12 11 10 8 7            0</p> <table border="1" style="margin-left: 40px;"> <tr> <td style="width: 40px;">0 1 1 0</td> <td style="width: 40px;">S</td> <td style="width: 40px;">D</td> <td style="width: 40px;">S</td> <td style="width: 40px;">D</td> <td style="width: 40px;">0 0 0 0</td> <td style="width: 40px;">0 0 0 0</td> <td style="width: 40px;">0</td> </tr> </table> <p style="margin-left: 100px;">(\$ 6 X 0 0)</p> <p>COMMAND PARAMETERS</p> <p>15    8 7            0</p> <table border="1" style="margin-left: 40px;"> <tr> <td style="width: 80px;">0 0 0 0 0 0 0 0</td> <td style="width: 40px;">SAH (8 bits)</td> <td style="width: 40px;">0</td> </tr> </table> <p>15    0</p> <table border="1" style="margin-left: 40px;"> <tr> <td style="width: 120px;">SAL (12 bits)</td> <td style="width: 40px;">0 0 0 0</td> <td style="width: 40px;">0</td> </tr> </table> <p>15    0</p> <table border="1" style="margin-left: 40px;"> <tr> <td style="width: 160px;">AX (16 bits)</td> <td style="width: 40px;">0</td> </tr> </table> <p>15    0</p> <table border="1" style="margin-left: 40px;"> <tr> <td style="width: 160px;">AY (16 bits)</td> <td style="width: 40px;">0</td> </tr> </table>		0 1 1 0	S	D	S	D	0 0 0 0	0 0 0 0	0	0 0 0 0 0 0 0 0	SAH (8 bits)	0	SAL (12 bits)	0 0 0 0	0	AX (16 bits)	0	AY (16 bits)	0	WORD NUMBER Wn=5	EXECUTION CYCLES Cn=(6x+10)y+12 $\begin{cases} x =  AX  + 1 \\ y =  AY  + 1 \end{cases}$
0 1 1 0	S	D	S	D	0 0 0 0	0 0 0 0	0														
0 0 0 0 0 0 0 0	SAH (8 bits)	0																			
SAL (12 bits)	0 0 0 0	0																			
AX (16 bits)	0																				
AY (16 bits)	0																				
<p>&lt; DESCRIPTION &gt;</p> <p>The parameters to the command define the source area. The RWP must be predefined to point to the destination area (including screen number). The source area resides in the same screen as that of the destination area as defined in RWP.</p> <p>The source area is defined by the origin address (SAH/SAL) and physical frame buffer word (AX and AY) dimensions.</p> <p>To allow rotation and proper operation for overlapping during copying, the command code contains fields which define the source and destination scanning direction. The S (Source Scan Direction) and DSD (Destination Scan Direction) fields of the command code define the source and destination scanning direction respectively as shown next page.</p> <p>At the end of the CPY command, RWP is set to RWPe.</p>																					



Pss:

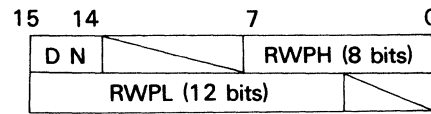


- (1) Pss (SAH, SAL) is set to be a 20-bit linear address separated into 2 words, high order SAH (8 bits) and low order SAL (12 bits).

DN:

DN	Screen Number
00	Upper Screen
01	Base Screen
10	Lower Screen
11	Window Screen

RWP:



The frame buffer 20-bit linear address is separated into high order RWPH (8 bits) and low order RWPL (12 bits).

Specify the Screen No. where drawing is executed.

Figure C14-1 Pss and RWP Set

< CPY Command Scan Direction >

As to CPY, the direction of pointer scanning is specified in command code. (The pointer functions in the unit of word).

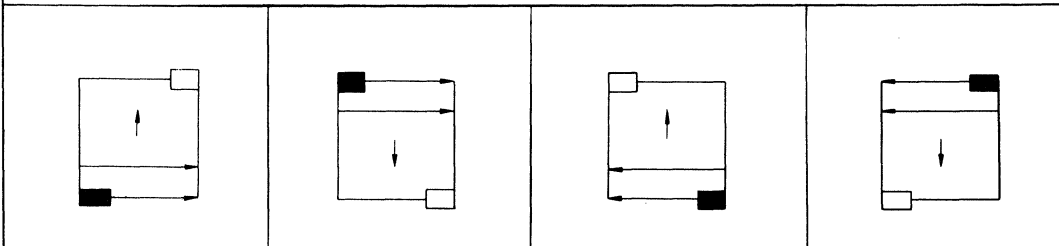
(a) Scanning Direction of Source Area (S: Source Scan Direction)

COMMAND CODE

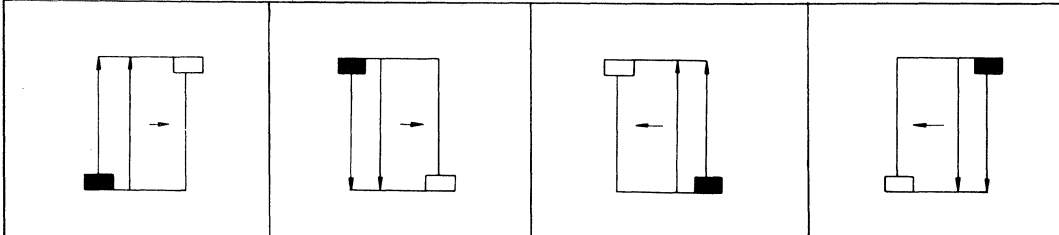


Table C14-1 Source Scan Direction

S = 0



S = 1

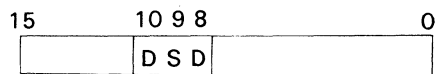


■ : Pss □ : Pse

As shown in Table C14-1, the scanning direction in frame buffer of the copy source area is decided by the relation between bit 11 in the command code and the Pss and the Pse.

(a) Scanning Direction of Destination Area (DSD: Destination Scan Direction)

COMMAND CODE



CPY			CPY
CPY (Copy)			PAGE CPY-4
Table C14-2 Destination Scan Direction			
DSD = 000	DSD = 001	DSD = 010	DSD = 011
DSD = 100	DSD = 101	DSD = 110	DSD = 111

As shown in Table C14-2, the scanning direction in frame buffer of the destination area is decided by the relation between bit 10 to 8 in the command code and the RWP.

Upon termination of the command, RWPe, end point of the RWP moves as shown in Table C14-2.

#### Relation to Linear Address

Fig. C14-2 provides the relation between CPY and specified value when  $S = 1$  and  $DSD = 000$ .

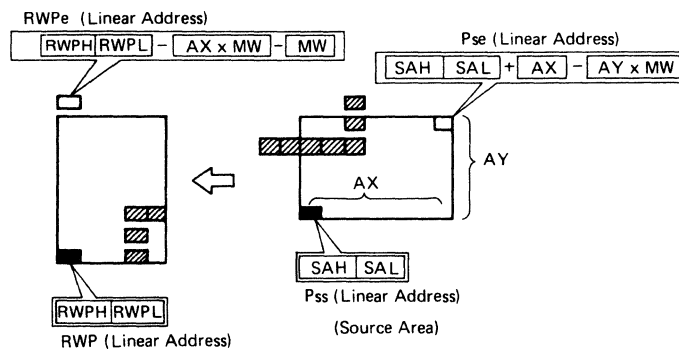


Figure C14-2 Relations with Linear Addresses

< EXAMPLE >

For this example 4 bits per logical pixel is used, the Memory Width (MW) is \$10 and the copy operation source area (SAH/SAL) start is frame buffer address \$89 while the copy destination area (RWP) start is frame buffer address \$B0 on screen 0.

The source area scanning direction is specified as S = 1 and the destination area scanning direction is specified as DSD = 000.

RWP

15 0  

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

 (\$ 0 0 0 0)

15 0  

0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

 (\$ 0 B 0 0)

Figure C14-3 Example of Read Write Pointer Setting

COMMAND CODE

15 0  

0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

 (\$ 6 8 0 0)

COMMAND PARAMETERS

15 0  

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

 (\$ 0 0 0 0)

15 0  

0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

 (\$ 0 8 9 0)

15 0  

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

 (\$ 0 0 0 3)

15 0  

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

 (\$ 0 0 0 6)

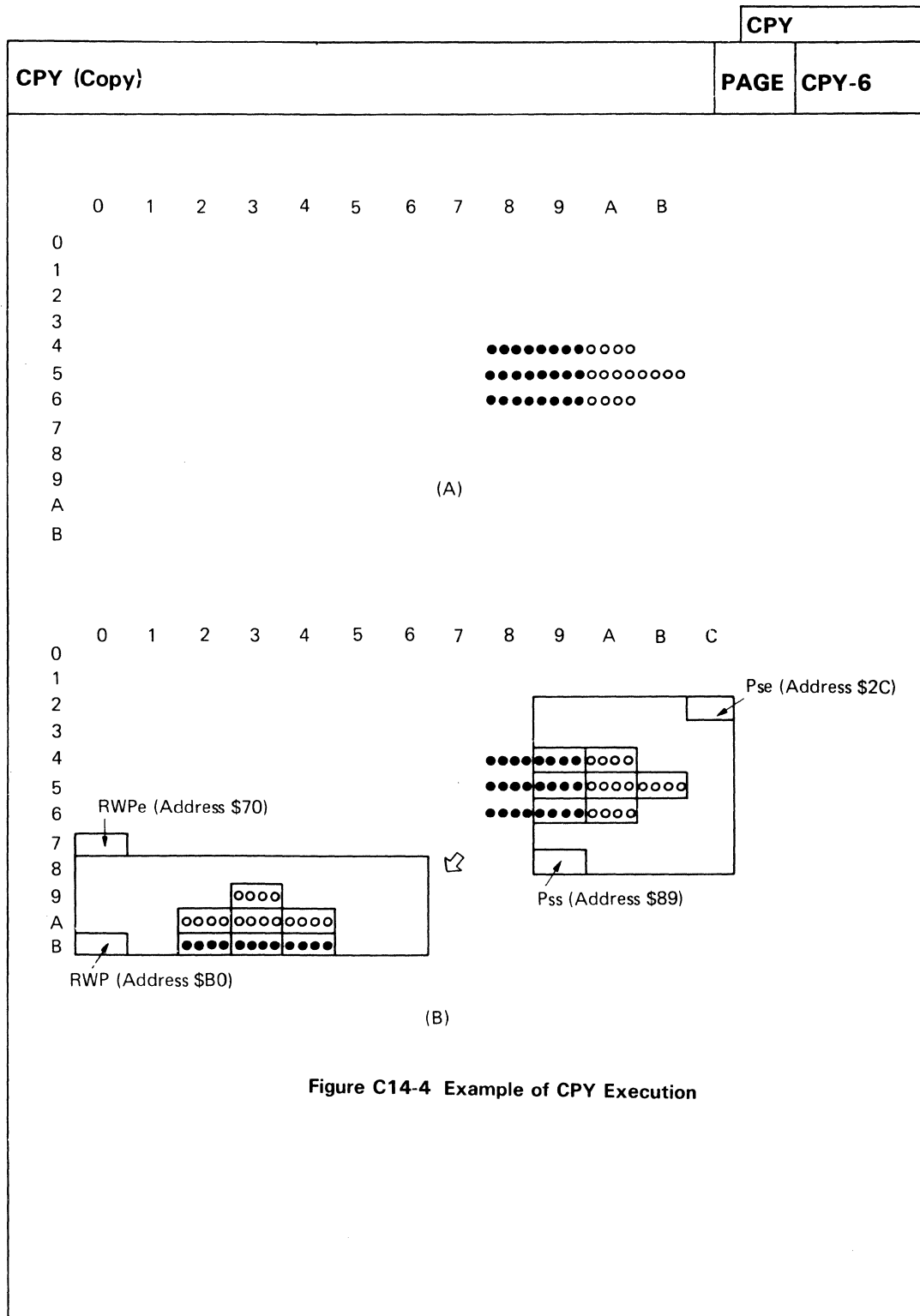


Figure C14-4 Example of CPY Execution



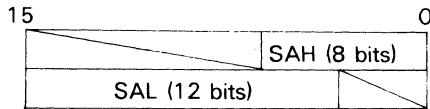
SCPY (Selective Copy)

PAGE

SCPY-2

The source address and Read/Write Pointer need to be specified as follows prior to the execution.

Pss:

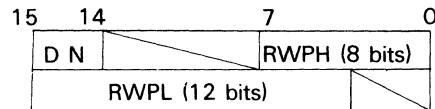


- (1) Pss (SAH, SAL) is set to be a 20-bit linear address separated into 2 words, high order SAH (8 bits) and low order SAL (12 bits).

DN:

DN	Screen Number
00	Upper Screen
01	Base Screen
10	Lower Screen
11	Window Screen

RWP:



The frame buffer 20-bit linear address is separated into high order RWPH (8 bits) and low order RWPL (12 bits).

Specify the Screen No. where drawing is executed.

Figure C15-1 P<sub>SS</sub> and RWP Set

< SCPY Command Scan Direction >

As to SCPY, the direction of pointer scanning is specified in command code. (The pointer functions in the unit of word)

(a) Scanning Direction of Source Area (S: Source Scan Direction)

COMMAND CODE



Table C15-1 Source Scan Direction

S = 0			
S = 1			
		■ : Pss	□ : Pse

As shown in Table C15-1, the scanning direction in frame buffer of the copy source area is decided by the relation between bit 11 in the command code and the Pss and the Pse.



SCPY (Selective Copy)

(b) Scanning Direction of Destination Area (DSD: Destination Scan Direction)

COMMAND CODE

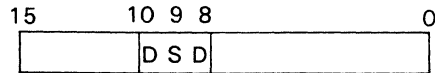


Table C15-2 Destination Scan Direction

DSD = 000	DSD = 001	DSD = 010	DSD = 011
DSD = 100	DSD = 101	DSD = 110	DSD = 111

■ : RWP □ : RWPe

As shown in Table C15-2, the scanning direction in frame buffer of the destination area is decided by the relation between bit 10 to 8 in the command code and the RWP.

Upon termination of the command, RWPe, end point of the RWP moves as shown in Table C15-2.

The operation is decided by the modify mode (MM) and is specified by bit "0" or "1" in the command code.

Relation to Linear Address

Fig. C15-2 provides the relation between SCPY and specified value when  $S = 1$  and  $DSD = 000$ .

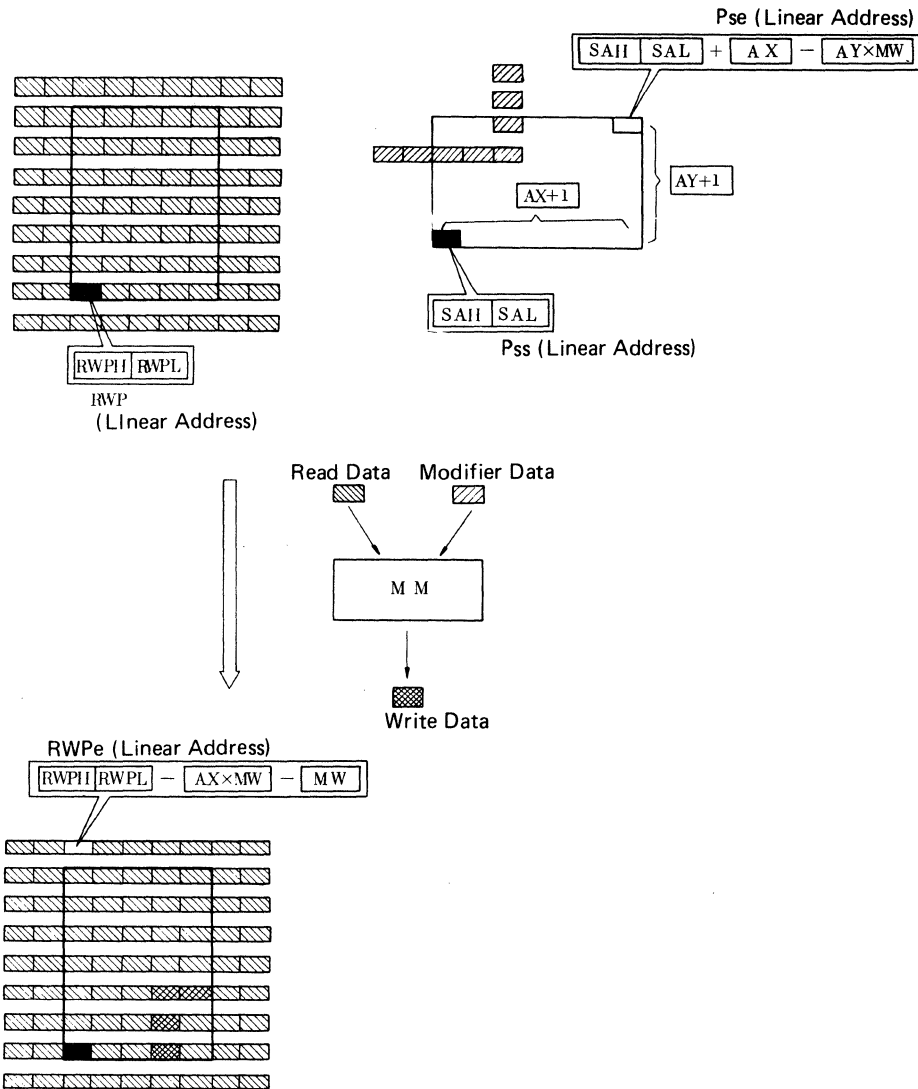


Figure C15-2 Relations with Linear Addresses

**SCPY (Selective Copy)**

PAGE

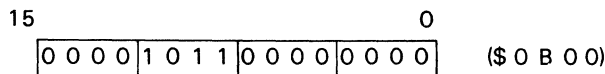
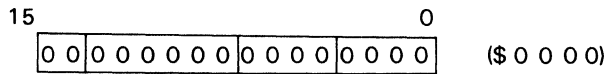
SCPY-6

<EXAMPLE>

For this example 4 bits per logical pixel is used, the Memory Width (MW) is \$10, the MASK register contains \$FOFO and the copy operation source area (SAH/SAL) start is frame buffer address \$85 while the copy destination area (RWP) start is frame buffer address \$B0 on screen 0.

The source area scanning direction is specified as S = 1 and the destination area scanning direction is specified as DSD = 000.

RWP



MASK

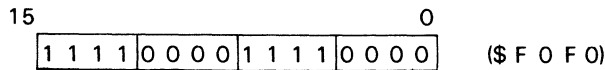


Figure C15-3 RWP and MASK Setting

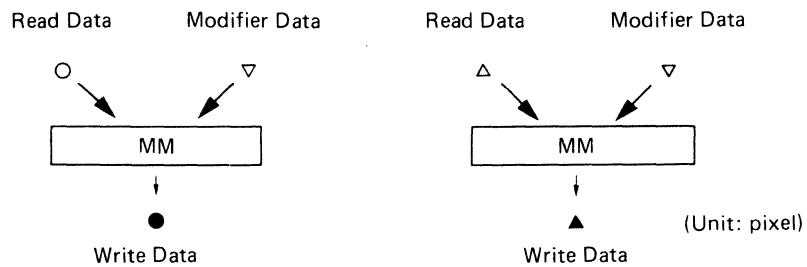


Figure C15-4 Operation of SCPY

SCPY (Selective Copy)

COMMAND CODE

15										0	
	0	1	1	1	1	0	0	0	0	0	MM

(\$ 7 8 0 X)

COMMAND PARAMETERS

15											0
	0	0	0	0	0	0	0	0	0	0	0

(\$ 0 0 0 0)

15											0
	0	0	0	0	1	0	0	0	0	1	0

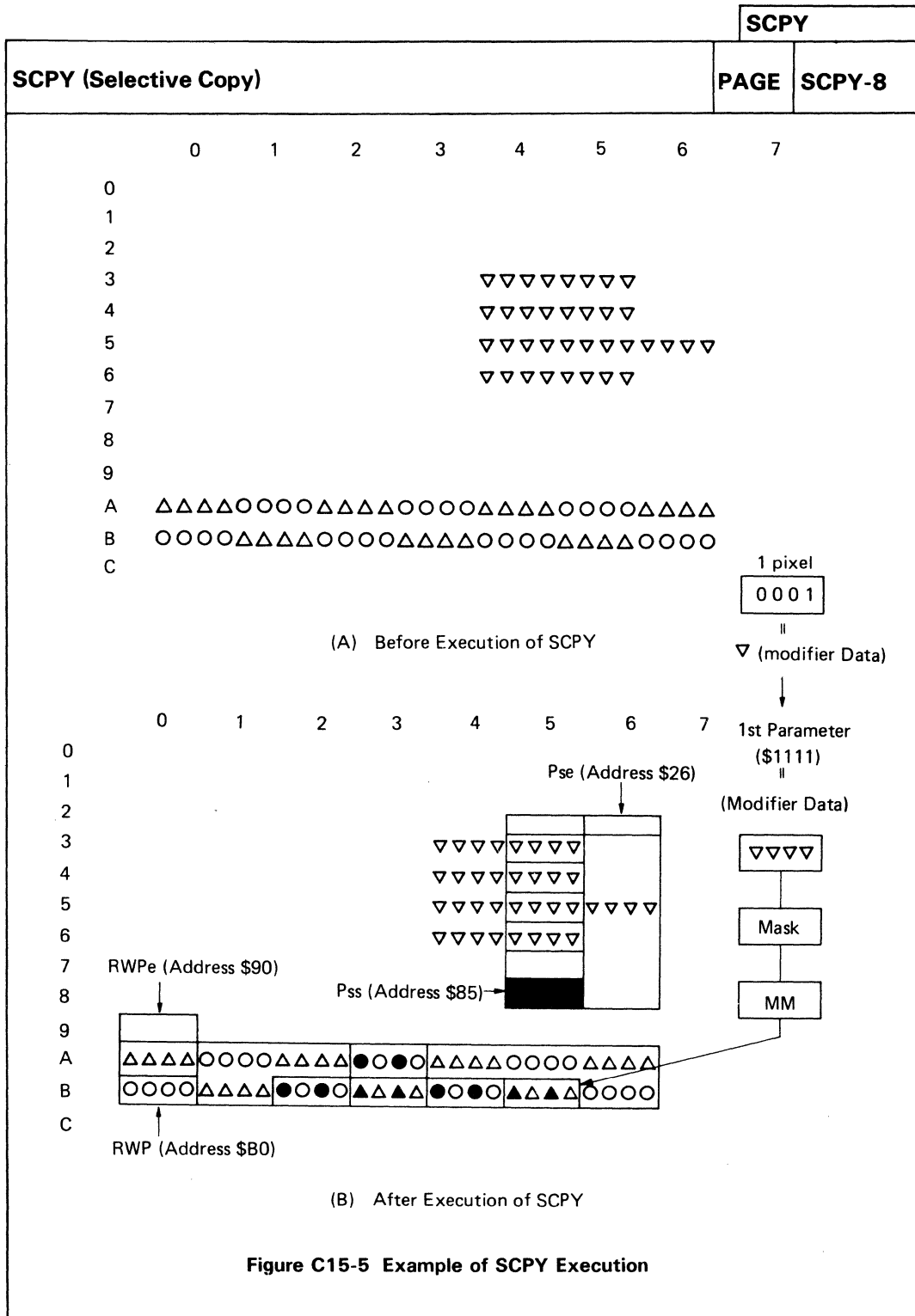
(\$ 0 8 5 0)

15											0
	0	0	0	0	0	0	0	0	0	0	1

(\$ 0 0 0 1)

15											0
	0	0	0	0	0	0	0	0	0	0	1

(\$ 0 0 0 6)



**AMOVE**

<p><b>[16] AMOVE (Absolute Move)</b></p>	<p><b>PAGE</b></p>	<p><b>AMOVE-1</b></p>																																																
<p>&lt; <b>FUNCTION</b> &gt; Move the Current Pointer (CP) to an absolute logical pixel X-Y address.</p> <p>&lt; <b>MNEMONIC</b> &gt; AMOVE X, Y</p>	<p><b>TYPE</b></p>	<p>Graphic Command</p>																																																
<p>&lt; <b>FORMAT</b> &gt;</p> <p><b>COMMAND CODE</b> <span style="float: right;">hexadecimal notation</span></p> <p>15 <span style="float: right;">0</span></p> <table border="1" style="width: 100%; text-align: center;"> <tr> <td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td> </tr> </table> <p style="text-align: right;">(\$ 8 0 0 0)</p> <p><b>COMMAND PARAMETERS</b></p> <p>15 <span style="float: right;">0</span></p> <table border="1" style="width: 100%; text-align: center;"> <tr> <td colspan="16">X (16 bits)</td> </tr> </table> <p>15 <span style="float: right;">0</span></p> <table border="1" style="width: 100%; text-align: center;"> <tr> <td colspan="16">Y (16 bits)</td> </tr> </table>	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X (16 bits)																Y (16 bits)																<p><b>WORD NUMBER</b> Wn=3</p> <p><b>EXECUTION CYCLES</b> Cn=56</p>	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																																			
X (16 bits)																																																		
Y (16 bits)																																																		
<p>&lt; <b>DESCRIPTION</b> &gt;</p> <p>The parameters (X, Y) of the AMOVE command specify the new value for the CP. The address is specified using logical pixel X-Y addresses relative to the origin defined by the ORG command.</p> <div style="text-align: center;"> </div> <p><b>Figure C16-1 Function of AMOVE Command</b></p>																																																		

**AMOVE (Absolute Move)**

**< EXAMPLE >**

If CP = (-13, -10) and AMOVE command is executed with parameters (X, Y) = (10, 2), then the CP is set to Pe as shown below.

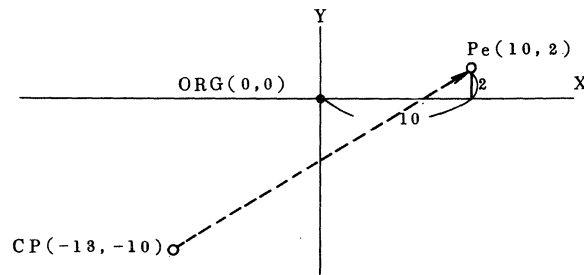
**COMMAND CODE**

15	0
1 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
	(\$ 8 0 0 0)

**COMMAND PARAMETERS**

15	0
1 0 0 0	0 0 0 0 0 0 0 0 0 1 0 1 0
	(\$ 0 0 0 A)

15	0
0 0 0 0	0 0 0 0 0 0 0 0 0 0 1 0
	(\$ 0 0 0 2)



**Figure C16-2 Example of AMOVE Execution**

**RMOVE**

<p><b>[17] RMOVE (Relative Move)</b></p>	<p><b>PAGE</b></p>	<p><b>RMOVE-1</b></p>																		
<p>&lt; <b>FUNCTION</b> &gt; Move the Current Pointer (CP) to a relative logical pixel X-Y address.</p> <p>&lt; <b>MNEMONIC</b> &gt; RMOVE dX, dY</p>	<p>TYPE</p>	<p>Graphic Command</p>																		
<p>&lt; <b>FORMAT</b> &gt;</p> <p>COMMAND CODE <span style="float: right;">hexadecimal notation</span></p> <p>15 <span style="float: right;">0</span></p> <table border="1" style="margin-left: 20px;"> <tr> <td style="width: 40px;">1</td><td style="width: 40px;">0</td><td style="width: 40px;">0</td><td style="width: 40px;">0</td><td style="width: 40px;">0</td><td style="width: 40px;">1</td><td style="width: 40px;">0</td><td style="width: 40px;">0</td><td style="width: 40px;">0</td><td style="width: 40px;">0</td><td style="width: 40px;">0</td><td style="width: 40px;">0</td><td style="width: 40px;">0</td><td style="width: 40px;">0</td><td style="width: 40px;">0</td><td style="width: 40px;">0</td> </tr> </table> <p style="margin-left: 100px;">(\$ 8 4 0 0)</p> <p>COMMAND PARAMETERS</p> <p>15 <span style="float: right;">0</span></p> <table border="1" style="margin-left: 20px;"> <tr> <td style="width: 100px; text-align: center;">dX (16 bits)</td> </tr> </table> <p>15 <span style="float: right;">0</span></p> <table border="1" style="margin-left: 20px;"> <tr> <td style="width: 100px; text-align: center;">dY (16 bits)</td> </tr> </table>	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	dX (16 bits)	dY (16 bits)	<p>WORD NUMBER Wn=3</p> <p>EXECUTION CYCLES Cn=56</p>	
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0					
dX (16 bits)																				
dY (16 bits)																				
<p>&lt; <b>DESCRIPTION</b> &gt;</p> <p>The parameters (dX, dY) of the RMOVE command are used to calculate the new value for the CP. The address is specified using logical pixel X-Y displacements relative to CP.</p> <div style="text-align: center;"> </div> <p><b>Figure C17-1 Function of RMOVE</b></p>																				

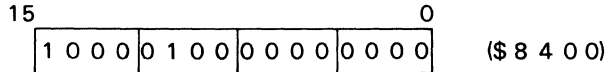


**RMOVE (Relative Move)**

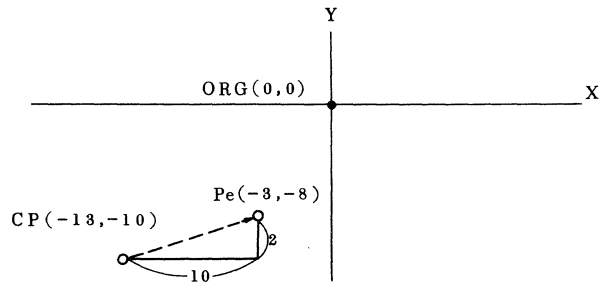
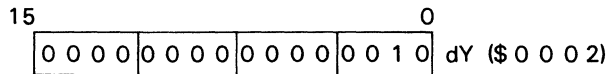
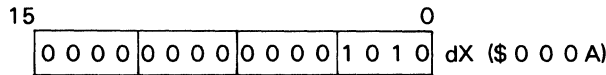
< EXAMPLE >

If CP = (-13, -10) and RMOVE command is executed with parameters (X, Y) = (10, 2), then the CP is set to Pe as shown below.

COMMAND CODE



COMMAND PARAMETERS



**Figure C17-2 Example of RMOVE Execution**

<b>ALINE</b>																											
<b>[18] ALINE (Absolute Line)</b>	<b>PAGE</b>	<b>ALINE-1</b>																									
<p>&lt; <b>FUNCTION</b> &gt;            Draw a straight line from CP to a command specified end point.</p> <p>&lt; <b>MNEMONIC</b> &gt;            ALINE (AREA, COL, OPM) X, Y</p>	<b>TYPE</b>	Graphic Command																									
<p>&lt; <b>FORMAT</b> &gt;</p> <p>COMMAND CODE <span style="float: right;">hexadecimal notation</span></p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%; text-align: right;">15</td> <td style="width: 15%;"></td> <td style="width: 15%; text-align: right;">8 7</td> <td style="width: 15%; text-align: right;">5 4</td> <td style="width: 15%; text-align: right;">3 2</td> <td style="width: 15%; text-align: right;">0</td> <td style="width: 15%;"></td> </tr> <tr> <td style="border: 1px solid black; text-align: center;">1 0 0 0</td> <td style="border: 1px solid black; text-align: center;">1 0 0 0</td> <td style="border: 1px solid black; text-align: center;">AREA</td> <td style="border: 1px solid black; text-align: center;">COL</td> <td style="border: 1px solid black; text-align: center;">OPM</td> <td></td> <td style="text-align: center;">(\$ 8 8 X X)</td> </tr> </table> <p>COMMAND PARAMETERS</p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%; text-align: right;">15</td> <td style="width: 70%;"></td> <td style="width: 15%; text-align: right;">0</td> </tr> <tr> <td></td> <td style="border: 1px solid black; text-align: center;">X (16 bits)</td> <td></td> </tr> <tr> <td style="text-align: right;">15</td> <td></td> <td style="text-align: right;">0</td> </tr> <tr> <td></td> <td style="border: 1px solid black; text-align: center;">Y (16 bits)</td> <td></td> </tr> </table>	15		8 7	5 4	3 2	0		1 0 0 0	1 0 0 0	AREA	COL	OPM		(\$ 8 8 X X)	15		0		X (16 bits)		15		0		Y (16 bits)		<p><b>WORD NUMBER</b>  <math>W_n = 3</math></p> <p><b>EXECUTION CYCLES</b>  <math>C_n = P \cdot L + 18</math></p>
15		8 7	5 4	3 2	0																						
1 0 0 0	1 0 0 0	AREA	COL	OPM		(\$ 8 8 X X)																					
15		0																									
	X (16 bits)																										
15		0																									
	Y (16 bits)																										
<p>&lt; <b>DESCRIPTION</b> &gt;</p> <p>The parameters (X, Y) define the line end point as absolute logical pixel X-Y addresses relative to the origin defined with the ORG command.</p> <p>As the line is drawn, CP is moved to Pe. However, the logical pixel at position Pe is not drawn.</p> <div style="text-align: center;"> </div>																											
<p><b>Figure C18-1 Function of ALINE</b></p>																											

## ALINE (Absolute Line)

PAGE

ALINE-2

## &lt; EXAMPLE &gt;

If  $CP = (-13, -10)$  and ALINE command is executed with parameters  $(X, Y) = (10, 2)$ , then a line is drawn and CP is set to  $Pe$  as shown below.

## COMMAND CODE

15		8	7	5	4	3	2	0	
1	0	0	0	1	0	0	0	AREA	COL
								OPM	(\$ 8 8 X X)

## COMMAND PARAMETERS

15								0	
0	0	0	0	0	0	0	0	1	0
									X (\$ 0 0 0 A)

15								0	
0	0	0	0	0	0	0	0	0	0
									Y (\$ 0 0 0 2)

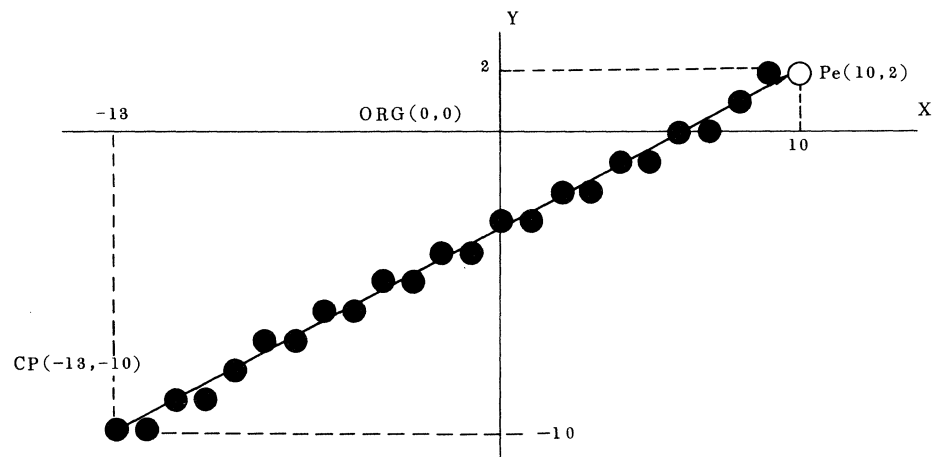


Figure C18-2 Example of ALINE Execution

		RLINE																							
[19] RLINE (Relative Line)		PAGE	RLINE-1																						
<p>&lt; <b>FUNCTION</b> &gt; Draw a straight line from CP to a command specified end point.</p> <p>&lt; <b>MNEMONIC</b> &gt; RLINE (AREA, COL, OPM) dX, dY</p>		TYPE	Graphic Command																						
<p>&lt; <b>FORMAT</b> &gt;</p> <p>COMMAND CODE <span style="float: right;">hexadecimal notation</span></p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%; text-align: right;">15</td> <td style="width: 15%;"></td> <td style="width: 15%; text-align: right;">8 7</td> <td style="width: 15%; text-align: right;">5 4</td> <td style="width: 15%; text-align: right;">3 2</td> <td style="width: 15%; text-align: right;">0</td> <td style="width: 20%;"></td> </tr> <tr> <td style="border: 1px solid black; text-align: center;">1 0 0 0</td> <td style="border: 1px solid black; text-align: center;">1 1 0 0</td> <td style="border: 1px solid black; text-align: center;">AREA</td> <td style="border: 1px solid black; text-align: center;">COL</td> <td style="border: 1px solid black; text-align: center;">OPM</td> <td></td> <td style="text-align: center;">(\$ 8 C X X)</td> </tr> </table> <p>COMMAND PARAMETERS</p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%; text-align: right;">15</td> <td style="width: 80%; border: 1px solid black; text-align: center;">dX (16 bits)</td> <td style="width: 5%;"></td> <td style="width: 10%; text-align: right;">0</td> </tr> <tr> <td style="text-align: right;">15</td> <td style="border: 1px solid black; text-align: center;">dY (16 bits)</td> <td></td> <td style="text-align: right;">0</td> </tr> </table>		15		8 7	5 4	3 2	0		1 0 0 0	1 1 0 0	AREA	COL	OPM		(\$ 8 C X X)	15	dX (16 bits)		0	15	dY (16 bits)		0	<p>WORD NUMBER <math>W_n = 3</math></p> <p>EXECUTION CYCLES <math>C_n = P \cdot L + 18</math></p>	
15		8 7	5 4	3 2	0																				
1 0 0 0	1 1 0 0	AREA	COL	OPM		(\$ 8 C X X)																			
15	dX (16 bits)		0																						
15	dY (16 bits)		0																						
<p>&lt; <b>DESCRIPTION</b> &gt;</p> <p>The parameters (dX, dY) define the line end point as relative logical pixel X-Y displacements from the CP.</p> <p>As the line is drawn, CP is moved to Pe. However, the logical pixel at position Pe is not drawn.</p> <div style="text-align: center;"> </div>																									
<p>Figure C19-1 Function of RLINE</p>																									

**RLINE (Relative Line)**

**< EXAMPLE >**

If CP = (-13, -10) and RLINE command is executed with parameters (dX, dY) = (10, 2), then a line is drawn and CP is set to Pe as shown below.

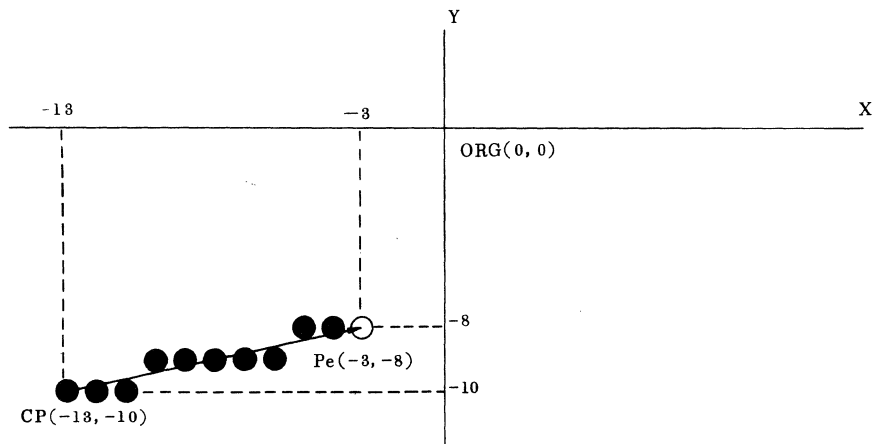
**COMMAND CODE**

15		8	7	5	4	3	2	0			
1	0	0	0	1	1	0	0	AREA	COL	OPM	(\$ 8 C X X)

**COMMAND PARAMETERS**

15									0				
0	0	0	0	0	0	0	0	0	1	0	1	0	(\$ 0 0 0 A)

15									0				
0	0	0	0	0	0	0	0	0	0	0	1	0	(\$ 0 0 0 2)



**Figure C19-2 Example of RLINE Execution**

**ARCT**

<p><b>[20] ARCT (Absolute Rectangle)</b></p>	<p><b>PAGE</b></p>	<p><b>ARCT-1</b></p>																														
<p>&lt; <b>FUNCTION</b> &gt;          Draw a rectangle defined by CP and the command specified diagonal point.           &lt; <b>MNEMONIC</b> &gt;          ARCT (AREA, COL, OPM) X, Y</p>	<p>TYPE</p>	<p>Graphic Command</p>																														
<p>&lt; <b>FORMAT</b> &gt;</p> <p>COMMAND CODE <span style="float: right;">hexadecimal notation</span></p> <table border="1" style="margin-left: 20px;"> <tr> <td style="text-align: right;">15</td> <td style="text-align: right;">8</td> <td style="text-align: right;">7</td> <td style="text-align: right;">5</td> <td style="text-align: right;">4</td> <td style="text-align: right;">3</td> <td style="text-align: right;">2</td> <td style="text-align: right;">0</td> <td></td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td style="text-align: center;">AREA COL OPM (\$ 9 0 X X)</td> </tr> </table> <p>COMMAND PARAMETERS</p> <table border="1" style="margin-left: 20px;"> <tr> <td style="text-align: right;">15</td> <td style="width: 100px;"></td> <td style="text-align: right;">0</td> </tr> <tr> <td colspan="3" style="text-align: center;">X (16 bits)</td> </tr> <tr> <td style="text-align: right;">15</td> <td style="width: 100px;"></td> <td style="text-align: right;">0</td> </tr> <tr> <td colspan="3" style="text-align: center;">Y (16 bits)</td> </tr> </table>	15	8	7	5	4	3	2	0		1	0	0	1	0	0	0	0	AREA COL OPM (\$ 9 0 X X)	15		0	X (16 bits)			15		0	Y (16 bits)			<p>WORD NUMBER Wn=3</p> <p>EXECUTION CYCLES Cn=2p(A+B)+54</p>	
15	8	7	5	4	3	2	0																									
1	0	0	1	0	0	0	0	AREA COL OPM (\$ 9 0 X X)																								
15		0																														
X (16 bits)																																
15		0																														
Y (16 bits)																																
<p>&lt; <b>DESCRIPTION</b> &gt;</p> <p>The parameters (X, Y) define the diagonal point of the rectangle as absolute logical pixel X-Y addresses relative to the origin defined by the ORG command.</p> <p>As the rectangle is drawn, CP is moved to Pe (which is the same as CP). However, the logical pixel at position Pe is not drawn.</p> <p>Drawing starts in the X direction first, and is drawn in the direction shown below. The initial X direction is determined by the relationship between CP and (X, Y).</p> <div style="text-align: center;"> </div> <p><b>Figure C20-1 Function of ARCT</b></p>																																

## ARCT (Absolute Polyline)

PAGE

ARCT-2

## &lt; EXAMPLE &gt;

If  $CP = (6, -6)$  and ACT command is executed with parameters  $(X, Y) = (-16, 10)$ , then a rectangle is drawn and CP is set to  $P_e$  as shown below.

## &lt; NOTE &gt;

Drawing starts from the X-axis direction.

## COMMAND CODE

15				8	7		5	4	3	2	0	
1	0	0	1	0	0	0	0	AREA	COL	OPM		(\$ 9 0 X X)

## COMMAND PARAMETERS

15											0	
1	1	1	1	1	1	1	1	1	1	1	0	0
												(\$ F F F 0)

15											0	
0	0	0	0	0	0	0	0	0	0	0	1	0
												(\$ 0 0 0 A)

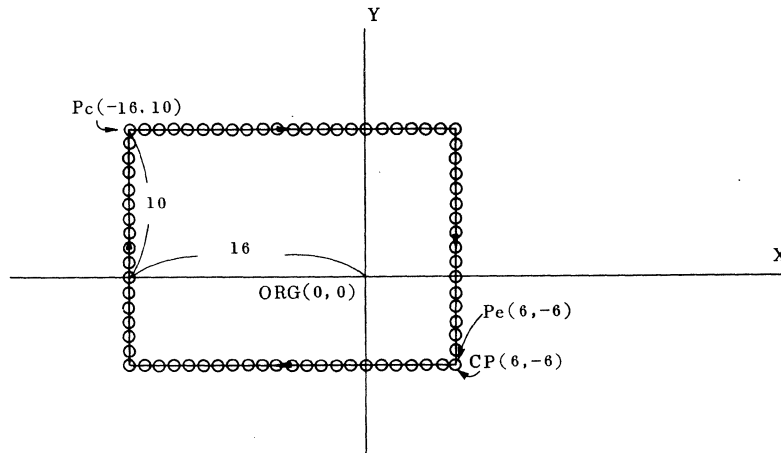
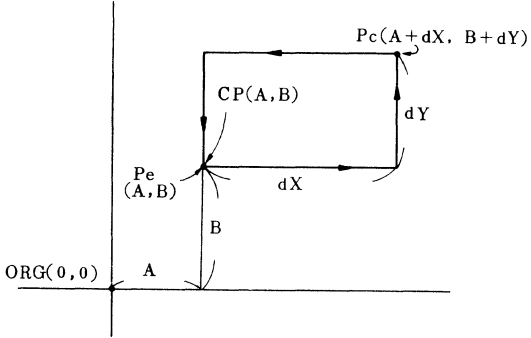


Figure C20-2 Example of ARCT Execution

**RRCT**

<p><b>[21] RRCT (Relative Rectangle)</b></p>	<p><b>PAGE</b></p>	<p><b>RRCT-1</b></p>																		
<p><b>&lt; FUNCTION &gt;</b>          Draw a rectangle defined by CP and the command specified diagonal point.</p> <p><b>&lt; MNEMONIC &gt;</b>          RRCT (AREA, COL, OPM) dX, dY</p>	<p>TYPE</p>	<p>Graphic Command</p>																		
<p><b>&lt; FORMAT &gt;</b></p> <p>COMMAND CODE <span style="float: right;">hexadecimal notation</span></p> <table border="1" style="margin-left: 20px;"> <tr> <td style="text-align: right;">15</td> <td style="text-align: right;">8 7</td> <td style="text-align: right;">5 4</td> <td style="text-align: right;">3 2</td> <td style="text-align: right;">0</td> <td></td> </tr> <tr> <td style="text-align: center;">1 0 0 1</td> <td style="text-align: center;">0 1 0 0</td> <td style="text-align: center;">AREA</td> <td style="text-align: center;">COL</td> <td style="text-align: center;">OPM</td> <td style="text-align: center;">(\$ 9 4 X X)</td> </tr> </table> <p>COMMAND PARAMETERS</p> <table border="1" style="margin-left: 20px;"> <tr> <td style="text-align: right;">15</td> <td style="text-align: right;">0</td> <td style="text-align: center;">dX (16 bits)</td> </tr> <tr> <td style="text-align: right;">15</td> <td style="text-align: right;">0</td> <td style="text-align: center;">dY (16 bits)</td> </tr> </table>	15	8 7	5 4	3 2	0		1 0 0 1	0 1 0 0	AREA	COL	OPM	(\$ 9 4 X X)	15	0	dX (16 bits)	15	0	dY (16 bits)	<p>WORD NUMBER  <math>W_n = 3</math></p> <p>EXECUTION CYCLES  <math>C_n = 2P(A+B) + 54</math></p>	
15	8 7	5 4	3 2	0																
1 0 0 1	0 1 0 0	AREA	COL	OPM	(\$ 9 4 X X)															
15	0	dX (16 bits)																		
15	0	dY (16 bits)																		
<p><b>&lt; DESCRIPTION &gt;</b></p> <p>The parameters (dX, dY) define the diagonal point of the rectangle as relative logical pixel X-Y displacements from the CP.</p> <p>As the rectangle is drawn, CP is moved to Pe. However, the logical pixel at position Pe is not drawn.</p> <p>Drawing starts in the X direction first, and is drawn in the direction show below. The initial X direction is determined by the relationship between CP and (dX, dY).</p> <div style="text-align: center;">  </div> <p><b>Figure C21-1 Function of RRCT</b></p>																				

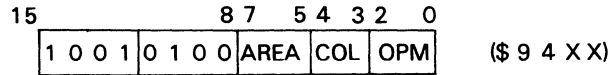


**RRCT (Relative Rectangular)**

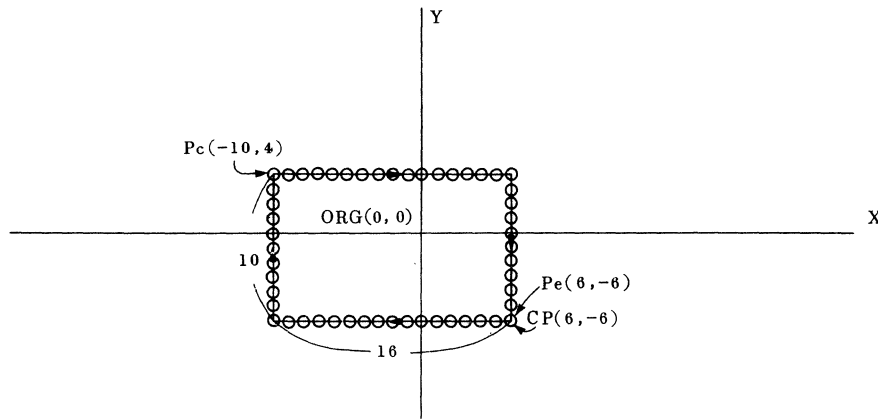
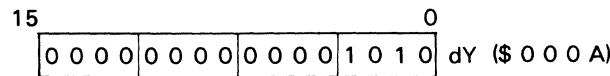
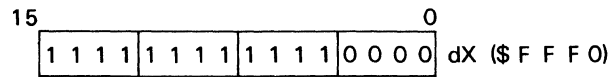
**< EXAMPLE >**

If CP = (6, -6) and RRCT command is executed with parameters (dX, dY) = (-16, 10), then a rectangle is drawn and CP is set to Pc as shown below.

**COMMAND CODE**



**COMMAND PARAMETERS**



**Figure C21-2 Example of RRCT**

<p><b>[22] APLL (Absolute Polyline)</b></p>	<p><b>PAGE</b></p>	<p><b>APLL-1</b></p>																																													
<p>&lt; <b>FUNCTION</b> &gt;          Draw a polyline (multiple contiguous segments) from the CP through command specified points.</p> <p>&lt; <b>MNEMONIC</b> &gt;          APLL (AREA, COL, OPM) n, X<sub>1</sub>, Y<sub>1</sub> ... X<sub>n</sub>, Y<sub>n</sub></p>	<p>TYPE</p>	<p>Graphic Command</p>																																													
<p>&lt; <b>FORMAT</b> &gt;</p> <p>COMMAND CODE <span style="float: right;">hexadecimal notation</span></p> <table border="1" style="margin-left: 20px;"> <tr> <td style="text-align: right;">15</td> <td style="text-align: center;">8 7 5 4 3 2 0</td> <td></td> </tr> <tr> <td style="text-align: center;">1 0 0 1</td> <td style="text-align: center;">1 0 0 0</td> <td style="text-align: center;">AREA COL OPM</td> </tr> </table> <p style="margin-left: 40px;">(\$ 9 8 X X)</p> <p>COMMAND PARAMETERS</p> <table border="1" style="margin-left: 20px;"> <tr> <td style="text-align: right;">15</td> <td style="text-align: center;">n (16 bits)</td> <td style="text-align: right;">0</td> <td></td> </tr> <tr> <td style="text-align: right;">15</td> <td style="text-align: center;">X<sub>1</sub> (16 bits)</td> <td style="text-align: right;">0</td> <td></td> </tr> <tr> <td style="text-align: right;">15</td> <td style="text-align: center;">Y<sub>1</sub> (16 bits)</td> <td style="text-align: right;">0</td> <td>P1</td> </tr> <tr> <td style="text-align: right;">15</td> <td style="text-align: center;">X<sub>2</sub> (16 bits)</td> <td style="text-align: right;">0</td> <td></td> </tr> <tr> <td style="text-align: right;">15</td> <td style="text-align: center;">Y<sub>2</sub> (16 bits)</td> <td style="text-align: right;">0</td> <td>P2</td> </tr> <tr> <td style="text-align: right;">15</td> <td style="text-align: center;">⋮</td> <td style="text-align: right;">0</td> <td style="text-align: center;">⋮</td> </tr> <tr> <td style="text-align: right;">15</td> <td style="text-align: center;">X<sub>n-1</sub> (16 bits)</td> <td style="text-align: right;">0</td> <td></td> </tr> <tr> <td style="text-align: right;">15</td> <td style="text-align: center;">Y<sub>n-1</sub> (16 bits)</td> <td style="text-align: right;">0</td> <td>Pn-1</td> </tr> <tr> <td style="text-align: right;">15</td> <td style="text-align: center;">X<sub>n</sub> (16 bits)</td> <td style="text-align: right;">0</td> <td></td> </tr> <tr> <td style="text-align: right;">15</td> <td style="text-align: center;">Y<sub>n</sub> (16 bits)</td> <td style="text-align: right;">0</td> <td>Pe</td> </tr> </table> <p>P<sub>2n+1</sub></p> <p>n is specified by the absolute value of a 16-bits binary number.</p>	15	8 7 5 4 3 2 0		1 0 0 1	1 0 0 0	AREA COL OPM	15	n (16 bits)	0		15	X <sub>1</sub> (16 bits)	0		15	Y <sub>1</sub> (16 bits)	0	P1	15	X <sub>2</sub> (16 bits)	0		15	Y <sub>2</sub> (16 bits)	0	P2	15	⋮	0	⋮	15	X <sub>n-1</sub> (16 bits)	0		15	Y <sub>n-1</sub> (16 bits)	0	Pn-1	15	X <sub>n</sub> (16 bits)	0		15	Y <sub>n</sub> (16 bits)	0	Pe	<p>WORD NUMBER  <math>W_n = 2n + 2</math></p> <p>EXECUTION CYCLES  <math>C_n = \sum \{P \cdot L + 16\} + 8</math></p>
15	8 7 5 4 3 2 0																																														
1 0 0 1	1 0 0 0	AREA COL OPM																																													
15	n (16 bits)	0																																													
15	X <sub>1</sub> (16 bits)	0																																													
15	Y <sub>1</sub> (16 bits)	0	P1																																												
15	X <sub>2</sub> (16 bits)	0																																													
15	Y <sub>2</sub> (16 bits)	0	P2																																												
15	⋮	0	⋮																																												
15	X <sub>n-1</sub> (16 bits)	0																																													
15	Y <sub>n-1</sub> (16 bits)	0	Pn-1																																												
15	X <sub>n</sub> (16 bits)	0																																													
15	Y <sub>n</sub> (16 bits)	0	Pe																																												

## APLL (Absolute Polyline)

PAGE APLL-2

## &lt; DESCRIPTION &gt;

The first parameter (n) specifies the number of line segments, that is,  $n = 1$  specifies one line segment. The following parameters ( $X_n, Y_n$ ) are absolute logical pixel X-Y addresses, which specify each segments end point relative to the origin defined by the ORG command.

As the polyline is drawn, CP is moved to Pe. However, the logical pixel at position Pe is not drawn.

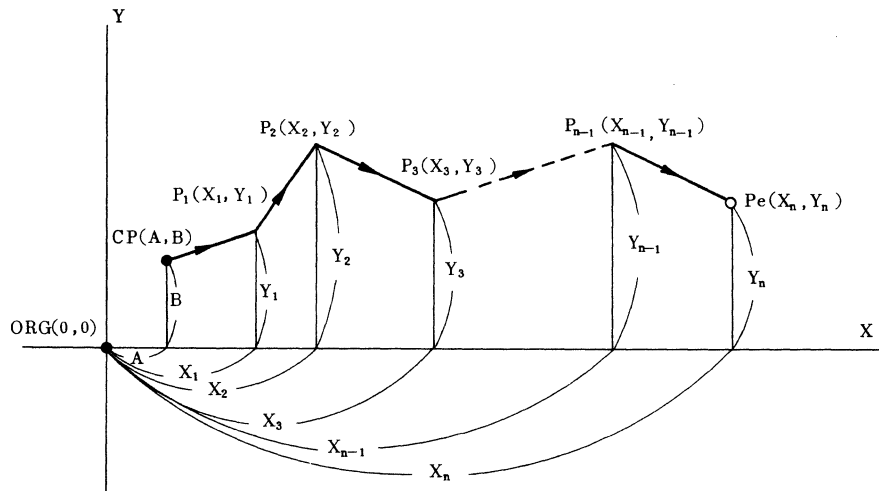


Figure C22-1 Function of APLL

## APLL (Absolute Polyline)

PAGE APLL-3

## &lt; EXECUTION EXAMPLE &gt;

If the CP is at  $(-8, -6)$  on the split screen,  $n$  is set to 3,  $X_1$  to  $-4$ ,  $Y_1$  to 4,  $X_2$  to 8,  $Y_2$  to 6,  $X_3$  to 16 and  $Y_3$  to  $-8$ , then the APLL command draws a poly line as shown below.

## COMMAND CODE

15		8	7	5	4	3	2	0
1	0	0	1	1	0	0	0	0
AREA		COL		OPM				

## COMMAND PARAMETERS

15								0	
0	0	0	0	0	0	0	0	0	0
									(\$ 0 0 0 3)

15								0	
1	1	1	1	1	1	1	1	1	0
									(\$ F F F C)

15								0	
0	0	0	0	0	0	0	0	0	1
									(\$ 0 0 0 4)

15								0	
0	0	0	0	0	0	0	0	1	0
									(\$ 0 0 0 8)

15								0	
0	0	0	0	0	0	0	0	0	1
									(\$ 0 0 0 6)

15								0	
0	0	0	0	0	0	1	0	0	0
									(\$ 0 0 1 0)

15								0	
1	1	1	1	1	1	1	1	1	0
									(\$ F F F 8)

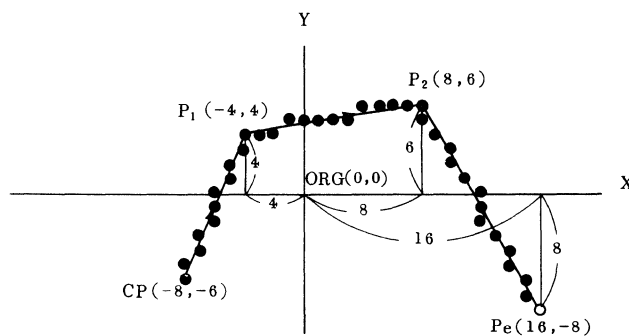


Figure C22-2 Example of APLL Execution

<p><b>[23] RPLL (Relative Polyline)</b></p>	<p><b>PAGE</b></p>	<p><b>RPLL-1</b></p>																																																																																																																																																																																			
<p>&lt; <b>FUNCTION</b> &gt;  RPLL command draws a polyline which connects the Start point, current pointer, and each relative coordinate point.</p> <p>&lt; <b>MNEMONIC</b> &gt;  RPLL (AREA, COL, OPM) n, dX<sub>1</sub>, dY<sub>1</sub>, ... dX<sub>n</sub>, dY<sub>n</sub></p>	<p><b>TYPE</b></p>	<p>Graphic Command</p>																																																																																																																																																																																			
<p>&lt; <b>FORMAT</b> &gt;</p> <p>COMMAND CODE <span style="float: right;">hexadecimal notation</span></p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: right; width: 15%;">15</td> <td style="width: 15%;"></td> <td style="text-align: right; width: 15%;">8 7</td> <td style="width: 15%;"></td> <td style="text-align: right; width: 15%;">5 4</td> <td style="width: 15%;"></td> <td style="text-align: right; width: 15%;">3 2</td> <td style="width: 15%;"></td> <td style="text-align: right; width: 15%;">0</td> <td style="width: 15%;"></td> </tr> <tr> <td>C</td> <td style="border: 1px solid black; text-align: center;">1 0 0 1</td> <td style="border: 1px solid black; text-align: center;">1 1 0 0</td> <td style="border: 1px solid black; text-align: center;">AREA</td> <td style="border: 1px solid black; text-align: center;">COL</td> <td style="border: 1px solid black; text-align: center;">OPM</td> <td colspan="4" style="text-align: right;">(\$ 9 8 X X)</td> </tr> </table> <p>COMMAND PARAMETERS</p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: right; width: 15%;">15</td> <td style="width: 15%;"></td> <td style="text-align: right; width: 15%;">0</td> <td style="width: 15%;"></td> <td style="width: 15%;"></td> <td style="width: 15%;"></td> <td style="width: 15%;"></td> <td style="width: 15%;"></td> <td style="width: 15%;"></td> <td style="width: 15%;"></td> </tr> <tr> <td></td> <td style="border: 1px solid black; text-align: center;">n (16 bits)</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td style="text-align: right;">15</td> <td></td> <td style="text-align: right;">0</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td style="border: 1px solid black; text-align: center;">dX<sub>1</sub> (16 bits)</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td style="text-align: right;">15</td> <td></td> <td style="text-align: right;">0</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td style="text-align: right;">P<sub>1</sub></td> </tr> <tr> <td></td> <td style="border: 1px solid black; text-align: center;">dY<sub>1</sub> (16 bits)</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td style="text-align: right;">15</td> <td></td> <td style="text-align: right;">0</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td style="border: 1px solid black; text-align: center;">dX<sub>2</sub> (16 bits)</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td style="text-align: right;">P<sub>2</sub></td> </tr> <tr> <td style="text-align: right;">15</td> <td></td> <td style="text-align: right;">0</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td style="text-align: center;">⋮</td> </tr> <tr> <td></td> <td style="border: 1px solid black; text-align: center;">dX<sub>n-1</sub> (16 bits)</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td style="text-align: center;">⋮</td> </tr> <tr> <td style="text-align: right;">15</td> <td></td> <td style="text-align: right;">0</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td style="text-align: right;">P<sub>n-1</sub></td> </tr> <tr> <td></td> <td style="border: 1px solid black; text-align: center;">dY<sub>n-1</sub> (16 bits)</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td style="text-align: right;">15</td> <td></td> <td style="text-align: right;">0</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td style="border: 1px solid black; text-align: center;">dX<sub>n</sub> (16 bits)</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td style="text-align: right;">P<sub>e</sub></td> </tr> <tr> <td style="text-align: right;">15</td> <td></td> <td style="text-align: right;">0</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td style="border: 1px solid black; text-align: center;">dY<sub>n</sub> (16 bits)</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </table> <p>P<sub>2n+1</sub></p> <p>Set "n" in binary absolute values of 16 bits.</p>	15		8 7		5 4		3 2		0		C	1 0 0 1	1 1 0 0	AREA	COL	OPM	(\$ 9 8 X X)				15		0									n (16 bits)									15		0									dX <sub>1</sub> (16 bits)									15		0							P <sub>1</sub>		dY <sub>1</sub> (16 bits)									15		0									dX <sub>2</sub> (16 bits)								P <sub>2</sub>	15		0							⋮		dX <sub>n-1</sub> (16 bits)								⋮	15		0							P <sub>n-1</sub>		dY <sub>n-1</sub> (16 bits)									15		0									dX <sub>n</sub> (16 bits)								P <sub>e</sub>	15		0									dY <sub>n</sub> (16 bits)									<p><b>WORD NUMBER</b>  W<sub>n</sub> = 2n + 2</p> <p><b>EXECUTION CYCLES</b>  C<sub>n</sub> = Σ {P·L + 16} + 8</p>
15		8 7		5 4		3 2		0																																																																																																																																																																													
C	1 0 0 1	1 1 0 0	AREA	COL	OPM	(\$ 9 8 X X)																																																																																																																																																																															
15		0																																																																																																																																																																																			
	n (16 bits)																																																																																																																																																																																				
15		0																																																																																																																																																																																			
	dX <sub>1</sub> (16 bits)																																																																																																																																																																																				
15		0							P <sub>1</sub>																																																																																																																																																																												
	dY <sub>1</sub> (16 bits)																																																																																																																																																																																				
15		0																																																																																																																																																																																			
	dX <sub>2</sub> (16 bits)								P <sub>2</sub>																																																																																																																																																																												
15		0							⋮																																																																																																																																																																												
	dX <sub>n-1</sub> (16 bits)								⋮																																																																																																																																																																												
15		0							P <sub>n-1</sub>																																																																																																																																																																												
	dY <sub>n-1</sub> (16 bits)																																																																																																																																																																																				
15		0																																																																																																																																																																																			
	dX <sub>n</sub> (16 bits)								P <sub>e</sub>																																																																																																																																																																												
15		0																																																																																																																																																																																			
	dY <sub>n</sub> (16 bits)																																																																																																																																																																																				

## &lt; DESCRIPTION &gt;

As shown in figure below, the relative poly line command (RPLL) draws a poly line which connects the Start point CP, and each relative coordinate ( $P_1, P_2, P_3, \dots, P_{n-1}, P_n$ ). The total number of points is set in the 1st command parameter ( $n$ ). X and Y components of each point are set in the command parameters in the order the lines are drawn. CP moves to the End point  $P_n$  as the lines are drawn. However, a dot is not drawn at  $P_n$ .

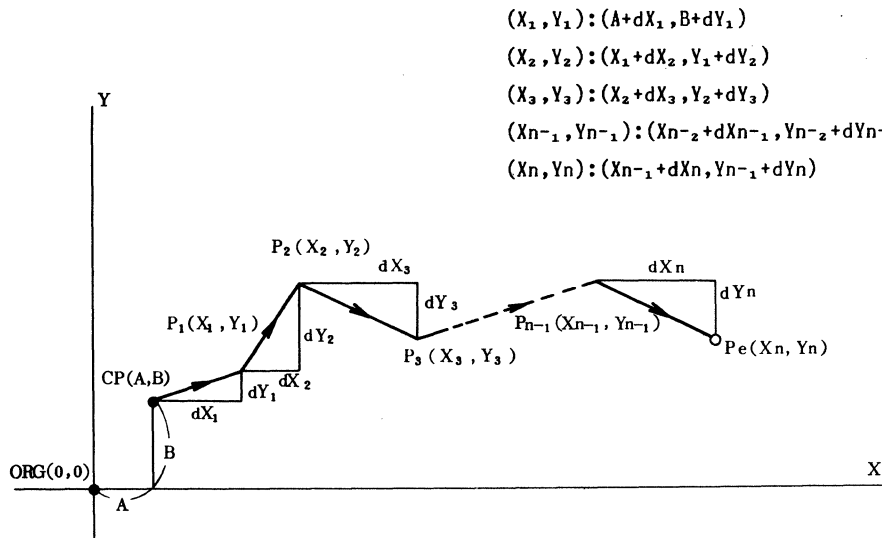


Figure C23-1 Function of RPLL

## &lt; EXECUTION EXAMPLE &gt;

If the CP is at  $(-8, -6)$  on the split screen,  $dX_1$  is set to  $-4$ ,  $dY_1$  to  $4$ ,  $dX_2$  to  $8$ ,  $dY_2$  to  $6$ ,  $dX_3$  to  $16$  and  $dY_3$  to  $-8$ , then the RPLL command draws a poly line as shown below.

## COMMAND CODE

15		8	7	5	4	3	2	0		
1	0	0	1	1	1	0	0	AREA	COL	OPM

(\$ 9 C X X)

## COMMAND PARAMETERS

15									0	
0	0	0	0	0	0	0	0	0	0	1

(\$ 0 0 0 3)

15									0	
1	1	1	1	1	1	1	1	1	1	0

(\$ F F F C)

15									0	
0	0	0	0	0	0	0	0	0	1	0

(\$ 0 0 0 4)

15									0	
0	0	0	0	0	0	0	0	1	0	0

(\$ 0 0 0 8)

15									0	
0	0	0	0	0	0	0	0	0	1	1

(\$ 0 0 0 6)

15									0	
0	0	0	0	0	0	0	1	0	0	0

(\$ 0 0 1 0)

15									0	
1	1	1	1	1	1	1	1	1	0	0

(\$ F F F 8)

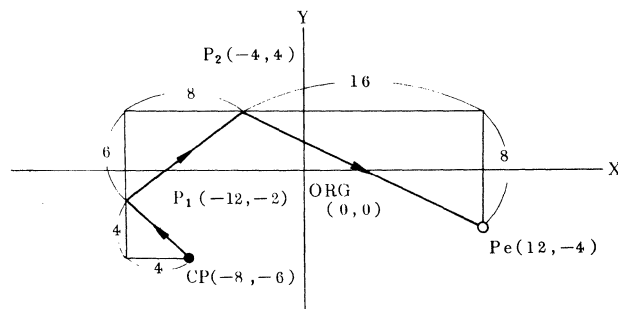


Figure C23-2 Example of RPLL Execution

**APLG**

<p><b>[24] APLG (Absolute Polygon)</b></p>	<p><b>PAGE</b></p>	<p><b>APLG-1</b></p>																																																																																																																							
<p><b>&lt; FUNCTION &gt;</b>          APLG draws a polygon which connects the initial point, CP, and each absolute coordinate.</p> <p><b>&lt; MNEMONIC &gt;</b>          APLG (AREA, COL, OPM) n, X<sub>1</sub>, Y<sub>1</sub> ..... X<sub>n</sub>, Y<sub>n</sub></p>	<p>TYPE</p>	<p>Graphic Command</p>																																																																																																																							
<p><b>&lt; FORMAT &gt;</b></p> <p>COMMAND CODE <span style="float: right;">hexadecimal notation</span></p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: right; width: 10%;">15</td> <td style="width: 10%;"></td> <td style="text-align: center; width: 10%;">8</td> <td style="text-align: center; width: 10%;">7</td> <td style="text-align: center; width: 10%;">5</td> <td style="text-align: center; width: 10%;">4</td> <td style="text-align: center; width: 10%;">3</td> <td style="text-align: center; width: 10%;">2</td> <td style="text-align: center; width: 10%;">0</td> <td style="width: 10%;"></td> </tr> <tr> <td style="border: 1px solid black; text-align: center;">1</td> <td style="border: 1px solid black; text-align: center;">0</td> <td style="border: 1px solid black; text-align: center;">1</td> <td style="border: 1px solid black; text-align: center;">0</td> <td style="border: 1px solid black; text-align: center;">0</td> <td style="border: 1px solid black; text-align: center;">0</td> <td style="border: 1px solid black; text-align: center;">0</td> <td style="border: 1px solid black; text-align: center;">0</td> <td style="border: 1px solid black; text-align: center;">0</td> <td style="text-align: center;">(\$ A 0 X X)</td> </tr> </table> <p>COMMAND PARAMETERS</p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: right; width: 10%;">15</td> <td style="width: 10%;"></td> <td style="text-align: right; width: 10%;">0</td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> </tr> <tr> <td style="border: 1px solid black; text-align: center;">n</td> <td style="border: 1px solid black; text-align: center;">(16 bits)</td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> </tr> <tr> <td style="text-align: right;">15</td> <td style="border: 1px solid black; text-align: center;">X<sub>1</sub></td> <td style="text-align: right;">0</td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> </tr> <tr> <td style="text-align: right;">15</td> <td style="border: 1px solid black; text-align: center;">Y<sub>1</sub></td> <td style="text-align: right;">0</td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> </tr> <tr> <td style="text-align: right;">15</td> <td style="border: 1px solid black; text-align: center;">X<sub>2</sub></td> <td style="text-align: right;">0</td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> </tr> <tr> <td style="text-align: right;">15</td> <td style="border: 1px solid black; text-align: center;">Y<sub>2</sub></td> <td style="text-align: right;">0</td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> </tr> <tr> <td style="text-align: right;">15</td> <td style="border: 1px solid black; text-align: center;">X<sub>n-1</sub></td> <td style="text-align: right;">0</td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> </tr> <tr> <td style="text-align: right;">15</td> <td style="border: 1px solid black; text-align: center;">Y<sub>n-1</sub></td> <td style="text-align: right;">0</td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> </tr> <tr> <td style="text-align: right;">15</td> <td style="border: 1px solid black; text-align: center;">X<sub>n</sub></td> <td style="text-align: right;">0</td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> </tr> <tr> <td style="text-align: right;">15</td> <td style="border: 1px solid black; text-align: center;">Y<sub>n</sub></td> <td style="text-align: right;">0</td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> <td style="width: 10%;"></td> </tr> </table> <p style="margin-top: 10px;">P<sub>2n+1</sub></p> <p>Set "n" in binary absolute values of 16 bits.</p>	15		8	7	5	4	3	2	0		1	0	1	0	0	0	0	0	0	(\$ A 0 X X)	15		0								n	(16 bits)									15	X <sub>1</sub>	0								15	Y <sub>1</sub>	0								15	X <sub>2</sub>	0								15	Y <sub>2</sub>	0								15	X <sub>n-1</sub>	0								15	Y <sub>n-1</sub>	0								15	X <sub>n</sub>	0								15	Y <sub>n</sub>	0								<p>WORD NUMBER  <math>W_n = 2n + 2</math></p> <p>EXECUTION CYCLES  <math>C_n = \sum \{P-L + 16\} + P-L_0 + 20</math></p>
15		8	7	5	4	3	2	0																																																																																																																	
1	0	1	0	0	0	0	0	0	(\$ A 0 X X)																																																																																																																
15		0																																																																																																																							
n	(16 bits)																																																																																																																								
15	X <sub>1</sub>	0																																																																																																																							
15	Y <sub>1</sub>	0																																																																																																																							
15	X <sub>2</sub>	0																																																																																																																							
15	Y <sub>2</sub>	0																																																																																																																							
15	X <sub>n-1</sub>	0																																																																																																																							
15	Y <sub>n-1</sub>	0																																																																																																																							
15	X <sub>n</sub>	0																																																																																																																							
15	Y <sub>n</sub>	0																																																																																																																							



## &lt; DESCRIPTION &gt;

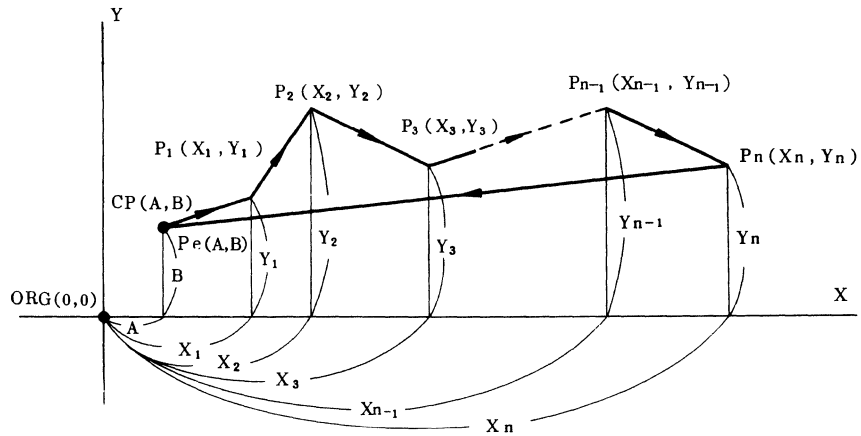


Figure C24-1 Function of APLG

As shown in above figure, the APLG command draws a polygon line which connects the start point, CP, and each absolute coordinate ( $P_1, P_2, \dots, P_{n-1}, P_n$ ), then back to CP. The total number of points are set in the first command parameter. X and Y components of each point are set in the command parameters in the order the lines are drawn. CP moves to the end point CPe to draw a poly line. However a dot is not drawn at Pe. CP is the same point as Pe.

**APLG (Absolute Polygon)**

< EXAMPLE >

If the CP is at (-8, -6) on the split screen, n is set to 3, X<sub>1</sub> to -4, Y<sub>1</sub> to 4, X<sub>2</sub> to 8, Y<sub>2</sub> to 6, X<sub>3</sub> to 16 and Y<sub>3</sub> to -8 in the command parameter. The APLG command draws a polygon line as shown below.

COMMAND CODE

15									0
	1	0	1	0	0	0	0	0	0

AREA COL OPM (\$ A O X X)

COMMAND PARAMETERS

15									0
	0	0	0	0	0	0	0	0	0

(\$ 0 0 0 3)

15									0
	1	1	1	1	1	1	1	1	0

(\$ F F F C)

15									0
	0	0	0	0	0	0	0	0	0

(\$ 0 0 0 4)

15									0
	0	0	0	0	0	0	0	1	0

(\$ 0 0 0 8)

15									0
	0	0	0	0	0	0	0	0	1

(\$ 0 0 0 6)

15									0
	0	0	0	0	0	0	0	1	0

(\$ 0 0 1 0)

15									0
	1	1	1	1	1	1	1	1	0

(\$ F F F 8)

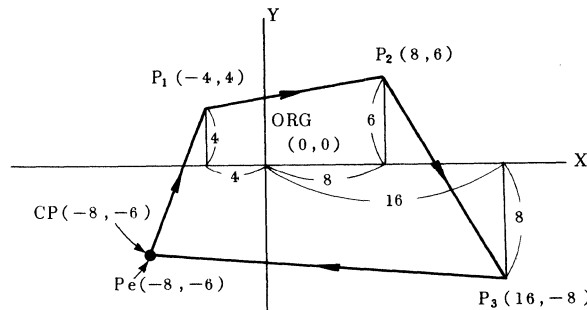


Figure C24-2 Example of APLG Execution

[25] RPLG (Relative Polygon)	PAGE	RPLG-1																																																																																																																																	
<p>&lt; FUNCTION &gt;                      APLG draws a polygon which connects the initial point, CP, and each relative coordinate.</p> <p>&lt; MNEMONIC &gt;                      RPLG (AREA, COL, OPM) n, dX<sub>1</sub>, dY<sub>1</sub>, ... dX<sub>n</sub>, dY<sub>n</sub></p>	TYPE	Graphic Command																																																																																																																																	
<p>&lt; FORMAT &gt;</p> <p>COMMAND CODE <span style="float: right;">hexadecimal notation</span></p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: right; width: 10%;">15</td> <td style="width: 10%;"></td> <td style="text-align: center; width: 10%;">87</td> <td style="width: 10%;"></td> <td style="text-align: center; width: 10%;">54</td> <td style="width: 10%;"></td> <td style="text-align: center; width: 10%;">32</td> <td style="width: 10%;"></td> <td style="text-align: right; width: 10%;">0</td> <td style="width: 10%;"></td> </tr> <tr> <td style="border: 1px solid black; text-align: center;">1</td> <td style="border: 1px solid black; text-align: center;">0</td> <td style="border: 1px solid black; text-align: center;">1</td> <td style="border: 1px solid black; text-align: center;">0</td> <td style="border: 1px solid black; text-align: center;">0</td> <td style="border: 1px solid black; text-align: center;">1</td> <td style="border: 1px solid black; text-align: center;">0</td> <td style="border: 1px solid black; text-align: center;">0</td> <td style="border: 1px solid black; text-align: center;">0</td> <td style="border: 1px solid black; text-align: center;">0</td> </tr> <tr> <td colspan="4"></td> <td style="text-align: center;">AREA</td> <td colspan="2"></td> <td style="text-align: center;">COL</td> <td colspan="2"></td> </tr> <tr> <td colspan="10" style="text-align: right;">(\$ A 4 X X)</td> </tr> </table> <p>COMMAND PARAMETERS</p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: right; width: 10%;">15</td> <td style="width: 10%;"></td> <td style="width: 80%;"></td> <td style="text-align: right; width: 10%;">0</td> <td style="width: 10%;"></td> </tr> <tr> <td colspan="5" style="text-align: center; border: 1px solid black;">n (16 bits)</td> </tr> <tr> <td style="text-align: right;">15</td> <td></td> <td style="border: 1px solid black;"></td> <td style="text-align: right;">0</td> <td></td> </tr> <tr> <td colspan="5" style="text-align: center; border: 1px solid black;">dX<sub>1</sub> (16 bits)</td> </tr> <tr> <td style="text-align: right;">15</td> <td></td> <td style="border: 1px solid black;"></td> <td style="text-align: right;">0</td> <td style="text-align: right;">P<sub>1</sub></td> </tr> <tr> <td colspan="5" style="text-align: center; border: 1px solid black;">dY<sub>1</sub> (16 bits)</td> </tr> <tr> <td style="text-align: right;">15</td> <td></td> <td style="border: 1px solid black;"></td> <td style="text-align: right;">0</td> <td></td> </tr> <tr> <td colspan="5" style="text-align: center; border: 1px solid black;">dX<sub>2</sub> (16 bits)</td> </tr> <tr> <td style="text-align: right;">15</td> <td></td> <td style="border: 1px solid black;"></td> <td style="text-align: right;">0</td> <td style="text-align: right;">P<sub>2</sub></td> </tr> <tr> <td colspan="5" style="text-align: center; border: 1px solid black;">dY<sub>2</sub> (16 bits)</td> </tr> <tr> <td style="text-align: right;">15</td> <td></td> <td style="text-align: center;">⋮</td> <td style="text-align: right;">0</td> <td style="text-align: center;">⋮</td> </tr> <tr> <td colspan="5" style="text-align: center; border: 1px solid black;">dX<sub>n-1</sub> (16 bits)</td> </tr> <tr> <td style="text-align: right;">15</td> <td></td> <td style="border: 1px solid black;"></td> <td style="text-align: right;">0</td> <td style="text-align: right;">P<sub>n-1</sub></td> </tr> <tr> <td colspan="5" style="text-align: center; border: 1px solid black;">dY<sub>n-1</sub> (16 bits)</td> </tr> <tr> <td style="text-align: right;">15</td> <td></td> <td style="border: 1px solid black;"></td> <td style="text-align: right;">0</td> <td></td> </tr> <tr> <td colspan="5" style="text-align: center; border: 1px solid black;">dX<sub>n</sub> (16 bits)</td> </tr> <tr> <td style="text-align: right;">15</td> <td></td> <td style="border: 1px solid black;"></td> <td style="text-align: right;">0</td> <td style="text-align: right;">P<sub>n</sub></td> </tr> <tr> <td colspan="5" style="text-align: center; border: 1px solid black;">dY<sub>n</sub> (16 bits)</td> </tr> </table> <p>P<sub>2n+1</sub></p> <p>Set "n" in binary absolute values of 16 bits.</p>	15		87		54		32		0		1	0	1	0	0	1	0	0	0	0					AREA			COL			(\$ A 4 X X)										15			0		n (16 bits)					15			0		dX <sub>1</sub> (16 bits)					15			0	P <sub>1</sub>	dY <sub>1</sub> (16 bits)					15			0		dX <sub>2</sub> (16 bits)					15			0	P <sub>2</sub>	dY <sub>2</sub> (16 bits)					15		⋮	0	⋮	dX <sub>n-1</sub> (16 bits)					15			0	P <sub>n-1</sub>	dY <sub>n-1</sub> (16 bits)					15			0		dX <sub>n</sub> (16 bits)					15			0	P <sub>n</sub>	dY <sub>n</sub> (16 bits)					<p>WORD NUMBER  <math>W_n = 2n + 2</math></p> <p>EXECUTION CYCLES  <math>C_n = \sum (P \cdot L + 16) + P \cdot L_0 + 20</math></p>
15		87		54		32		0																																																																																																																											
1	0	1	0	0	1	0	0	0	0																																																																																																																										
				AREA			COL																																																																																																																												
(\$ A 4 X X)																																																																																																																																			
15			0																																																																																																																																
n (16 bits)																																																																																																																																			
15			0																																																																																																																																
dX <sub>1</sub> (16 bits)																																																																																																																																			
15			0	P <sub>1</sub>																																																																																																																															
dY <sub>1</sub> (16 bits)																																																																																																																																			
15			0																																																																																																																																
dX <sub>2</sub> (16 bits)																																																																																																																																			
15			0	P <sub>2</sub>																																																																																																																															
dY <sub>2</sub> (16 bits)																																																																																																																																			
15		⋮	0	⋮																																																																																																																															
dX <sub>n-1</sub> (16 bits)																																																																																																																																			
15			0	P <sub>n-1</sub>																																																																																																																															
dY <sub>n-1</sub> (16 bits)																																																																																																																																			
15			0																																																																																																																																
dX <sub>n</sub> (16 bits)																																																																																																																																			
15			0	P <sub>n</sub>																																																																																																																															
dY <sub>n</sub> (16 bits)																																																																																																																																			

## &lt; DESCRIPTION &gt;

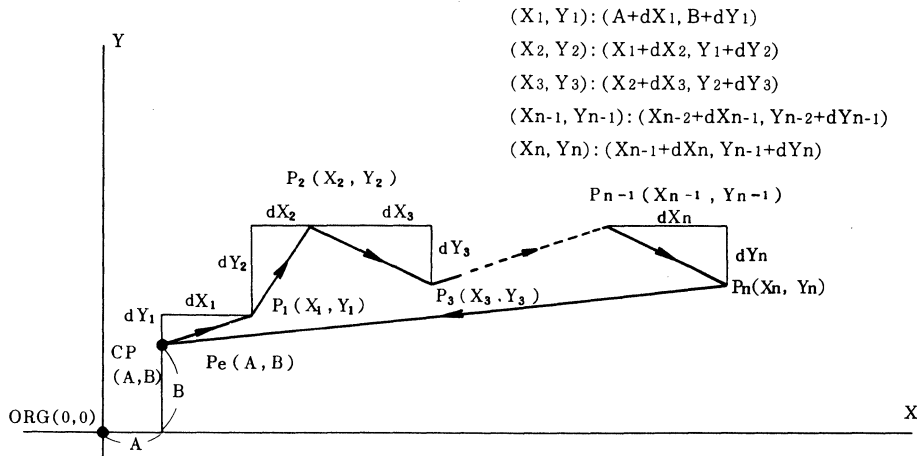


Figure C25-1 Function of RPLG

As shown in above figure, the RPLG command draws a polygon line which connects the start point, CP, and each related coordinate ( $P_1, P_2, P_3, \dots, P_{n-1}, P_n$ ), then back to CP. The total number of points are set in the first command parameter. X and Y components of each point are set in the command parameters in the order the lines are drawn. CP moves to the end point Pe as the lines are drawn. However a dot is not drawn at Pe. CP is the same point as Pe.

**RPLG (Relative Polygon)**

**<EXAMPLE>**

If the CP is at (-8, -6) on the split screen, n is set to 3, dX<sub>1</sub> to -4, dY<sub>1</sub> to 4, dX<sub>2</sub> to 8, dY<sub>2</sub> to 6, dX<sub>3</sub> to 16 and dY<sub>3</sub> to -8 in the command parameter, then the RPLG command draws a polygon line as shown below.

**COMMAND CODE**

15	0
1 0 1 0	0 1 0 0
AREA	COL OPM

(\$ A 4 X X)

**COMMAND PARAMETERS**

15	0
0 0 0 0	0 0 0 0
0 0 0 0	0 0 0 1

(\$ 0 0 0 3)

15	0
1 1 1 1	1 1 1 1
1 1 1 1	1 1 0 0

(\$ F F F C)

15	0
0 0 0 0	0 0 0 0
0 0 0 0	0 1 0 0

(\$ 0 0 0 4)

15	0
0 0 0 0	0 0 0 0
0 0 0 0	1 0 0 0

(\$ 0 0 0 8)

15	0
0 0 0 0	0 0 0 0
0 0 0 0	0 1 1 0

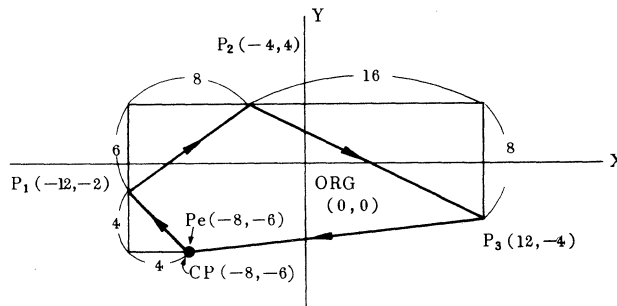
(\$ 0 0 0 6)

15	0
0 0 0 0	0 0 0 1
0 0 0 0	0 0 0 0

(\$ 0 0 1 0)

15	0
1 1 1 1	1 1 1 1
1 1 1 1	1 0 0 0

(\$ F F F 8)



**Figure C25-2 Example of RPLG Execution**

		CRCL																							
[26] CRCL (Circle Command)	PAGE	CRCL-1																							
<p>&lt; <b>FUNCTION</b> &gt; CRCL Command draws a circle of the radius R placing the CP at the center.</p> <p>&lt; <b>MNEMONIC</b> &gt; CRCL (C, AREA, COL, OPM) r</p>	TYPE	Graphic Command																							
<p>&lt; <b>FORMAT</b> &gt;</p> <p>COMMAND CODE <span style="float: right;">hexadecimal notation</span></p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: right; width: 10%;">15</td> <td style="width: 10%;"></td> <td style="text-align: right; width: 10%;">9 8 7</td> <td style="width: 10%;"></td> <td style="text-align: right; width: 10%;">5 4</td> <td style="width: 10%;"></td> <td style="text-align: right; width: 10%;">3 2</td> <td style="width: 10%;"></td> <td style="text-align: right; width: 10%;">0</td> </tr> <tr> <td style="border: 1px solid black; text-align: center;">1 0 1 0</td> <td style="border: 1px solid black; text-align: center;">1 0 0</td> <td style="border: 1px solid black; text-align: center;">C</td> <td style="border: 1px solid black; text-align: center;">AREA</td> <td style="border: 1px solid black; text-align: center;">COL</td> <td style="border: 1px solid black; text-align: center;">OPM</td> <td colspan="3"></td> </tr> </table> <p style="margin-left: 40px;">C = 1 : (\$ A 9 X X) C = 0 : (\$ A 8 X X)</p> <p>COMMAND PARAMETERS</p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: right; width: 10%;">15</td> <td style="width: 80%;"></td> <td style="text-align: right; width: 10%;">0</td> </tr> <tr> <td colspan="3" style="border: 1px solid black; text-align: center; padding: 5px;">r (16 bits)</td> </tr> </table>	15		9 8 7		5 4		3 2		0	1 0 1 0	1 0 0	C	AREA	COL	OPM				15		0	r (16 bits)			<p>WORD NUMBER <math>W_n = 2</math></p> <p>EXECUTION CYCLES <math>C_n = 8d + 66</math></p>
15		9 8 7		5 4		3 2		0																	
1 0 1 0	1 0 0	C	AREA	COL	OPM																				
15		0																							
r (16 bits)																									
<p>&lt; <b>DESCRIPTION</b> &gt;</p> <p>The Circle Command (CRCL) draws a circle placing the Current Pointer (CP) at the center. The command parameter r specifies a radius in units of pixels.</p> <p>First the CP moves in the X-direction from the center for the length of the radius r. Now this point is named P<sub>s</sub>. The circle drawing starts at P<sub>s</sub> and finishes at P<sub>1</sub> (=P<sub>s</sub>). But, a dot is not drawn at P<sub>1</sub>. After the circle has been drawn, the CP moves back to the center and the command is finished. The position of the CP and P<sub>e</sub> are the same.</p> <p>Bit 8 (C) of the command code specifies whether a circle is drawn clockwise or counterclockwise. When C=1, it is drawn clockwise, when C = 0, counterclockwise as shown next page.</p> <p>The parameter radius r is allocated 16 bits, but only the low order 13 bits are effective.</p>																									

CRCL (Circle Command)

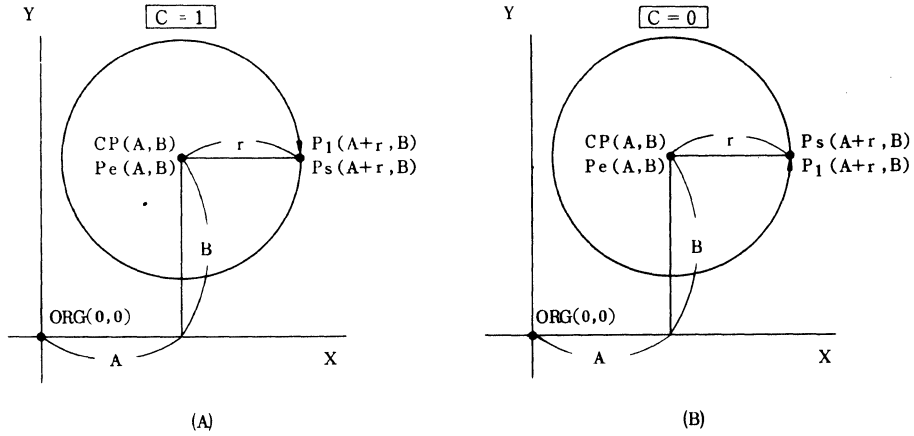
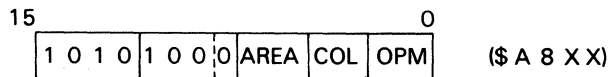


Figure C26-1 Function of CRCL

< EXAMPLE >

If the CP is (0, 0) on the split screen, and r is set 7 in the command parameter, then the CRCL Command draws a circle as shown in figure below.

COMMAND CODE



COMMAND PARAMETERS

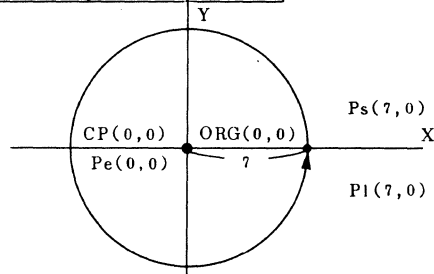
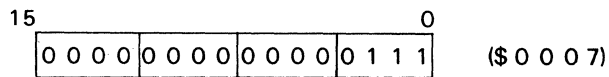


Figure C26-2 Example of CRCL Execution

**ELPS**

<p><b>[27] ELPS (Ellipse Command)</b></p>	<p><b>PAGE</b></p>	<p><b>ELPS-1</b></p>																			
<p>&lt; <b>FUNCTION</b> &gt;          ELPS Command draws an ellipse placing the CP at the center.</p> <p>&lt; <b>MNEMONIC</b> &gt;          ELPS (C, AREA, COL, OPM) a, b, DX</p>	<p>TYPE</p>	<p>Graphic Command</p>																			
<p><b>COMMAND CODE</b></p> <p>hexadecimal notation</p> <table border="1" style="margin-left: 20px;"> <tr> <td style="text-align: right;">15</td> <td style="text-align: center;">9 8 7</td> <td style="text-align: center;">5 4</td> <td style="text-align: center;">3 2</td> <td style="text-align: right;">0</td> </tr> <tr> <td style="text-align: center;">1 0 1 0</td> <td style="text-align: center;">1 1 0</td> <td style="text-align: center;">C</td> <td style="text-align: center;">AREA</td> <td style="text-align: center;">COL</td> <td style="text-align: center;">OPM</td> </tr> </table> <p>C = 1 : (\$ A D X X)          C = 0 : (\$ A C X X)</p> <p><b>COMMAND PARAMETERS</b></p> <table style="margin-left: 20px;"> <tr> <td style="text-align: right;">15</td> <td style="width: 100px; border: 1px solid black; text-align: center;">a (16 bits)</td> <td style="text-align: right;">0</td> </tr> <tr> <td style="text-align: right;">15</td> <td style="width: 100px; border: 1px solid black; text-align: center;">b (16 bits)</td> <td style="text-align: right;">0</td> </tr> <tr> <td style="text-align: right;">15</td> <td style="width: 100px; border: 1px solid black; text-align: center;">dX (16 bits)</td> <td style="text-align: right;">0</td> </tr> </table>	15	9 8 7	5 4	3 2	0	1 0 1 0	1 1 0	C	AREA	COL	OPM	15	a (16 bits)	0	15	b (16 bits)	0	15	dX (16 bits)	0	<p><b>WORD NUMBER</b>          Wn=4</p> <p><b>EXECUTION CYCLES</b>          Cn=10d+90</p>
15	9 8 7	5 4	3 2	0																	
1 0 1 0	1 1 0	C	AREA	COL	OPM																
15	a (16 bits)	0																			
15	b (16 bits)	0																			
15	dX (16 bits)	0																			
<p>&lt; <b>DESCRIPTION</b> &gt;</p> <p>The Ellipse Command (ELPS) draws an ellipse placing the current pointer (CP) at the center.</p> <p>On the X-Y coordinates, if the center of an ellipse is CP (A, B), the major axis is dX, and the minor axis is dY. An ellipse is drawn according to Equation (1) as shown next page.</p> $\frac{(X-A)^2}{dX^2} + \frac{(Y-B)^2}{dY^2} = 1 \dots\dots\dots (1)$ <p>In Equation (1), letting the ratio of squared dX and dY be a, b;</p> $a : b = dX^2 : dY^2 \dots\dots\dots (2)$ <p>Then substituting (2) for Equation (1);</p> $\frac{(X-A)^2}{a} + \frac{(Y-B)^2}{b} = \frac{dX^2}{a} \dots\dots\dots (3)$																					



ELPS (Ellipse Command)

The ELPS Command draws an ellipse according to Equation (3). The a, b, dX are specified in units of pixels.

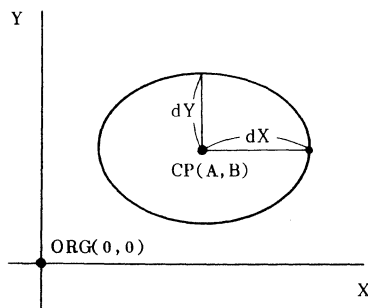


Figure C27-1 Function of ELPS

As shown in figure below, the CP moves in the X-direction from the center for the length of dX. This point is named Ps. The ellipse drawing starts at Ps and finishes at P1 (=Ps). But, the dot is not drawn at P1. After the ellipse has been drawn, the CP moves back to the center, and the command is finished. The first position of the CP and Pe are the same.

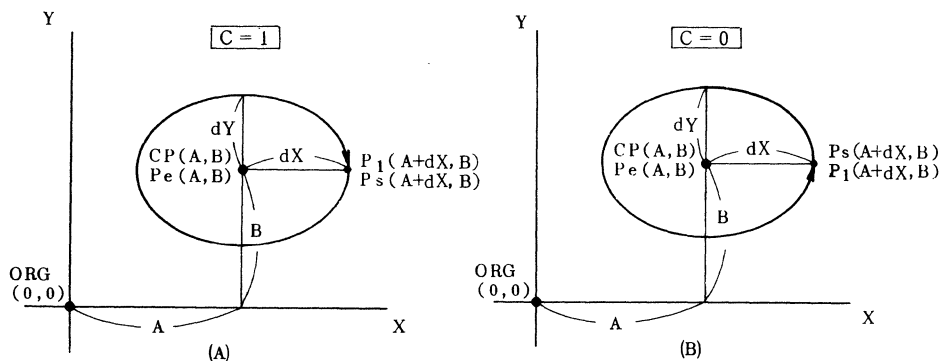


Figure C27-2 Drawing Direction of ELPS

< EXAMPLE >

Bit 8 (c) of the command code specifies whether an ellipse is drawn clockwise or counterclockwise. When C = 1, it is drawn clockwise, when C = 0, counterclockwise as shown in previous page.

If the bit length of a, b, dX are  $\mathcal{L}a$ ,  $\mathcal{L}b$ ,  $\mathcal{L}dX$ , then the bit length of these parameters must be as follows;

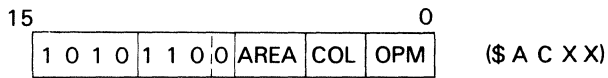
$$\mathcal{L}a + \mathcal{L}dX \leq 13$$

$$\mathcal{L}b + \mathcal{L}dX \leq 13$$

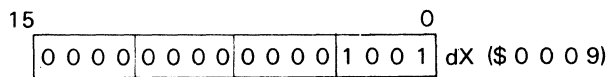
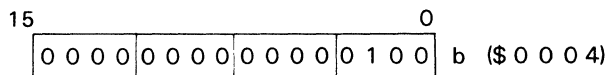
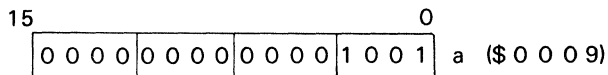
< EXECUTION EXAMPLE >

If the absolute coordinate of CP is (16, 10) on the split screen, a is set to 9, b to 4, dX to 9 in the command parameter, then the ELPS Command (C = 0) draws an ellipse as shown below.

COMMAND CODE



COMMAND PARAMETERS



$$9 : 4 = 9^2 : 6^2$$

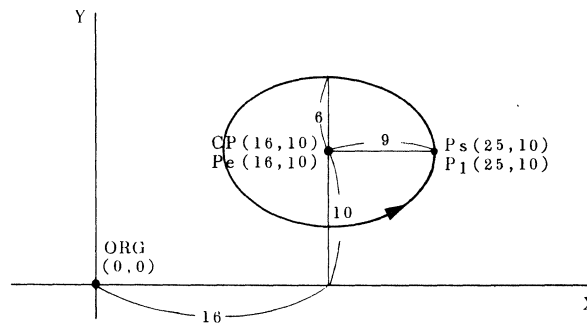


Figure C27-3 Example of ELPS Execution

		AARC																																																
[28] AARC (Absolute Arc)		PAGE	AARC-1																																															
<p>&lt; FUNCTION &gt;  AARC draws an arc by current pointer (start point), end point, and center point of the absolute coordinate.</p> <p>&lt; MNEMONIC &gt;  AARC (C, AREA, COL, OPM) Xc, Yc, Xe, Ye</p>		TYPE	Graphic Command																																															
<p>&lt; FORMAT &gt;</p> <p>COMMAND CODE</p> <table style="margin-left: 20px;"> <tr> <td style="text-align: right;">15</td> <td style="text-align: center;">9 8 7</td> <td style="text-align: center;">5 4</td> <td style="text-align: center;">3 2</td> <td style="text-align: right;">0</td> <td></td> </tr> <tr> <td style="border: 1px solid black; padding: 2px;">1</td> <td style="border: 1px solid black; padding: 2px;">0</td> <td style="border: 1px solid black; padding: 2px;">0</td> <td style="border: 1px solid black; padding: 2px;">0</td> <td style="border: 1px solid black; padding: 2px;">C</td> <td style="border: 1px solid black; padding: 2px;">AREA</td> </tr> <tr> <td style="border: 1px solid black; padding: 2px;"></td> <td style="border: 1px solid black; padding: 2px;"></td> <td style="border: 1px solid black; padding: 2px;"></td> <td style="border: 1px solid black; padding: 2px;"></td> <td style="border: 1px solid black; padding: 2px;"></td> <td style="border: 1px solid black; padding: 2px;">COL</td> </tr> <tr> <td style="border: 1px solid black; padding: 2px;"></td> <td style="border: 1px solid black; padding: 2px;"></td> <td style="border: 1px solid black; padding: 2px;"></td> <td style="border: 1px solid black; padding: 2px;"></td> <td style="border: 1px solid black; padding: 2px;"></td> <td style="border: 1px solid black; padding: 2px;">OPM</td> </tr> </table> <p style="margin-left: 20px;">hexadecimal notation  C = 1 : (\$ B 1 X X)  C = 0 : (\$ B 0 X X)</p> <p>COMMAND PARAMETERS</p> <table style="margin-left: 20px;"> <tr> <td style="text-align: right;">15</td> <td style="border: 1px solid black; width: 150px; height: 20px;"></td> <td style="text-align: right;">0</td> </tr> <tr> <td></td> <td style="text-align: center;">Xc (16 bits)</td> <td></td> </tr> <tr> <td style="text-align: right;">15</td> <td style="border: 1px solid black; width: 150px; height: 20px;"></td> <td style="text-align: right;">0</td> </tr> <tr> <td></td> <td style="text-align: center;">Yc (16 bits)</td> <td></td> </tr> <tr> <td style="text-align: right;">15</td> <td style="border: 1px solid black; width: 150px; height: 20px;"></td> <td style="text-align: right;">0</td> </tr> <tr> <td></td> <td style="text-align: center;">Xe (16 bits)</td> <td></td> </tr> <tr> <td style="text-align: right;">15</td> <td style="border: 1px solid black; width: 150px; height: 20px;"></td> <td style="text-align: right;">0</td> </tr> <tr> <td></td> <td style="text-align: center;">Ye (16 bits)</td> <td></td> </tr> </table>		15	9 8 7	5 4	3 2	0		1	0	0	0	C	AREA						COL						OPM	15		0		Xc (16 bits)		15		0		Yc (16 bits)		15		0		Xe (16 bits)		15		0		Ye (16 bits)		<p>WORD NUMBER  Wn=5</p> <p>EXECUTION CYCLES  Cn=8d+18</p>
15	9 8 7	5 4	3 2	0																																														
1	0	0	0	C	AREA																																													
					COL																																													
					OPM																																													
15		0																																																
	Xc (16 bits)																																																	
15		0																																																
	Yc (16 bits)																																																	
15		0																																																
	Xe (16 bits)																																																	
15		0																																																
	Ye (16 bits)																																																	
<p>&lt; DESCRIPTION &gt;</p> <p>As shown in Fig. C28-1, the AARC command draws an arc from the current pointer, CP, to Pe of the absolute coordinate, the absolute coordinates CC (Xc, Yc) being the center point. The X and Y components of the absolute coordinates CC and Pe are set in the first and second parameters in units of pixels. After the arc drawing, current pointer moves to Pe. However a dot is not drawn at Pe. The command code bit 8 (C) selects whether an arc is drawn clockwise or counterclockwise. When C is "1", the arc is drawn clockwise, and when C is "0", the arc is drawn counterclockwise as shown in Fig. C28-1.</p>																																																		

The command parameters are allocated 16 bits, but only the low order 13 bits are effective.

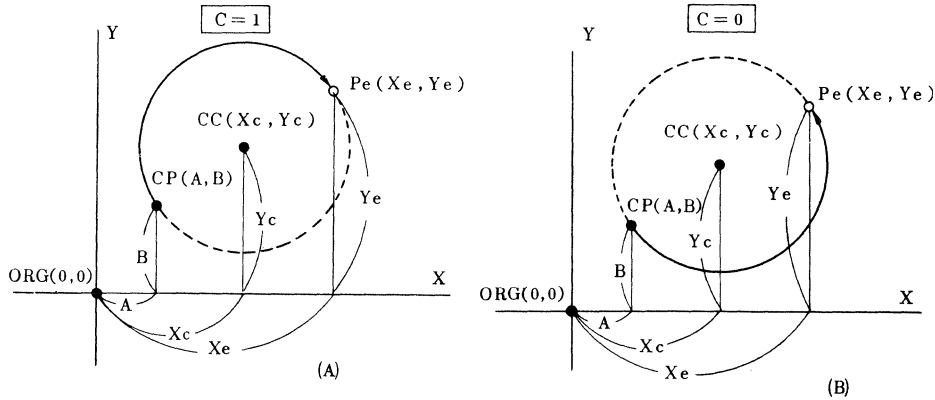
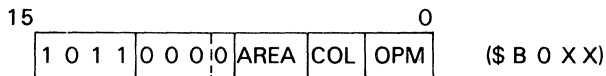


Figure C28-1 Function of AARC Command

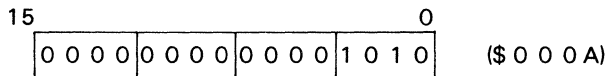
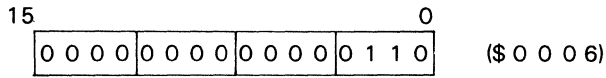
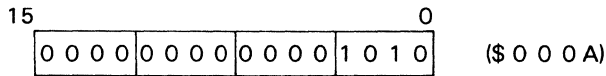
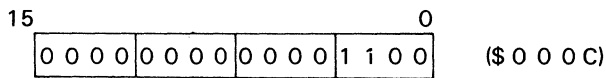
< EXAMPLE >

If the coordinate of CP is at (12, 4) on the split screen, Xc is set to 12, Yc to 10, Xe to 6, and Ye to 10 in the command parameter, then the AARC Command (C = 0) draws an arc as shown in figure next page.

COMMAND CODE



COMMAND PARAMETERS



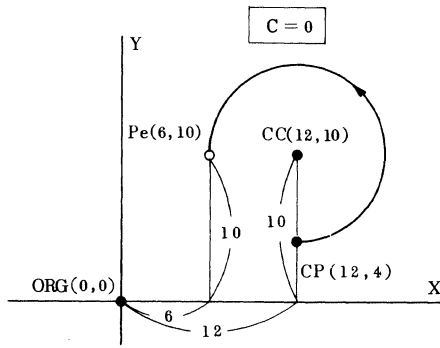


Figure C28-2 Example of AARC Execution

		RARC																																						
[29] RARC (Relative Arc)		PAGE	RARC-1																																					
<p>&lt; <b>FUNCTION</b> &gt; RARC draws an arc by current pointer (start point), end point, and center point of the relative coordinate.</p> <p>&lt; <b>MNEMONIC</b> &gt; RARC (C, AREA, COL, OPM) dXc, dYc, dXe, dYe</p>		TYPE	Graphic Command																																					
<p>&lt; <b>FORMAT</b> &gt;</p> <p>COMMAND CODE <span style="float: right;">hexadecimal notation</span></p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: right; width: 5%;">15</td> <td style="width: 15%;"></td> <td style="text-align: center; width: 5%;">9</td> <td style="text-align: center; width: 5%;">8</td> <td style="text-align: center; width: 5%;">7</td> <td style="width: 15%;"></td> <td style="text-align: center; width: 5%;">5</td> <td style="text-align: center; width: 5%;">4</td> <td style="text-align: center; width: 5%;">3</td> <td style="text-align: center; width: 5%;">2</td> <td style="text-align: center; width: 5%;">0</td> <td style="width: 15%;"></td> </tr> <tr> <td></td> <td style="border: 1px solid black; text-align: center;">1 0 1 1</td> <td style="border: 1px solid black; text-align: center;">0 1 0</td> <td style="border: 1px solid black; text-align: center;">C</td> <td style="border: 1px solid black; text-align: center;">AREA</td> <td style="border: 1px solid black; text-align: center;">COL</td> <td style="border: 1px solid black; text-align: center;">OPM</td> <td colspan="5"></td> <td style="vertical-align: top;">           C = 1 : (\$ B 5 X X)            C = 0 : (\$ B 4 X X)         </td> </tr> </table> <p>COMMAND PARAMETERS</p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: right; width: 5%;">15</td> <td style="width: 85%; border: 1px solid black; text-align: center;">dXc (16 bits)</td> <td style="text-align: right; width: 5%;">0</td> </tr> <tr> <td style="text-align: right;">15</td> <td style="border: 1px solid black; text-align: center;">dYc (16 bits)</td> <td style="text-align: right;">0</td> </tr> <tr> <td style="text-align: right;">15</td> <td style="border: 1px solid black; text-align: center;">dXe (16 bits)</td> <td style="text-align: right;">0</td> </tr> <tr> <td style="text-align: right;">15</td> <td style="border: 1px solid black; text-align: center;">dYe (16 bits)</td> <td style="text-align: right;">0</td> </tr> </table>		15		9	8	7		5	4	3	2	0			1 0 1 1	0 1 0	C	AREA	COL	OPM						C = 1 : (\$ B 5 X X) C = 0 : (\$ B 4 X X)	15	dXc (16 bits)	0	15	dYc (16 bits)	0	15	dXe (16 bits)	0	15	dYe (16 bits)	0	<p>WORD NUMBER Wn=5</p> <p>EXECUTION CYCLES Cn=8d+18</p>	
15		9	8	7		5	4	3	2	0																														
	1 0 1 1	0 1 0	C	AREA	COL	OPM						C = 1 : (\$ B 5 X X) C = 0 : (\$ B 4 X X)																												
15	dXc (16 bits)	0																																						
15	dYc (16 bits)	0																																						
15	dXe (16 bits)	0																																						
15	dYe (16 bits)	0																																						
<p>&lt; <b>DESCRIPTION</b> &gt;</p> <p>As shown in Fig. C29-1, the RARC command draws an arc from the current pointer, CP, to Pe (A+dXe, B+dYe) of the relative coordinates, the relative coordinates CC (A+dXc, B+dYc) being the center points. The X and Y components of the relative coordinates CC and Pe are set in the first and second parameters in units of pixels. CP moves to the end point Pe when an arc is drawn. However a dot is not drawn at Pe. The command code bit 8(C) selects whether an arc is drawn clockwise or counterclockwise. When C is "1", the arc is drawn clockwise, and when C is "0", the arc is drawn counterclockwise as shown in Fig. C29-2.</p>																																								

The command parameters are allocated 16 bits, but only the low order 13 bits are effective.

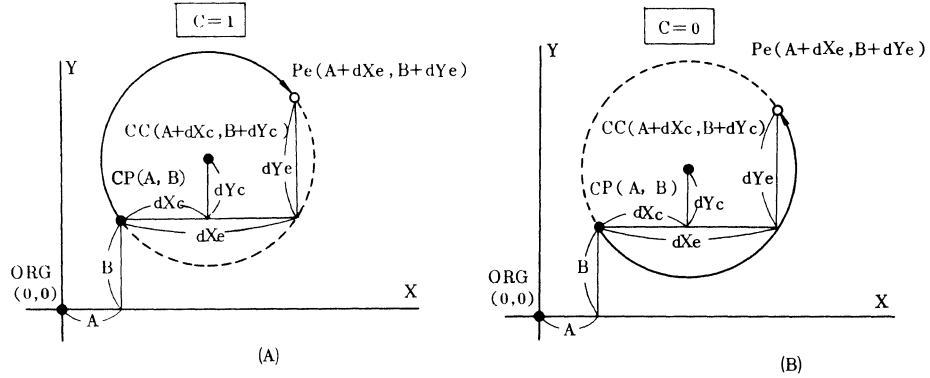


Figure C29-1 Function of RARC

< EXAMPLE >

If the coordinate of CP is at (6, 10) on the split screen, dXc is set to 6, dYc to 0, dXe to 6, and dYe to 6 in the command parameter, then the RARC command (C = 0) draws an arc as shown next page.

COMMAND CODE

15	0
1 0 1 1	0 1 0 0
AREA	COL ODM

(\$ B 4 X X)

COMMAND PARAMETERS

15	0
0 0 0 0	0 0 0 0
0 0 0 0	0 1 1 0

(\$ 0 0 0 6)

15	0
0 0 0 0	0 0 0 0
0 0 0 0	0 0 0 0

(\$ 0 0 0 0)

15	0
0 0 0 0	0 0 0 0
0 0 0 0	0 1 1 0

(\$ 0 0 0 6)

15	0
0 0 0 0	0 0 0 0
0 0 0 0	0 1 1 0

(\$ 0 0 0 6)

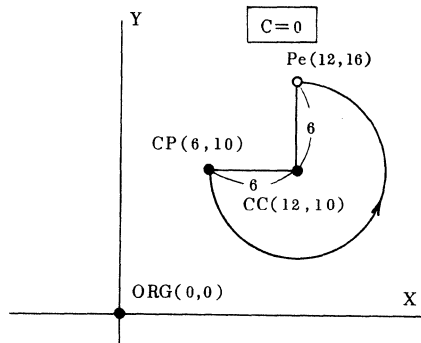


Figure C29-2 Example of RARC Execution



**AEARC**

<b>[30] AEARC (Absolute Ellipse ARC)</b>	<b>PAGE</b>	<b>AEARC-1</b>																																									
<p>&lt; <b>FUNCTION</b> &gt; AEARC draws an ellipse ARC.</p> <p>&lt; <b>MNEMONIC</b> &gt; AEARC (C, AREA, COL, OPM) a, b, Xc, Yc, Xe, Ye</p>	TYPE	Graphic Command																																									
<p>&lt; <b>FORMAT</b> &gt;</p> <p>COMMAND CODE <span style="float: right;">hexadecimal notation</span></p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: right; width: 5%;">15</td> <td style="width: 10%;"></td> <td style="text-align: center; width: 10%;">9 8 7</td> <td style="width: 10%;"></td> <td style="text-align: center; width: 10%;">5 4</td> <td style="width: 10%;"></td> <td style="text-align: center; width: 10%;">3 2</td> <td style="width: 10%;"></td> <td style="text-align: right; width: 5%;">0</td> </tr> <tr> <td></td> <td style="border: 1px solid black; text-align: center;">1 0 1 1</td> <td style="border: 1px solid black; text-align: center;">1 0 0</td> <td style="border: 1px solid black; text-align: center;">C</td> <td style="border: 1px solid black; text-align: center;">AREA</td> <td style="border: 1px solid black; text-align: center;">COL</td> <td style="border: 1px solid black; text-align: center;">OPM</td> <td></td> <td></td> </tr> </table> <p style="margin-left: 100px;">C = 1 : (\$ B 9 X X) C = 0 : (\$ B 8 X X)</p> <p>COMMAND PARAMETERS</p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: right; width: 5%;">15</td> <td style="width: 10%;"></td> <td style="width: 80%; border: 1px solid black; text-align: center;">a (16 bits)</td> <td style="text-align: right; width: 5%;">0</td> </tr> <tr> <td style="text-align: right;">15</td> <td></td> <td style="border: 1px solid black; text-align: center;">b (16 bits)</td> <td style="text-align: right;">0</td> </tr> <tr> <td style="text-align: right;">15</td> <td></td> <td style="border: 1px solid black; text-align: center;">Xc (16 bits)</td> <td style="text-align: right;">0</td> </tr> <tr> <td style="text-align: right;">15</td> <td></td> <td style="border: 1px solid black; text-align: center;">Yc (16 bits)</td> <td style="text-align: right;">0</td> </tr> <tr> <td style="text-align: right;">15</td> <td></td> <td style="border: 1px solid black; text-align: center;">Xe (16 bits)</td> <td style="text-align: right;">0</td> </tr> <tr> <td style="text-align: right;">15</td> <td></td> <td style="border: 1px solid black; text-align: center;">Ye (16 bits)</td> <td style="text-align: right;">0</td> </tr> </table>	15		9 8 7		5 4		3 2		0		1 0 1 1	1 0 0	C	AREA	COL	OPM			15		a (16 bits)	0	15		b (16 bits)	0	15		Xc (16 bits)	0	15		Yc (16 bits)	0	15		Xe (16 bits)	0	15		Ye (16 bits)	0	<p>WORD NUMBER <math>W_n = 7</math></p> <p>EXECUTION CYCLES <math>C_n = 10d + 96</math></p>
15		9 8 7		5 4		3 2		0																																			
	1 0 1 1	1 0 0	C	AREA	COL	OPM																																					
15		a (16 bits)	0																																								
15		b (16 bits)	0																																								
15		Xc (16 bits)	0																																								
15		Yc (16 bits)	0																																								
15		Xe (16 bits)	0																																								
15		Ye (16 bits)	0																																								
<p>&lt; <b>DESCRIPTION</b> &gt;</p> <p>The AEARC command draws an arc from the current pointer, CP, to Pe of the absolute coordinate, the absolute coordinates CC (Xc, Yc) being the center point. the X and Y components of the absolute coordinates CC and Pe are set in the command parameters in units of pixels.</p> <p>CP moves to the end point Pe when an arc is drawn. However a dot is not drawn at Pe.</p>																																											

The command code bit 8(C) selects whether an arc is drawn clockwise or counterclockwise. When C is "1", the arc is drawn clockwise, and when C is "0", the arc is drawn counterclockwise as shown in Fig. C30-1.

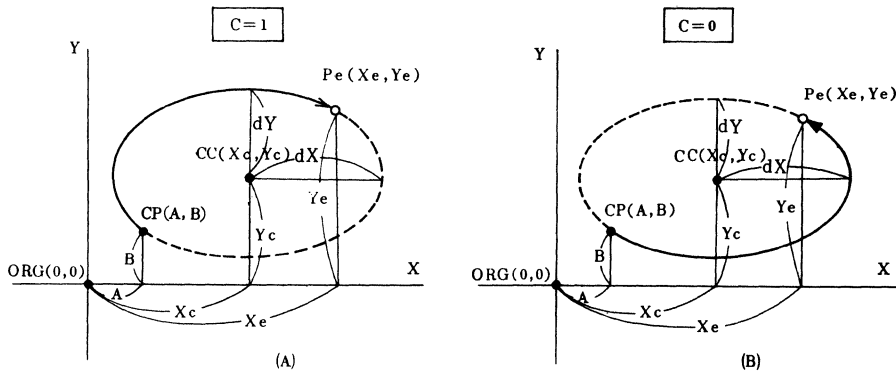


Figure C30-1 Function of AEARC

< RELATED EQUATIONS >

In the X-Y coordinate, let the center point of the ellipse be  $CC(X_c, Y_c)$ , let the length of the X-axis be  $dX$ , and let the length of the Y-axis be  $dY$ . Depending on (1), an ellipse ARC is drawn as shown in Fig. C30-2.

$$\frac{(X - X_c)^2}{dX^2} + \frac{(Y - Y_c)^2}{dY^2} = 1 \dots\dots\dots (1)$$

When letting  $dX^2$  and  $dY^2$  be  $a$  and  $b$ ,  
then  $a : b = dX^2 : dY^2 \dots\dots\dots (2)$

by substituting (2) for (1), the result is

$$\frac{(X - X_c)^2}{a} + \frac{(Y - Y_c)^2}{b} = \frac{dX^2}{a} \dots\dots\dots (3)$$

The AEARC draws an ellipse ARC according to Equation (3).

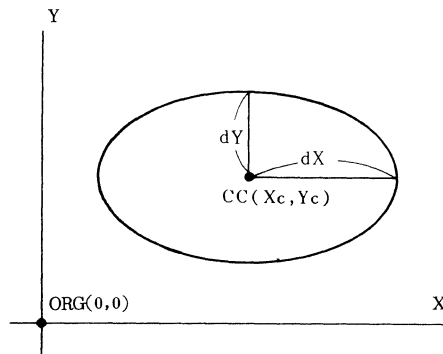


Figure C30-2 Notation of an Ellipse (1)

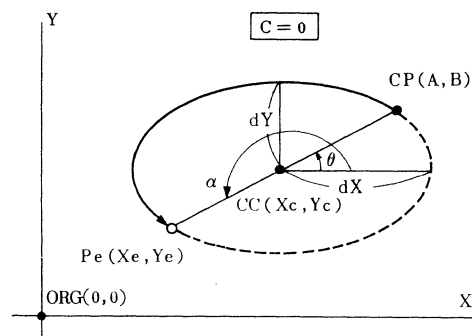


Figure C30-3 Notation of an Ellipse (2)

When setting CP (A, B) and CPe (Xe, Ye) as shown in Fig. C30-3 for an ellipse arc drawing, the following equations are applicable.

AEARC (Absolute Ellipse ARC)	PAGE	AEARC-4
------------------------------	------	---------

$$A = \frac{dXdY \cos \theta}{\sqrt{dX^2 \sin^2 \theta + dY^2 \cos^2 \theta}} + Xc \dots\dots\dots (4)$$

$$B = \frac{dXdY \sin \theta}{\sqrt{dX^2 \sin^2 \theta + dY^2 \cos^2 \theta}} + Yc \dots\dots\dots (5)$$

$$Xe = \frac{dXdY \cos \alpha}{\sqrt{dX^2 \sin^2 \alpha + dY^2 \cos^2 \alpha}} + Xc \dots\dots\dots (6)$$

$$Ye = \frac{dXdY \sin \alpha}{\sqrt{dX^2 \sin^2 \alpha + dY^2 \cos^2 \alpha}} + Yc \dots\dots\dots (7)$$

a, b, Xc, Yc, Xe and Ye are given as a parameter to the AEARC command in units of pixels. When setting the command parameters, CC (Xc, Yc) of an ellipse, and CP (A, B) and Pe (Xe, Ye) and ellipse ARC must meet the above (4), (5), (6) and (7) equations.

		REARC																																																											
[31] REARC (Relative Ellipse ARC)		PAGE	REARC-1																																																										
<p>&lt; FUNCTION &gt; REARC draws an ellipse ARC.</p> <p>&lt; MNEMONIC &gt; REARC (C, AREA, COL, OPM) a, b, dXc, dYc, dXe, dYe</p>		TYPE	Graphic Command																																																										
<p>&lt; FORMAT &gt;</p> <p>COMMAND CODE <span style="float: right;">hexadecimal notation</span></p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: right; width: 10%;">15</td> <td style="width: 10%;"></td> <td style="text-align: center; width: 10%;">9 8 7</td> <td style="width: 10%;"></td> <td style="text-align: center; width: 10%;">5 4</td> <td style="width: 10%;"></td> <td style="text-align: center; width: 10%;">3 2</td> <td style="width: 10%;"></td> <td style="text-align: right; width: 10%;">0</td> <td style="width: 10%;"></td> </tr> <tr> <td colspan="10" style="border: 1px solid black; text-align: center;"> <div style="display: flex; justify-content: space-between; align-items: center;"> <span>1 0 1 1</span> <span>1 1 0</span> <span>C</span> <span>AREA</span> <span>COL</span> <span>OPM</span> </div> </td> </tr> <tr> <td colspan="5"></td> <td colspan="5">C = 1 : (\$ B D X X)</td> </tr> <tr> <td colspan="5"></td> <td colspan="5">C = 0 : (\$ B C X X)</td> </tr> </table> <p>COMMAND PARAMETERS</p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: right; width: 10%;">15</td> <td style="width: 80%; border: 1px solid black; text-align: center;">a (16 bits)</td> <td style="text-align: right; width: 10%;">0</td> </tr> <tr> <td style="text-align: right;">15</td> <td style="border: 1px solid black; text-align: center;">b (16 bits)</td> <td style="text-align: right;">0</td> </tr> <tr> <td style="text-align: right;">15</td> <td style="border: 1px solid black; text-align: center;">dXc (16 bits)</td> <td style="text-align: right;">0</td> </tr> <tr> <td style="text-align: right;">15</td> <td style="border: 1px solid black; text-align: center;">dYc (16 bits)</td> <td style="text-align: right;">0</td> </tr> <tr> <td style="text-align: right;">15</td> <td style="border: 1px solid black; text-align: center;">dXe (16 bits)</td> <td style="text-align: right;">0</td> </tr> <tr> <td style="text-align: right;">15</td> <td style="border: 1px solid black; text-align: center;">dYe (16 bits)</td> <td style="text-align: right;">0</td> </tr> </table>		15		9 8 7		5 4		3 2		0		<div style="display: flex; justify-content: space-between; align-items: center;"> <span>1 0 1 1</span> <span>1 1 0</span> <span>C</span> <span>AREA</span> <span>COL</span> <span>OPM</span> </div>															C = 1 : (\$ B D X X)										C = 0 : (\$ B C X X)					15	a (16 bits)	0	15	b (16 bits)	0	15	dXc (16 bits)	0	15	dYc (16 bits)	0	15	dXe (16 bits)	0	15	dYe (16 bits)	0	WORD NUMBER Wn=7	EXECUTION CYCLES Cn=10d+96
15		9 8 7		5 4		3 2		0																																																					
<div style="display: flex; justify-content: space-between; align-items: center;"> <span>1 0 1 1</span> <span>1 1 0</span> <span>C</span> <span>AREA</span> <span>COL</span> <span>OPM</span> </div>																																																													
					C = 1 : (\$ B D X X)																																																								
					C = 0 : (\$ B C X X)																																																								
15	a (16 bits)	0																																																											
15	b (16 bits)	0																																																											
15	dXc (16 bits)	0																																																											
15	dYc (16 bits)	0																																																											
15	dXe (16 bits)	0																																																											
15	dYe (16 bits)	0																																																											
<p>&lt; DESCRIPTION &gt;</p> <p>As shown in Fig. C31-1, the REARC command draws an arc from the current pointer, CP, to Pe (dXe, dYe) of the relative coordinate, the relative coordinates CC (dXc, dYc) being the center point.</p> <p>The X and Y components of the relative coordinates CC and Pe are set in the command parameters in units of pixels.</p>																																																													

The command code bit 8 (C) selects whether an arc is drawn clockwise or counterclockwise. When C is "1", the arc is drawn clockwise, and when C is "0", the arc is drawn counterclockwise as shown in Fig. C31-1.

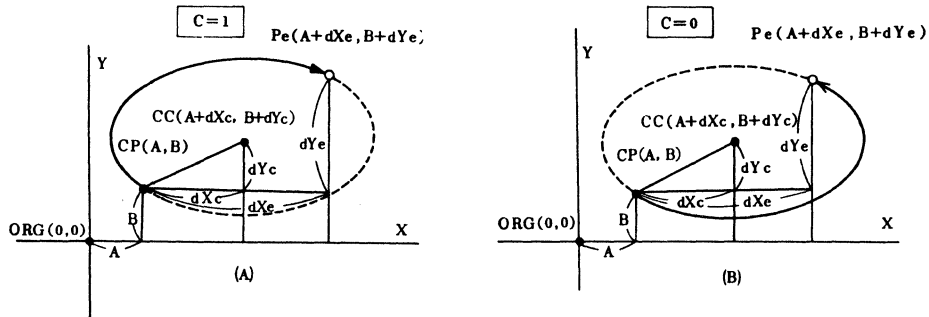


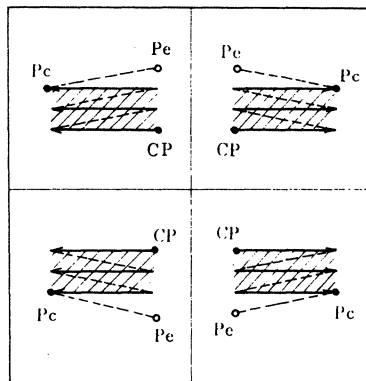
Figure C31-1 Function of REARC

**AFRCT**

<p><b>[32] AFRCT (Absolute Filled Rectangle)</b></p>	<p><b>PAGE</b></p>	<p><b>AFRCT-1</b></p>																										
<p>&lt; <b>FUNCTION</b> &gt;          AFRCT command paints the rectangular area specified with CP (Current Pointer) and the command parameter (the absolute coordinates) according to a figure pattern stored in the Pattern RAM.</p> <p>&lt; <b>MNEMONIC</b> &gt;          AFRC (AREA, COL, OPM) X, Y</p>	<p><b>TYPE</b></p>	<p>Graphic Command</p>																										
<p>&lt; <b>FORMAT</b> &gt;</p> <p>COMMAND CODE <span style="float: right;">hexadecimal notation</span></p> <table border="1" style="margin-left: 20px;"> <tr> <td style="text-align: center;">15</td> <td style="text-align: center;">8 7</td> <td style="text-align: center;">5 4</td> <td style="text-align: center;">3 2</td> <td style="text-align: center;">0</td> <td></td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> </tr> <tr> <td></td> <td></td> <td style="text-align: center;">AREA</td> <td style="text-align: center;">COL</td> <td style="text-align: center;">OPM</td> <td style="text-align: center;">(\$ C 0 X X)</td> </tr> </table> <p>COMMAND PARAMETERS</p> <table border="1" style="margin-left: 20px;"> <tr> <td style="text-align: center;">15</td> <td style="text-align: center;">0</td> </tr> <tr> <td colspan="2" style="text-align: center;">X (16 bits)</td> </tr> </table> <table border="1" style="margin-left: 20px;"> <tr> <td style="text-align: center;">15</td> <td style="text-align: center;">0</td> </tr> <tr> <td colspan="2" style="text-align: center;">Y (16 bits)</td> </tr> </table>	15	8 7	5 4	3 2	0		1	1	0	0	0	0			AREA	COL	OPM	(\$ C 0 X X)	15	0	X (16 bits)		15	0	Y (16 bits)		<p><b>WORD NUMBER</b>  <math>W_n = 3</math></p> <p><b>EXECUTION CYCLES</b>  <math>C_n = (P \cdot A + 8)B + 18</math></p>	
15	8 7	5 4	3 2	0																								
1	1	0	0	0	0																							
		AREA	COL	OPM	(\$ C 0 X X)																							
15	0																											
X (16 bits)																												
15	0																											
Y (16 bits)																												
<p>&lt; <b>DESCRIPTION</b> &gt;</p> <p>The Absolute Filled Rectangle Command (AFRCT) paints the rectangle area according to the color information in the pattern RAM. The sizes of the rectangle are parallel to the coordinate axis. Two corner points on the diagonal are CP and Pc (X, Y) at the absolute coordinate point from the origin.</p> <p>Pc (X, Y) expressed in the absolute X-Y coordinates from the origin are given by the command parameter in units of pixels.</p> <div style="text-align: center;"> </div> <p><b>Figure C32-1 Function of AFRCT</b></p>																												

**AFRCT (Absolute Filled Rectangle)**

Painting in a rectangular area depends on the position of CP and Pc, as shown in Fig. C32-2. In Fig. C32-2, painting between CP and Pc is performed. CP is moved to Pe at the termination of the command. The drawing at the end point Pe is not performed.

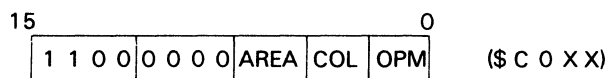


**Figure C32-2 Painting Direction of AFRCT**

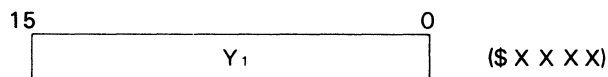
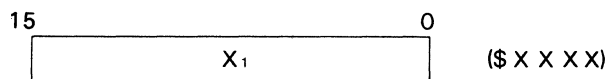
**< EXAMPLE >**

If the absolute coordinate of CP is (A, B) on the split screen, X is set to X<sub>1</sub> and Y to Y<sub>1</sub> in the command parameter, and the drawing parameter register for the pattern RAM is set to the following, the pattern start point (PSX, PSY), the pattern end point (PEX, PEY), the graphic pattern pointer (PPX, PPY), then, the rectangular area is painted with the AFRCT command as shown next page.

**COMMAND CODE**



**COMMAND PARAMETERS**





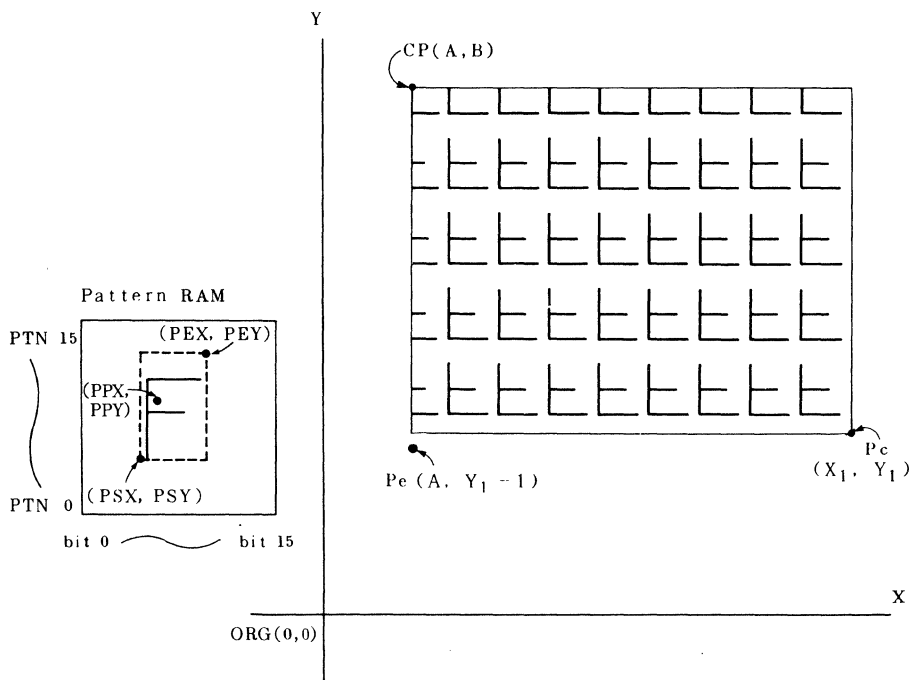


Figure C32-3 Example of AFRCT Execution

		<b>RFRCT</b>																															
<b>[33] RFRCT (Relative Filled Rectangle)</b>		<b>PAGE</b>	<b>RFRCT-1</b>																														
<p>&lt; <b>FUNCTION</b> &gt;  RFRCT command paints in the rectangular area specified with CP (Current Pointer) and the command parameter (the relative coordinates) according to a figure pattern stored in the Pattern RAM.</p> <p>&lt; <b>MNEMONIC</b> &gt;  RFRCT (AREA, COL, OPM) dX, dY</p>		TYPE	Graphic Command																														
<p>&lt; <b>FORMAT</b> &gt;</p> <p>COMMAND CODE <span style="float: right;">hexadecimal notation</span></p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: right;">15</td> <td style="text-align: center;">8 7</td> <td style="text-align: center;">5 4</td> <td style="text-align: center;">3 2</td> <td style="text-align: center;">0</td> <td></td> </tr> <tr> <td style="border: 1px solid black; text-align: center;">1</td> <td style="border: 1px solid black; text-align: center;">1</td> <td style="border: 1px solid black; text-align: center;">0</td> <td style="border: 1px solid black; text-align: center;">0</td> <td style="border: 1px solid black; text-align: center;">0</td> <td style="border: 1px solid black; text-align: center;">0</td> </tr> <tr> <td></td> <td style="text-align: center;">AREA</td> <td style="text-align: center;">COL</td> <td style="text-align: center;">OPM</td> <td></td> <td style="text-align: center;">(\$ C 4 X X)</td> </tr> </table> <p>COMMAND PARAMETERS</p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: right;">15</td> <td style="border: 1px solid black; width: 150px; height: 20px;"></td> <td style="text-align: right;">0</td> </tr> <tr> <td></td> <td style="text-align: center;">dX (16 bits)</td> <td></td> </tr> <tr> <td style="text-align: right;">15</td> <td style="border: 1px solid black; width: 150px; height: 20px;"></td> <td style="text-align: right;">0</td> </tr> <tr> <td></td> <td style="text-align: center;">dY (16 bits)</td> <td></td> </tr> </table>		15	8 7	5 4	3 2	0		1	1	0	0	0	0		AREA	COL	OPM		(\$ C 4 X X)	15		0		dX (16 bits)		15		0		dY (16 bits)		<p>WORD NUMBER  <math>W_n = 3</math></p> <p>EXECUTION CYCLES  <math>C_n = (P \cdot A + 8)B + 18</math></p>	
15	8 7	5 4	3 2	0																													
1	1	0	0	0	0																												
	AREA	COL	OPM		(\$ C 4 X X)																												
15		0																															
	dX (16 bits)																																
15		0																															
	dY (16 bits)																																
<p>&lt; <b>DESCRIPTION</b> &gt;</p> <p>The Relative Filled Rectangle Command (RFRCT) paints the rectangular area according to the color information in the pattern RAM. The sizes of the rectangle are parallel to the coordinates axis. Two corner points on the diagonal are CP and Pe (A + dX, B + dY) at the relative coordinate point from CP.</p> <p>Pe (dX, dY) expressed in the relative coordinate from CP is given by the command parameter in units of pixels.</p> <div style="text-align: center;"> </div>																																	
<p><b>Figure C33-1 Function of RFRCT</b></p>																																	

## RFRCT (Relative Filled Rectangle)

PAGE

RFRCT-2

Painting in a rectangular area depends on the position of CP and Pe, as shown in Fig. C33-2. In Fig. C33-2, painting between CP and Pe is performed. CP is moved to Pe at the termination of the command. The drawing at the end point Pe is not performed.

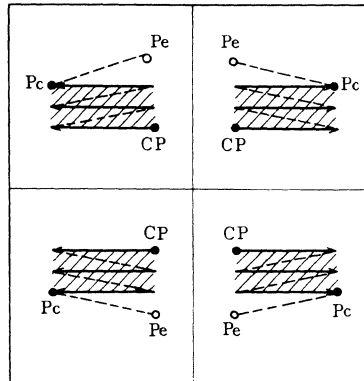


Figure C33-2 Painting Direction of RFRCT

## &lt; EXAMPLE &gt;

If the absolute coordinate of CP is (A, B) on the split screen, dX is set to dX<sub>1</sub> and dY to dY<sub>1</sub> in the command parameter, and the drawing parameter register for the pattern RAM is set to the following, the pattern start point (PSX, PSY), the pattern end point (PEX, PEY), the graphic pattern pointer (PPX, PPY), then, the rectangular area is painted with the RFRCT command, as shown in Fig. C33-3.

## COMMAND CODE

15	0
1 1 0 0 0 1 0 0	AREA COL OPM (\$ C 4 X X)

## COMMAND PARAMETERS

15	0
dX <sub>1</sub>	(\$ X X X X)

15	0
dY <sub>1</sub>	(\$ X X X X)

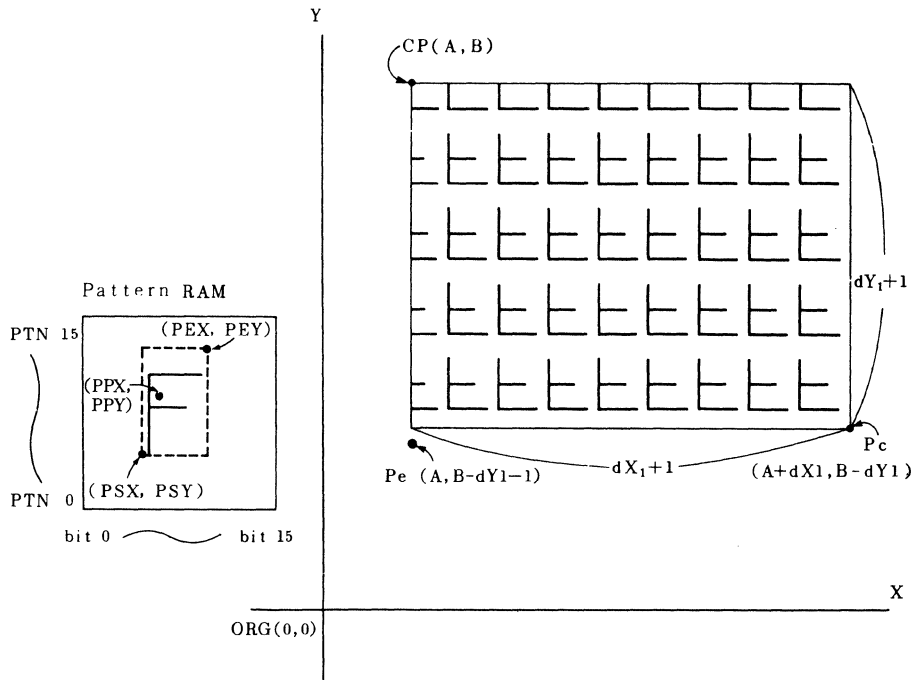
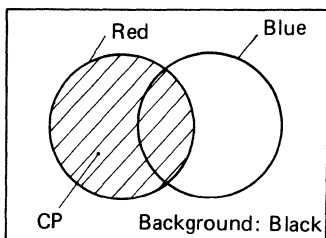


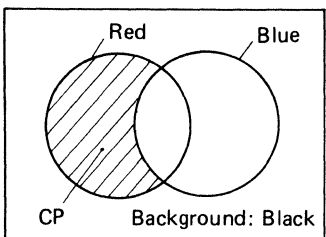
Figure C33-3 Example of RFRCT Execution

		PAINT	
[34] PAINT (Paint)	PAGE	PAINT-1	
<p>&lt; <b>FUNCTION</b> &gt;            PAINT command paints the closed area surrounded by edge color using the figure pattern stored in the pattern RAM.</p> <p>&lt; <b>MNEMONIC</b> &gt;            PAINT (AREA, COL, OPM)</p>		TYPE	Graphic Command
<p>&lt; <b>FORMAT</b> &gt;</p> <p>COMMAND CODE <span style="float: right;">hexadecimal notation</span></p> <div style="display: flex; align-items: center; justify-content: center;"> <div style="text-align: center; margin-right: 10px;">           15            1 1 0 0         </div> <div style="text-align: center; margin-right: 10px;">           9 8 7            1 0 0         </div> <div style="text-align: center; margin-right: 10px;">           5 4            E         </div> <div style="text-align: center; margin-right: 10px;">           3 2            AREA         </div> <div style="text-align: center; margin-right: 10px;">           0            COL         </div> <div style="text-align: center; margin-right: 10px;">           OPM         </div> <div style="margin-left: 20px;">           (\$ C X X X)         </div> </div> <p>COMMAND PARAMETERS            – NON –</p>		<p>WORD NUMBER  <math>W_n = 1</math></p> <p>Command execution cycle number</p> <p><math>C_n = (18 \cdot A + 102)B - 58</math></p> <p>(When painting rectangle)</p>	
<p>&lt; <b>DESCRIPTION</b> &gt;</p> <p>The "Paint" command (PAINT) paints the closed area surrounded by edge color defined in the parameter register (EDG: edge color), using the figure pattern stored in the pattern RAM. If the CP is inside the closed area, the paint operation is performed only inside the closed area. If the CP is outside, the paint operation is performed outside the closed area. Color code stored in color registers (CLO or CL1) are also considered to be an edge during PAINT execution. (See &lt; Complex Figure Painting &gt;.) When an unpaintable area is detected during this command, the coordinates are put in the Read FIFO and painting is continued. Therefore, a complex figure can be completely painted by re-issuing PAINT commands using the coordinate data put in the Read FIFO.</p> <p>&lt; <b>Definition of Edge Color</b> &gt;</p> <p>E = 0 : The edge color is defined by the data in the EDG register. (See figure next page)</p> <p>E = 1 : The edge color is defined to be all colors except for the color in the EDG register. (See figure next page)</p>			



E = 0 : "red" is set to the EDG register.  
PAINT is executed at E=0.

Figure C34-1 Paint Function (E=0)



E = 1 : "black" is set to the EDG register.  
PAINT is executed at E=1.

Figure C34-2 Paint Function (E=1)

< Paint Using a Pattern >

The PAINT command paints using a pattern stored in the pattern RAM. As the scan point in the pattern RAM moves corresponding to the movement of the drawing point, the figure is repeatedly drawn.

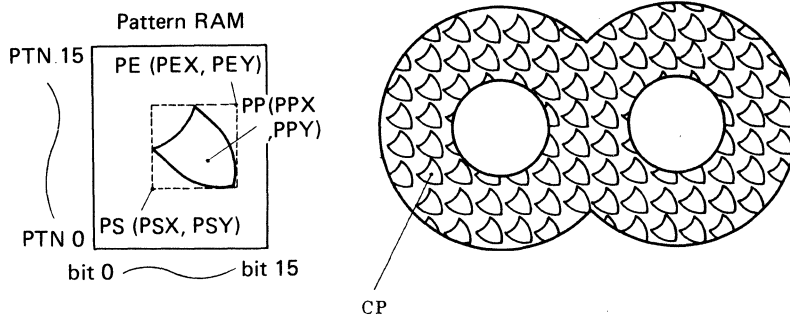


Figure C34-3 Paint Function Using Figure Pattern

## &lt; Paint Procedure &gt;

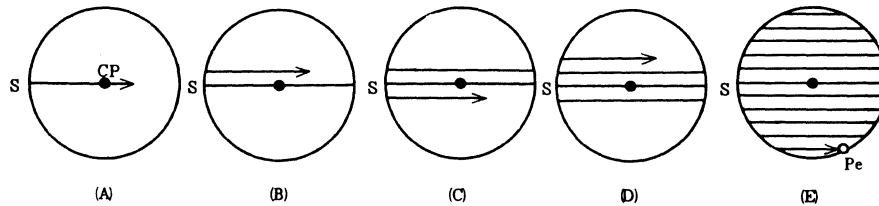


Figure C34-4 Paint Procedure

Painting is continuously performed parallel to the X axis (left to right), and in the Y direction, dot by dot. Fig. C34-2 shows an example of painting the encircled area. First, painting begins from points S on a line which is parallel to the X axis from CP. Next, painting is executed on the adjacent line which is above or below the first line. This drawing is repeated and painting proceeds. In this way, the whole encircled area is painted. The current pointer, CP, moves to the end point Pe at the finish.

## &lt; Complex Figure Painting &gt;

The PAINT command checks the outlined area for any un-painted areas during painting. If there are any during painting, the coordinates of the areas are pushed into the internal stack. Figure below shows a case of four coordinates being pushed into the stack.

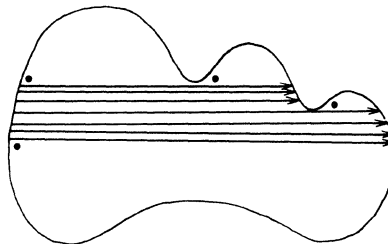
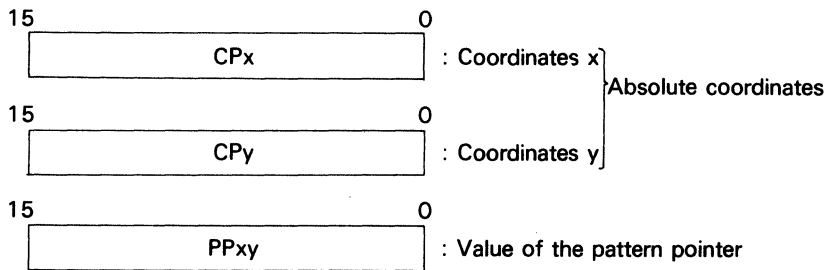


Figure C34-5 Paint Stack Function

## PAINT (Paint)

The ACRTC can store four such coordinates. If the points are within four, one PAINT command can completely paint a complex figure.

If the points are five or more all coordinates cannot be pushed into the stack. The un-stacked coordinates are put in the Read FIFO to be read out by the MPU. the MPU reads out the coordinates and issues another PAINT command to paint the un-painted areas using these coordinates after the initial PAINT command is finished. The coordinate for one point put in the Read FIFO consists of the following 3 words.



If the Read FIFO is full, the command execution remains halted until the MPU reads out the coordinates. When the Read FIFO has data before "PAINT" is instructed, only two or less coordinates can be pushed into the stack. Therefore, it is recommended that the read FIFO be empty before instructing A "PAINT" command.

The following two cases are the state of termination of the command.

- ① Data is not written in the Read FIFO  
(The outlined area is completely painted.)
- ② Data is written in the read FIFO  
(An un-paintable area exists.)

In the case of ②, any un-painted area should be painted by issuing the PAINT command again.



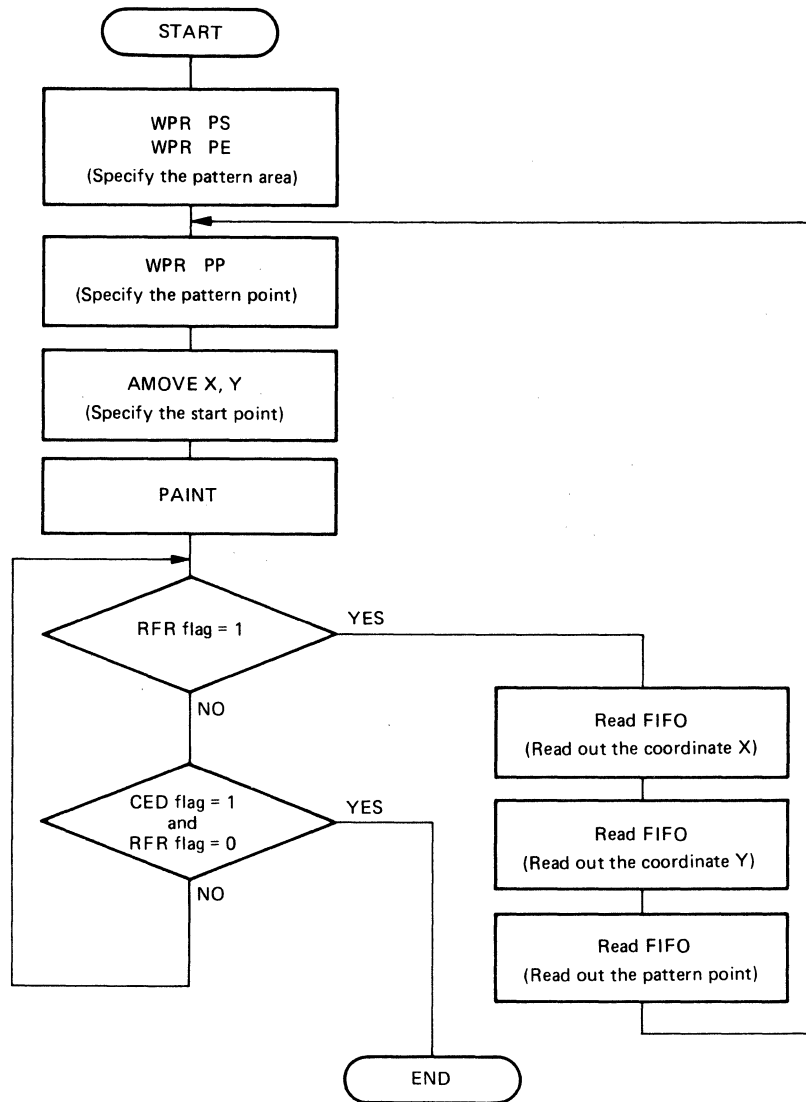


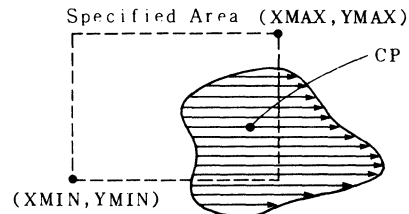
Figure C34-6 Paint Flow of Complex Figures Using PAINT Command

## &lt; PAINT Area Detection Mode &gt;

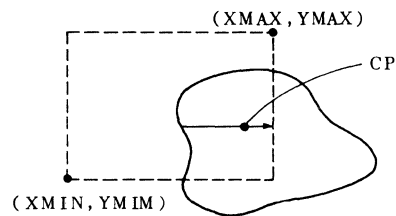
PAINT Area Detection modes have each of the following functions.

AREA	PAINT Command Execution
000	Not check the specified area.
001	AREA flag is set and the command execution is truncated, if CP moves outside the specified area during painting.
010	Paint only inside the specified area. AREA flag is not set.
011	Paint only inside the specified area. If CP meets the edge of the specified area, AREA flag is set.
100	Not check the specified area.
101	AREA flag is set and the command execution is truncated, if CP moves inside the specified area.
110	Paint only outside the specified area. AREA flag is not set.
111	Paint only outside the specified area. If CP meets the edge of the specified area, AREA flag is set.

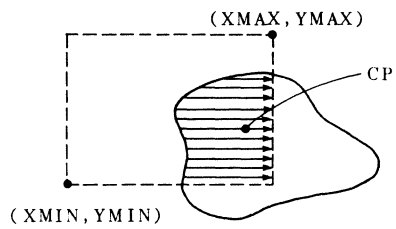
- (i) AREA = 0 0 0  
AREA = 1 0 0



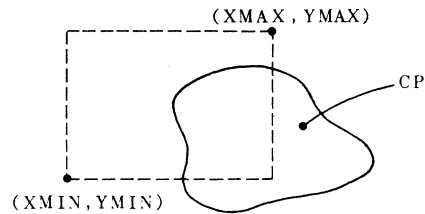
- (ii) AREA = 0 0 1  
(AREA flag is set.)



- (iii) AREA = 0 1 0  
(AREA flag is not changed.)  
AREA = 0 1 1  
(AREA flag is set.)



- (iv) AREA = 1 0 1  
(AREA flag is set.)



- (v) AREA = 1 1 0  
(AREA flag is not changed.)  
AREA = 1 1 1  
(AREA flag is set.)

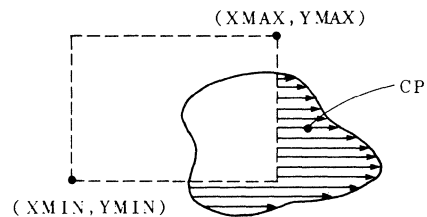


Figure C34-6A Paint Command Example with AREA Modes

< EXAMPLE > (In the case of E = "0")

If a circle of the same color as specified in the edge color register (EDG) is drawn on the split screen, the pattern shown in Fig. C34-7 fetched from the pattern RAM is used and the pattern pointer (PP) is in the position shown in Fig. C34-7. Then the PAINT command with bit-8 = "0", CP in the position shown in Fig. C34-8 is executed as shown in Fig. C34-8.

COMMAND CODE

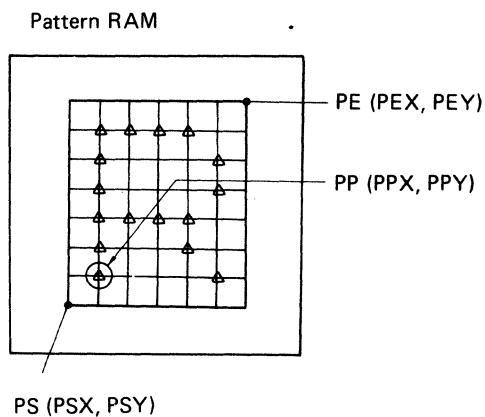
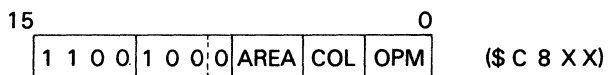


Figure C34-7 Setting of Pattern RAM

< EXECUTION EXAMPLE > (In the case of E = "0")

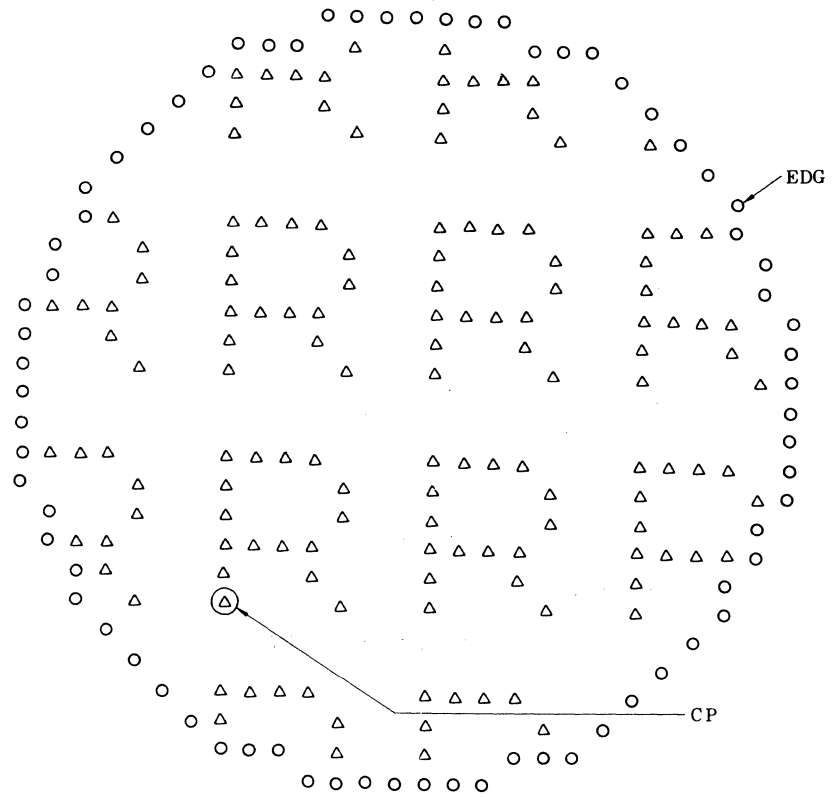


Figure C34-8 Example of PAINT Execution (E = "0")

< EXAMPLE > (In the case of  $E = "1"$ )

If a circle of the same color as specified in the edge color parameter register (EDG) is drawn on the split screen and the inside of the circle is also painted in the same color and the surround of the circle is not the same color as the edge, Fig. C34-10 (A), and the pattern shown in Fig. C34-9 is in the pattern RAM, the pattern pointer (PP) is in the position shown in Fig. C34-9. Then the PAINT command with bit 8 = "1", CP in the position shown in Fig. C34-10 (A) is executed as shown in Fig. C34-10 (B).

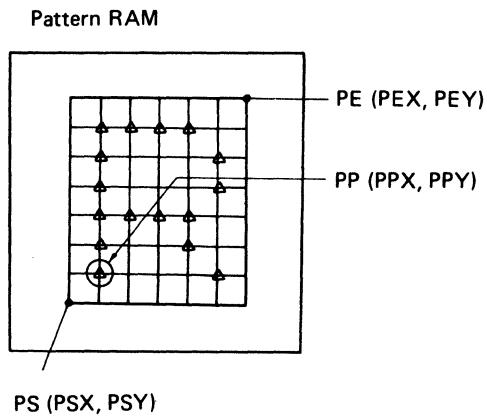
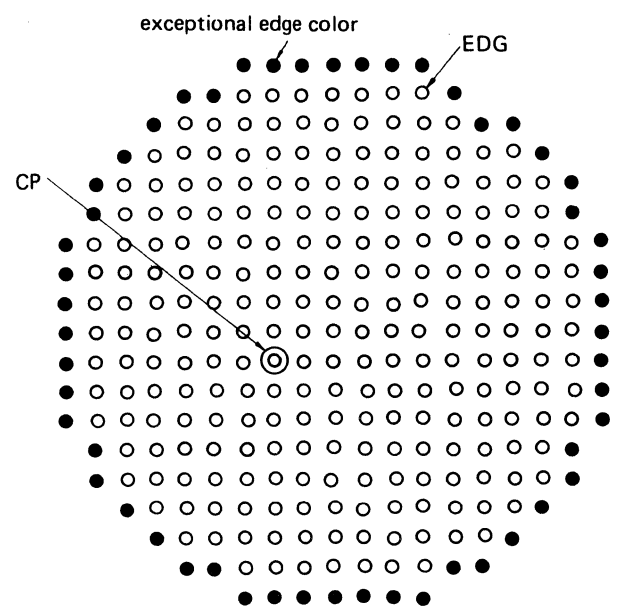
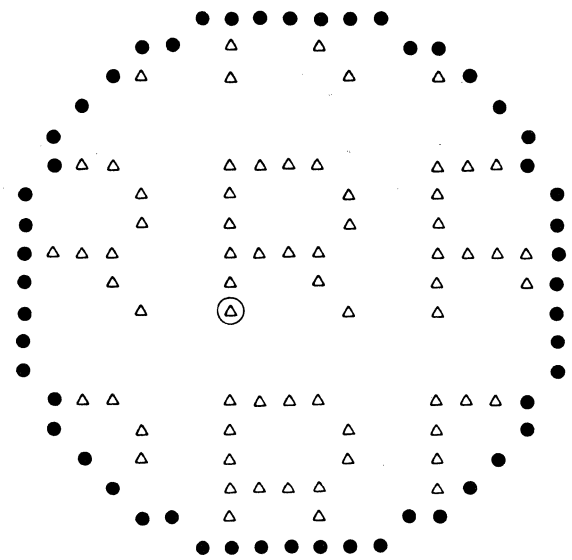


Figure C34-9 Setting of Pattern RAM



(A)



(B)

Figure C34-10 Example of PAINT Execution (E = "1")

		DOT												
[35] DOT (Dot Command)		PAGE	DOT-1											
<p>&lt; <b>FUNCTION</b> &gt;            DOT Command marks a dot on the coordinates where the CP points.</p> <p>&lt; <b>MNEMONIC</b> &gt;            DOT (AREA, COL, OPM)</p>		TYPE	Graphic Command											
<p>&lt; <b>FORMAT</b> &gt;</p> <p>COMMAND CODE</p> <p style="text-align: center;">15                      8 7    5 4    3 2    0</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="padding: 2px;">1</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;">0</td> <td style="padding: 2px;">0</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;">0</td> <td style="padding: 2px;">0</td> <td style="padding: 2px;">AREA</td> <td style="padding: 2px;">COL</td> <td style="padding: 2px;">OPM</td> </tr> </table> <p style="text-align: right;">hexadecimal notation (\$ C C X X)</p> <p>COMMAND PARAMETERS            - NON -</p>		1	1	0	0	1	1	0	0	AREA	COL	OPM	<p>WORD NUMBER  <math>W_n = 1</math></p> <p>EXECUTION CYCLES  <math>C_n = 8</math></p>	
1	1	0	0	1	1	0	0	AREA	COL	OPM				
<p>&lt; <b>DESCRIPTION</b> &gt;</p> <p>The Dot Command (DOT) marks a dot on the coordinate where the Current Pointer (CP) indicates. After dot drawing, the CP doesn't move. So, Pe, the dotting-finishing point, is the same point as the CP.</p> <div style="text-align: center;"> </div>														
<p><b>Figure C35-1 Function of DOT</b></p>														





		PTN												
[36] PTN (Pattern)	PAGE	PTN-1												
<p>&lt; FUNCTION &gt; The graphic pattern defined in the pattern RAM is drawn onto the rectangular area specified by the current pointer and by the pattern size.</p> <p>&lt; MNEMONIC &gt; PTN (SL, SD, AREA, COL, OPM) S</p>		TYPE	Graphic Command											
<p>&lt; FORMAT &gt;</p> <p>COMMAND CODE <span style="float: right;">hexadecimal notation</span></p> <p>15            12 11 10 8 7        5 4 3 2 0</p> <table border="1" style="margin-left: 40px;"> <tr> <td>1</td><td>0</td><td>1</td><td>SL</td><td>SD</td><td>AREA</td><td>COL</td><td>OPM</td> </tr> </table> <p style="margin-left: 100px;">(\$ D X X X)</p> <p>COMMAND PARAMETERS</p> <p>15                                    8 7                                    0</p> <table border="1" style="margin-left: 40px;"> <tr> <td style="width: 100px;">SZ</td> <td style="width: 100px;">SZy</td> <td style="width: 100px;">SZx</td> </tr> </table> <p style="margin-left: 100px;">SZx, SZy Setting: 0~255 Meaning: 1~256 in units of pixels</p>		1	0	1	SL	SD	AREA	COL	OPM	SZ	SZy	SZx	<p>WORD NUMBER Wn=2</p> <p>EXECUTION CYCLES Cn=(P·A+10)·B+20</p>	
1	0	1	SL	SD	AREA	COL	OPM							
SZ	SZy	SZx												
<p>&lt; DESCRIPTION &gt;</p> <p>As shown in Fig. C36-1, the Pattern command (PTN) is used to draw the graphic pattern defined in the pattern RAM onto the rectangular area specified by the current pointer (CP) and by the parameter (SZ: SZy, SZx). The pattern to be taken out of the pattern RAM is set by the pattern start point (PS) and pattern end point (PE).</p> <p>The point at which to start pattern RAM scan to obtain color information is set by the pattern pointer (PP). The color information is set on color registers "0" and "1" for execution of pattern drawing.</p> <p>Parameter SZ is divided into X component (SZx) and Y component (SZy), each component being set in units of pixels.</p> <p>The PTN command has the CP scan direction set up in units of 45° in the operation code, together with the choice of 45° slanted pattern drawing. After pattern drawing, the CP is moved to the Pe (see Table C36-1).</p>														

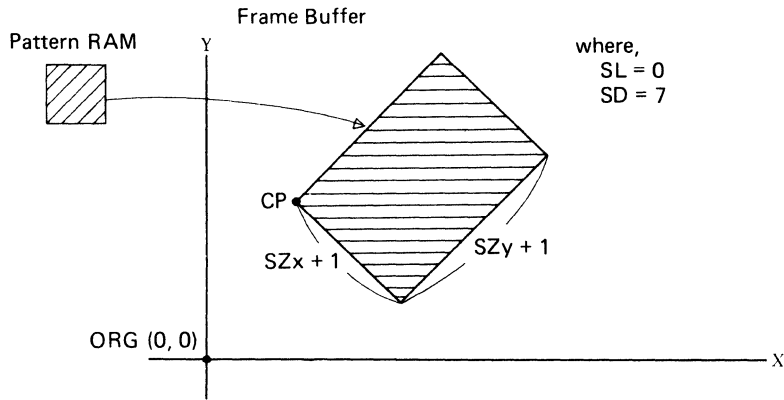


Figure C36-1 Function of PTN

Table C36-1 Directions of CP Scan

SL \ SD	0 0 0	0 0 1	0 1 0	0 1 1
0				
SL \ SD	1 0 0	1 0 1	1 1 0	1 1 1
0				
SL \ SD	0 0 0	0 0 1	0 1 0	0 1 1
1				
SL \ SD	1 0 0	1 0 1	1 1 0	1 1 1
1				

• : CP    ◦ : Pe

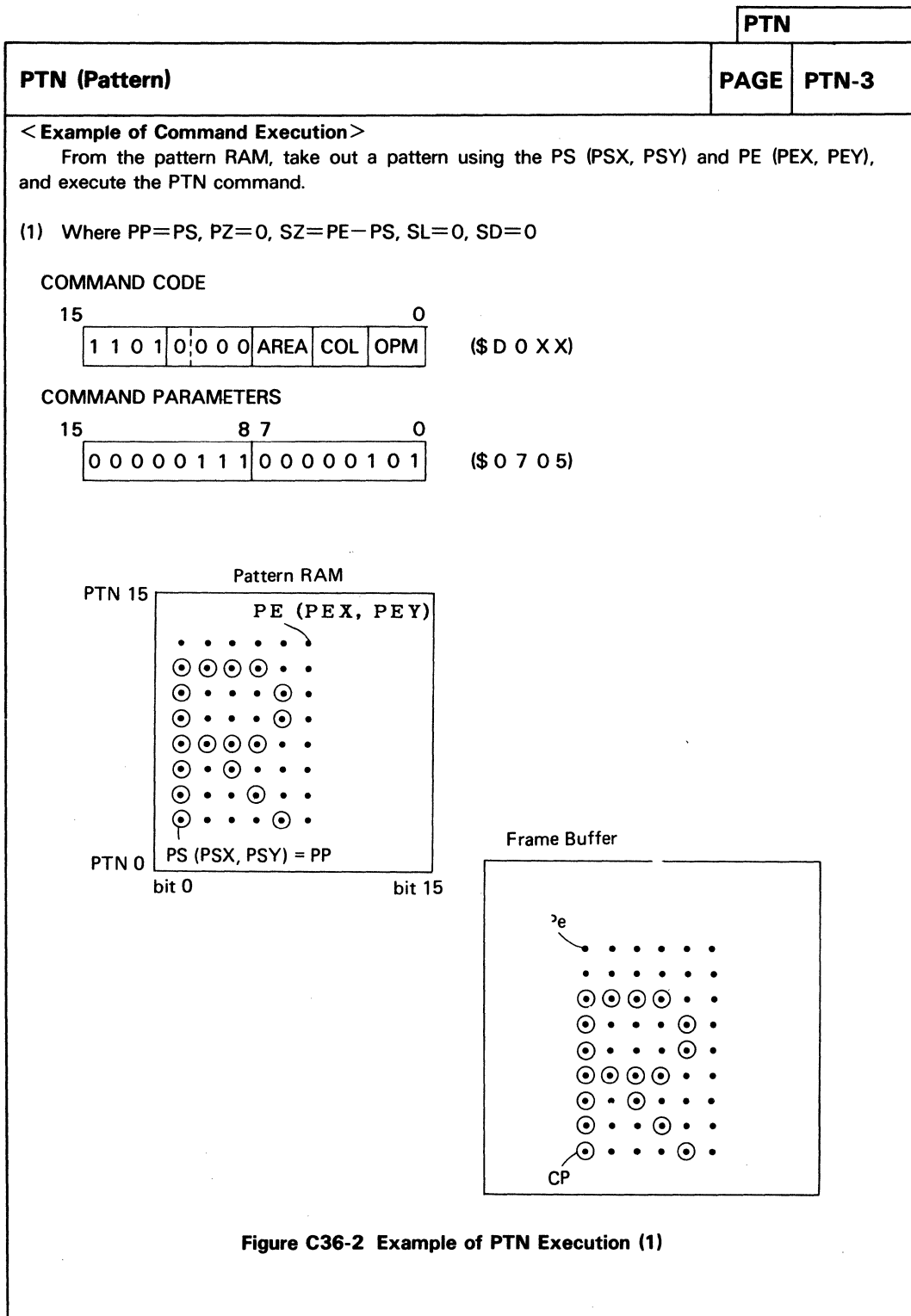
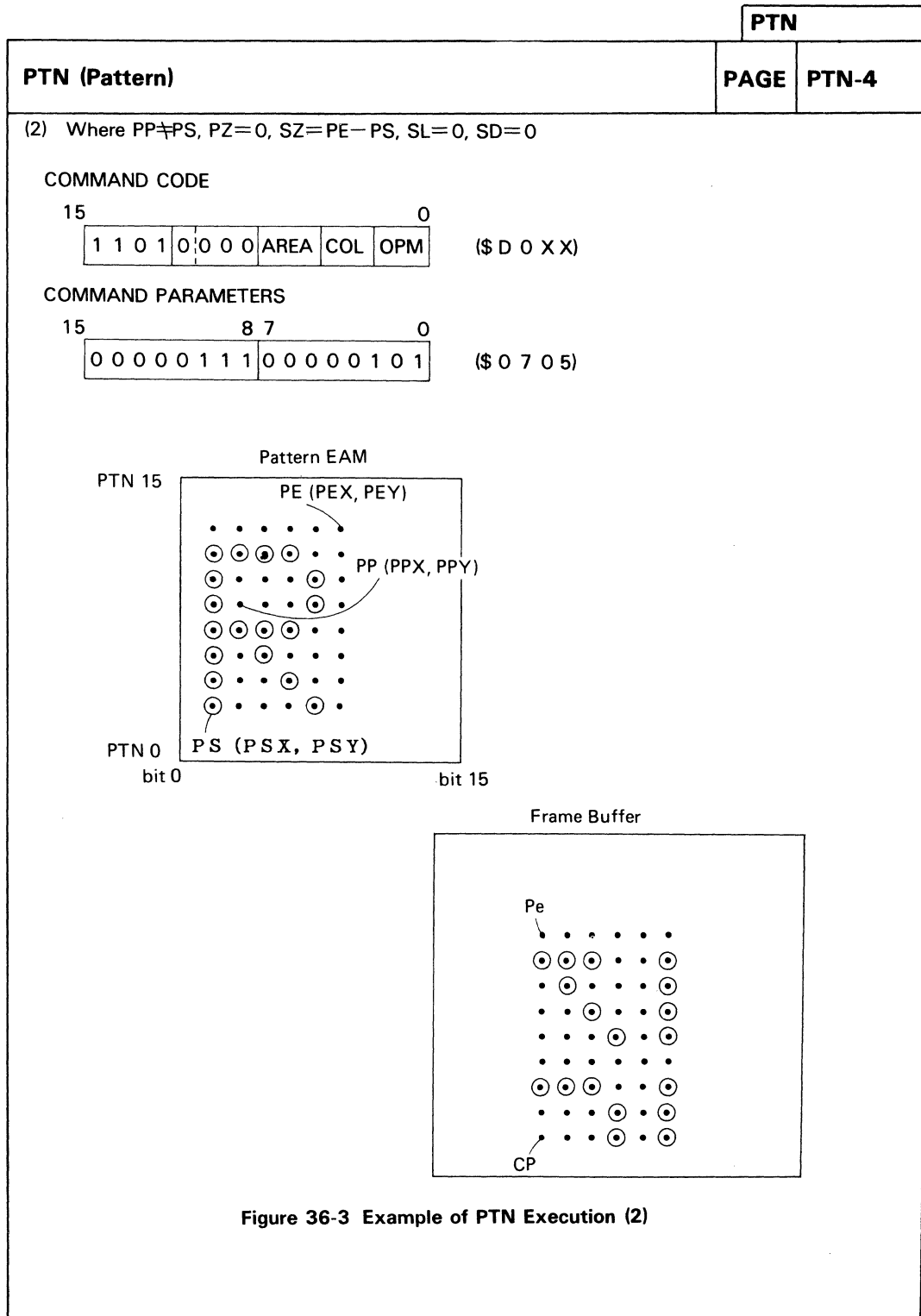


Figure C36-2 Example of PTN Execution (1)



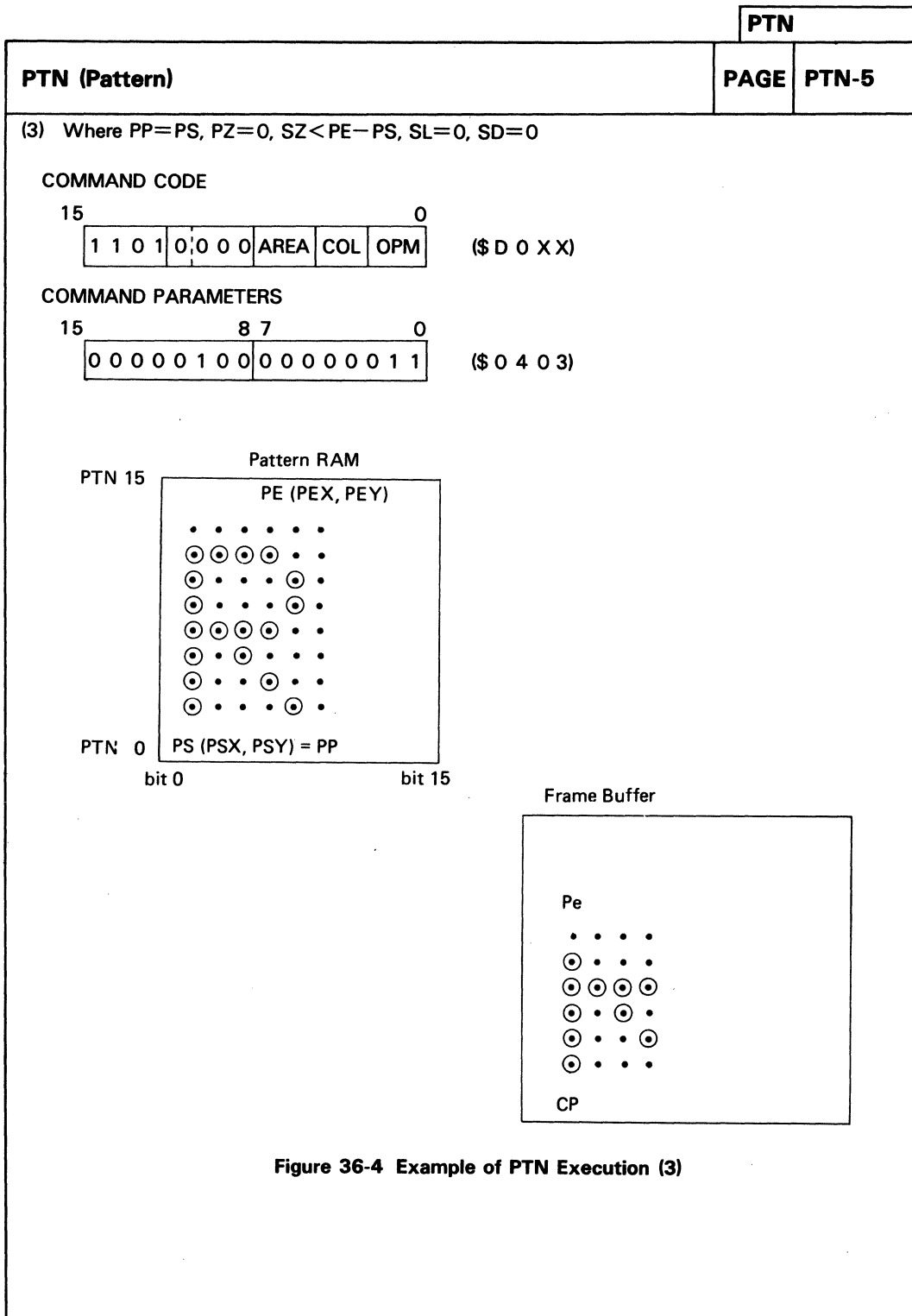


Figure 36-4 Example of PTN Execution (3)

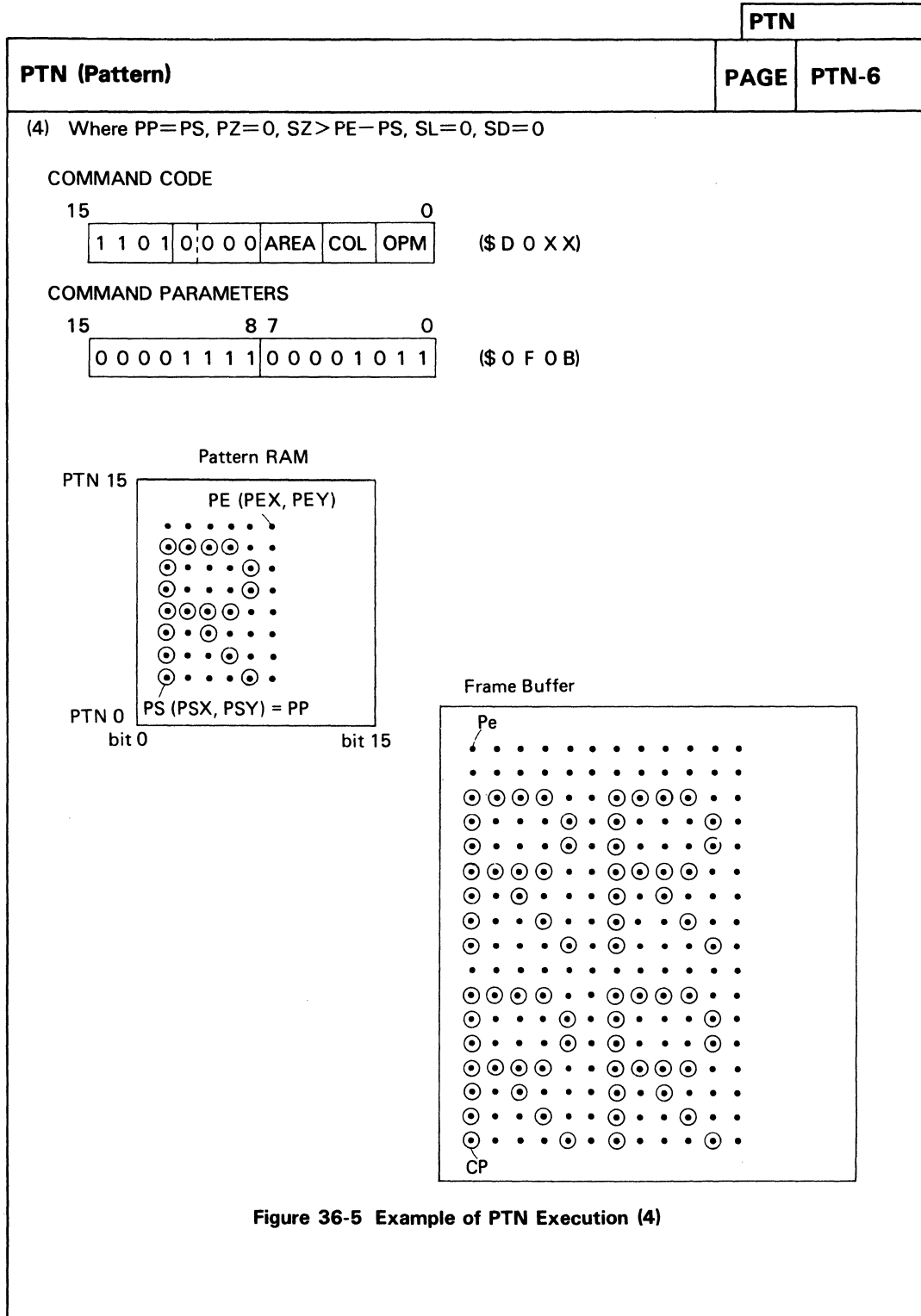
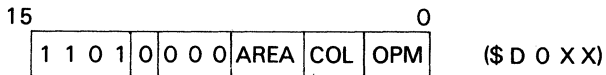


Figure 36-5 Example of PTN Execution (4)

PTN (Pattern)

(5) Where PP=PS, PZX=1, PZY=1, SZ>PE-PS, SL=0, SD=0

COMMAND CODE



COMMAND PARAMETERS

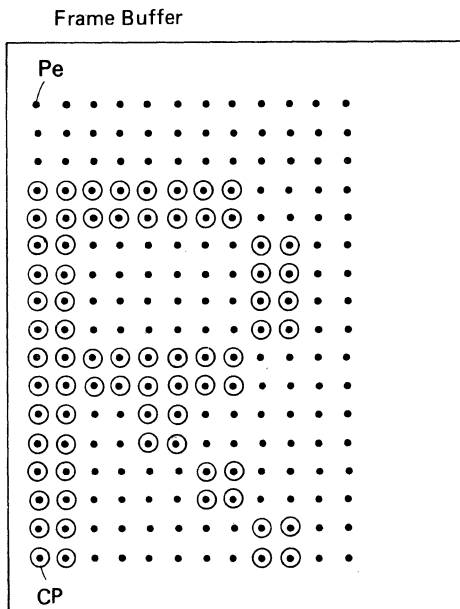
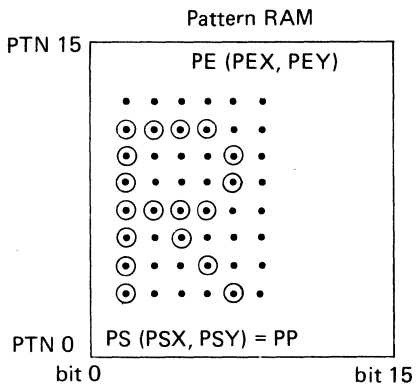
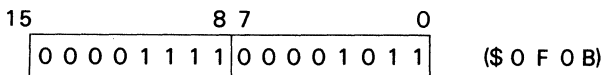
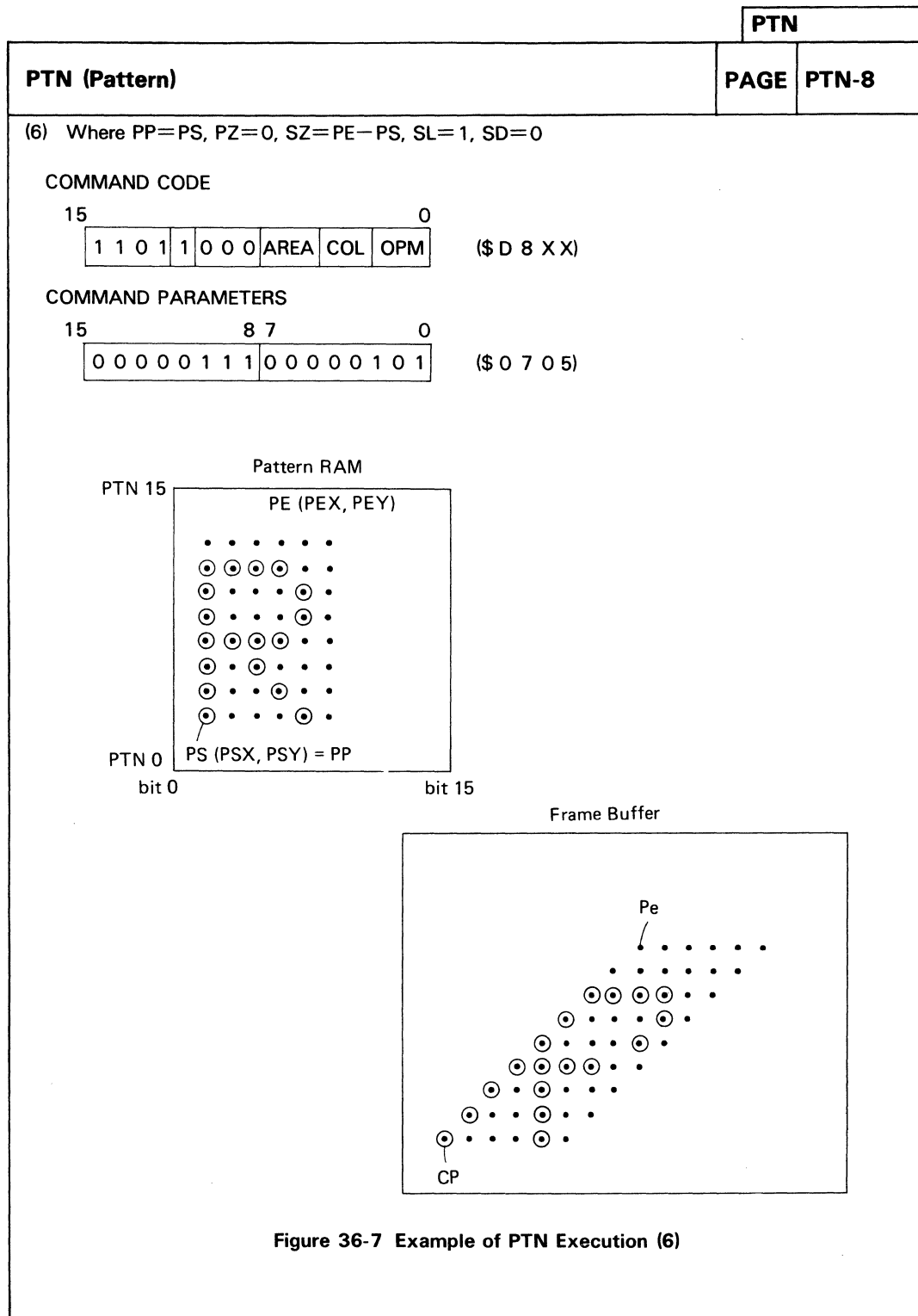


Figure 36-6 Example of PTN Execution (5)

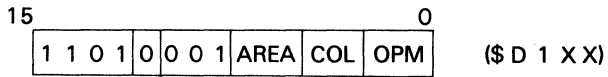




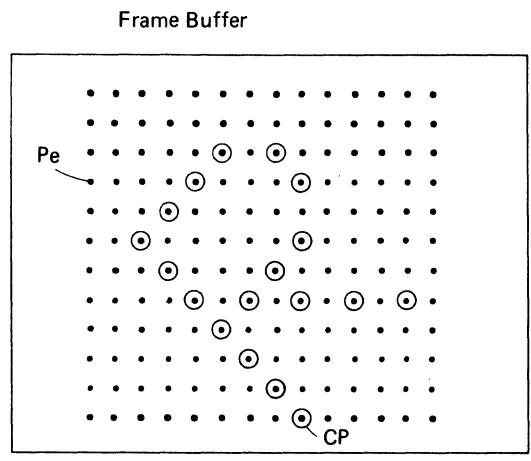
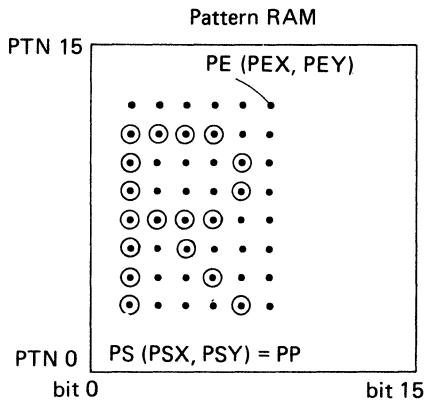
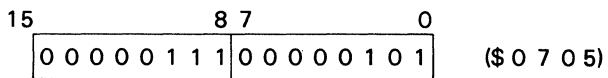
**PTN (Pattern)**

(7) Where  $PP=PS$ ,  $PZ=0$ ,  $SZ=PE-PS$ ,  $SL=0$ ,  $SD=1$

COMMAND CODE



COMMAND PARAMETERS



**Figure 36-8 Example of PTN Execution (7)**

		AGCPY																											
[37] AGCPY (Absolute Graphic Copy)		PAGE	AGCPY-1																										
<p>&lt; <b>FUNCTION</b> &gt;  AGCPY command copies a rectangular area specified by the absolute coordinates to the address specified by CP (Current Pointer)</p> <p>&lt; <b>MNEMONIC</b> &gt;  AGCPY (S, DSD, AREA, COL, OPM) Xs, Ys, DX, DY</p>		TYPE	Graphic Command																										
<p>&lt; <b>FORMAT</b> &gt;</p> <p>COMMAND CODE</p> <table style="margin-left: 40px;"> <tr> <td style="text-align: right;">15</td> <td style="text-align: center;">12 11 10</td> <td style="text-align: center;">8 7</td> <td style="text-align: center;">5 4</td> <td style="text-align: center;">3 2</td> <td style="text-align: center;">0</td> <td style="text-align: left;">hexadecimal notation</td> </tr> <tr> <td></td> <td style="border: 1px solid black; padding: 2px;">1 1 0</td> <td style="border: 1px solid black; padding: 2px;">S D S D</td> <td style="border: 1px solid black; padding: 2px;">AREA</td> <td style="border: 1px solid black; padding: 2px;">0 0</td> <td style="border: 1px solid black; padding: 2px;">OPM</td> <td style="text-align: left;">(\$ E X X X)</td> </tr> </table> <p>COMMAND PARAMETERS</p> <table style="margin-left: 40px;"> <tr> <td style="text-align: right;">15</td> <td style="border: 1px solid black; padding: 2px; width: 100px;">Xs</td> <td style="text-align: left;">0</td> </tr> <tr> <td style="text-align: right;">15</td> <td style="border: 1px solid black; padding: 2px; width: 100px;">Ys</td> <td style="text-align: left;">0</td> </tr> <tr> <td style="text-align: right;">15</td> <td style="border: 1px solid black; padding: 2px; width: 100px;">DX</td> <td style="text-align: left;">0</td> </tr> <tr> <td style="text-align: right;">15</td> <td style="border: 1px solid black; padding: 2px; width: 100px;">DY</td> <td style="text-align: left;">0</td> </tr> </table>		15	12 11 10	8 7	5 4	3 2	0	hexadecimal notation		1 1 0	S D S D	AREA	0 0	OPM	(\$ E X X X)	15	Xs	0	15	Ys	0	15	DX	0	15	DY	0	<p>WORD NUMBER  <math>W_n = 5</math></p> <p>EXECUTION CYCLES  <math>C_n = \{(P + 2)A + 10\}B + 70</math></p>	
15	12 11 10	8 7	5 4	3 2	0	hexadecimal notation																							
	1 1 0	S D S D	AREA	0 0	OPM	(\$ E X X X)																							
15	Xs	0																											
15	Ys	0																											
15	DX	0																											
15	DY	0																											
<p>&lt; <b>DESCRIPTION</b> &gt;</p> <p>The Absolute Graphic Copy Command (AGCPY) copies data from an rectangular area in the frame buffer (the source area) to another location in the frame buffer (the destination area) with the initial starting point CP. The size of the source rectangular area is parallel to the coordinate axis. Two diagonal corner points are Pss (Xs, Ys) at the absolute coordinate point from the origin and Pse (Xs+DX, Ys+DY) at the relative coordinate point from Pss.</p> <p>Pss (Xs, Ys) expressed by absolute X-Y coordinates from the origin are set in the command parameter in units of pixels.</p> <p>Pse (DX, DY) expressed by relative X-Y coordinates from Pss are set in the command parameter in units of pixels.</p>																													

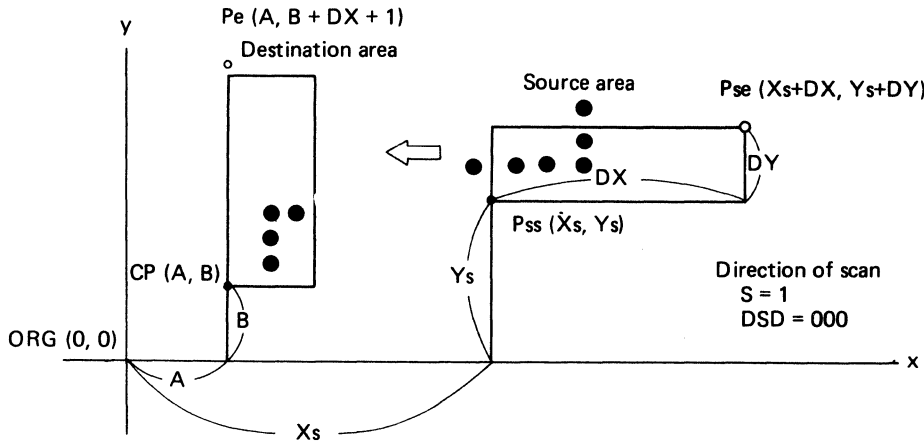


Figure C37-1 Function of AGCPY

< DIRECTION OF POINTER SCAN >

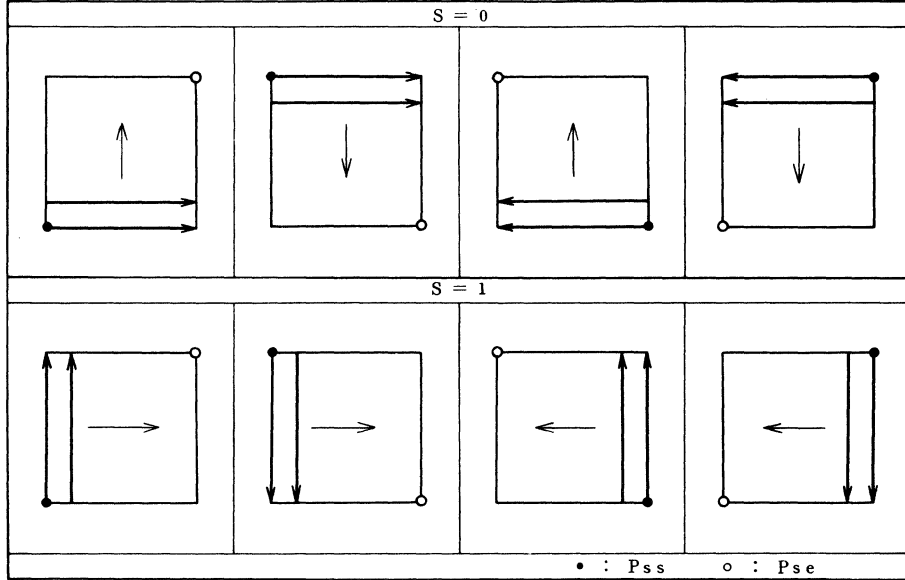
The direction of pointer scan is determined by S bit and DSD bit in the command code through the AGCPY command.

(a) S (Source Scan Direction)

COMMAND CODE



Table C37-1 Direction of Source Data Scan



The direction of scan on the frame buffer in the source area is determined with bit 11 in the command code and the position of P<sub>ss</sub> and P<sub>se</sub>, as shown in Table C37-1.

(b) DSD (Destination Scan Direction)

COMMAND CODE

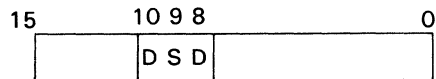


Table C37-2 Direction of Destination Data Scan

DSD=000	DSD=001	DSD=010	DSD=011
DSD=100	DSD=101	DSD=110	DSD=111

• : CP    ○ : Pe

As shown Table C37-2, the direction of scan on the frame buffer in the destination area is determined with bits 10 through 8 in the command code and position of CP and Pe.

After termination of the command, Pe, the end point of CP, is moved to the point shown in Table C37-2.

< EXAMPLE >

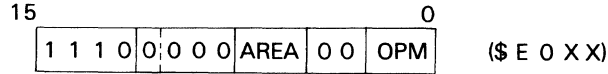
If the absolute coordinates of CP is (4, 2) on the split screen, Xs is set to 18, Ys is set to 2, DX is set to 13 and DY is set to 7 in the command parameter. Then, the drawing is copied by the AGCPY command (S = 1, DSD = 000), as shown in Fig. C37-2 (B).

**AGCPY (Absolute Graphic Copy)**

PAGE

AGCPY-5

COMMAND CODE



COMMAND PARAMETERS

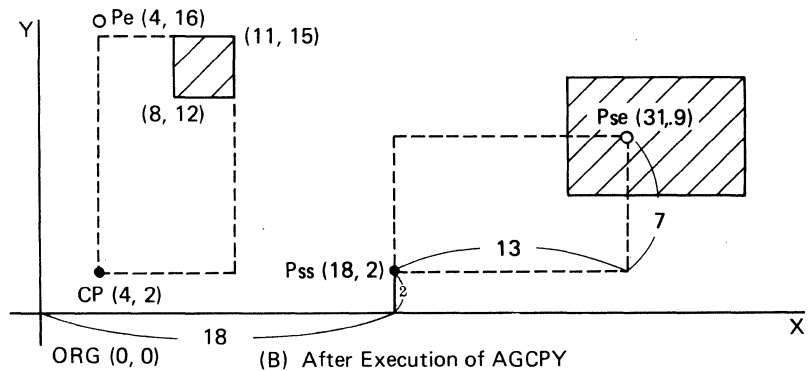
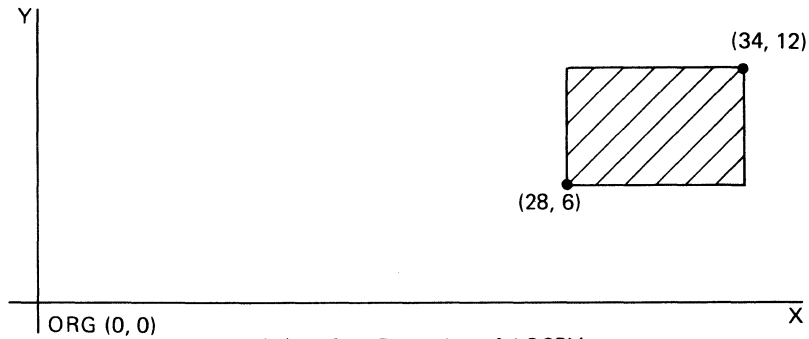
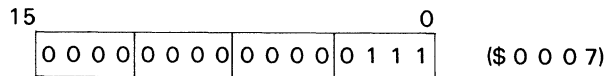
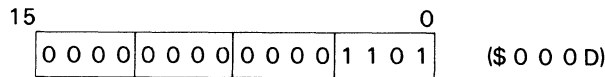
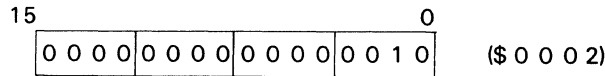
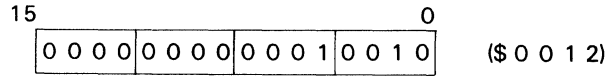


Figure C37-2 Example of AGCPY Execution

		<b>RGCPY</b>																																			
<b>[38] RGCPY (Relative Graphic Copy)</b>		<b>PAGE</b>	<b>RGCPY-1</b>																																		
<p><b>&lt; FUNCTION &gt;</b>            RGCPY command copy a rectangular area specified by the reative coordinates based on CP (Current Pointer) to an address specified by CP.</p> <p><b>&lt; MNEMONIC &gt;</b>            RGCPY (S, DSD, AREA, COL, OPM) dXs, dYs, DX, DY</p>		TYPE	Graphic Command																																		
<p><b>&lt; FORMAT &gt;</b></p> <p>COMMAND CODE <span style="float: right;">hexadecimal notation</span></p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: right;">15</td> <td style="text-align: center;">12 11 10 8 7</td> <td style="text-align: center;">5 4 3 2</td> <td style="text-align: right;">0</td> <td></td> </tr> <tr> <td></td> <td style="text-align: center;">1 1 1 1</td> <td style="text-align: center;">S D S D</td> <td style="text-align: center;">AREA 0 0</td> <td style="text-align: center;">OPM</td> </tr> </table> <p style="text-align: right;">(\$ F X X X)</p> <p>COMMAND PARAMETERS</p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: right;">15</td> <td style="width: 80%;"></td> <td style="text-align: right;">0</td> </tr> <tr> <td></td> <td style="text-align: center;">dXs</td> <td></td> </tr> <tr> <td style="text-align: right;">15</td> <td style="width: 80%;"></td> <td style="text-align: right;">0</td> </tr> <tr> <td></td> <td style="text-align: center;">dYs</td> <td></td> </tr> <tr> <td style="text-align: right;">15</td> <td style="width: 80%;"></td> <td style="text-align: right;">0</td> </tr> <tr> <td></td> <td style="text-align: center;">DX</td> <td></td> </tr> <tr> <td style="text-align: right;">15</td> <td style="width: 80%;"></td> <td style="text-align: right;">0</td> </tr> <tr> <td></td> <td style="text-align: center;">DY</td> <td></td> </tr> </table>		15	12 11 10 8 7	5 4 3 2	0			1 1 1 1	S D S D	AREA 0 0	OPM	15		0		dXs		15		0		dYs		15		0		DX		15		0		DY		<p>WORD NUMBER Wn=5</p> <p>EXECUTION CYCLES Cn= (P+2)A+10 B+70</p>	
15	12 11 10 8 7	5 4 3 2	0																																		
	1 1 1 1	S D S D	AREA 0 0	OPM																																	
15		0																																			
	dXs																																				
15		0																																			
	dYs																																				
15		0																																			
	DX																																				
15		0																																			
	DY																																				
<p><b>&lt; DESCRIPTION &gt;</b></p> <p>The Relative Graphic Copy Command (RGCPY) copies data from an rectangular area in the frame buffer (the source area) to another location in the frame buffer (the destination area) with the initial starting point CP. The size of the source rectangular area is parallel to the coordinate axis. Two diagonal corner points are Pss (A+dXs, B+dYs) at the absolute coordinate point from CP and Pse (A+dXs+DX, B+dYs+DY) at the relative coordinate point from Pss.</p> <p>Pss (dXs, dYs) expressed by the relative X-Y coordinates from CP are set in the command parameter in units of pixels.</p> <p>Pse (DX, DY) expressed by the relative X-Y coordinates from Pss are set in the command parameter in units of pixels.</p>																																					



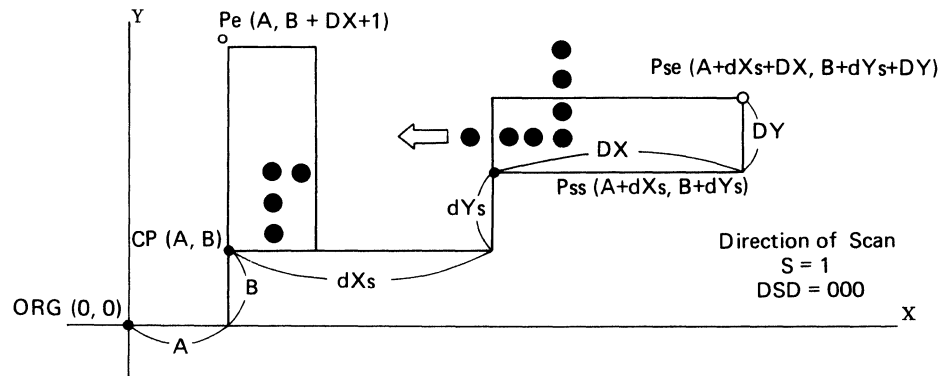


Figure C38-1 Function of RGCPY

## &lt; DIRECTION OF POINTER SCAN &gt;

S-bit and DSD bit in the RGCPY command have the same function as those in the AGCPY command. Refer to the description about the AGCPY command for details.

## &lt; EXECUTION EXAMPLE &gt;

If the absolute coordinate of CP is (4, 2) on the split screen, dXs is set to 18, dYs to 2, DX to 12 and DY to 6 in the command parameter. Then, the drawing is executed by the RGCPY command (S = 1, DSD = 000), as shown in Fig. C38-2 (B).



○ Use of Arcs and Ellipse Arcs Commands

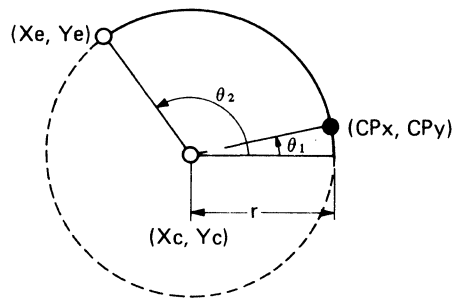
How to Calculate Parameters of Arc Commands

AARC Xc, Yc, Xe, Ye;  
 RARC dXc, dYc, dXe, dYe;

(Command Issuing Procedure)

CP is moved to the start point  
 (CPx, CPy) by MOVE, then  
 ARC is issued.

[Example 1] Given center coordinates  
 (Xc, Yc), radius r, drawing  
 start angle  $\theta_1$  and drawing  
 end angle  $\theta_2$ , calculate as  
 follows (counterclockwise  
 rotation):



(Parameter calculation: ① absolute addressing)

- Calculate the start point (CPx, CPy):

$$\begin{aligned} \text{CPx} &= \text{Xc} + [r \cos \theta_1 \downarrow] \\ \text{CPy} &= \text{Yc} + [r \sin \theta_1 \downarrow] \end{aligned}$$

- Calculate the end point (Xe, Ye):

$$\begin{aligned} \text{Xe} &= \text{Xc} + [R \cos \theta_2 \downarrow] \\ \text{Ye} &= \text{Yc} + [R \sin \theta_2 \downarrow] \quad (\text{where, } R = \sqrt{(\text{CPx} - \text{Xc})^2 + (\text{CPy} - \text{Yc})^2} \doteq r) \end{aligned}$$

(Parameter calculation: ② relative addressing)

- Calculate the start point (CPx, CPy):

$$\begin{aligned} \text{CPx} &= \text{Xc} + [r \cos \theta_1 \downarrow] \\ \text{CPy} &= \text{Yc} + [r \sin \theta_1 \downarrow] \quad \text{Same as in absolute addressing} \end{aligned}$$

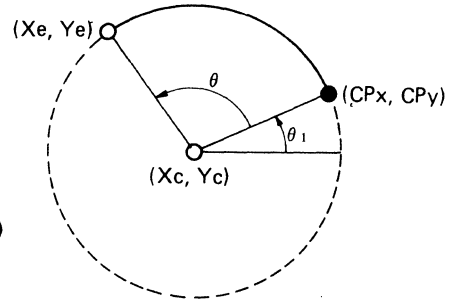
- Calculate the center coordinates (dXc, dYc):

$$\begin{aligned} \text{dXc} &= - [r \cos \theta_1 \downarrow] \\ \text{dYc} &= - [r \sin \theta_1 \downarrow] \end{aligned}$$

- Calculate the end point (dXe, dYe):

$$\begin{aligned} \text{dXe} &= \text{dXc} + [R \cos \theta_2 \downarrow] \\ \text{dYe} &= \text{dYc} + [R \sin \theta_2 \downarrow] \quad \text{where, } (R = \sqrt{(\text{CPx} - \text{Xc})^2 + (\text{CPy} - \text{Yc})^2} \doteq r) \end{aligned}$$

[Example 2] Given center coordinates  $(X_c, Y_c)$ , start point  $(CP_x, CP_y)$  and drawing angle  $\theta$ , calculate as follows (counterclockwise rotation):



(Parameter calculation: ① absolute addressing)

- Calculate the end point  $(X_e, Y_e)$ :

$$X_e = X_c + [R \cos (\theta + \theta_1) \downarrow]$$

$$Y_e = Y_c + [R \sin (\theta + \theta_1) \downarrow]$$

$$\left( \begin{array}{l} \text{where, } R = \sqrt{(CP_x - X_c)^2 + (CP_y - Y_c)^2} \\ \theta_1 = \tan^{-1} \left( \frac{CP_y - Y_c}{CP_x - X_c} \right) \end{array} \right)$$

(Parameter calculation: ② relative addressing)

- Calculate the center coordinates  $(dX_c, dY_c)$ :

$$dX_c = X_c - CP_x$$

$$dY_c = Y_c - CP_y$$

- Calculate the end point  $(dX_e, dY_e)$ :

$$dX_e = dX_c + [R \cos (\theta + \theta_1) \downarrow]$$

$$dY_e = dY_c + [R \sin (\theta + \theta_1) \downarrow]$$

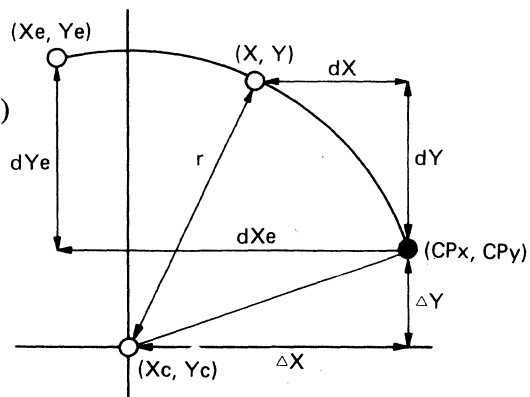
$$\left( \begin{array}{l} \text{where, } R = \sqrt{(CP_x - X_c)^2 + (CP_y - Y_c)^2} \\ \theta_1 = \tan^{-1} \left( \frac{CP_y - Y_c}{CP_x - X_c} \right) \end{array} \right)$$

[Example 3] Calculate parameters for an Arc that passes 3 points,  $(CP_x, CP_y)$ ,  $(X, Y)$  and  $(X_e, Y_e)$ .

(Parameter calculation: Relative addressing)

$$\left\{ \begin{array}{l} dX = X - CP_x \\ dY = Y - CP_y \end{array} \right.$$

$$\left\{ \begin{array}{l} dX_e = X_e - CP_x \\ dY_e = Y_e - CP_y \end{array} \right.$$



- Calculate the center coordinates (dXc, dYc):

$$\begin{cases} \Delta X^2 + \Delta Y^2 = r^2 \\ (\Delta X + dX)^2 + (\Delta Y + dY)^2 = r^2 \\ (\Delta X + dXe)^2 + (\Delta Y + dYe)^2 = r^2 \end{cases}$$

where,

$$\begin{cases} dXc = [-\Delta X] \\ \quad = \left[ \frac{1}{2} \cdot \frac{(dX^2 + dY^2) \cdot dYe - (dXe^2 + dYe^2) \cdot dY}{dX \cdot dYe - dXe \cdot dY} \right] \\ \\ dYc = [-\Delta Y] \\ \quad = \left[ \frac{1}{2} \cdot \frac{(dXe^2 + dYe^2) \cdot dX - (dX^2 + dY^2) \cdot dXe}{dX \cdot dYe - dXe \cdot dY} \right] \end{cases}$$

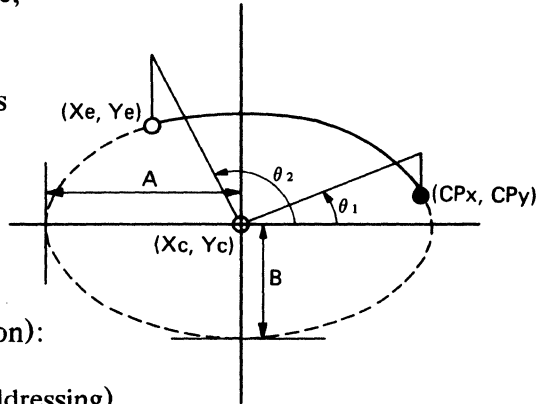
○ Calculating Parameters of Ellipse Arc Commands

AEARC a, b, Xc, Yc, Xe, Ye;

REARC a, b, dXc, dYc, dXe, dYe;

(Command Issuing Procedure)

[Example 1] Given center coordinates (Xc, Yc), X direction axial length A, Y direction axial length B, drawing start angle  $\theta_1$  and drawing end angle  $\theta_2$ , calculate as follows (counterclockwise rotation):



(Parameter calculation: ① absolute addressing)

- Calculate the axial length square ratio (a/b):

The ratio should be an integral ratio satisfying  $a/b = A^2/B^2$ .

- Calculate the start point (CPx, CPy):

$$\begin{cases} \text{CPx} = \text{Xc} + [A \cos \theta_1 \downarrow] \\ \text{CPy} = \text{Yc} + [B \sin \theta_1 \downarrow] \end{cases}$$

- Calculate the end point (Xe, Ye):

$$\begin{cases} \text{Xe} = \text{Xc} + [\sqrt{a} R' \cos \theta_2 \downarrow] \\ \text{Ye} = \text{Yc} + [\sqrt{b} R' \sin \theta_2 \downarrow] \end{cases}$$

$$\text{where } R' = \sqrt{\frac{(\text{CPx} - \text{Xc})^2}{a} + \frac{(\text{CPy} - \text{Yc})^2}{b}} \doteq \frac{A}{\sqrt{a}} \text{ or } \frac{B}{\sqrt{b}}$$

(Parameter calculation: ② relative addressing)

- Calculate the start point (CPx, CPy):

$$\begin{cases} \text{CPx} = \text{Xc} + [A \cos \theta_1 \downarrow] \\ \text{CPy} = \text{Yc} + [B \sin \theta_1 \downarrow] \end{cases} \quad \leftarrow \text{Same as in absolute addressing}$$

- Calculate the center coordinates (dXc, dYc):

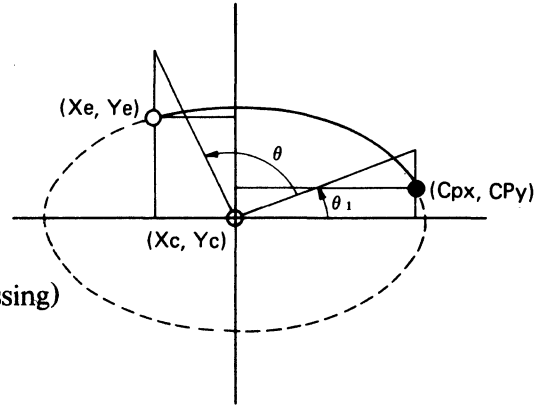
$$\begin{cases} \text{dXc} = - [A \cos \theta_1 \downarrow] \\ \text{dYc} = - [B \sin \theta_1 \downarrow] \end{cases}$$

- Calculate the end point (dXe, dYe):

$$\begin{cases} \text{dXe} = \text{dXc} + [\sqrt{a} R' \cos \theta_2 \downarrow] \\ \text{dYe} = \text{dYc} + [\sqrt{b} R' \sin \theta_2 \downarrow] \end{cases}$$

$$\text{where } R' = \sqrt{\frac{(\text{CPx} - \text{Xc})^2}{a} + \frac{(\text{CPy} - \text{Yc})^2}{b}} \doteq \frac{A}{\sqrt{a}} \text{ or } \frac{B}{\sqrt{b}}$$

[Example 2] Given center coordinates  $(X_c, Y_c)$ , axial length square ratio  $a/b$ , drawing start point  $(CP_x, CP_y)$  and drawing angle  $\theta$ , calculate as follows (counterclockwise rotation):



(parameter calculation: ① absolute addressing)

• Calculate the end point  $(X_e, Y_e)$ :

$$\begin{cases} X_e = X_c + [\sqrt{a} R' \cos(\theta + \theta_1)] \\ Y_e = Y_c + [\sqrt{b} R' \sin(\theta + \theta_1)] \end{cases}$$

$$\text{where } R' = \sqrt{\frac{(CP_x - X_c)^2}{a} + \frac{(CP_y - Y_c)^2}{b}}$$

$$\theta_1 = \tan^{-1} \left( \sqrt{\frac{a}{b}} \cdot \frac{CP_y - Y_c}{CP_x - X_c} \right)$$

(Parameter calculation: ② relative addressing)

• Calculate the center coordinates  $(dX_c, dY_c)$ :

$$\begin{cases} dX_c = X_c - CP_x \\ dY_c = Y_c - CP_y \end{cases}$$

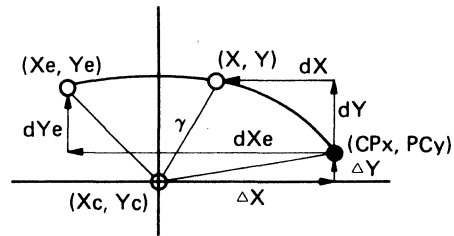
• Calculate the end point  $(dX_e, dY_e)$ :

$$\begin{cases} dX_e = dX_c + [\sqrt{a} R' \cos(\theta + \theta_1)] \\ dY_e = dY_c + [\sqrt{b} R' \sin(\theta + \theta_1)] \end{cases}$$

$$\text{where } R' = \sqrt{\frac{(CP_x - X_c)^2}{a} + \frac{(CP_y - Y_c)^2}{b}}$$

$$\theta_1 = \tan^{-1} \left( \sqrt{\frac{a}{b}} \cdot \frac{CP_y - Y_c}{CP_x - X_c} \right)$$

[Example 3] Calculate parameters for an ellipse arc that passes 3 points,  $(CP_x, CP_y)$ ,  $(X, Y)$  and  $(X_e, Y_e)$  (axial length square ratio:  $a/b$ ).



(Parameter calculation: Relative addressing)

$$\begin{cases} dX = X - CP_x \\ dY = Y - CP_y \end{cases} \quad \begin{cases} dX_e = X_e - CP_x \\ dY_e = Y_e - CP_y \end{cases}$$

- Calculate the center coordinates (dXc, dYc):

$$\begin{cases} \frac{\Delta X^2}{a} + \frac{\Delta Y^2}{b} = r^2 \\ \frac{(\Delta X + dX)^2}{a} + \frac{(\Delta Y + dY)^2}{b} = r^2 \\ \frac{(\Delta X + dXe)^2}{a} + \frac{(\Delta Y + dYe)^2}{b} = r^2 \end{cases}$$

we get

$$\begin{aligned} dXc &= [-\Delta X \uparrow] \\ &= \left[ \frac{1}{2b} \cdot \frac{(b \cdot dX^2 + a \cdot dY^2) \cdot dYe - (b \cdot dXe^2 + a \cdot dYe^2) \cdot dY}{dX \cdot dYe - dXe \cdot dY} \uparrow \right] \end{aligned}$$

$$\begin{aligned} dYc &= [-\Delta Y \uparrow] \\ &= \left[ \frac{1}{2a} \cdot \frac{(b \cdot dXe^2 + a \cdot dYe^2) \cdot dX - (b \cdot dX^2 + a \cdot dY^2) \cdot dXe}{dX \cdot dYe - dXe \cdot dY} \uparrow \right] \end{aligned}$$

Note:

[  $\uparrow$  ]: With sign unchanged, rounding the absolute value to the integer.

[  $\uparrow$  ]: With sign unchanged, round up the absolute value to the integer.

[  $\downarrow$  ]: With sign unchanged, truncate the absolute value to the integer.



# COPIES OF DATA SHEETS

FORCE COMPUTERS Inc./GmbH  
All Rights Reserved

This document shall not be duplicated, nor its contents used  
for any purpose, unless express permission has been granted.

Copyright by FORCE Computers®



TABLE OF CONTENTS

	<b>Page</b>
1. 63484 ACRTC.....	
2. 68153 BIM.....	
3. 8151 GCP.....	



# HD 63484

Advanced CRT Controller (ACRTC)



The Advanced CRT Controller (ACRTC) is a CMOS VLSI microcomputer peripheral device capable of controlling raster scan type CRTs to display both graphics and characters. The ACRTC is also a new generation CRT controller that is based on a bit-mapped technology and has more display control functions than those of an HD6845S (CRTC).

The ACRTC prepares the mechanisms to use at one of three modes; character only, graphic only and multiplexed character/graphic modes. Therefore, the ACRTC can be applied to many applications, from character only display devices to large full-graphic systems, as the key devices.

The ACRTC can reduce a CPU software overhead and enhance system throughput.

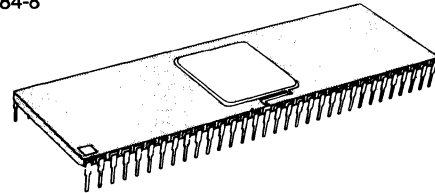
## ■ FEATURES

- High speed graphic drawings
  - Drawing rate : Maximum 500 ns/pixel (Color drawing)
  - Drawn graphics : Dot, Line, Rectangle, Poly-line, Poly-gon, Circle, Ellipse, Paint, Copy, etc.
  - Drawn colors : 16-bit/word  
1-, 2-, 4-, 8-, 16-bit/pixel (5 types)  
monochrome to max. 64k colors.
- Large frame memory space
  - Maximum 2M bytes graphic memory  
128k bytes character memory  
separated from the MPU memory
  - Available to maximum 4096 x 4096 high-resolution CRT (1 bit/pixel mode)
- Various CRT display controls
  - Split screens (3 displays and 1 window)
  - Zooming up (1 to 16 times)
  - Scroll (Vertical and horizontal)
- External synchronization
  - Synchronization between ACRTCs or between the ACRTC and external device (ex. TV system or other controller)
- DMA interface
- Two programmable cursors
- Three scan modes
  - Non-interlace, Interlace Sync. and Interlace Sync. & Video modes
- Interrupt request to MPU
- 256 characters/line, 32 rasters/line, 4096 rasters/screen
- Maximum clock frequency 8 MHz
- CMOS, +5V single power supply

## ■ TYPE OF PRODUCTS

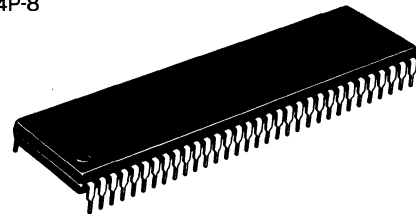
ACRTC	Clock Frequency (2CLK)
HD63484-4	4 MHz
HD63484-6	6 MHz
HD63484-8	8 MHz

HD63484-4, HD63484-6,  
HD63484-8



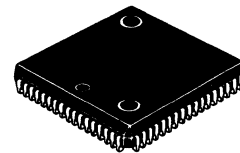
(DC-64)

HD63484P-4, HD63484P-6,  
HD63484P-8



(DP-64)

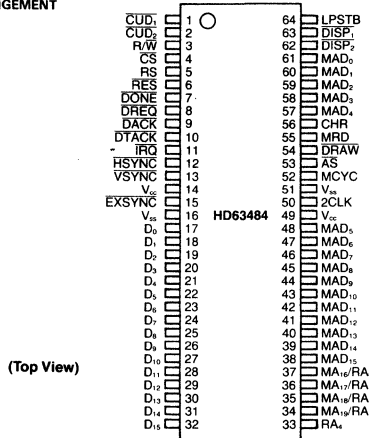
HD63484CP-4, HD63484CP-6,  
HD63484CP-8



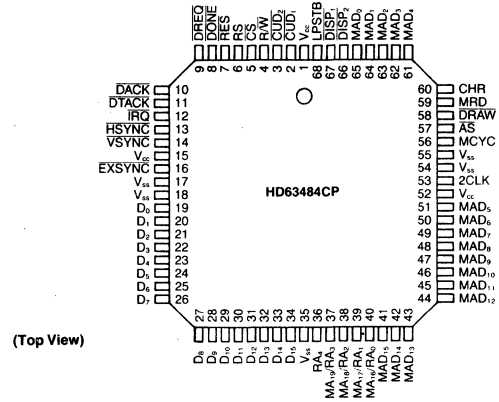
(CP-68)

■ PIN ARRANGEMENT

● HD63484, HD63484P PIN ARRANGEMENT



● HD63484CP PIN ARRANGEMENT



■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Rating	Unit
Supply Voltage	$V_{CC}^*$	-0.3 ~ +7.0	V
Input Voltage	$V_{in}^*$	-0.3 ~ $V_{CC} + 0.3$	V
Allowable Output Current	$ I_O ^{**}$	5	mA
Total Allowable Output Current	$ \sum I_O ^{***}$	120	mA
Operating Temperature	$T_{opr}$	0 ~ +70	°C
Storage Temperature	$T_{stg}$	-55 ~ +150	°C

- \* This value is in reference to  $V_{SS} = 0V$ .
  - \*\* The allowable output current is the maximum current that may be drawn from, or flow out to, one output terminal or one input/output common terminal.
  - \*\*\* The total allowable output current is the total sum of currents that may be drawn from, or flow out to, output terminals or input/output common terminals.
- (Note) Using an LSI beyond its maximum ratings may result in its permanent destruction. LSI's should usually be used under recommended operating conditions. Exceeding any of these conditions may adversely affect its reliability.

■ RECOMMENDED OPERATING CONDITIONS

Item	Symbol	min	typ	max	Unit
Supply Voltage	$V_{CC}^*$	4.75	5.0	5.25	V
Input "Low" Level Voltage	$V_{IL}^*$	0	—	0.7	V
Input "High" Level Voltage	$V_{IH}^*$	2.2	—	$V_{CC}$	V
Operating Temperature	$T_{opr}$	0	25	70	°C

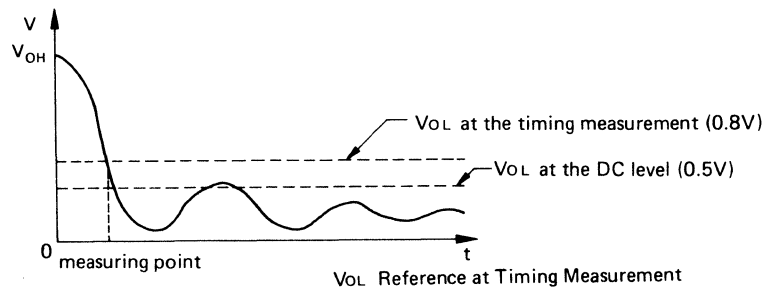
\* This value is in reference to  $V_{SS} = 0V$ .

■ Timing Measurement

The timing measurement point for the output "low" level is defined at 0.8V throughout this specification.

The output "low" level at stable condition (DC characteristics) is defined at 0.5V.

The output "high" level is defined at  $V_{CC} - 2.0V$ .



■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS ( $V_{CC} = 5.0V \pm 5\%$ ,  $V_{SS} = 0V$ ,  $T_a = 0$  to  $+70^\circ C$  unless otherwise noted)

Item	Symbol	Measuring Condition	4 MHz Version		6 MHz Version		8 MHz Version		Unit	
			HD63484-4		HD63484-6		HD63484-8			
			min	max	min	max	min	max		
Input "High" Level Voltage	All Inputs	$V_{IH}$							V	
Input "Low" Level Voltage	All Inputs	$V_{IL}$	-0.3	0.7	-0.3	0.7	-0.3	0.7	V	
Input Leak Current	$\overline{R/W}$ , $\overline{CS}$ , $\overline{RS}$ , $\overline{RES}$ , $\overline{DACK}$ , $2\overline{CLK}$ , $\overline{LPSTB}$	$I_{in}$	$V_{in}=0 \sim V_{CC}$	-2.5	2.5	-2.5	2.5	-2.5	2.5	$\mu A$
Three State (Off State) Input Current	$\overline{D_0} \sim \overline{D_{15}}$ , $\overline{EXSYNC}$ , $\overline{MAD_0} \sim \overline{MAD_{15}}$	$I_{TSI}$	$V_{in}=0.4 \sim V_{CC}$	-10	10	-10	10	-10	10	$\mu A$
Output "High" Level Voltage	$\overline{D_0} \sim \overline{D_{15}}$ , $\overline{MAD_0} \sim \overline{MAD_{15}}$ , $\overline{CUD_1}$ , $\overline{CUD_2}$ , $\overline{DREQ}$ , $\overline{DTACK}$ , $\overline{HSYNC}$ , $\overline{VSYNC}$ , $\overline{EXSYNC}$ ,	$V_{OH}$	$I_{OH} = -400\mu A$	$V_{CC}$ -1.0	-	$V_{CC}$ -1.0	-	$V_{CC}$ -1.0	-	V
Output "Low" Level Voltage	$\overline{DISP_1}$ , $\overline{DISP_2}$ , $\overline{CHR}$ , $\overline{MRD}$ , $\overline{DRAW}$ , $\overline{AS}$ , $\overline{MCYC}$ , $\overline{RA_4}$ , $\overline{MA_{16}}/\overline{RA_0} \sim$ $\overline{MA_{19}}/\overline{RA_3}$	$V_{OL}$	$I_{OL} = 2.2mA$	-	0.5	-	0.5	-	0.5	V
Output Leak Current (Off State)	$\overline{IRQ}$ , $\overline{DONE}$	$I_{LOD}$	$V_{OH} = V_{CC}$	-	10	-	10	-	10	$\mu A$
Input Capacity	$\overline{D_0} \sim \overline{D_{15}}$ , $\overline{EXSYNC}$ , $\overline{MAD_0} \sim \overline{MAD_{15}}$	$C_{in}$	$V_{in}=0V$ , $T_a=25^\circ C$ , $f=1.0MHz$	-	17	-	17	-	17	pF
				$\overline{R/W}$ , $\overline{CS}$ , $\overline{RS}$ , $\overline{RES}$ , $\overline{DACK}$ , $2\overline{CLK}$ , $\overline{LPSTB}$	-	17	-	17	-	
Output Capacity	$\overline{IRQ}$ , $\overline{DONE}$	$C_{out}$	$V_{in}=0V$ , $T_a=25^\circ C$ , $f=1.0MHz$	-	15	-	15	-	15	pF
Current Consumption		$I_{CC}$	<ul style="list-style-type: none"> <li>• Chip not selected</li> <li>• Display in progress</li> </ul>	-	60	-	80	-	100	mA
				<ul style="list-style-type: none"> <li>• Data bus in read/write operation</li> <li>• Display in progress</li> <li>• Command execution in progress</li> </ul>	-	60	-	80	-	

• AC CHARACTERISTICS ( $V_{CC} = 5.0V \pm 5\%$ ,  $V_{SS} = 0V$ ,  $T_a = 0$  to  $+70^\circ C$  unless otherwise noted)

No.	Item	Symbol	Measuring Condition	4 MHz Version		6 MHz Version		8 MHz Version		Unit
				HD63484-4		HD63484-6		HD63484-8		
				min	max	min	max	min	max	
	Operation Frequency of 2CLK	f		1	4	1	6	1	8	MHz
1	Clock Cycle Time	$t_{cyc}$	Fig. 1	250	1000	167	1000	125	1000	ns
2	Clock "High" Level Pulse Width	$t_{PWCH}$		115	500	75	500	55	500	ns
3	Clock "Low" Level Pulse Width	$t_{PWCL}$		115	500	75	500	55	500	ns
4	Clock Rise Time	$t_{Cr}$		—	10	—	10	—	10	ns
5	Clock Fall Time	$t_{Cf}$		—	10	—	10	—	10	ns
6	R/W Setup Time	$t_{RWS}$	Fig. 2, Fig. 3	70	—	60	—	50	—	ns
7	R/W Hold Time	$t_{RWH}$		0	—	0	—	0	—	ns
8	RS Setup Time	$t_{RSS}$		70	—	60	—	50	—	ns
9	RS Hold Time	$t_{RSH}$		0	—	0	—	0	—	ns
10	CS Setup Time	$t_{CSS}$		50	—	40	—	40	—	ns
11	CS "High" Level Width	$t_{WCSH}$		80	—	70	—	60	—	ns
12										
13	Read Wait Time	$t_{RWAI}$	Fig. 2	0	—	0	—	0	—	ns
14	Read Data Access Time	$t_{RDAC}$		—	120	—	100	—	80	ns
15	Read Data Hold Time	$t_{RDH}$		10	—	10	—	10	—	ns
16	Read Data Turn Off Time	$t_{RDZ}$		—	60	—	60	—	60	ns
17	$\overline{DTACK}$ Delay Time (Z to L)	$t_{DTKZL}$	Fig. 2, Fig. 3	—	90	—	80	—	70	ns
18	$\overline{DTACK}$ Delay Time (D to L)	$t_{DTKDL}$	Fig. 2	0	—	0	—	0	—	ns
19	$\overline{DTACK}$ Release Time (L to H)	$t_{DTKLH}$	Fig. 2, Fig. 3	—	100	—	90	—	80	ns
20	$\overline{DTACK}$ Turn Off Time (H to Z)	$t_{DTKZ}$		—	100	—	100	—	100	ns
21	Data Bus 3 State Recovery Time 1	$t_{DBRT1}$	Fig. 2	0	—	0	—	0	—	ns
22	Write Wait Time	$t_{WWAI}$	Fig. 3	0	—	0	—	0	—	ns
23	Write Data Setup Time	$t_{WDS}$		80	—	60	—	40	—	ns
24	Write Data Hold Time	$t_{WDH}$		10	—	10	—	10	—	ns
25	$\overline{DREQ}$ Delay Time 1	$t_{DRQD1}$	Fig. 4, Fig. 5	—	150	—	130	—	110	ns
26	$\overline{DREQ}$ Delay Time 2	$t_{DRQD2}$		—	90	—	80	—	70	ns
27	DMA R/W Setup Time	$t_{DMRWS}$		70	—	60	—	50	—	ns
28	DMA R/W Hold Time	$t_{DMRWH}$		0	—	0	—	0	—	ns
29	$\overline{DACK}$ Setup Time	$t_{DAKS}$		50	—	40	—	40	—	ns
30	$\overline{DACK}$ "High" Level Width	$t_{WDAKH}$		80	—	70	—	60	—	ns
31										
32	DMA Read Wait Time	$t_{DRW}$	Fig. 4	0	—	0	—	0	—	ns
33	DMA Read Data Access Time	$t_{DRDAC}$		—	120	—	100	—	80	ns
34	DMA Read Data Hold Time	$t_{DRDH}$		10	—	10	—	10	—	ns
35	DMA Read Data Turn Off Time	$t_{DRDZ}$		—	60	—	60	—	60	ns
36	DMA $\overline{DTACK}$ Delay Time (Z to L)	$t_{DDTZL}$	Fig. 4, Fig. 5	—	90	—	80	—	70	ns
37	DMA $\overline{DTACK}$ Delay Time (D to L)	$t_{DDTDL}$	Fig. 4	0	—	0	—	0	—	ns
38	DMA $\overline{DTACK}$ Release Time (L to H)	$t_{DDTLH}$	Fig. 4, Fig. 5	—	100	—	90	—	80	ns
39	DMA $\overline{DTACK}$ Turn Off Time (H to Z)	$t_{DDTHZ}$		—	100	—	100	—	100	ns
40	DONE Output Delay Time	$t_{DND}$		—	90	—	80	—	70	ns
41	DONE Output Turn Off Time (L to Z)	$t_{DNLZ}$		—	100	—	90	—	80	ns

(to be continued)



No.	Item	Symbol	Measuring Condition	4 MHz Version		6 MHz Version		8 MHz Version		Unit
				HD63484-4		HD63484-6		HD63484-8		
				min	max	min	max	min	max	
42	Data Bus 3 State Recovery Time 2	$t_{DBRT2}$	Fig. 4	0	—	0	—	0	—	ns
43	$\overline{DONE}$ Input Pulse Width	$t_{DNPW}$	Fig. 4, Fig. 5	2	—	2	—	2	—	Clk. Cyc.
44	DMA Write Wait Time	$t_{DWW}$	Fig. 5	0	—	0	—	0	—	ns
45	DMA Write Data Setup Time	$t_{DWD S}$		80	—	60	—	40	—	ns
46	DMA Write Data Hold Time	$t_{DWDH}$		10	—	10	—	10	—	ns
47										
48	$\overline{AS}$ "Low" Level Pulse Width	$t_{PWASL}$	Fig. 6~ Fig. 9	80	—	40	—	25	—	ns
49	Memory Address Hold Time 2	$t_{MAH2}$	Fig. 7, Fig. 8	10	—	10	—	10	—	ns
50	$\overline{AS}$ Delay Time 1	$t_{ASD1}$	Fig. 6~ Fig. 9	—	90	—	75	—	65	ns
51	$\overline{AS}$ Delay Time 2	$t_{ASD2}$		—	90	—	75	—	65	ns
52	Memory Address Delay Time	$t_{MAD}$		—	95	—	80	—	70	ns
53	Memory Address Hold Time 1	$t_{MAH1}$		10	—	10	—	10	—	ns
54	Memory Address Turn Off Time (A to Z)	$t_{MAAZ}$	Fig. 6, 7 Fig. 9	—	50	—	50	—	50	ns
55	Memory Address Data Setup Time	$t_{MRDS}$	Fig. 7	60	—	50	—	40	—	ns
56	Memory Read Data Hold Time	$t_{MRDH}$		10	—	10	—	10	—	ns
57	MA/RA Delay Time	$t_{MARAD}$	Fig. 6~ Fig. 9	—	100	—	90	—	80	ns
58	MA/RA Hold Time	$t_{MARA H}$	Fig. 6~ Fig. 8	10	—	10	—	10	—	ns
59	MCYC Delay Time	$t_{MCYCD}$	Fig. 6~ Fig. 10	—	60	—	50	—	50	ns
60	MRD Delay Time	$t_{MRDD}$	Fig. 6~ Fig. 9	—	90	—	80	—	70	ns
61	MRD Hold Time	$t_{MRH}$		10	—	10	—	10	—	ns
62	$\overline{DRAW}$ Delay Time	$t_{DRWD}$		—	90	—	80	—	70	ns
63	$\overline{DRAW}$ Hold Time	$t_{DRWH}$		10	—	10	—	10	—	ns
64	Memory Write Data Delay Time	$t_{MWDD}$	Fig. 8	—	90	—	80	—	70	ns
65	Memory Write Data Hold Time	$t_{MWDH}$		10	—	10	—	10	—	ns
66										
67	$\overline{HSYNC}$ Delay Time	$t_{HSD}$	Fig. 9~ Fig. 11	—	90	—	80	—	70	ns
68	$\overline{VSYNC}$ Delay Time	$t_{VSD}$	Fig. 10	—	90	—	80	—	70	ns
69	$\overline{DISP_1}, \overline{DISP_2}$ Delay Time	$t_{DSPD}$		—	90	—	80	—	70	ns
70	$\overline{CUD_1}, \overline{CUD_2}$ Delay Time	$t_{CUDD}$		—	90	—	80	—	70	ns
71	$\overline{EXSYNC}$ Output Delay Time	$t_{EXD}$		20	90	20	80	20	70	ns
72	CHR Delay Time	$t_{CHD}$		—	90	—	80	—	70	ns
73										
74										
75	$\overline{EXSYNC}$ Input Pulse Width	$t_{EXSW}$	Fig. 11	3	—	3	—	3	—	Clk. Cyc.
76	$\overline{EXSYNC}$ Input Setup Time	$t_{EXS}$		60	—	60	—	50	—	ns
77	$\overline{EXSYNC}$ Input Hold Time	$t_{EXH}$		30	—	30	—	30	—	ns
78	LPSTB Uncertain Time 1	$t_{LPD1}$	Fig. 12	70	—	70	—	70	—	ns
79	LPSTB Uncertain Time 2	$t_{LPD2}$		10	—	10	—	10	—	ns
80	LPSTB Input Hold Time	$t_{LPH}$		10	—	10	—	10	—	ns
81	LPSTB Input Inhibit time	$t_{LPI}$		4	—	4	—	4	—	Clk. Cyc.

No.	Item	Symbol	Measuring Condition	4 MHz Version		6 MHz Version		8 MHz Version		Unit
				HD63484-4		HD63484-6		HD63484-8		
				min	max	min	max	min	max	
82	DACK Setup Time for $\overline{\text{RES}}$	$t_{\text{DAKSR}}$	Fig. 13	100	—	100	—	100	—	ns
83	DACK Hold Time for $\overline{\text{RES}}$	$t_{\text{DAKHR}}$		0	—	0	—	0	—	ns
84	$\overline{\text{RES}}$ Input Pulse Width	$t_{\text{RES}}$		10	—	10	—	10	—	Clk. Cyc.
85	$\overline{\text{IRQ}}$ Delay Time 1	$t_{\text{IRQ1}}$	Fig. 14	—	250	—	200	—	150	ns
86	$\overline{\text{IRQ}}$ Delay Time 2	$t_{\text{IRQ2}}$		—	500	—	500	—	500	ns
87	ATR Delay Time 1	$t_{\text{ATRD1}}$	Fig. 9	—	100	—	90	—	80	ns
88	ATR Hold Time 1	$t_{\text{ATRH1}}$		10	—	10	—	10	—	ns
89										
90	ATR Delay Time 2	$t_{\text{ATRD2}}$	Fig. 9	—	100	—	90	—	80	ns
91	ATR Hold Time 2	$t_{\text{ATRH2}}$		10	—	10	—	10	—	ns
100	$\overline{\text{CS}}$ Cycle Time	$t_{\text{CSC}}$	Fig. 2, Fig. 3	4	—	4	—	4	—	Clk. Cyc.
101	$\overline{\text{CS}}$ "Low" Level Width	$t_{\text{WCSL}}$		2	—	2	—	2	—	Clk. Cyc.
102	$\overline{\text{CS}}$ "High" Level Width	$t_{\text{WCSH}}$		2	—	2	—	2	—	Clk. Cyc.
104	DACK Cycle Time	$t_{\text{DACKC}}$	Fig. 4A, Fig. 5A	4	—	4	—	4	—	Clk. Cyc.
105	DACK "Low" Level Width	$t_{\text{WDACKL}}$		2	—	2	—	2	—	Clk. Cyc.
106	DACK "High" Level Width	$t_{\text{WDACKH}}$		2	—	2	—	2	—	Clk. Cyc.

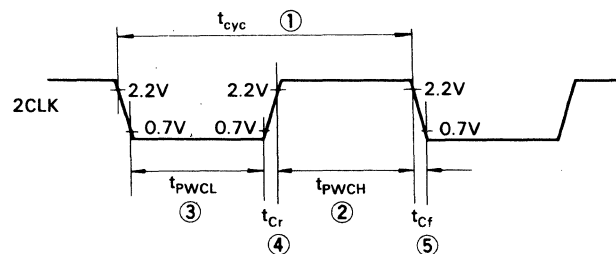
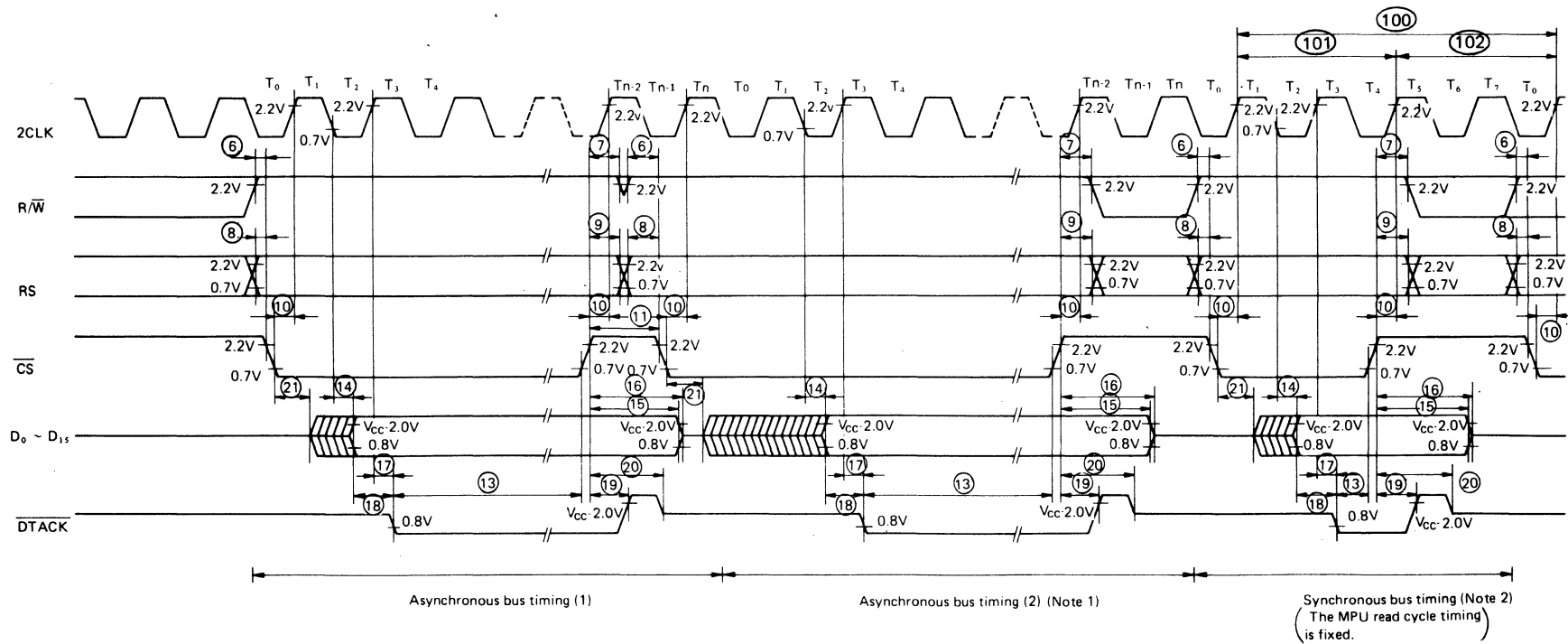


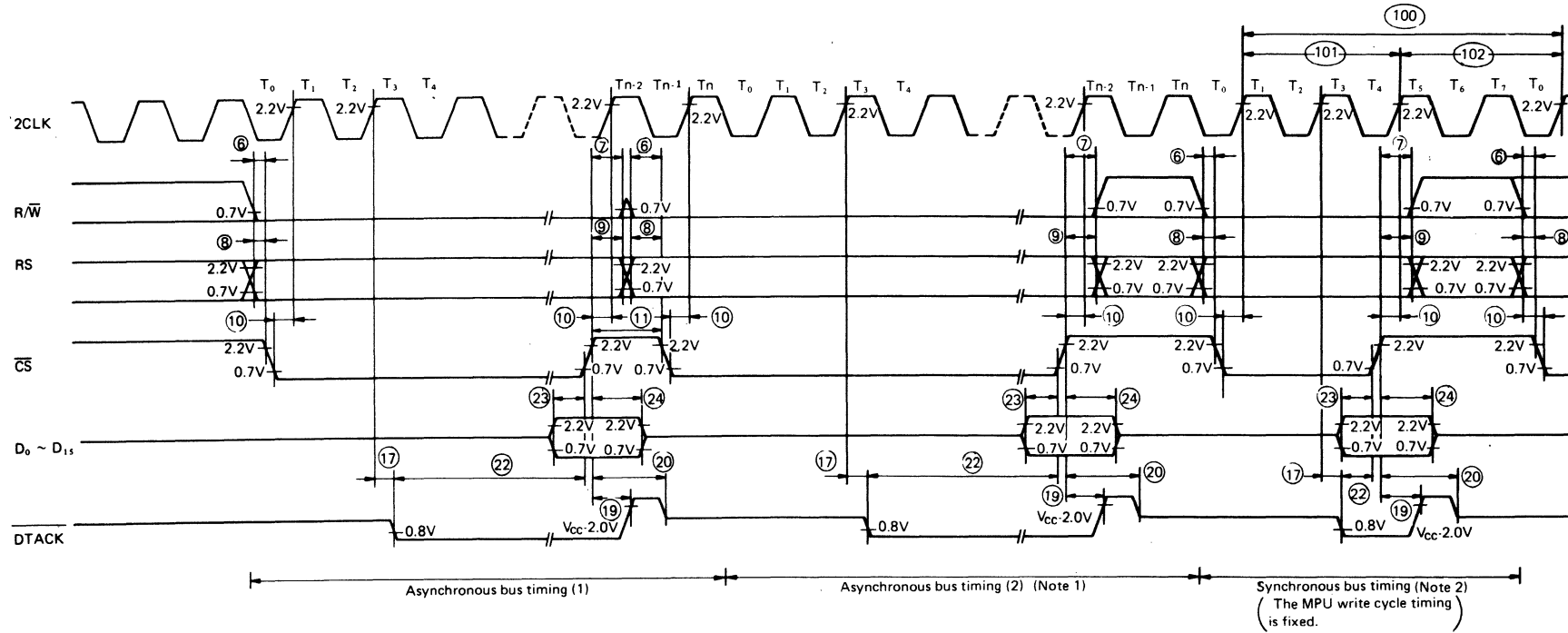
Figure 1 2CLK Waveform



(Note 1)  $\overline{CS}$  "high" width must satisfy the specification (11).  
Unless satisfying the spec (102), DTACK and read data responses to the succeeding cycle are delayed.

(Note 2) When the ACRTC is used with the synchronous bus timing, the specifications (100) (101) and (102) must be satisfied.

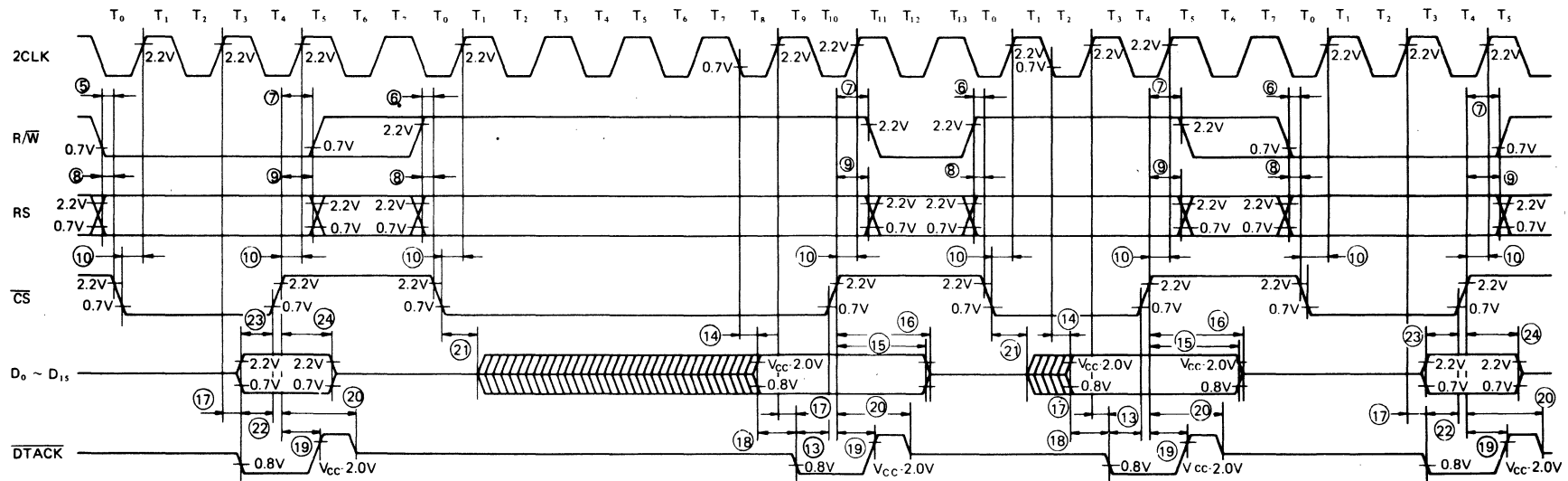
Figure 2 MPU Read Cycle Timing (MPU ← ACRTC)



(Note 1)  $\overline{CS}$  "high" width must satisfy the specification (11).  
Unless satisfying the spec. (102), DTACK response to the succeeding cycle is delayed.

(Note 2) When the ACRTC is used with the synchronous bus timing, the specifications (100) (101) and (102) must be satisfied.

Figure 3 MPU Write Cycle Timing (MPU → ACRTC)



(Note) When the MPU read cycle immediately follows the MPU write cycle execution,  $\overline{DTACK}$  and the read data responses are delayed (by 3 cycles of 2CLK) even though the spec. (102) is satisfied.

Figure 3A MPU Read/Write Cycle Timing (MPU → ACRTC)

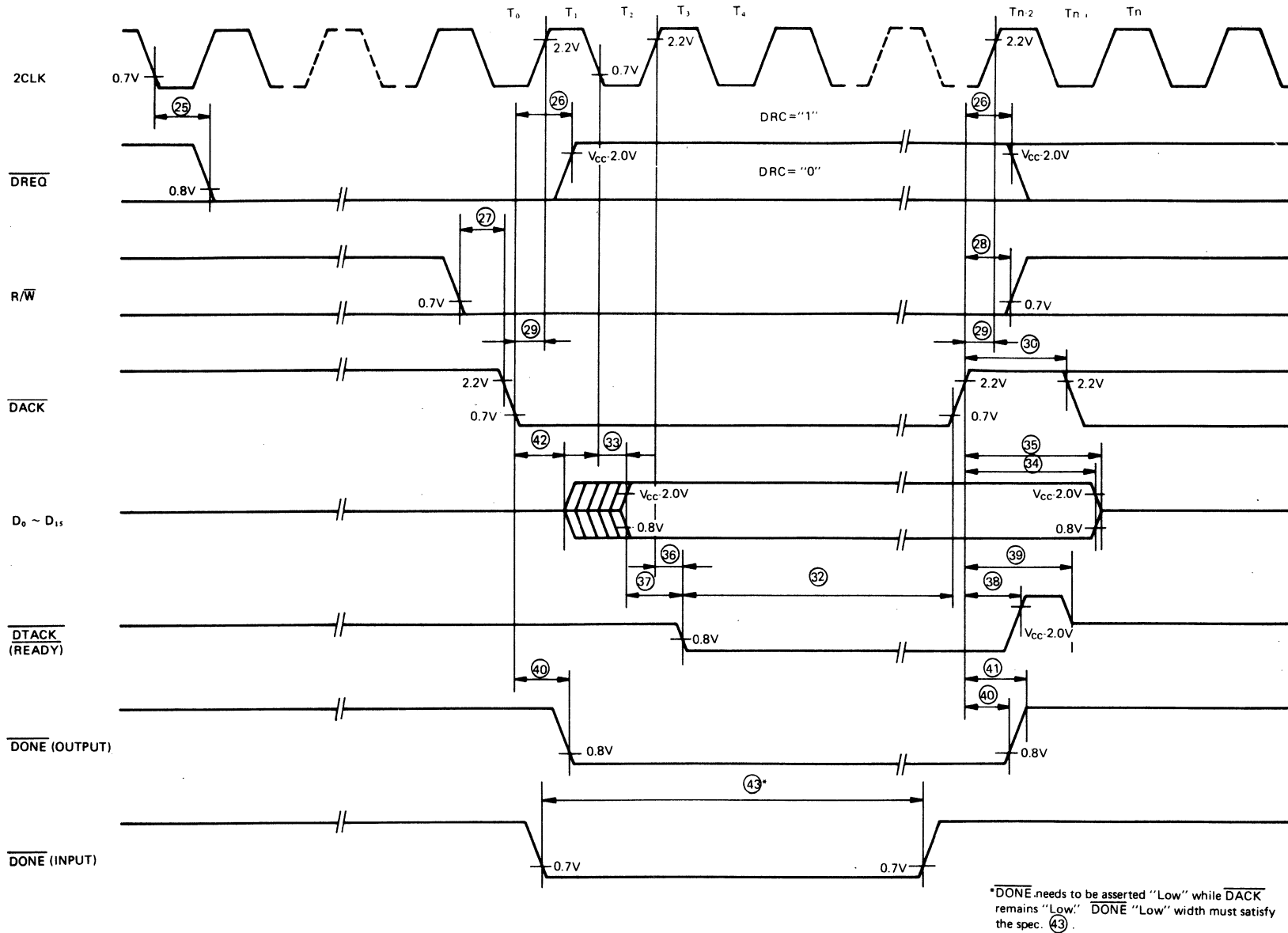
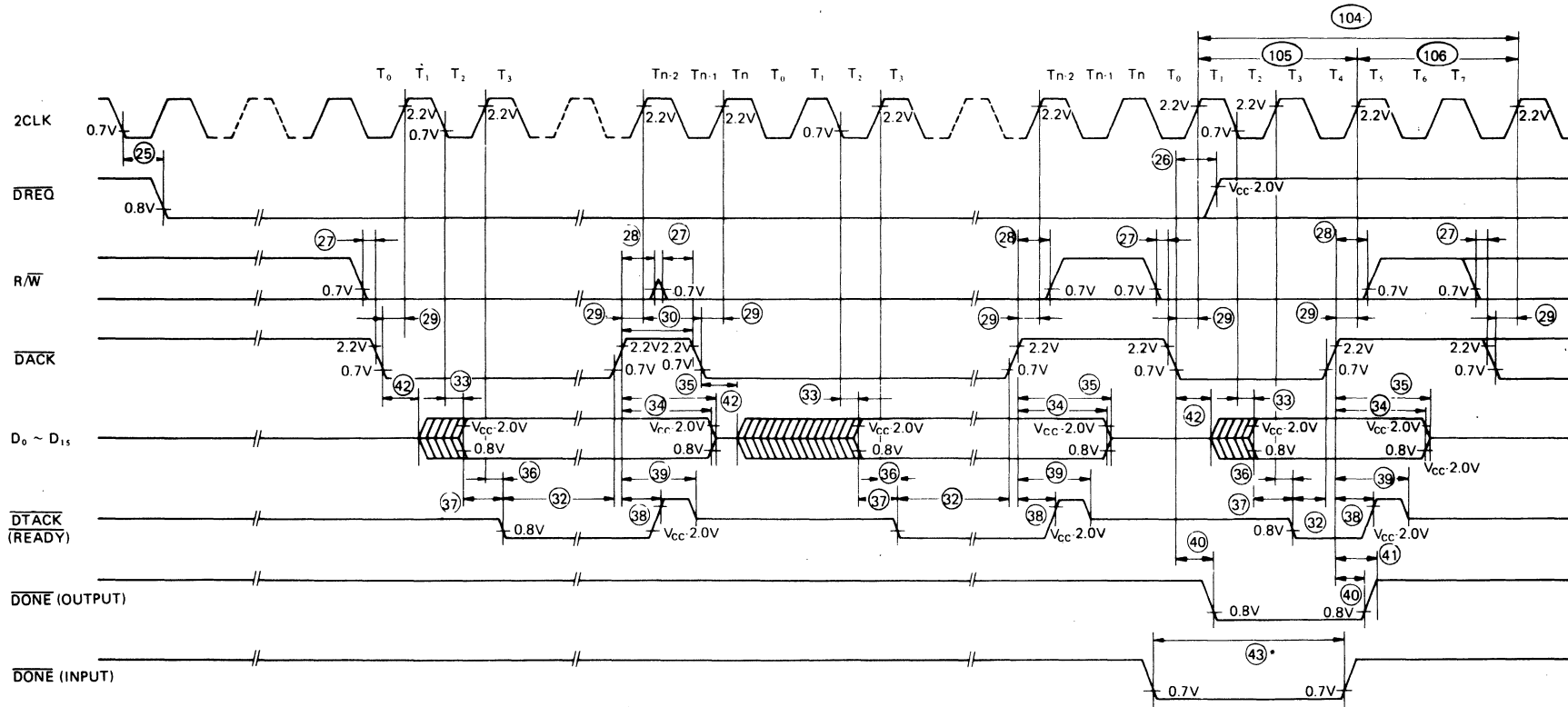


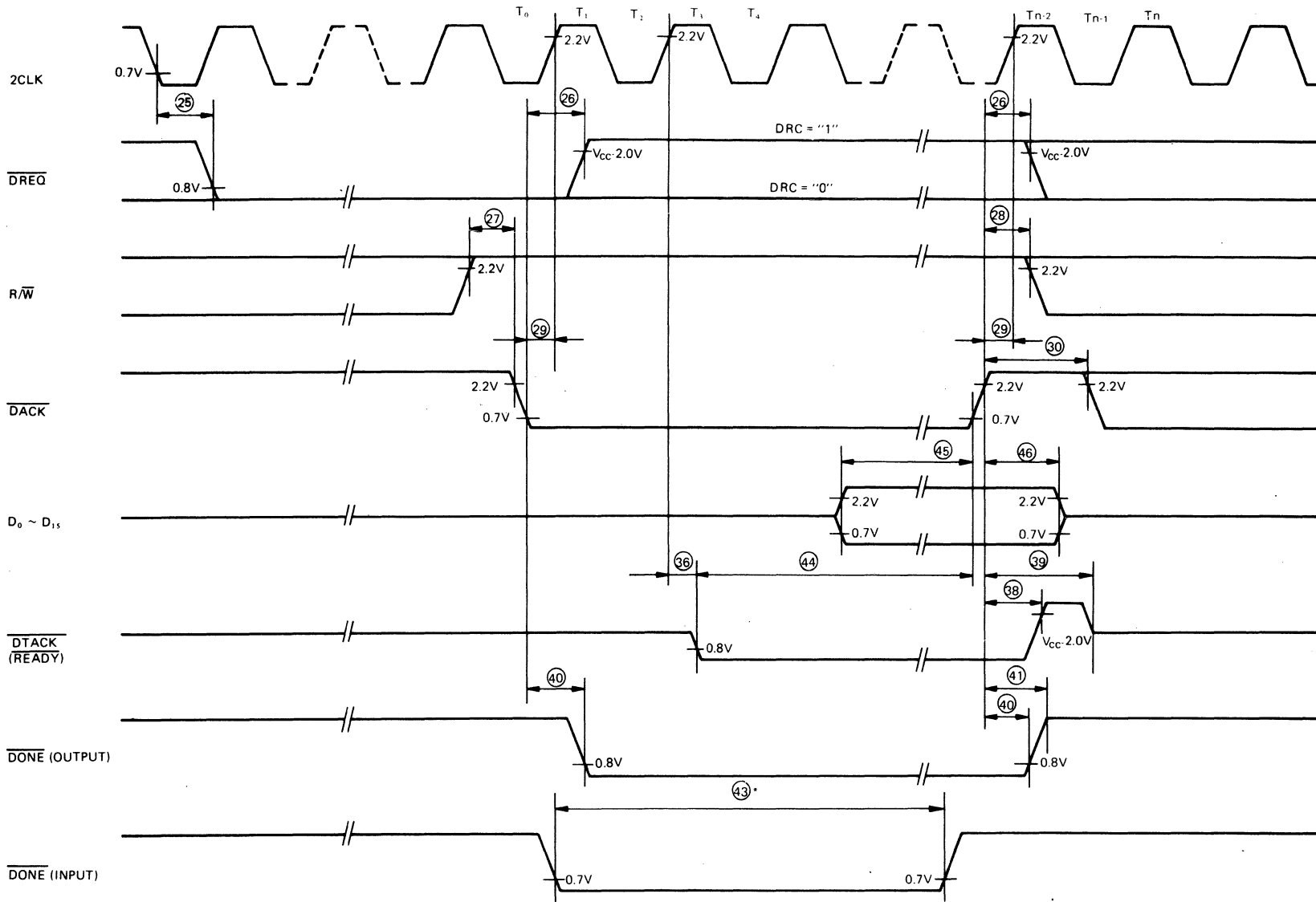
Figure 4 DMA Read Cycle Timing (Memory ← ACRTC)



\*DONE needs to be asserted "Low" while DACK remains "Low".  
DONE "Low" width must satisfy the spec (43).

(Note) DACK ("high" width must satisfy the spec (30). Unless satisfying the spec (106),  
DTACK and the read data responses to the succeeding cycle are delayed.  
 When the ACRTC is used with the synchronous bus timing, the specifications (104), (105)  
 and (106) must be satisfied.

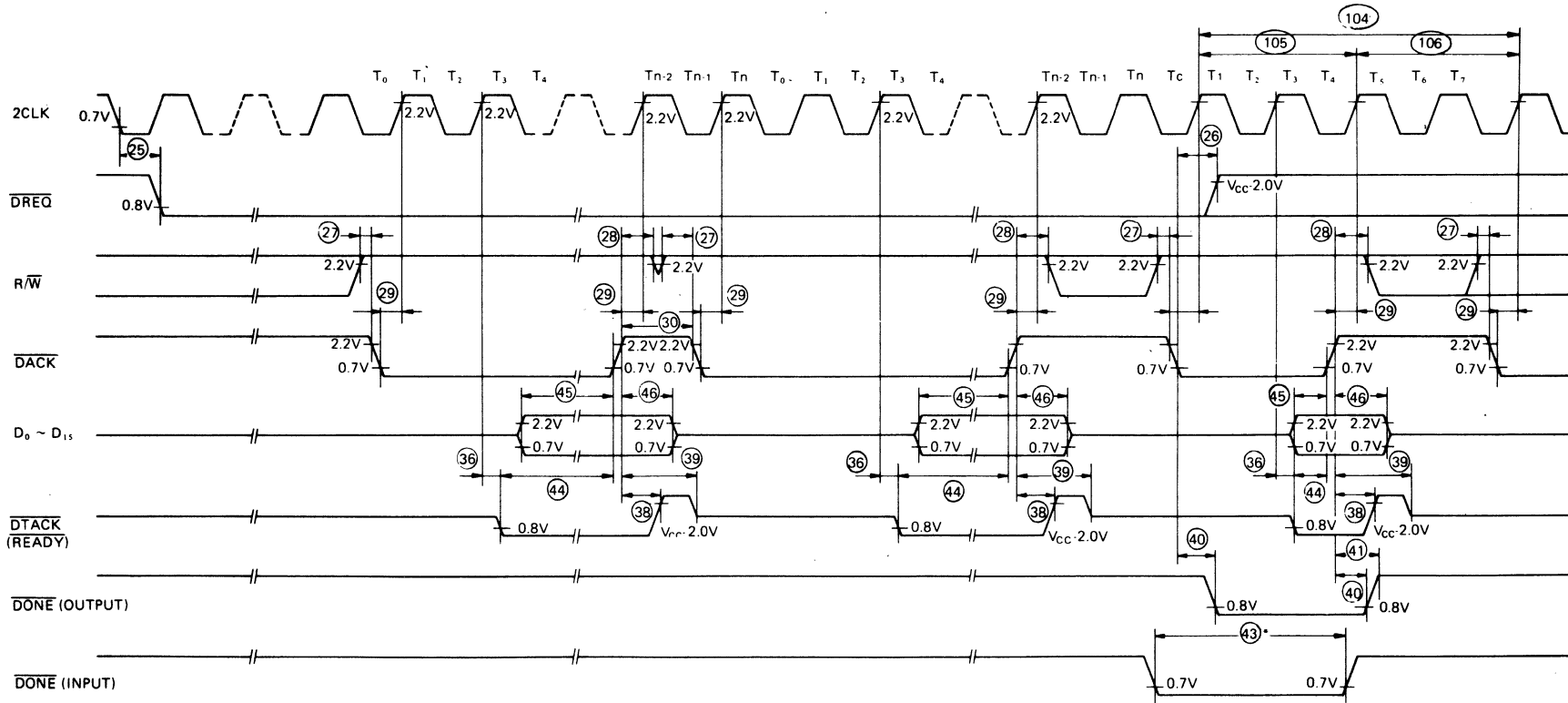
Figure 4A DMA Read Cycle Timing (Memory ← ACRTC): Burst Mode



\*DONE needs to be asserted "Low" while DACK remains "Low". DONE "Low" width must satisfy the spec. 43.

Figure 5 DMA Write Cycle Timing (Memory → ACRTC)





\*DONE needs to be asserted "Low" while DACK remains "low".  
DONE "low" width must satisfy the spec. 43.

(Note) DACK "high" width must satisfy the spec. 30, unless satisfying the spec. 106.  
DTACK response to the succeeding cycle is delayed.  
 When the ACRTC is used with the synchronous bus timing, the specifications 104, 105 and 106 must be satisfied.

Figure 5A DMA Write Cycle Timing (Memory → ACRTC): Burst Mode

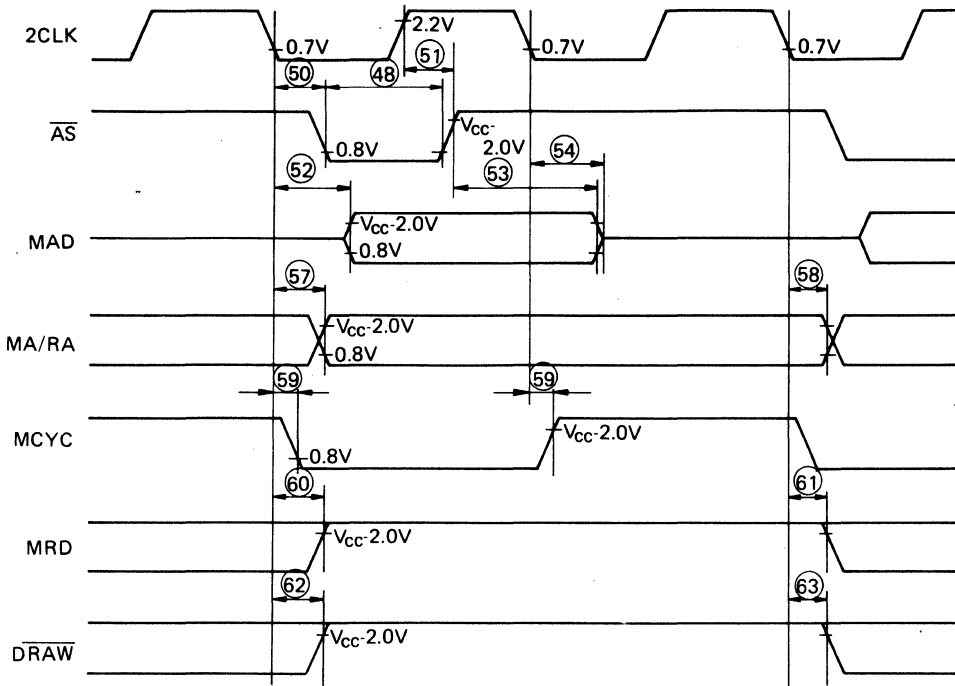


Figure 6 Screen Display Cycle Timing

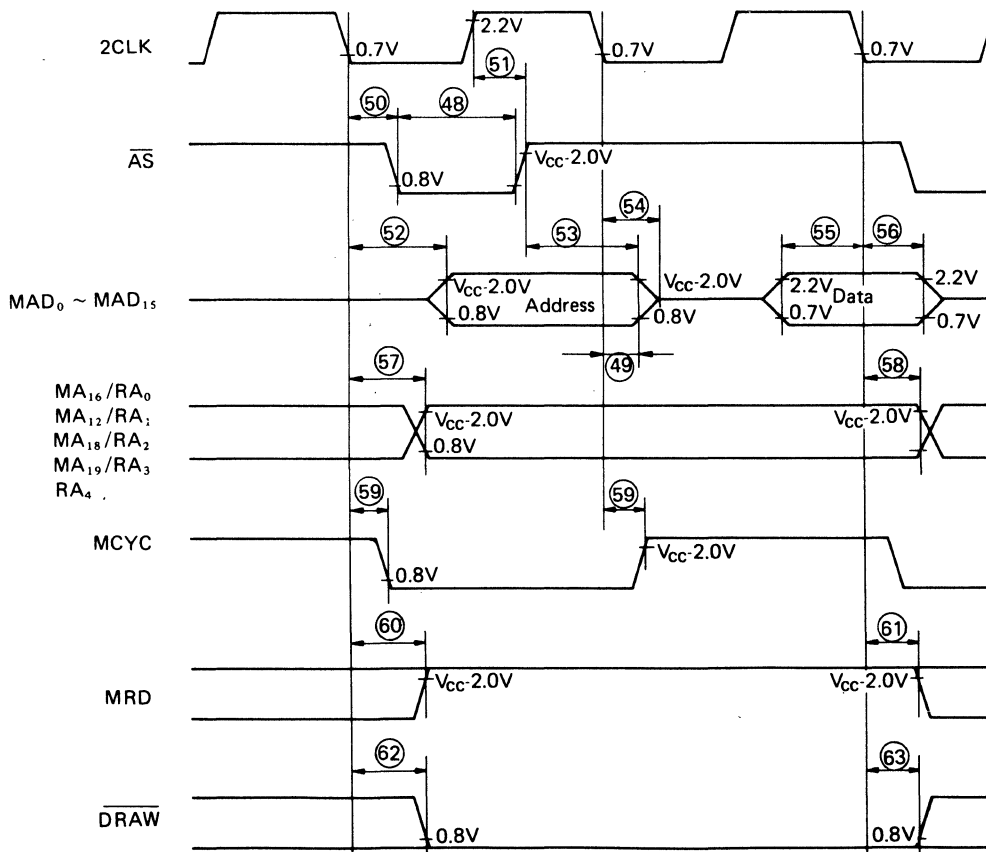


Figure 7 Frame Memory Read Cycle Timing (ACRTC ← Frame Memory)

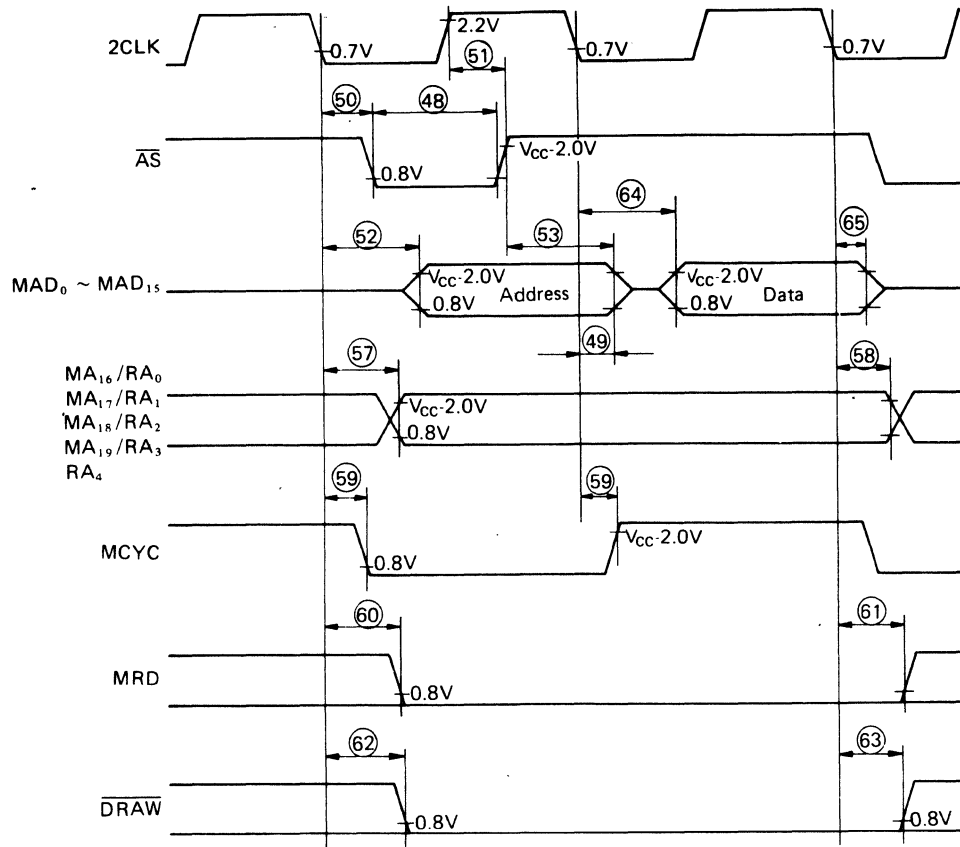
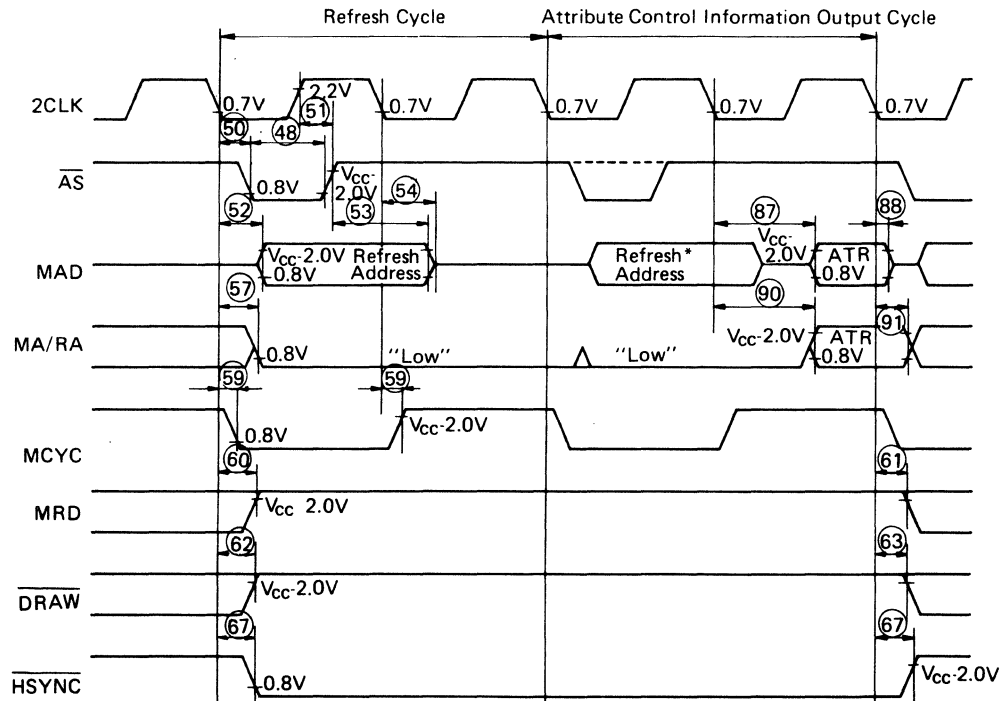


Figure 8 Frame Memory Write Cycle Timing (ACRTC → Frame Memory)



\*When AS is "High", a "0" output is given.

Figure 9 Frame Memory Refresh/Attribute Control Information Output Cycle Timing

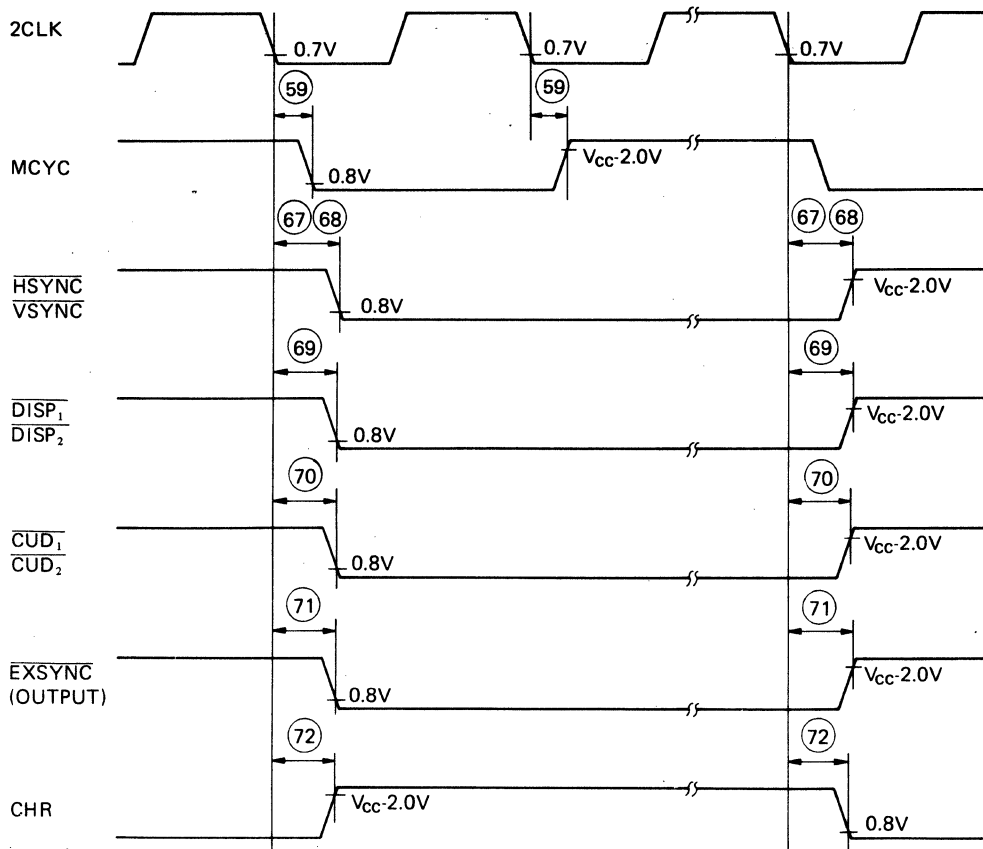
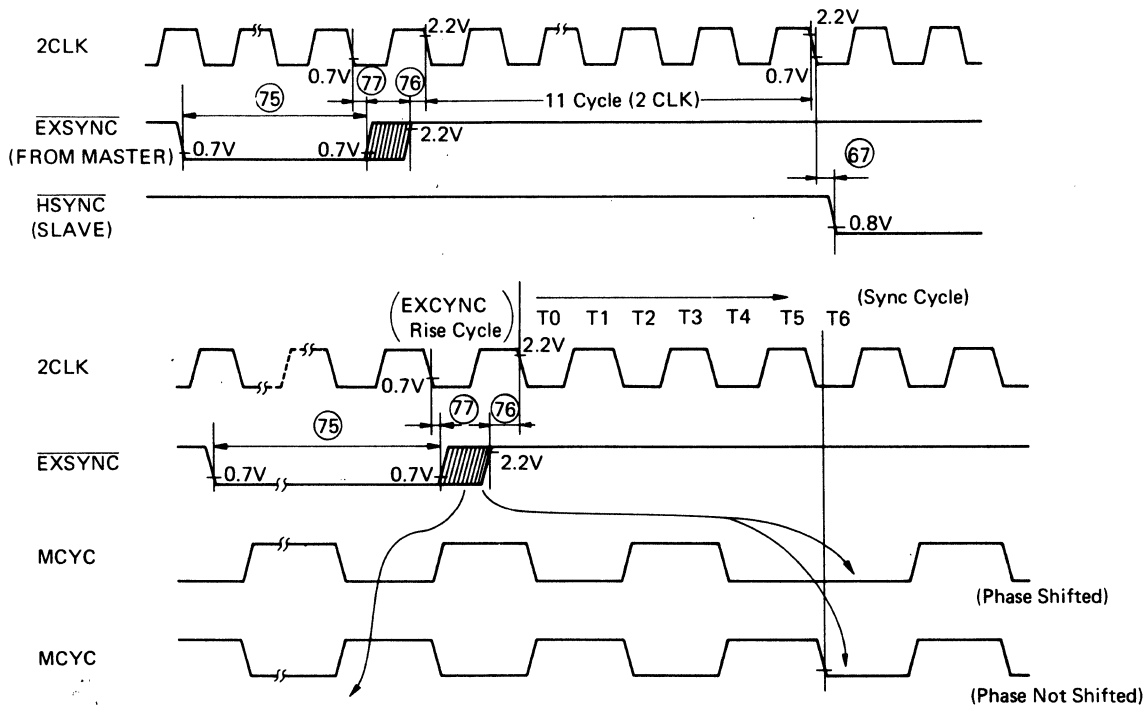


Figure 10 Display Control Signal Output Timing



(When the leading edge of EXSYNC enters this period, ACRTC shifts the internal phase according to the above sequence.)

Figure 11 EXSYNC Input Timing

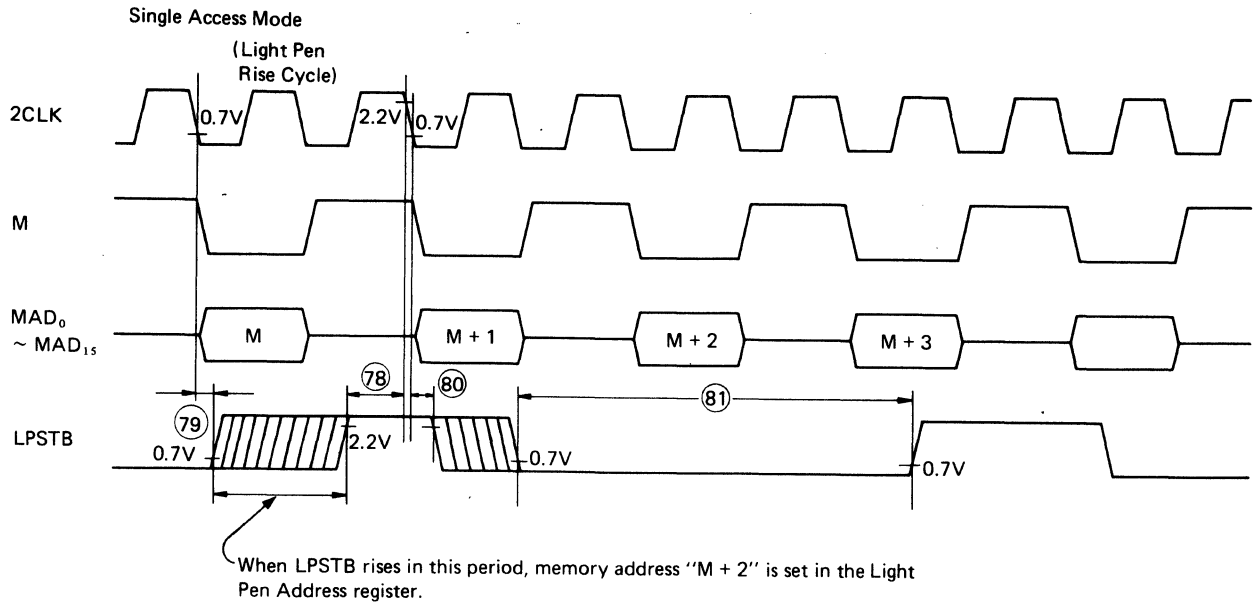
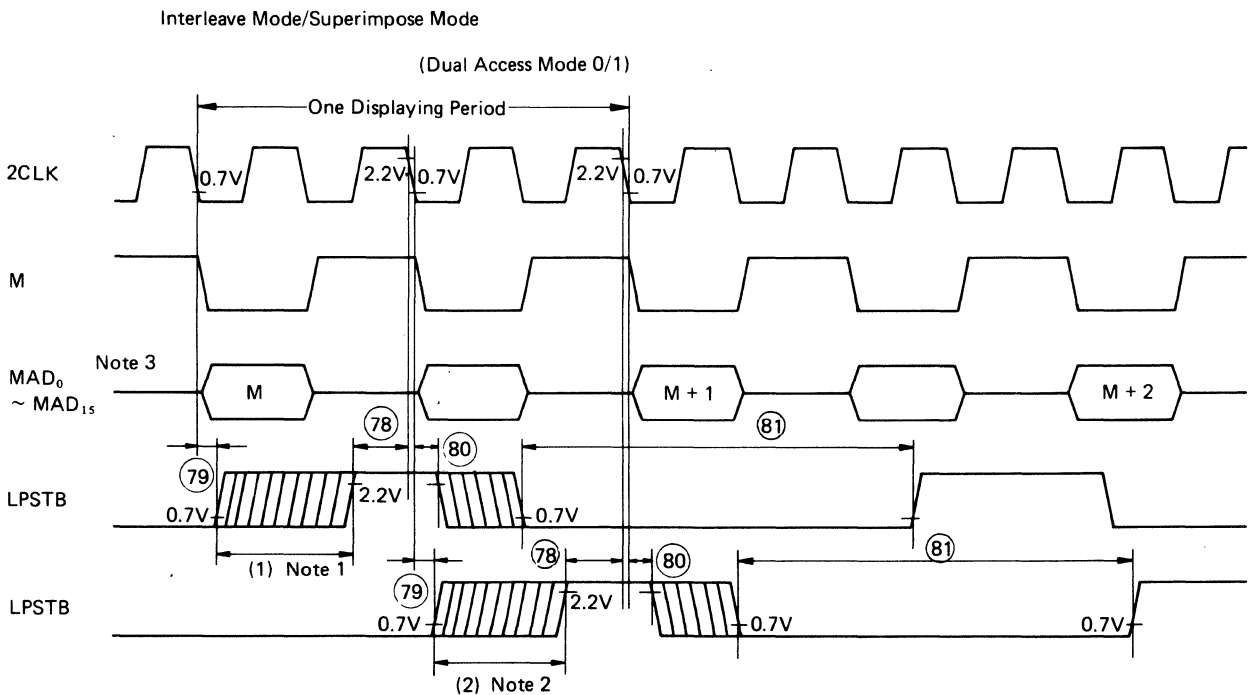


Figure 12 LPSTB Input Timing



Note 1: When LPSTB rises in the period (1), memory address "M + 1" is set in the Light Pen Address register.

Note 2: When LPSTB rises in the period (2), memory address "M + 2" is set in the "Ligyt Pen Address register.

Note 3: In the Interleave Mode, memory address "M", "M + 1", "M + 2" denote the display address.  
In the Superimpose Mode, memory address "M", "M + 1", M + 2" denote the display address of the background screen.

Figure 12A LPSTB Input Timing

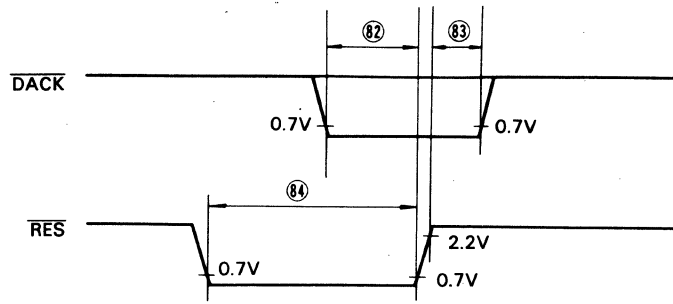


Figure 13  $\overline{\text{RES}}$  Input and  $\overline{\text{DACK}}$  Input Timing  
(System Reset and 16-bit/8-bit Selection)

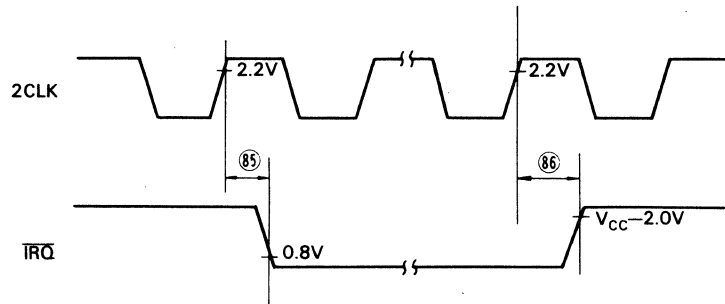
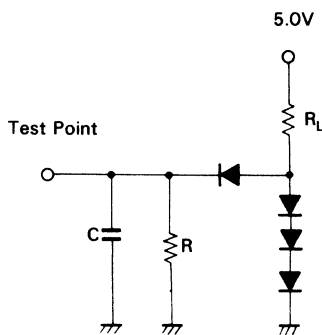


Figure 14  $\overline{\text{IRQ}}$  Output Timing



Signal	Load Condition
$D_0 \sim D_{15}$ $\overline{\text{DTACK}}$ $\overline{\text{DREQ}}$ $\text{MAD}_0 \sim \text{MAD}_{15}$ $\text{MA}_{16}/\text{RA}_0 \sim \text{MA}_{19}/\text{RA}_3$ $\text{RA}_4$ $\overline{\text{VSYNC}}, \overline{\text{HSYNC}}$ $\overline{\text{EXSYNC}}$ $\text{MCYC}, \overline{\text{AS}}, \text{MRD}$ $\overline{\text{DRAW}}, \text{CHR}$ $\overline{\text{DISP}}_1, \overline{\text{DISP}}_2$ $\overline{\text{CUD}}_1, \overline{\text{CUD}}_2$	$R_L = 1.8\text{k}\Omega$ $C = 40\text{pF}$ $R = 10\text{k}\Omega$  All diodes are 1S2074(H)'s or the equivalent.

Figure 15 Test Load Circuit A

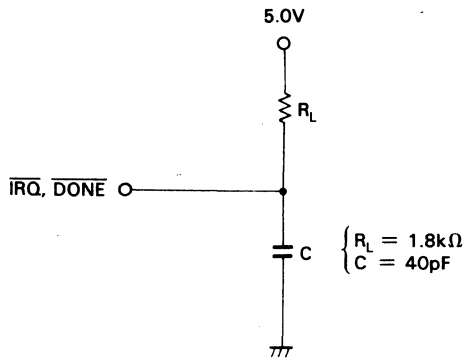


Figure 16 Test Load Circuit B

**NOTE FOR USE**

- (1) Power-on Sequence  
Following condition needs to be satisfied when the power turns-on.

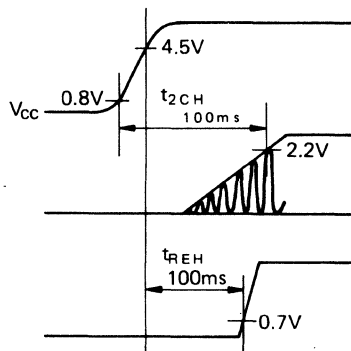


Figure 17 Power-on Sequence

- (2) Output Waveform

In case that ringing noise occurs beyond tolerance on CRT data buses ( $MAD_0 \sim MAD_{15}$ ,  $MA_{16}/RA_0 \sim MA_{19}/RA_3$ ,  $RA_4$ ), damping resistors may be required for data buses as shown in the figures below.

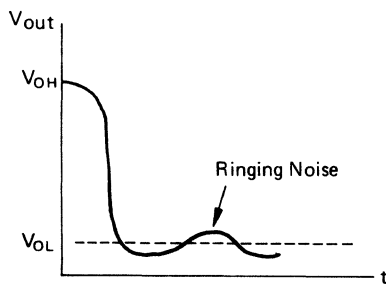


Figure 18 Ringing Noise

Note: The ringing level depends on the load capacity, and it can be  $V_{OH} + 0.1V$ .

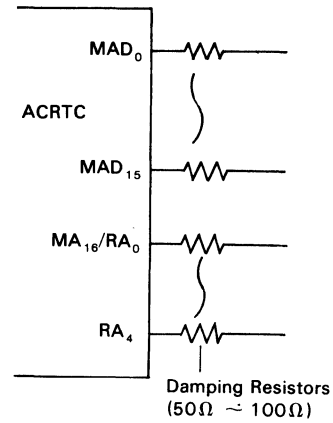


Figure 19 Damping Resistors

- (3) Power Supply Circuit

When designing  $V_{CC}$  and  $V_{SS}$  pattern of the circuit board, the capacitors need to be located nearest to pin 14 ( $V_{CC}$ ) and pin 16 ( $V_{SS}$ ) or pin 51 ( $V_{SS}$ ) and pin 49 ( $V_{CC}$ ) as shown in the figure 20.

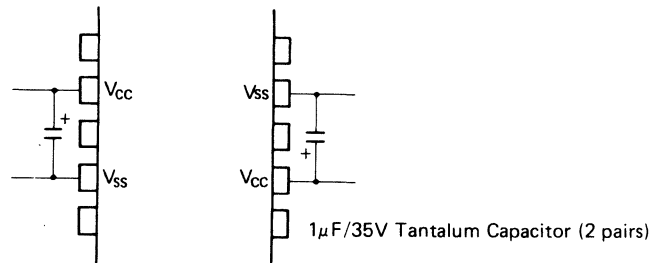


Figure 20 Note for Designing Power Supply Circuit

**■ ACRTC INTRODUCTION**

**● HD63484 (ACRTC) FEATURES**

Powerful visual interfaces are a key component of advanced system architectures. A proven technique uses raster scanned CRT technology for the display of graphics and text information.

Systems which use first generation CRT Controllers (CRTC) are constrained by hardware/software design time, manufacturing cost, and limited MPU bandwidth.

To meet the functional requirements for powerful visual interfaces, and to support their use in high volume, cost sensitive applications, advanced circuit design and VLSI CMOS manufacturing technologies have been used to create a next generation CRTC, the HD63484 ACRTC (Advanced CRT Controller).

The ACRTC concept is to incorporate major functionality, formerly requiring external hardware and software, on-chip. In this way, both higher performance and reduced system cost benefits are achieved.

\* High Level Command Language Increases Performance and Reduces Software Development Cost.

- ACRTC Converts Logical X-Y Coordinates to Physical Frame Buffer Addresses.
- 38 Commands including 23 Graphic Drawing Commands - LINE, RECTANGLE, POLYLINE, POLYGON, CIRCLE, ELLIPSE, ARC, ELLIPSE ARC, FILLED RECTANGLE, PAINT, PATTERN and COPY.
- On-chip 32 Byte Pattern RAM.
- Conditional Drawing function (8 conditions) for Drawing Patterns, Color Mixing and Software Windowing.
- Drawing Area Control with Hardware Clipping and Hitting.
- Maximum Drawing Speed of 2 Million Logical Pixels per

- Second is the same for Monochrome and Color applications.
- \* High Resolution Display with Advanced Screen Control
  - Up to 4096 by 4096 Bit Map GRAPHIC Display and/or 256 Line by 256 Character by 32 Raster CHARACTER Display.
  - Separate Bit Map GRAPHIC (2M byte) and CHARACTER (128k byte) Address Spaces with Combined GRAPHIC/CHARACTER Display.
  - Three Horizontal Split Screens and One Window Screen. Size and Position Fully Programmable.
  - Independent Horizontal and Vertical Smooth Scroll for each Screen.
  - 1 to 16 Zoom Magnitude - Independent X and Y Zoom Factors.
  - Logical Pixel Specification as 1, 2, 4, 8 or 16 Bits for Monochrome, Gray Scale and Color Displays.
  - Programmable Address Increment Supports Frame Buffer Memory Widths to 256 Bits for Video Bit Rates > 500 MHz. (ACRTC R MASK is limit to 128 Bits.)
  - Unique Interleaved Access Mode for Screen Superimposition or 'Flashless' Displays.
  - ACRTC provides Dynamic RAM Refresh Address.
- \* High Performance MPU Interface
  - Optimized Interface with the HD68000 MPU and HD68450 DMAC.
  - 8 or 16 Bit Bus - Compatible With Other MPUs.
  - Separate on-chip 16 Byte READ and WRITE FIFOs.
  - Maskable Interrupts Including FIFO status.
- \* Versatile CRT Interface
  - Full Programmability of CRT Timing Signals.
  - Three Raster Scanning Modes.
  - Master or Slave Synchronization to Multiple ACRTCs or Other Video Generating Devices.
  - Two Hardware Cursors. Three Cursor Modes.
  - Programmable Cursor and Display Timing Skew.
  - Eight User Defineable Video Attributes.
  - Light Pen Detection.
- \* VLSI CMOS Process

OS or application software. At this layer, a number of popular standards have emerged including GKS, CORE, NAPLP, GSX and others.

Figure 21 shows how the ACRTC performs the key functions or logical drawing algorithm and physical drawing execution. Formerly, these function were performed by external hardware and/or MPU software.

As shown, the ACRTC reduces the 'gap' between device functionality and high level graphics procedures. Since the ACRTC device itself provides capabilities closely related to those of high level graphics packages, the effort (hardware and software design time and cost) required to develop a visual interface is significantly reduced.

Noting the traditional and emerging applications for visual interfaces, Figure 22 shows that a single ACRTC is suitable for a broad range of products in both alphanumeric and graphics areas.

Multiple ACRTCs can achieve performance beyond that of any first generation CRTC configuration.

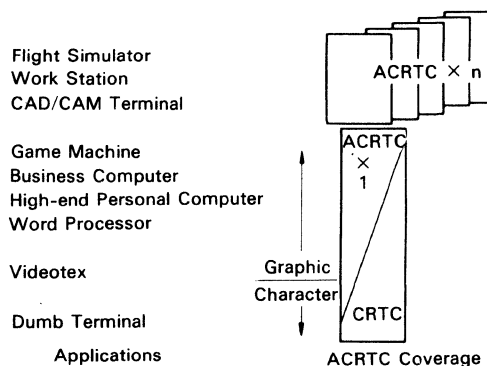


Figure 22 Application Spectrum

• APPLICATIONS

Drawing Procedures	MPU	MPU
Co-ordinates conversion		Software
Drawing Pre-process		Software
Drawing Process (Algorithms)	ACRTC	ACRTC
Drawing Execution		
Display Control	CRTC	CRTC
Synchronizing		
Signals Generation		
Others		

Figure 21 ACRTC vs. CRTC

The overall function of a visual interface is logically partitioned into layers. At the lowest layer are CRT timing and control signal generation. At the top layer are general purpose drawing procedures which provide a high-level interface to the users

• SYSTEM CONFIGURATION

Existing CRTCs provide a single bus interface to the frame buffer which must be shared with the host MPU. However, the refresh of large frame buffers and the requirement to access the frame buffer for drawing operations can quickly saturate this shared bus bandwidth.

As shown, the ACRTC uses separate host MPU and frame buffer bus interfaces. This allows the ACRTC full access to the frame buffer for display refresh, DRAM refresh and drawing operations while minimizing the ACRTCs usage of the MPU system bus. Thus, overall system performance is maximized. A related benefit is that a large frame buffer (2M byte for each ACRTC) is useable even if the host MPU has a smaller address space or segment size restriction.

The ACRTC can utilize an external DMA Controller. This increases system throughput when large amounts of command, parameter and data information must be transferred to the ACRTC. Also, advanced DMAC features, such as the HD68450 DMACs 'chaining' modes, can be used to develop powerful graphics system architectures.

However, more cost sensitive or less performance sensitive applications do not require a DMAC. The interface to the ACRTC can be handled completely under MPU software control.

While both ACRTC bus interfaces (Host MPU and Frame Buffer) exploit 16 bit data paths for maximum performance, the ACRTC also offers an 8 bit MPU mode for easy connection to popular 8 bit bus structures.



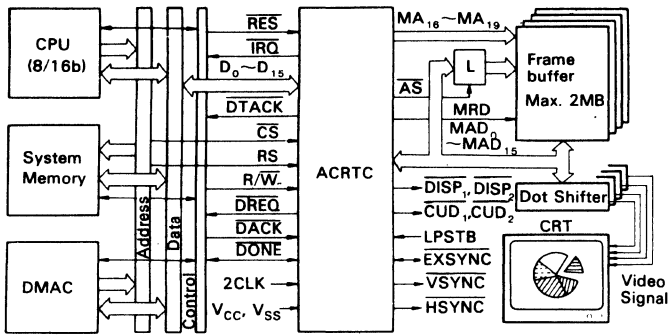


Figure 23 System Configuration

■ INTERNAL FUNCTIONS

● BLOCK DIAGRAM

The ACRTC consists of five major functional blocks. These functional blocks operate in parallel to achieve maximum performance. Two of the blocks perform the external bus interface for the host MPU and CRT respectively.

○ MPU Interface

Manages the asynchronous host MPU interface including the programmable interrupt control unit and DMA handshaking

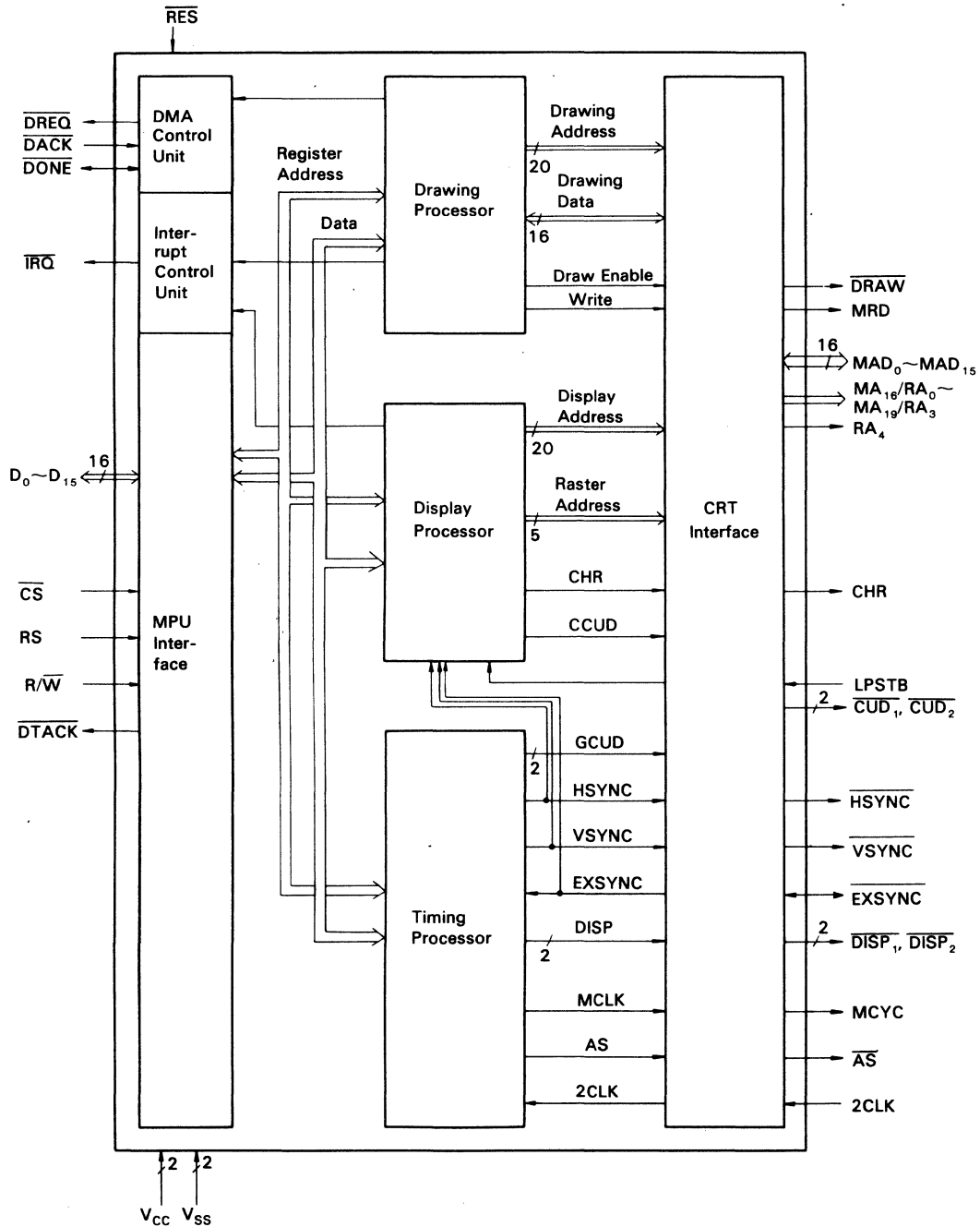


Figure 24 Block Diagram

- control unit.
- **CRT Interface**  
 Manages the frame buffer bus and CRT timing input and output control signals. Also, the selection of either display refresh address or drawing address outputs is performed.  
 The other three blocks are separately microprogrammed processors which operate in parallel to perform the major functions of drawing, display control and timing.
- **Drawing Processor**  
 Interprets commands and command parameters issued by the host bus (MPU and/or DMAC) and performs the drawing operations on the frame buffer memory. This processor is responsible for the execution of ACRTC drawing algorithms and conversion of logical pixel X-Y addresses to physical frame buffer addresses.  
 Communication with the host bus is via separate 16 byte read and write FIFOs.
- **Display Processor**  
 Manages frame buffer refresh addressing based on the user programmed specification of display screen organization. Combines and displays as many as 4 independent screen segments (3 horizontal splits and 1 window) using an internal high speed address calculation unit. Controls display refresh address outputs based on GRAPHIC (physical frame buffer address) or CHARACTER (physical frame buffer address + row address) display modes.
- **Timing Processor**  
 Generates the CRT synchronization signals and other timing signals used internally by the ACRTC.  
 The ACRTCs software visible registers are similarly partitioned and reside in the appropriate internal processor depending on function. The registers in the Display and Timing processors are loaded with basic display parameters during system initialization. During operation, the host primarily communicates with the ACRTCs Drawing processor via the on-chip FIFOs.

● **SIGNAL DESCRIPTION**

Following is a brief description of the ACRTC pin functions organized as MPU Interface, DMAC Interface, CRT Interface and Power Supply.

**MPU INTERFACE**

- $\overline{RES}$  – Input  
 Hardware reset input to the ACRTC.
- $D_0 \sim D_{15}$  – Input/Output  
 The bidirectional data bus for communication with the host MPU or DMAC. In 8 bit data bus mode,  $D_0 \sim D_7$  are used.
- $R/\overline{W}$  – Input  
 Controls the direction of host ↔ ACRTC transfers.
- $\overline{CS}$  – Input  
 Enables data transfers between the host and the ACRTC.
- $RS$  – Input  
 Selects the ACRTC register to be accessed and is normally connected to the least significant bit of the host address bus.
- $\overline{DTACK}$  – Output  
 Provides asynchronous bus cycle timing and is compatible with the HD68000 MPU  $\overline{DTACK}$  input.
- $\overline{IRQ}$  – Output  
 Generates interrupt service requests to the host MPU.

**DMAC INTERFACE**

- $\overline{DREQ}$  – Output  
 Generates DMA service requests to the host DMAC.
- $\overline{DACK}$  – Input  
 Receives DMA acknowledge timing from the host DMAC.
- $\overline{DONE}$  – Input/Output  
 Terminates DMA transfer and is compatible with the HD68450 DMAC  $\overline{DONE}$  signal.

**CRT INTERFACE**

- $2CLK$  – Input  
 Basic ACRTC operating clock derived from the dot clock.
- $MAD_0 \sim MAD_{15}$  – Input/Output  
 Multiplexed frame buffer address/data bus.
- $\overline{AS}$  – Output  
 Address strobe for demultiplexing the frame buffer address/data bus ( $MAD_0 \sim MAD_{15}$ ).
- $MA_{16}/RA_0 \sim MA_{19}/RA_3$  – Output  
 The high order address bits for graphic screens and the raster address outputs for character screens.
- $RA_4$  – Output  
 Provides the high order raster address bit (up to 32 rasters) for character screens.
- $\overline{CHR}$  – Output  
 Indicates whether a graphic or character screen is being accessed.
- $\overline{MCYC}$  – Output  
 Frame buffer memory access timing – one half the frequency of  $2CLK$ .
- $\overline{MRD}$  – Output  
 Frame Buffer data bus direction control.
- $\overline{DRAW}$  – Output  
 Differentiates between drawing cycles and CRT display refresh cycles.
- $\overline{DISP_1}, \overline{DISP_2}$  – Output  
 Programmable display enable timing used to selectively enable, disable and blank logical screens.
- $\overline{CUD_1}, \overline{CUD_2}$  – Output  
 Provides cursor timing determined by ACRTC programmed parameters such as cursor definition, cursor mode, cursor address, etc.
- $\overline{VSYNC}$  – Output  
 CRT device vertical synchronization pulse.
- $\overline{HSYNC}$  – Output  
 CRT device horizontal synchronization pulse.
- $\overline{EXSYNC}$  – Input/Output  
 For synchronization between multiple ACRTCs and other video signal generating devices.
- $\overline{LPSTB}$  – Input  
 Connection to an external light pen.

**VIDEO ATTRIBUTES**

The ACRTC outputs 20 bits of video attributes on  $MAD_0 \sim MAD_{15}$  and  $MA_{16}/RA_0 \sim MA_{19}/RA_3$ . These attributes are out-

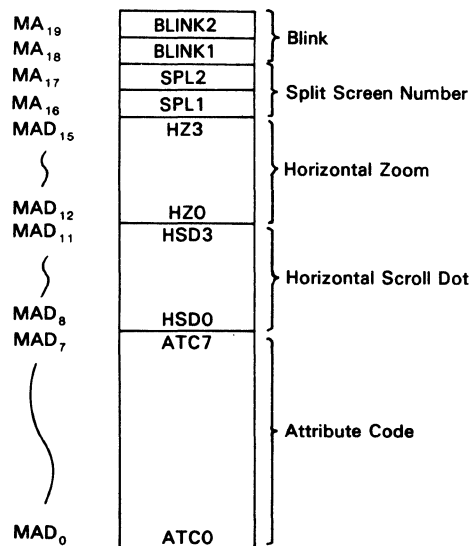


Figure 25 Video Attributes

put at the last cycle prior to the rising edge of  $\overline{\text{HSYNC}}$  and should be latched externally. Thus, video attributes can be set on a raster by raster basis.

**Attribute Code (ATC0~ATC7:MA<sub>D0</sub> ~ MA<sub>D7</sub>)**

These are user defined attributes. The programmed contents of the Attribute Control bits (ATR) of the Display Control Register (DCR) are output on these lines.

Note) The data written into ATR can be externally used after the completion of current raster scanning.

**Attribute Code (ATC7~ATC0) Application**

ATC is one of the function to provide the with application to the user and appropriate data need to be employed depending on the system requirement.

Followings show some of application example.

- (1) Amount of horizontal dot shift for window smooth scroll.
- (2) Horizontal width of crosshair cursor and the amount of horizontal dot shift (including Block cursor).
- (3) Frame buffer specification in blocks (used for the base register).
- (4) Back screen color or character color code.
- (5) Display screen selection during screen blink (used with SPL).
- (6) Interrupt vector address storage.
- (7) Polarity selection of horizontal/vertical synchronization signal etc.
- (8) Blinking signal like lamps used in the system.
- (9) Code storage (max. 8 bit) or selection signal etc.

**Horizontal Scroll Dot (HSD0~HSD3:MA<sub>D8</sub> ~ MA<sub>D11</sub>)**

These are used in conjunction with external circuitry to implement smooth horizontal scroll. These lines contain the encoded start dot address which is used to control the external shift register load timing and data. HSD usually corresponds to the start dot address of the background screens. However, if the window smooth scroll (SWS) bit of OMR (Operation Mode Register) is set to 1, HSD outputs the start dot address of the window screen segment.

Note) HSD outputs the valid value only within the specified raster area. Changing the register contents during the scanning does not cause any external effects, because the value loaded at the beginning of the area is reserved.

**Horizontal Zoom Factor (HZ0~HZ3:MA<sub>D12</sub> ~ MA<sub>D15</sub>)**

These lines output the encoded (1-16) horizontal zoom factor as stored in the Zoom Factor Register (ZFR). Horizontal zoom is

accomplished by the ACRTC repeating a single display address and using the HZ outputs to control the external shift register clock. Horizontal zoom can only be applied to the Base screen.

**Split Position (SPL1~SPL2:MA<sub>16</sub> ~ MA<sub>17</sub>)**

These lines present the encoded information showing the enabled background screen currently being displayed by the ACRTC.

SPL2	SPL1	
0	0	Background Screen not enabled or displayed
0	1	Base Screen
1	0	Upper Screen
1	1	Lower Screen

Even if the split screen display is prohibited, SPL is output if the area is specified.

**Blink (BLINK1~BLINK2:MA<sub>18</sub> ~ MA<sub>19</sub>)**

The lines alternate from high to low periodically as defined in the Blink Control Register (BCR). the blink frequency is specified in units of 4 field times. A field is defined as the period between successive VSYNC pulses. These lines are used to implement character and screen blink.

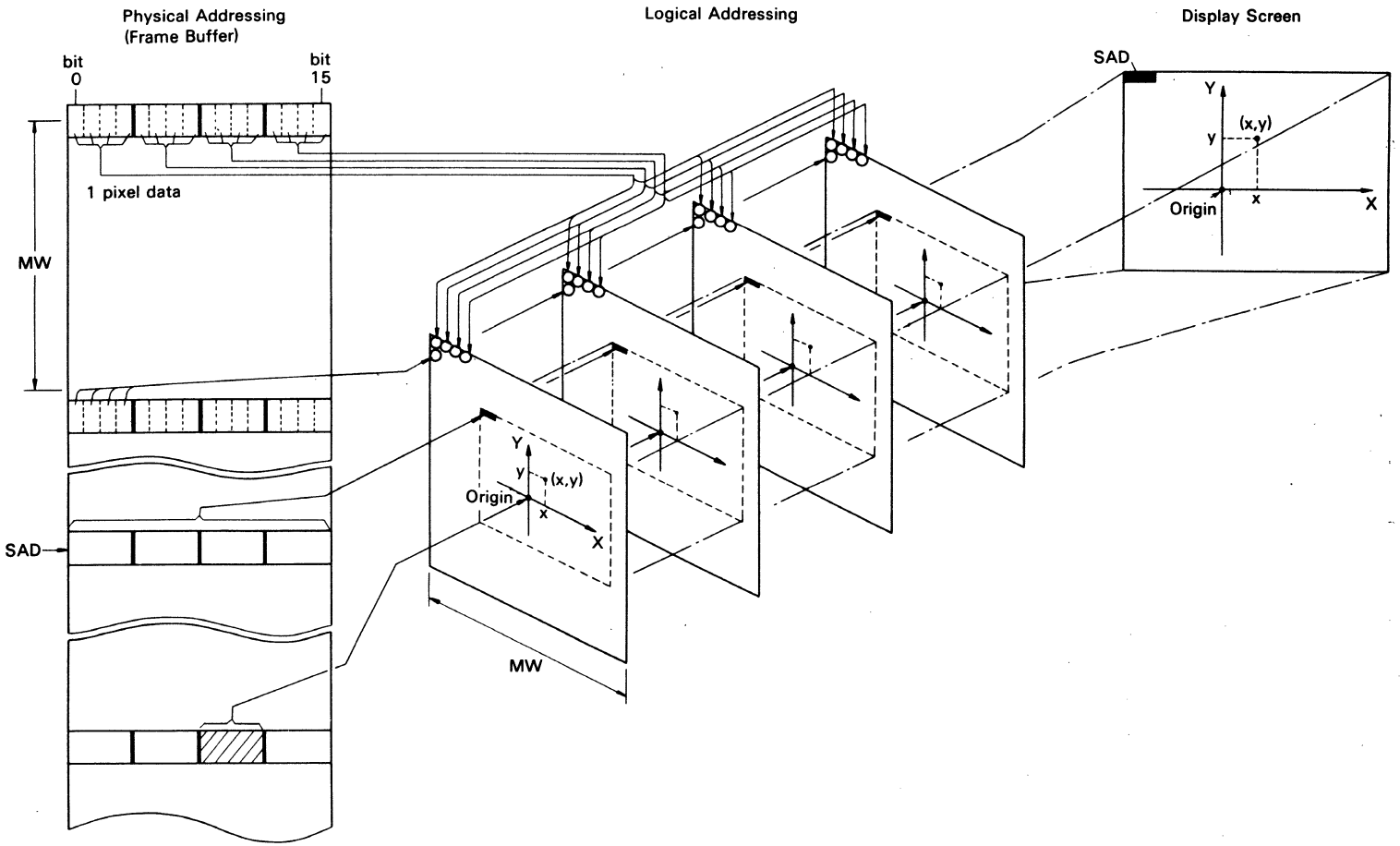
• **ADDRESS SPACE**

The ACRTC allows the host to issue commands using logical X-Y coordinate addressing. The ACRTC converts these to physical linear word addresses with bit field offsets in the frame buffer.

Figure 26 shows the relationship between a logical X-Y screen address and the frame buffer memory, organized as sequential 16 bit words. The host may specify that a logical pixel consists of 1, 2, 4, 8 or 16 physical bits in the frame buffer. In the example, 4 bits per logical pixel is used allowing 16 colors or tones to be selected.

Up to four logical screens (Upper, Base, Lower and Window) are mapped into the ACRTC physical address space. The host specifies a logical screen physical start address, logical screen physical memory width (number of memory words per raster), logical pixel physical memory width (number of bits per pixel) and the logical origin physical address. Then, logical pixel X-Y addresses issued by the host or by the ACRTC Drawing processor are converted to physical frame buffer addresses. The ACRTC also performs bit extraction and masking to map logical pixel operations (in the example, 4 bits) to 16 bit word frame buffer accesses.

Figure 26 Logical/Physical Addressing



• REGISTERS

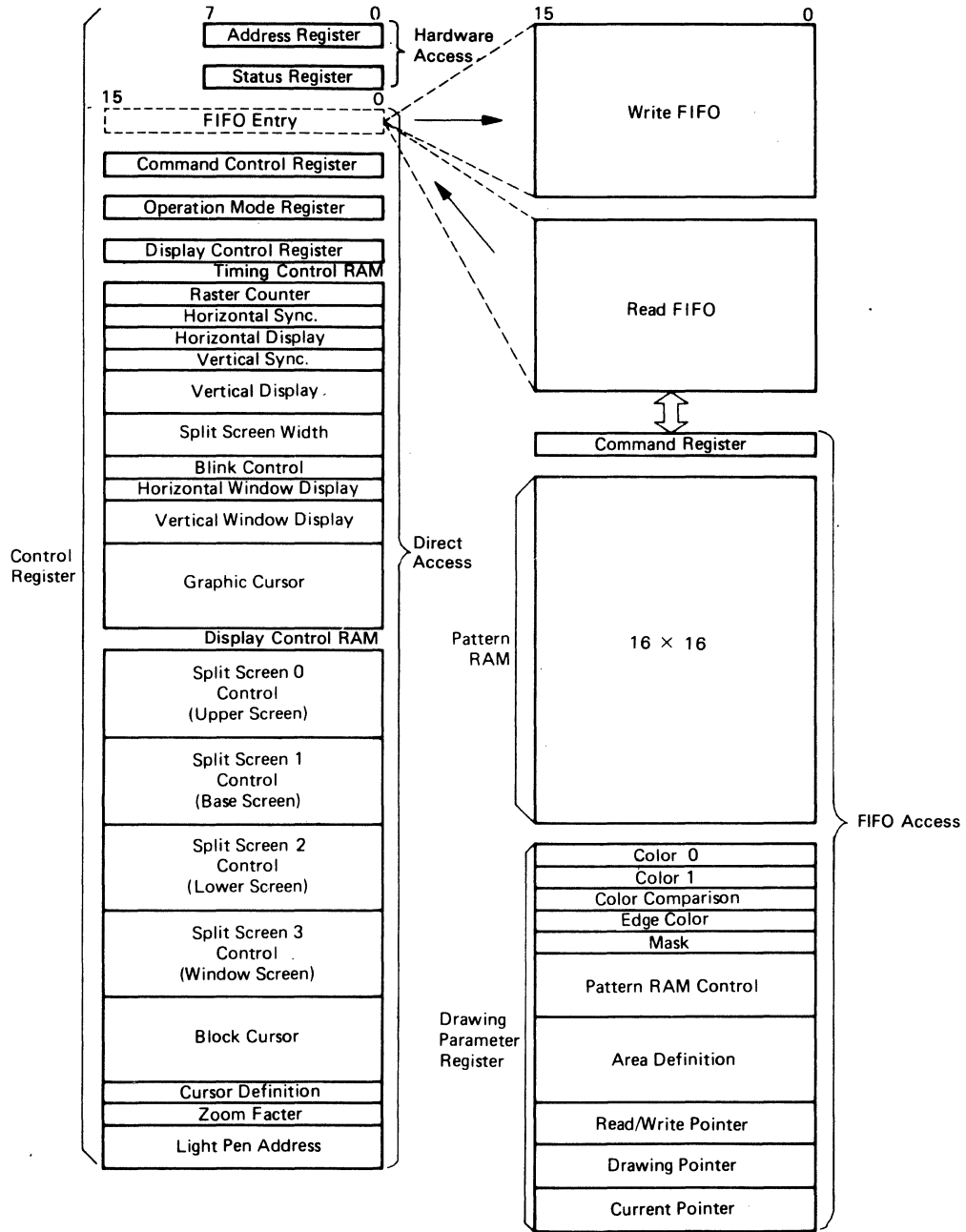


Figure 27 Programming Model

Table 1 Programming Model (Hardware Access, Direct Access Registers)

CS	RS	RW	Reg. No.	Register Name	Abbr.	DATA (H)								DATA (L)								
						15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
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	0	r00	FIFO Entry	FE	F E																
1	0	0	r02	Command Control	CCR	ABT	PSE	DDM	CDM	DRC	GBM	CRE	ARE	CEE	LPE	RFE	RRE	WRE	WEE			
1	0	0	r04	Operation Mode	OMR	M/S	STR	ACP	WSS	CSK	DSK	RAM	GAI	ACM	RSM							
1	0	0	r06	Display Control	DCR	DSP	SE1	SE0	SE2	SE3	A T R											
-	-	-	r08	(undefined)	-	-----																
1	0	0	r80	Raster Count	RCR	R C																
1	0	0	r82	Horizontal Sync.	HSR	H C								H S W								
1	0	0	r84	Horizontal Display	HDR	H D S								H D W								
1	0	0	r86	Vertical Sync.	VSR	V C																
1	0	0	r88	Vertical Display	VDR	V D S								V S W								
1	0	0	r8A	Split Screen Width	SSW	S P 1																
1	0	0	r8C			S P 0																
1	0	0	r8E			S P 2																
1	0	0	r90	Blink Control	BCR	BON1				BOFF1				BON2				BOFF2				
1	0	0	r92	Horizontal Window Display	HWR	H W S								H W W								
1	0	0	r94	Vertical Window Display	VWR	V W S																
1	0	0	r96			V W W																
1	0	0	r98	Graphic Cursor	GCR	C X E								C X S								
1	0	0	r9A			C Y S																
1	0	0	r9C			C Y E																
-	-	-	r9E	(undefined)	-	-----																
-	-	-	rA0	(undefined)	-	-----																
-	-	-	rBE	(undefined)	-	-----																
0	1	0	rC0	Raster Addr.0	RAR0	L R A 0								F R A 0								
1	0	0	rC2	Upper Memory Width 0	MWR0	CHR	M W 0								S A 0 L							
1	0	0	rC4	Screen	Start Addr.0	SAR0	S D A 0								S A 0 L							
1	0	0	rC6				S A 0 L								S A 0 L							
1	0	0	rC8	Raster Addr.1	RAR1	L R A 1								F R A 1								
1	0	0	rCA	Base Memory Width 1	MWR1	CHR	M W 1								S A 1 L							
1	0	0	rCC	Screen	Start Addr. 1	SAR1	S D A 1								S A 1 L							
1	0	0	rCE				S A 1 L								S A 1 L							
1	0	0	rD0	Raster Addr.2	RAR2	L R A 2								F R A 2								
1	0	0	rD2	Lower Memory Width 2	MWR2	CHR	M W 2								S A 2 L							
1	0	0	rD4	Screen	Start Addr.2	SAR2	S D A 2								S A 2 L							
1	0	0	rD6				S A 2 L								S A 2 L							
1	0	0	rD8	Raster Addr.3	RAR3	L R A 3								F R A 3								
1	0	0	rDA	Window Memory Width 3	MWR3	CHR	M W 3								S A 3 L							
1	0	0	rDC	Screen	Start Addr.3	SAR3	S D A 3								S A 3 L							
1	0	0	rDE				S A 3 L								S A 3 L							
1	0	0	rE0	Block Cursor 1	BCUR1	B C W 1				B C S R 1				B C A 1				B C E R 1				
1	0	0	rE2			B C W 2				B C S R 2				B C A 2				B C E R 2				
1	0	0	rE6	Block Cursor 2	BCUR2	B C W 2				B C S R 2				B C A 2				B C E R 2				
1	0	0	rE8			B C W 2				B C S R 2				B C A 2				B C E R 2				
1	0	0	rE8	Cursor Definition	CDR	C M	CON1				COFF1				CON2				COFF2			
1	0	0	rEA	Zoom Factor	ZFR	H Z F				V Z F												
1	0	0	rEC	Light Pen Address	LPAR	L P A L								L P A H								
1	0	0	rEE			L P A L								L P A H								
1	0	0	rF0	(undefined)	-	-----																
-	-	-	rFE	(undefined)	-	-----																

Note : 1 ..... "High" level  
0 ..... "Low" level

ABT	: Abort	SPO, SP1, SP2	: Split Screen 0 Width, Split Screen 1 Width, Split Screen 2 Width
ACM	: Access Mode	BON1, BON2	: Blink ON 1, Blink ON 2
ACP	: Access Priority	BOFF1, BOFF2	: Blink OFF 1, Blink OFF 2
Address	: Register No. of the control register	HWS	: Horizontal Window Start
ARD	: Area Detect	HWW	: Horizontal Window Width
ARE	: Area Detect Interrupt Enable	VWS	: Vertical Window Start
ATR	: Attribute Control	VWW	: Vertical Window Width
CDM	: Command DMA Mode	CXS, CYS	: Cursor X Start, Cursor Y Start
CED	: Command End	CXE, CYE	: Cursor X End, Cursor Y End
CEE	: Command End Interrupt Enable	FRA	: First Raster Address
CER	: Command Error	LRA	: Last Raster Address
CRE	: Command Error Interrupt Enable	CHR	: Character
CSK	: Cursor Display Skew	MW	: Memory Width
DDM	: Data DMA Mode	SDA	: Start Dot Address
DRC	: DMA Request Control	SAH/SRA	: Start Address "High"/Start Raster Address
DSK	: DISP Skew	SAL	: Start Address "Low"
DSP	: DISP Signal Control	BCW1, BCW2	: Block Cursor Width 1, Block Cursor Width 2
FE	: FIFO Entry	BCSR1, BCSR2	: Block Cursor Start Raster 1, Block Cursor Start Raster 2
GAI	: Graphic Address Increment Mode	BCER1, BCER2	: Block Cursor End Raster 1, Block Cursor End Raster 2
GBM	: Graphic Bit Mode	BCA1, BCA2	: Block Cursor Address 1, Block Cursor Address 2
HC	: Horizontal Cycle	CM	: Cursor Mode
HDS	: Horizontal Display Start	CON1, CON2	: Cursor ON 1, Cursor ON 2
HDW	: Horizontal Display Width	COFF1, COFF2	: Cursor OFF 1, Cursor OFF 2
HSW	: Horizontal Sync. Width	HZF, VZF	: Horizontal Zoom Factor, Vertical Zoom Factor
LPD	: Light Pen Strobe Detect	LPAH	: Light Pen Address "High"
LPE	: Light Pen Strobe Interrupt Enable	LPAL	: Light Pen Address "Low"
M/S	: Master/Slave		
PSE	: Pause		
RAM	: RAM Mode		
RC	: Raster Count		
RFE	: Read FIFO Full Interrupt Enable		
RFF	: Read FIFO Full		
RFR	: Read FIFO Ready		
RRE	: Read FIFO Ready Interrupt Enable		
RSM	: Raster Scan Mode		
SE0	: Split Enable 0		
SE1	: Split Enable 1		
SE2	: Split Enable 2		
SE3	: Split Enable 3		
STR	: Start		
VC	: Vertical Cycle		
VDS	: Vertical Display Start		
VSW	: Vertical Sync. Width		
WEE	: Write FIFO Empty Interrupt Enable		
WFE	: Write FIFO Empty		
WFR	: Write FIFO Ready		
WRE	: Write FIFO Ready Interrupt Enable		
WSS	: Window Smooth Scroll		

Table 1 (cont.) Programming Model (Drawing Parameter Registers)

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

■ ... Always set to "0"

\*\* ..... Set binary complements for negative values of X and Y axis.

DRAWING PARAMETER REGISTER

- R : Register which can be read by Read Parameter Register Command (RPR)
- W : Register which can be written into by Write Parameter Register Command (WPR)
- : Access is not allowed
- CLO : Defines the color data used for the drawing when logical drawing data=0
- CL1 : Defines the color data used for the drawing when logical drawing data=1
- CCMP : Defines the comparative color of the drawing operation
- EDG : Defines the edge color
- MASK : Defines the bit pattern used to mask bits upon which data transfer should not be performed
- PSX, PSY : Pattern Start Point
- PEX, PEY : Pattern End Point
- PPX, PPY : Pattern Scan Start Point
- PZX, PZY : Pattern Zoom
- PZCX, PZCY : Pattern Zoom Count
- XMIN, YMIN : Start point of Area definition
- XMAX, YMAX : End point of Area definition
- DN : Screen Number
- RWPH : High-order 8 bit of Read Write Pointer Address
- RWPL : Low-order 12 bit of Read Write Pointer Address
- DPAH : High-order 8 bit of Drawing Pointer Address
- DPAL : Low-order 12 bit of Drawing Pointer Address
- DPD : Drawing Pointer Dot Address
- X, Y : Position indicated by Current Pointer on X-Y coordinate



The ACRTC has over two hundred bytes of accessible registers. These are organized as Hardware, Directly and FIFO accessible.

○ **Hardware Accessible**

The ACRTC is connected to the host MPU as a standard peripheral which occupies two word locations of the host address space. The RS (Register Select) pin selects one of these two locations. When RS is low, reads access the Status Register and writes access the Address Register.

The Status Register summarizes the ACRTC state and is used by the MPU to monitor the overall operation of the ACRTC. The Address Register is used to program the ACRTC with the address of the specific directly accessible register which the MPU wishes to access.

○ **Directly Accessible**

These registers are accessed by prior loading of the Address Register with the chosen register address. Then, when the MPU accesses the ACRTC with RS=1, the chosen register is accessed.

The FIFO entry enables access to FIFO accessible registers using the ACRTC read and write FIFOs.

The Command Control Register is used to control overall ACRTC operation such as aborting or pausing commands, defining DMA protocols, enabling/disabling interrupt sources, etc.

The Operation Mode Register defines basic parameters of ACRTC operation such as frame buffer access mode, display or drawing priority, cursor and display timing skew factors, raster scan mode, etc.

The Display Control Register allows the independent enabling and disabling of each of the four ACRTC logical display screens (Base, Upper, Lower and Window). Also, this register contains the 8 bits of user defineable video attributes.

The Timing Control RAM contains registers which define ACRTC timing. This includes timing specification for CRT control signals (e.g. HSYNC, VSYNC), logical display screen size and display period, blink timing, etc.

The Display Control RAM contains registers which define logical screen display parameters such as start addresses, raster addresses and memory width. Also included are the cursor(s) definition, zoom factor and light pen registers.

○ **FIFO Accessible**

For high performance drawing, key Drawing Processor registers are coupled to the host via the ACRTCs separate 16 byte read and write FIFOs.

ACRTC commands are sent from the MPU via the write FIFO to the Command register. As the ACRTC completes command execution, the next command is automatically fetched from the FIFO into the Command register.

The Pattern RAM is used to define drawing and painting 'patterns'. The Pattern RAM is accessed using the ACRTCs Read Pattern RAM (RPTN) and Write Pattern RAM (WPTN) register access commands.

The Drawing Parameter Registers define detailed parameters of the drawing process, such as color control, area control (hitting/clipping) and Pattern RAM pointers. The Drawing Parameter Registers are accessed using the ACRTCs Read Parameter Register (RPR) and Write Parameter Register (WPR) register access commands.

■ **COMMANDS**

The ACRTC has 38 commands classified into three groups — REGISTER ACCESS, DATA TRANSFER and GRAPHIC DRAWING.

Five REGISTER ACCESS commands allow access to Drawing processor Drawing Parameter Registers and the Pattern RAM.

Ten DATA TRANSFER commands are used to move data between the host system memory and the frame buffer, or within the frame buffer.

Twenty three GRAPHIC DRAWING commands cause the ACRTC to perform drawing operations. Parameters for these commands are specified using logical X-Y addressing.

All the above commands, parameters and data are transferred via the ACRTC read and write FIFOs.

Assuming the ACRTC has been properly initialized, the MPU must perform two steps to cause graphic drawing.

First, the MPU must specify certain drawing parameters which define a number of details associated with the drawing process. For example, to draw a figure or paint an area, the MPU must specify the drawing or painting 'pattern' by initializing the ACRTC Pattern RAM and related pointers. Also, if clipping and hitting control are desired, the MPU specifies the 'area' to be monitored during drawing by initializing area definition registers. Other drawing parameters include color, edge definition, etc.

After the drawing parameters have been specified, the MPU issues a graphic drawing command and any required command parameters, such as the CRCL (Circle) command with a radius parameter. The ACRTC then performs the specified drawing operation by reading, modifying and rewriting the contents of the frame buffer.

Table 2 . ACRTC Command Table

TYPE	MNEMONIC	COMMAND NAME	OPERATION CODE	PARAMETER	# (words)	~ (cycles) *3)
Register Access Command	ORG	Origin	0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0	DPH DPL	3	8
	WPR	Write Parameter Register	0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0	RN	2	6
	RPR	Read Parameter Register	0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0	RN	1	6
	WPTN	Write Pattern RAM	0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0	PRA n D <sub>1</sub> , ..., D <sub>n</sub>	n+2	4n+8
	RPTN	Read Pattern RAM	0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0	PRA n	2	4n+10
Data Transfer Command	DRD	DMA Read	0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0	AX AY	3	(4x+8)y+12[x·y/8]+(62~68)
	DWT	DMA Write	0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0	AX AY	3	(4x+8)y+16[x·y/8]+34
	DMOD	DMA Modify	0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0	AX AY	3	(4x+8)y+16[x·y/8]+34
	RD	Read	0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0		1	12
	WT	Write	0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0	D	2	8
	MOD	Modify	0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0	MM D	2	8
	CLR	Clear	0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0	D AX AY	4	(2x+8)y+12
	SCLR	Selective Clear	0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0	MM D AX AY	4	(4x+6)y+12
	CPY	Copy	0 1 1 0 0 S DSD 0 0 0 0 0 0 0 0 0 0	SAH SAL AX AY	5	(6x+10)y+12
	SCPY	Selective Copy	0 1 1 1 0 S DSD 0 0 0 0 0 0 0 0 0 0	SAH SAL AX AY	5	(6x+10)y+12
Graphic Command	AMOVE	Absolute Move	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	X Y	3	56
	RMOVE	Relative Move	1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0	dX dY	3	56
	ALINE	Absolute Line	1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0	AREA:COL:OPM X Y	3	P·L+18
	RLINE	Relative Line	1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0	AREA:COL:OPM dX dY	3	P·L+18
	ARCT	Absolute Rectangle	1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0	AREA:COL:OPM X Y	3	2P(A+B)+54
	RRCT	Relative Rectangle	1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0	AREA:COL:OPM dX dY	3	2P(A+B)+54
	APLL	Absolute Polyline	1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0	AREA:COL:OPM n X <sub>1</sub> , Y <sub>1</sub> , ..., X <sub>n</sub> , Y <sub>n</sub>	2n+2	Σ[P·L+16]+8
	RPLL	Relative Polyline	1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0	AREA:COL:OPM n dX <sub>1</sub> , dY <sub>1</sub> , ..., dX <sub>n</sub> , dY <sub>n</sub>	2n+2	Σ[P·L+16]+8
	APLG	Absolute Polygon	1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0	AREA:COL:OPM n X <sub>1</sub> , Y <sub>1</sub> , ..., X <sub>n</sub> , Y <sub>n</sub>	2n+2	Σ[P·L+16]+P·Lo+20
	RPLC	Relative Polygon	1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0	AREA:COL:OPM n dX <sub>1</sub> , dY <sub>1</sub> , ..., dX <sub>n</sub> , dY <sub>n</sub>	2n+2	Σ[P·L+16]+P·Lo+20
	CRCL	Circle	1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0	AREA:COL:OPM r	2	8d+66
	ELPS	Ellipse	1 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0	AREA:COL:OPM a b dX	4	10d+90
	AARC	Absolute Arc	1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0	AREA:COL:OPM dXc Yc dXe Ye	5	8d+18
	RARC	Relative Arc	1 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0	AREA:COL:OPM dXc dYc dXe dYe	5	8d+18
	AEARC	Absolute Ellipse Arc	1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0	AREA:COL:OPM a b dXc Yc dXe Ye	7	10d+96
	REARC	Relative Ellipse Arc	1 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0	AREA:COL:OPM a b dXc dYc dXe dYe	7	10d+96
	AFRCT	Absolute Filled Rectangle	1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	AREA:COL:OPM X Y	3	(P·A+B)B+18
	RFRC	Relative Filled Rectangle	1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0	AREA:COL:OPM dX dY	3	(P·A+B)B+18
	PAINT	Paint	1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0	AREA:COL:OPM	1	(18A+102)B-58 *1)
	DOT	Dot	1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0	AREA:COL:OPM	1	8
PTN	Pattern	1 1 0 1 0 S L SD	AREA:COL:OPM SZ	*2)	(P·A+10)B+20	
AGCPY	Absolute Graphic Copy	1 1 1 0 0 S DSD	AREA:COL:OPM Xs Ys DX DY	5	((P+2)A+10)B+70	
RGCPY	Relative Graphic Copy	1 1 1 1 0 S DSD	AREA:COL:OPM dXs dYs DX DY	5	((P+2)A+10)B+70	

\*1) In case of rectangular filling

\*2) SZ:  $\begin{matrix} 15 & 87 & 0 \\ \text{SZy} & \text{SZx} & \end{matrix}$  SZy, SZx: Pattern Size

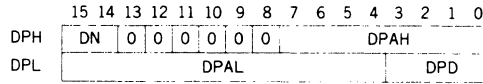
n: number of repetition X/Y: drawing words of X-direction/Y-direction  
 L/Lo/d: sum of drawing dots A/B: drawing dots of main/sub direction  
 E: [E=0 (Stop at Edge color), E=1 (Stp at excepting Edge color)] C: [C=1 (clockwise), C=0 (reverse)]  
 [↑]: rounding up  
 P = 4: OPM-000~011  
 6: OPM-100~111

\*3) cycles: 2clock cycle time

**REGISTER ACCESS COMMAND**

Mnemonic	Operation Code	Parameter	#(words)	~ (cycles)
ORG	0 0 0 0:0 1:0 0:0 0 0 0:0 0 0 0	DPH DPL	3	8
WPR	0 0 0 0:1 0:0 0:0 0 0	RN D	2	6
RPR	0 0 0 0:1 1:0 0:0 0 0	RN	1	6
WPTN	0 0 0 1:1 0:0 0:0 0 0	PRA n D <sub>1</sub> .....D <sub>n</sub>	n+2	4n+8
RPTN	0 0 0 1:1 1:0 0:0 0 0	PRA n	2	4n+10

RN : Register number of the drawing parameter register(\$0-\$13)  
 PRA : Pattern RAM address at which Read/Write operation starts(\$0-\$F)  
 DPH : Drawing pointer register High word  
 DPL : Drawing pointer register Low word



DPAH : Higher 8 bits of Drawing Pointer address  
 DPAL : Lower 12 bits of Drawing Pointer address  
 DPD : Dot position in the memory address

D, D<sub>1</sub>,....., D<sub>n</sub> : Write data  
 n : Number of Read/Write data

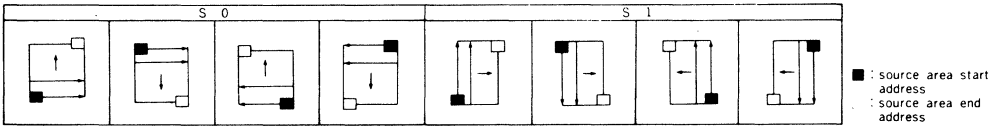
DN	Screen No.
00	Upper Screen
01	Base Screen
10	Lower Screen
11	Window Screen

**DATA TRANSFER COMMAND**

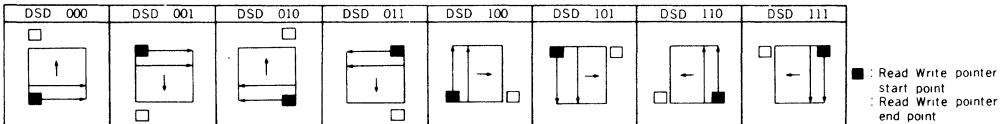
MM : Modify Mode

MM	Function
00	Replace Replace drawing point data with modifier information
01	OR OR drawing point data with modifier data and rewrite the result data to the frame buffer
10	AND AND drawing point data with modifier data and rewrite the result data to the frame buffer
11	EOR EOR drawing point data with modifier data and rewrite the result data to the frame buffer

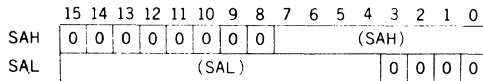
S : Source Scan Direction



DSD : Destination Scan Direction



AX : Number of word in X-axis direction - 1  
 AY : Number of word in Y-axis direction - 1  
 D : Write data  
 SAH : Source Start Address High word  
 SAL : Source Start Address Low word



(SAH) : Memory address Higher 8 bits  
 (SAL) : Memory address Lower 12 bits

x : Number of word in X-axis direction  
 y : Number of word in Y-axis direction  
 ↑ : Rounding up

**GRAPHIC DRAWING COMMAND**

AREA : Area Mode  
 COL : Color Mode  
 OPM : Operation Mode

C : Circling Direction

C	Direction
0	Counterclockwise
1	Clockwise

E : Definition of Edge color

E	Definition
0	Edge color is defined by the data in the edge color register.
1	Edge color is defined by the data excluding the above.

SL : Slant, SD : Scan Direction

SL \ SD	000	001	010	011	100	101	110	111
0								
1								

● : current pointer start point  
 ○ : current pointer end point

S : Source Scan Direction

S	0	1
0		

● : source area start dot position  
 ○ : source area end dot position

DSD : Destination Scan Direction

DSD	000	001	010	011	100	101	110	111
0								

● : current pointer start point  
 ○ : current pointer end point

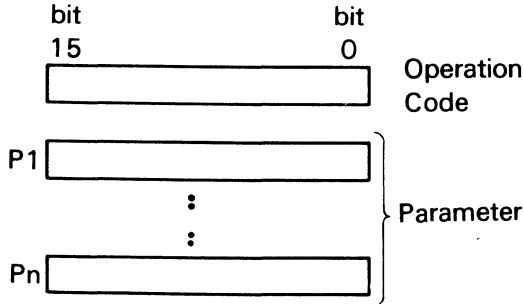
- X, X1, ..., Xn : Absolute X-address from the origin point
- Y, Y1, ..., Yn : Absolute Y-address from the origin point
- dX : Relative X-address from the current pointer
- dY : Relative Y-address from the current pointer
- n : Number of nodes
- dX1, ..., dXn : Relative X-address from each node
- dY1, ..., dYn : Relative Y-address from each node
- r : Dot number on radius
- a, b :  $(DX)^2 : (DY)^2 = a : b$
- DX : X-direction dot number
- DY : Y-direction dot number
- Xc : Absolute X-address of the center point of arc/ellipse
- Yc : Absolute Y-address of the center point of arc/ellipse
- dXc : Relative X-address from the current pointer to the center point of arc/ellipse
- dYc : Relative Y-address from the current pointer to the center point of arc/ellipse
- Xe : Absolute X-address of the end point of arc/ellipse
- Ye : Absolute Y-address of the end point of arc/ellipse
- dXe : Relative X-address from the current pointer to the end point of arc/ellipse
- dYe : Relative Y-address from the current pointer to the end point of arc/ellipse
- Xs : Absolute X-address of the start dot position in source area
- Ys : Absolute Y-address of the start dot position in source area
- dXs : Relative X-address from the current pointer to the start dot position in source area
- dYs : Relative Y-address from the current pointer to the start dot position in source area
- P :  $4(OPM=000\sim011)/6(OPM=100\sim111)$
- L, L0 : Dot number on straight line
- d : total dot number
- A : Scan main direction dot number
- B : Scan sub direction dot number

**• COMMAND FORMAT**

ACRTC commands consist of a 16 bit op-code, optionally followed by 1 or more 16 bit parameters. When 8 bit MPU mode is used, commands, parameters and data are sent to and from the ACRTC in the order of high byte, low byte.

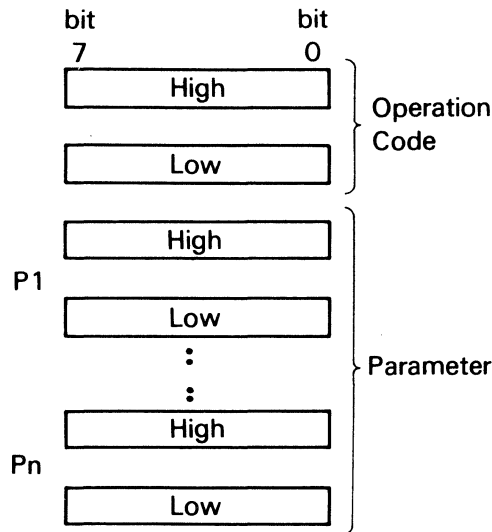
(a) 16 bit interface

In the case of 16 bit interface, first move the 16 bit operation code and then move necessary 16 bit parameters one by one.



(b) 8 bit interface

In the case of 8-bit interface, first move the operation code's high byte followed by low byte and then move those of parameters in the same order.



**PROGRAM TRANSFER**

Program Transfer occurs when the MPU specifies the FIFO

entry address and then writes commands/parameters to the write FIFO under program control (RS = high, R/W, CS = low). The MPU writes are normally synchronized with ACRTC FIFO status by software polling or interrupts.

- Software Polling (WFR, WFE interrupts disabled)
    - a) MPU program checks the SR (Status Register) for Write FIFO Ready (WFR) flag = 1, and the writes 1-word op-code/parameters.
    - b) MPU program checks the SR (Status Register) for Write FIFO Empty (WFE) flag = 1, and then writes 1 to 8-word op-code/parameters.
  - Interrupt Driven (WFR, WFE interrupts enabled)
    - a) MPU WFR interrupt service routine writes 1-word op-code/parameters.
    - b) MPU WFE interrupt service routine writes 1 to 8-word op-code/parameters.
- In the specific case of Register Access Commands and an initially empty write FIFO, MPU writes need not be synchronized to the write FIFO status. The ACRTC can fetch and execute these commands faster than the MPU can issue them.

**COMMAND DMA TRANSFER**

Commands and parameters can be transferred from MPU system memory using in external DMAC. The MPU initiates and terminates Command DMA Transfer mode under software control (CDM bit of CCR). Command DMA can also be terminated by assertion of the ACRTC DONE signal. DONE is treated as an input in Command DMA Transfer Mode.

Using Command DMA Transfer, the ACRTC will issue cycle stealing DMA requests to the DMAC when the write FIFO is empty. The DMA data is automatically sent from system memory to the ACRTC write FIFO regardless of the contents of the Address Register.

- Note) • Make sure that the write FIFO is empty and all the commands are terminated before starting the Command DMA Transfer.
- The Data DMA Command cannot be executed in the Command DMA Transfer Mode.
  - In the R mask and S mask version, the Command DMA Transfer is not in use.

**• REGISTER ACCESS COMMANDS**

Registers associated with the Drawing processor (Pattern RAM and Drawing Parameter Registers) are accessed through the read and write FIFOs using the Register Access Commands.

**• DATA TRANSFER COMMANDS**

Data Transfer Commands are used to move blocks of data between the MPU system memory and the ACRTC frame buffer or within the frame buffer itself. Before issuing these commands, a physical 20 bit frame buffer address must be specified in the RWP (Read Write Pointer) Drawing Parameter Register.

Table 3 Register Access Commands

Command	Function
ORG	Initialize the relation between the origin point in the X-Y coordinates and the physical address.
WPR	Write into the parameter register
RPR	Read the parameter register
WPTN	Write into the pattern RAM
RPTN	Read the pattern RAM

Table 4 . Data Transfer Commands

Command	Function
DRD	Transfer data, by DMA transfer, from the frame buffer to the MPU system memory.
DWT	Transfer data, by DMA transfer, from the MPU system memory to the frame buffer.
DMOD	Transfer data, by DMA transfer, from the MPU system to the frame buffer subject to logical modification. (bit maskable)
RD	Read one word of data from the frame buffer specified by the read/write pointer (RWP), and load the word into Read FIFO.
WT	Write one word of data to the frame buffer specified by the read/write pointer (RWP).
MOD	Perform logical operation on one word in the frame buffer specified by the read/write pointer (RWP). (bit maskable)
CLR	Clear a rectangular area of the frame buffer with a data in the command parameter.
SCLR	Initialize a rectangular area of the frame buffer with 1-word data subject to logical operation. (bit maskable)
CPY	Copy frame buffer data from one area (source area) to another area (destination area) specified by the read/write pointer (RWP).
SCPY	Copy frame buffer data from one area (source area) to another area (destination area) subject to logical modification by word. The source and destination areas must reside on the same screen. (bit maskable)

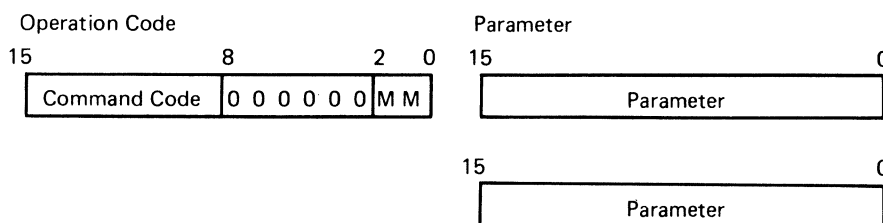


Figure 28 Data Transfer Command Format

**MODIFY MODE**

The DMOD, MOD, SCLR and SCPY commands allow 4 types of bit level logical operations to be applied to frame buffer data. The modify mode is encoded in the lower two bits (MM) of

these op-codes. The bit positions within each frame buffer word to be modified are selectable using the mask register (MASK). Bits set to 1 are modifiable, ones to 0 are masked and not modifiable.

MM	Modify Mode
0 0	REPLACE frame buffer data with command parameter data.
0 1	OR frame buffer data with command parameter data and rewrite to the frame buffer.
1 0	AND frame buffer data with command parameter data and rewrite to the frame buffer.
1 1	EOR frame buffer data with command parameter data and rewrite to the frame buffer.

• **GRAPHIC DRAWING COMMANDS**

The ACRTC has 23 separate graphic drawing commands. Graphic drawing is performed by modifying the contents of the frame buffer based upon microcoded drawing algorithms in the ACRTC drawing processor.

Most coordinate parameters for graphic drawing commands are specified using logical pixel X-Y addressing. The complex task of translating a logical pixel address to a linear frame buffer word address, and further selecting the appropriate sub-field of the word (for example, a given logical pixel in 4 bits per logical pixel mode might reside in bits 8-11 of a frame buffer word) is performed at high speed by ACRTC hardware.

Many instructions allow specification of X-Y coordinates with either absolute or relative X-Y coordinates (e.g. ALINE and RLINE). In both cases, twos complement numbers are used to represent positive and negative values.

(a) **Absolute Coordinate Specification**

The screen address (X, Y) is specified in units of logical pixels relative to an origin point defined with the ORG command.

(b) **Relative Coordinate Specification**

The screen address (dX,dY) is specified in units of logical pixels relative to the current drawing pointer (CP) position.

A graphic drawing command consists of a 16 bit op-code and optionally 0 to 64k 16 bit parameters.

The 16 bit op-code consists of an 8 bit command code, an AREA Mode specifier (3 bits), a Color Mode specifier (2 bits) and an Operation Mode specifier (3 bits).

The Area Mode allows versatile clipping and hitting detection. A drawing area can be defined, and should drawing operations attempt to enter or leave that area, a number of programmable actions can be taken by the ACRTC.

The Color Mode determines whether the Pattern RAM is used indirectly to select Color Registers or is directly used as the color information.

The Operation Mode defines one of eight logical operations to be performed between the frame buffer read data and the color data in the Pattern RAM to determine the drawing data to be rewritten into the frame buffer.

(i) **Absolute Coordinate Specification**  
 Specifies the addresses (x, y) based on the origin point set by the ORG command.

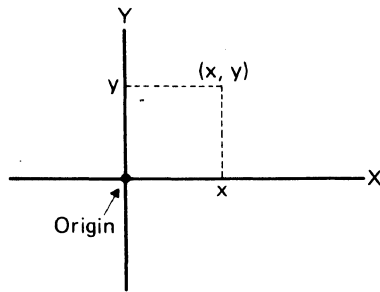


Figure 29 Absolute Coordinate Specification

(ii) **Relative Coordinate Specification**  
 Specifies the relative addresses ( $\Delta x$ ,  $\Delta y$ ) related to the current drawing point.

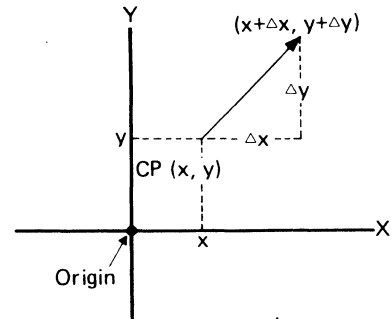


Figure 30 Relative Coordinate Specification

Table 5 Graphic Drawing Commands

Command	Function
AMOVE	Move the Current Pointer (CP) to an absolute logical pixel X-Y address.
RMOVE	Move the Current Pointer (CP) to a relative logical pixel X-Y address.
ALINE	Draw a straight line from the Current Pointer (CP) to a command specified end point of the absolute coordinates.
RLINE	Draw a straight line from the Current Pointer (CP) to a command specified end point of the relative coordinates.
ARCT	Draw a rectangle defined by the Current Pointer (CP) and a command specified diagonal point of the absolute coordinates.
RRCT	Draw a rectangle defined by the Current Pointer (CP) and a command specified diagonal point of the relative coordinates.
APLL	Draw a polyline (multiple contiguous segments) from the Current Pointer (CP) through command specified points of the absolute coordinates.
RPLL	Draw a polyline (multiple contiguous segments) from the Current Pointer (CP) through command specified points of the relative coordinates.
APLG	Draw a polygon which connects the start pointer (CP) and command specified points of the absolute coordinates.
RPLG	Draw a polygon which connects the start pointer (CP) and command specified points of the relative coordinates.
CRCL	Draw a circle of the radius R placing the Current Pointer (CP) at the center.
ELPS	Draw an ellipse whose shape is specified by command parameters, placing the Current Pointer (CP) at the center.
AARC	Draw an arc by using the Current Pointer (CP) as a start point with an end point and a center point of the absolute coordinates.
RARC	Draw an arc by using the Current Pointer (CP) as a start point with an end point and a center point of the relative coordinates.
AEARC	Draw an ellipse arc by using the Current Pointer (CP) as a start point with an end point and a center point of the absolute coordinates.
REARC	Draw an ellipse arc by using the Current Pointer (CP) as a start point with an end point and a center point of the relative coordinates.
AFRCT	Paint a rectangular area specified by the Current Pointer (CP) and command parameters (absolute coordinates) according to a figure pattern stored in the Pattern RAM. (Tiling)
RFRCT	Paint a rectangular area specified by the Current Point (CP) and command parameters (relative coordinates) according to a figure pattern stored in the Pattern RAM. (Tiling)
PAINT	Paint a closed area surrounded by edge color using a figure pattern stored in the Pattern RAM. (Tiling)
DOT	Mark a dot on the coordinates where the Current Point (CP) indicates.
PTN	Draw a graphic pattern defined in the Pattern RAM onto a rectangular area specified by the Current Pointer (CP) and by the pattern size. (rotation angle: 45°)
AGCPY	Copy a rectangular area specified by the absolute coordinates to the address specified by the Current Pointer (CP). (rotation angle: 90°/mirror turnover)
RGCPY	Copy a rectangular area specified by the relative coordinates to the address specified by the Current Pointer (CP). (rotation angle: 90°/mirror turnover)

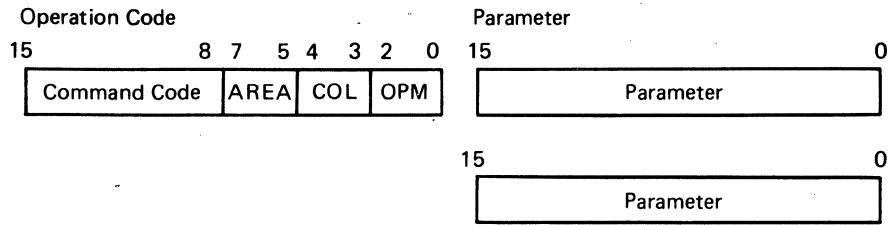


Figure 31 Graphic Drawing Command Format

**OPERATION MODE**

The Operation Mode (OPM bits) of the Graphic Drawing Command specify the logical drawing condition.

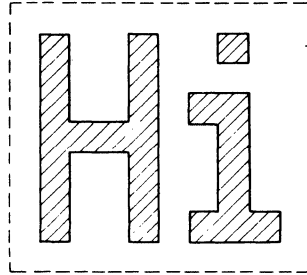
Figure 32 shows examples of a drawing pattern applied with various OPM modes.

OPM	Operation Mode
0 0 0 *	REPLACE: Replaces the frame buffer data with the color data.
0 0 1 *	OR: ORs the frame buffer data with the color data. The result is rewritten to the frame buffer.
0 1 0 *	AND: ANDs the frame buffer data with the color data. The result is rewritten to the frame buffer.
0 1 1 *	EOR: EORs the frame buffer data with the color data. The result is rewritten to the frame buffer.
1 0 0 *	CONDITIONAL REPLACE (Read Data=CCMP): When the frame buffer data at the drawing position is equal to the comparison color (CCMP), the frame buffer data is replaced with the color data.
1 0 1 *	CONDITIONAL REPLACE (Read Data≠CCMP): When the frame buffer data at the drawing position is not equal to the comparison color (CCMP), the frame buffer data is replaced with the color data.
1 1 0 * **	CONDITIONAL REPLACE (Read Data < CL): When the frame buffer data at the drawing position is less than the color register data (CL), the frame buffer data is replaced with the color data.
1 1 1 * **	CONDITIONAL REPLACE (Read Data > CL): When the frame buffer data at the drawing position is greater than the color register data (CL), the frame buffer data is replaced with the color data.

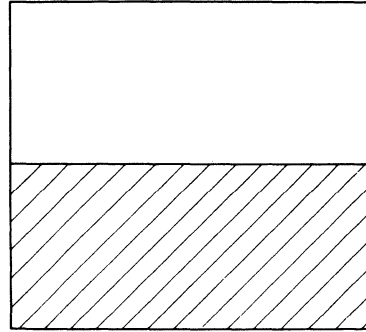
- \* Normally, the color register (CLO or CL1) selected by the pattern pointer (PPX, PPY) is used for the color data, but the source area data is used in the graphic copy commands (AGCPY and RGCPY).
- \*\* Normally, the color register (CLO or CL1) selected by the pattern pointer (PPX, PPY) is used for the color register data (CL), but the source area data is used in the graphic copy command (AGCPY and RGCPY).

Figure 32 shows examples of a drawing pattern applied with various OPM modes.

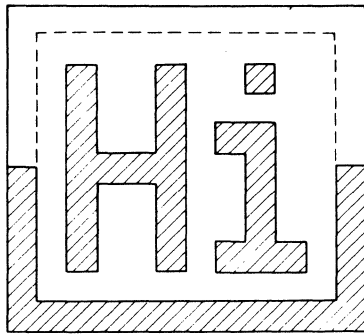




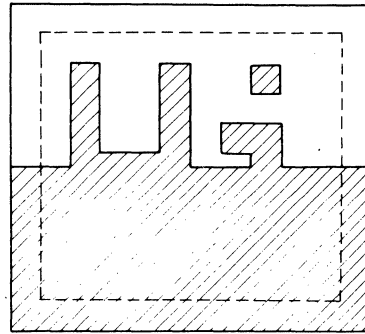
Drawing Pattern



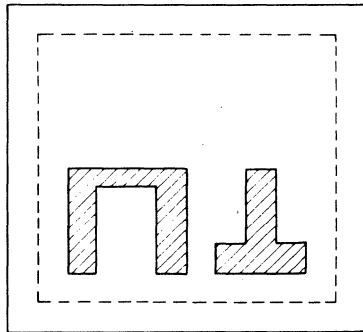
Frame Buffer Image before Drawing



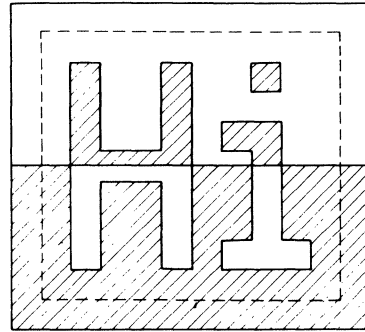
Replacement



OR



AND



EOR

Figure 32 Operation Mode Example

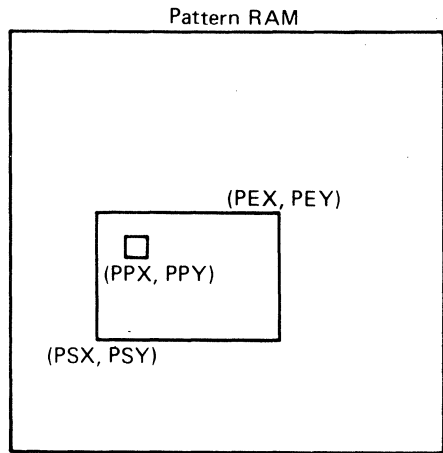
**COLOR MODE**

The Color Mode (COL bits) specify the source of the drawing

color data as directly or indirectly (using the Color Registers) determined by the contents of the Pattern RAM.

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

The Color Mode chooses the source for color information based on the contents (0 or 1) of a particular bit in the 16 bit by 16 bit (32 byte) Pattern RAM. A sub-pattern is specified by programming the Pattern RAM Control Register (PRC) with the



start (PSX, PSY) and end (PEX, PEY) points which define the diagonal of the sub-pattern. Furthermore, a specific starting point for Pattern RAM scanning is specified by PPX and PPY.

Normally, the color registers (CL) should be loaded with one color data based on the number of bits per pixel. For example, if 4 bits/pixel are used, the 4 bit color pattern (e.g. 0001) should be replicated four times in the color register, i.e.

Color Register = 

0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

In this way, color changes due to changing dot address are avoided.

**AREA MODE**

Prior to drawing, a drawing 'area' may be defined (Area Definition Register). Then, during Graphics Drawing operation the ACRTC will check if the drawing point is attempting to enter or exit the defined drawing area. Based on eight Area Modes, the ACRTC will take appropriate action for clipping or hitting.

AREA	Drawing Area Mode
0 0 0	Drawing is executed without Area checking.
0 0 1	When attempting to exit the Area, drawing is stopped after setting ABT (Abort Bit).
0 1 0	Drawing suppressed outside the Area – drawing operation continues and the ARD flag is not set.
0 1 1	Drawing suppressed outside the Area – drawing operation continues and the ARD flag is set at every drawing operation.
1 0 0	Same as AREA = 0 0 0.
1 0 1	When attempting to enter the Area, drawing is stopped after setting ABT (Abort Bit).
1 1 0	Drawing suppressed inside the Area – drawing operation continues and the ARD flag is not set.
1 1 1	Drawing suppressed inside the Area – drawing operation continues and the ARD flag is set at every drawing operation.

## ■ SYSTEM INTERFACE

### ● BASIC CLOCK

The ACRTC basic clock is 2CLK. 2CLK controls all primary ACRTC display and logic timing parameters.

2CLK, along with the specification of number of bits per logical pixel, the Graphic Address Increment mode and the Display Access mode, also determines the video data rate.

The basic clock must be input, noting its cycle, max. and min. of "High" and "Low" level width.

In any case, be careful not to stop the basic clock fixing it at "High" or "Low" or not to use 2CLK line in open state, which can destroy the LSI.

### ● CRT INTERFACE

#### FRAME BUFFER ACCESS

##### (1) Access Modes

The three ACRTC display memory access modes are Single, Interleaved and Superimposed.

##### (a) Single Access Mode

A display (or drawing) cycle is defined as two cycles of 2CLK. During the first 2CLK cycle, the frame buffer display or drawing address is output. During the second 2CLK cycle, the frame buffer data is read (display cycles and/or drawing cycles) or written (drawing cycles).

In this mode, display and drawing cycles contend for access to the frame buffer. The ACRTC allows the priority to be defined as display priority or drawing priority. If display priority, drawing cycles are only allowed to occur during horizontal/vertical flyback period. So, a 'flashless' display is

obtained at the expense of slower drawing. If drawing priority, drawing may occur during display so high speed drawing is obtained, however the display may flash.

##### (b) Interleaved Access Mode (Dual Access Mode 0)

In this mode, display cycles and drawing cycles are interleaved. A display/drawing cycle is defined as four cycles of 2CLK. During the first 2CLK cycle, the frame buffer display address is output. During the second 2CLK cycle, the display data is read from the frame buffer. During the third 2CLK cycle, the frame buffer drawing address is output. During the fourth 2CLK cycle, the drawing data is read or written.

Since there is no contention between display and drawing cycles, a 'flashless' display is obtained while maintaining full drawing speed. However, for a given configuration, frame buffer memory access time must be twice as fast as an equivalent Single Access Mode configuration.

##### (c) Superimposed Access Mode (Dual Access Mode 1)

In this mode, two separate logical screens are accessed during each display cycle. The display cycle is defined as four 2CLK cycles. During the first 2CLK cycle, the Background (Upper, Base or Lower) screen frame buffer address is output. During the second 2CLK cycle, the Background screen display data is read. During the third 2CLK cycle, the window screen frame buffer address or the drawing frame buffer address is output. During the fourth 2CLK cycle, the window screen display or drawing data is read (display or drawing) or written (drawing). Note that the third and fourth cycles can be used for drawing (similar to Interleaved mode) when these cycles are not used for Window display.

SA (SINGLE ACCESS MODE)

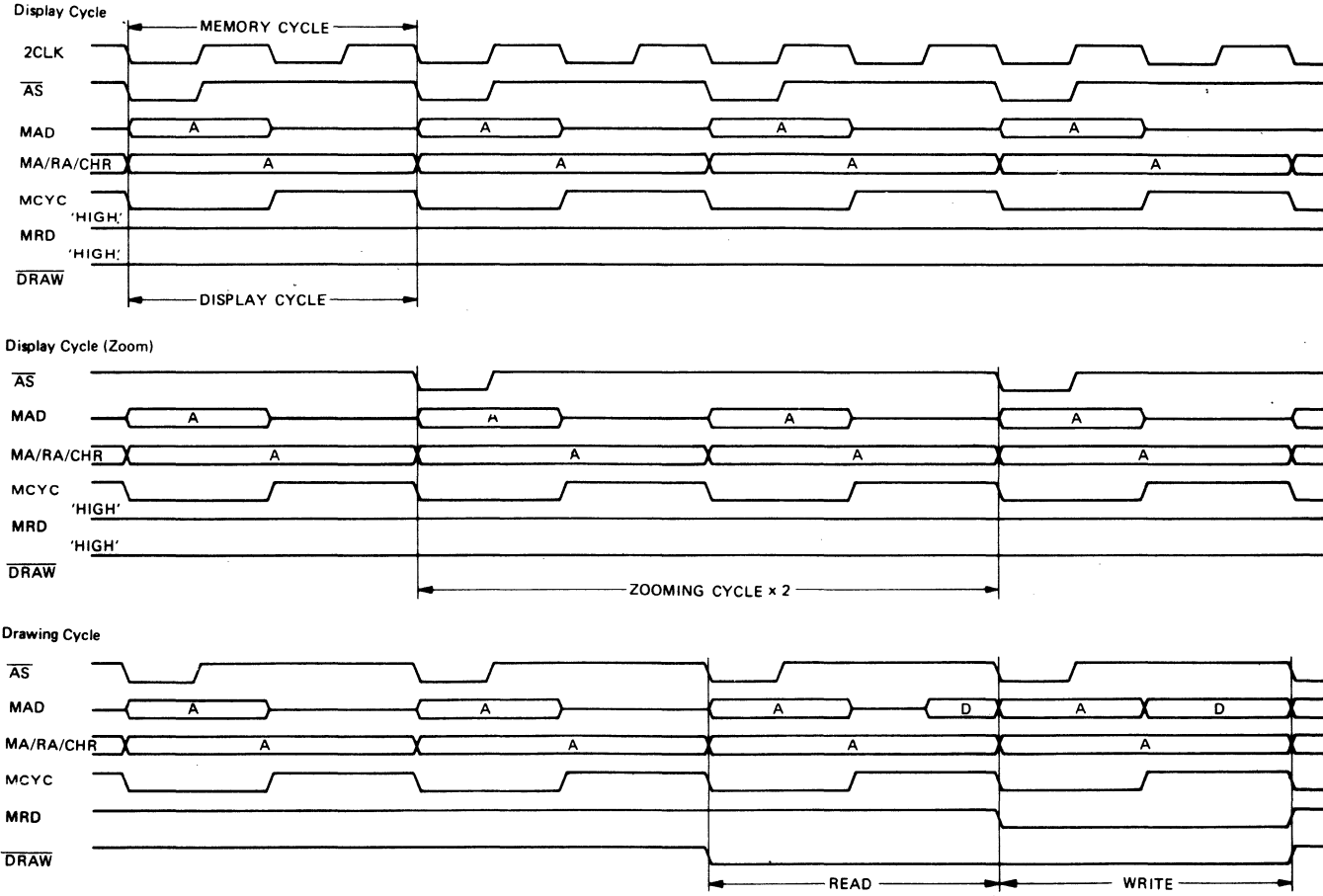


Figure 33 Access Mode Timing

INTERLEAVED ACCESS MODE

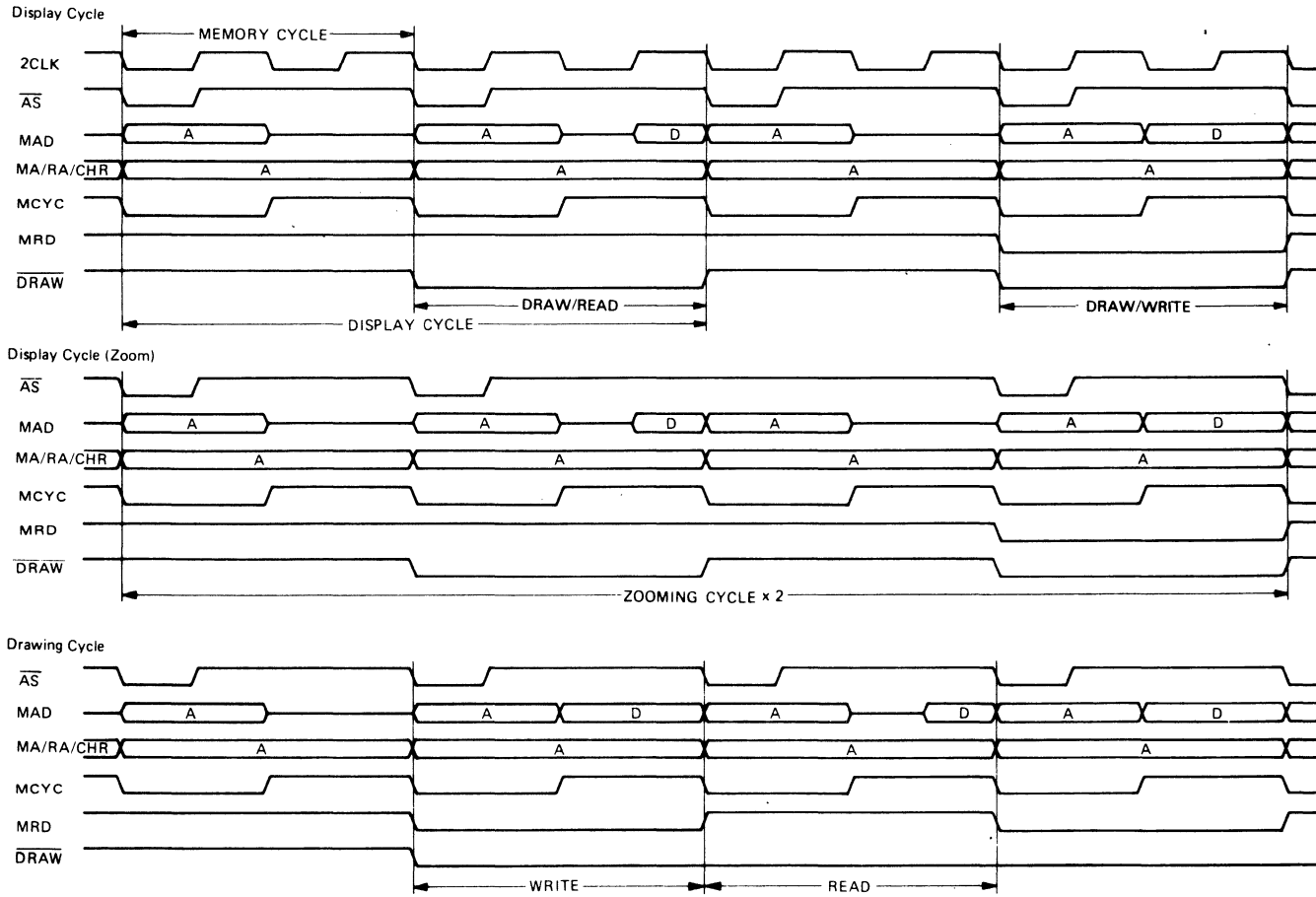


Figure 33A Access Mode Timing

SUPERIMPOSED ACCESS MODE

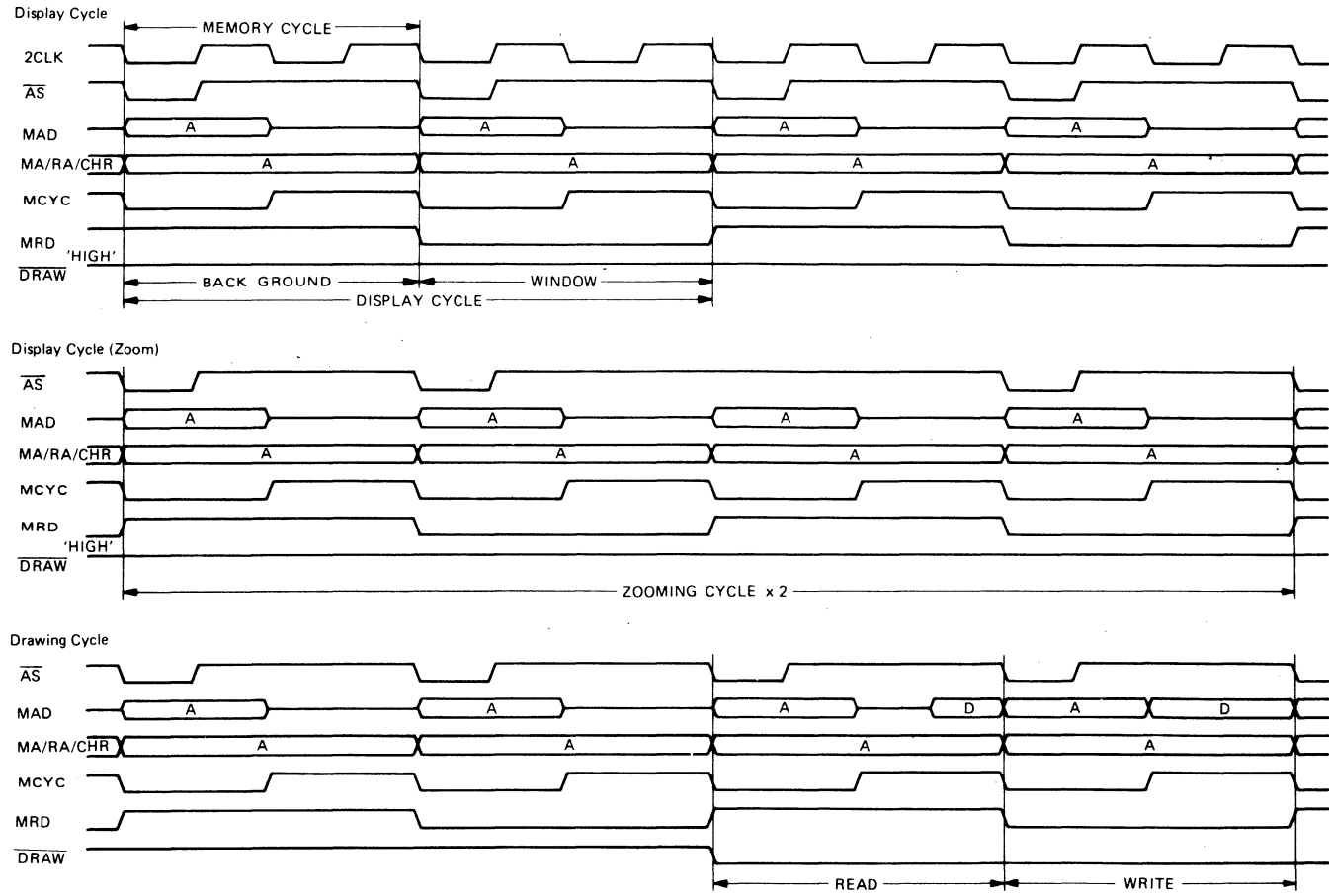
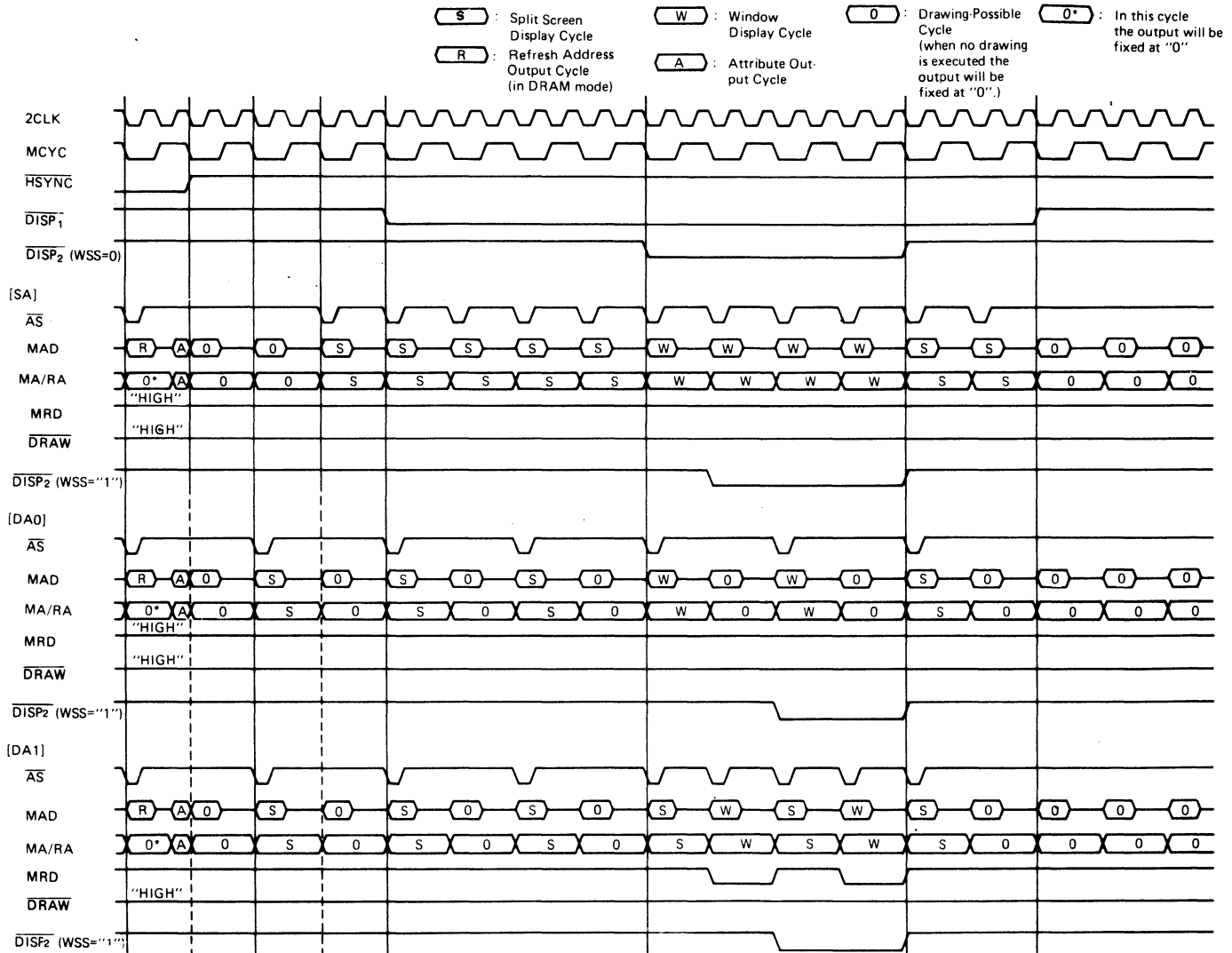


Figure 33B Access Mode Timing

Figure 33C Access Mode Timing



(2) Graphic Address Increment Mode (GAI)

During display operation, the ACRTC can be programmed to control the graphic display address in seven ways including increment by 1, 2, 4, 8 and 16\* words, 1 word every two display cycles and no increment.

Setting GAI to increment by 2, 4, 8 or 16\* words per display cycle achieves linear increases in the video data rate i.e. for a given configuration setting GAI to 2, 4, 8 or 16\* words will achieve 2, 4, 8 or 16\* times the video data rate corresponding to GAI=1. This allows increasing the number of bits/logical pixel and logical pixel resolution while meeting the 2CLK maximum frequency constraint.

Table 6 shows the summary relationship between 2CLK, Display Access Mode, Graphic Address Increment, # bits/logical pixel, memory access time and video data rate. The frame buffer cycle frequency (Fc) is shown by the following equation where:

- Fv = Dot Clock
- N = # bits/logical pixel
- D = Display Access Mode  
1 for Single Access Mode  
2 for interleaved and Superimposed Access Modes
- A = Graphic Address Increment (1/2, 1, 2, 4, 8, 16\*)
- Fc =  $(Fv \times N \times D) / (A \times 16)$

Table 6 Graphic Address Increment Modes

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

**DYNAMIC RAM REFRESH**

When dynamic RAMs (DRAMs) are used for the frame buffer memory, the ACRTC can automatically provide DRAM refresh addressing.

The ACRTC maintains an 8 bit DRAM refresh counter which is decremented on each frame buffer access. During HSYNC low, the ACRTC will output the sequential refresh addresses on MAD. The refresh address assignment depends on Graphic Address Increment (GAI) mode as shown in Table 7.

The ACRTC provides "0" output on the remaining address line of MAD and MA/RA.

DRAM refresh cycle timing must be factored into the determination of HSYNC low pulse width (HSW - specified in units of frame buffer memory cycles).

If the horizontal scan rate is Fh (kHz), number of DRAM refresh cycles is N and the DRAM refresh cycle time is Tr (msec) then horizontal sync width (HSW) is specified by the following equation:

Table 7 GAI and DRAM Refresh Addressing

Address Increment Mode	Refresh Address Output Terminal
+ 0 (GAI=101)	MAD <sub>0</sub> ~MAD <sub>7</sub>
+ 1 (GAI=000)	MAD <sub>0</sub> ~MAD <sub>7</sub>
+ 2 (GAI=001)	MAD <sub>1</sub> ~MAD <sub>8</sub>
+ 4 (GAI=010)	MAD <sub>2</sub> ~MAD <sub>9</sub>
+ 8 (GAI=011)	MAD <sub>3</sub> ~MAD <sub>10</sub>
+ 16 (GAI=100)*	MAD <sub>4</sub> ~MAD <sub>11</sub>
+ 1/2 (GAI= $\begin{matrix} 111 \\ 110 \end{matrix}$ )	MAD <sub>0</sub> ~MAD <sub>7</sub>

$$HSW \geq N / (Tr \times Fh)$$

For example, if the scan rate is 15.75 kHz and the DRAMS have 128 refresh cycles of 2 msec, HSW must be greater than or equal to 5.

$$HSW \geq 128 / (2 \times 15.75) = 4.06$$

(Note) \* ... HD63484 (R-type) does not support 16 words increment mode.



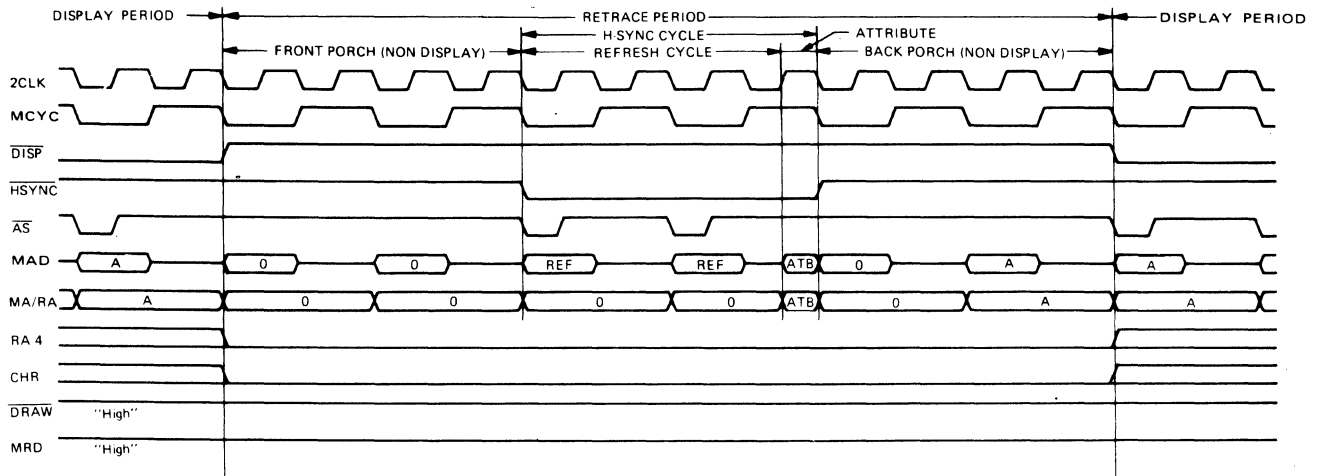


Figure 34 DRAM Refresh Timing

**EXTERNAL SYNCHRONIZATION**

The ACRTC EXSYNC pin allows synchronization of multiple ACRTCs or other video signal generators. The ACRTC may be programmed as a single Master device, or as one of a number of Slave devices.

To synchronize multiple ACRTCs, simply connect all the EXSYNC pins together.

For synchronizing to other video signals, the connection scheme depends on the raster scan mode. In Non-Interlace mode, EXSYNC corresponds to VSYNC. In Interlace modes, EXSYNC corresponds to VSYNC of the odd field.

Note) 1. The ACRTC performs the synchronization everytime it accepts the pulse input from EXSYNC in the slave mode.

It is recommended that the synchronous pulse should be input from EXSYNC only when the synchronization gap between the synchronous signal of the master device and that of ACRTC in the slave mode (HSYNC and VSYNC are output also in the slave mode.).

2. The ACRTC needs to be controlled not to execute the drawing operation during EXSYNC input.

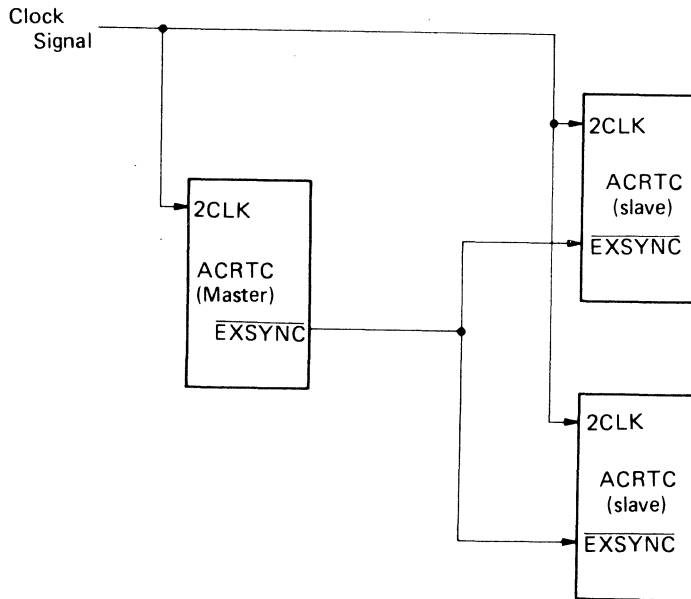


Figure 35 External Synchronization

**• MPU INTERFACE  
MPU BUS CYCLE**

The ACRTC interfaces to the MPU as a peripheral occupying two addresses in the MPU address space. The ACRTC can operate as an 8 or 16 bit peripheral as configured during RES.

An MPU bus cycle is initiated when CS is asserted (following the assertion of RS and R/W). The ACRTC responds to CS low by asserting DTACK low to complete the data transfer. DTACK will be returned to the MPU in between 1 and 1.5 2CLK cycles.

MPU WAIT states will be added in the following two cases.

(a) If the ACRTC 2CLK input is much slower than the MPU clock, continuous ACRTC accesses may be delayed due to internal processing of the previous bus cycle.

Note) Be careful for CS "High" width.

(b) If an ACRTC read cycle immediately follows an ACRTC write cycle, a WAIT state may occur due to ACRTC preparation for bus 'turn-around'. However, (68000 System: eg) MPUs normally have no instructions which immediately follow a write cycle with a read cycle.

For connection to synchronous bus interface MPUs, DTACK

can simply be left open assuming the system design guarantees that WAIT states cannot occur as described above. If WAIT states may occur, DTACK can be used with external logic to synthesize a READY signal.

**DMA TRANSFER**

The ACRTC can interface with an external DMA controller using three handshake signals, DMA Request (DREQ), DMA Acknowledge (DACK) and DMA Done (DONE).

The ACRTC uses the external DMAC for two types of transfers, Command/Parameter DMA and Data DMA. For both types, DMA transfers use the ACRTC read and write FIFOs.

(1) Command/Parameter DMA

The MPU initiates this mode by setting bit 12 (CDM) in the ACRTC Command Control Register to 1. Then, the ACRTC will automatically request DMA transfer for commands and their associated parameters as long the write FIFO has space. Only cycle steal request mode (DREQ pulses low for each data transfer) can be used. Command/Parameter DMA is terminated when the MPU resets bit 12 in CCR to 0 or the external DONE input is asserted.

Note) The R mask version and the S mask version can't perform Command/Parameter DMA transfer. So CDM (bit 12) should be set to 0.

(2) Data DMA

Data DMA is used to move data between the MPU system memory and the ACRTC frame buffer.

The MPU sets-up the transfer by specifying the frame buffer transfer address (and other parameters of the transfer, such as 'on-the fly' logical operations) to the ACRTC. Next, when the MPU issues a Data Transfer Command to the ACRTC, the ACRTC will request DMA transfer to and from system memory. The ACRTC will request DMA, automatically monitoring FIFO status, until the DMA Transfer Command is completed.

Data DMA request mode can be cycle steal (as in Command/Parameter DMA) or burst mode in which DREQ is a low level control output to the DMAC which allows multiple data transfers during each acquisition of the MPU bus.

**INTERRUPTS**

The ACRTC recognizes eight separate conditions which can generate an interrupt including command error detection, command end, drawing edge detection, light pen strobe and four FIFO status conditions. Each condition has an associated mask bit for enabling/disabling the associated interrupt. The ACRTC removes the interrupt request when the MPU performs appropriate interrupt service by reading or writing to the ACRTC.

■ **DISPLAY FUNCTION**

● **LOGICAL DISPLAY SCREENS**

The ACRTC allows division of the frame buffer into four separate logical screens.

Screen Number	Screen Name	Screen Group Name
0	Upper Screen	Background Screens
1	Base Screen	
2	Lower Screen	
3	Window Screen	

In the simplest case, only the Base screen parameters must be defined. Other screens may be selectively enabled, disabled and blanked under software control.

The Background (Upper, Base and Lower) screens partition the display into three horizontal splits whose position is fully programmable. A typical application might use the Base screen for the bulk of user interaction, using the Lower screen for a 'status line(s)' and the Upper screen for 'pull-down menu(s)'.

The Window screen is unique, since the ACRTC gives the Window screen higher priority than Background screens. thus, when the Window, whose size and position is fully programmable, overlaps a Background screen, the Window screen is displayed. One exception is the ACRTC Superimposed Access Mode, in which the Window has the same display priority as Background screens. In this case, the Window and Background screen are 'superimposed' on the display.

The ACRTC logical screen organization can be programmed to best suit a number of display applications.

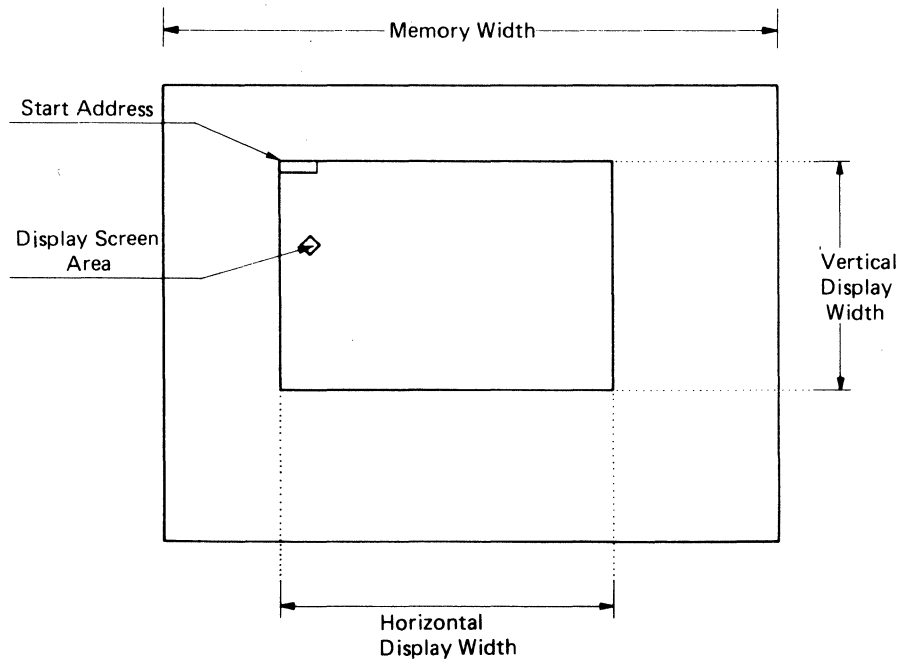


Figure 36 Display Screen/Frame Buffer Relationship

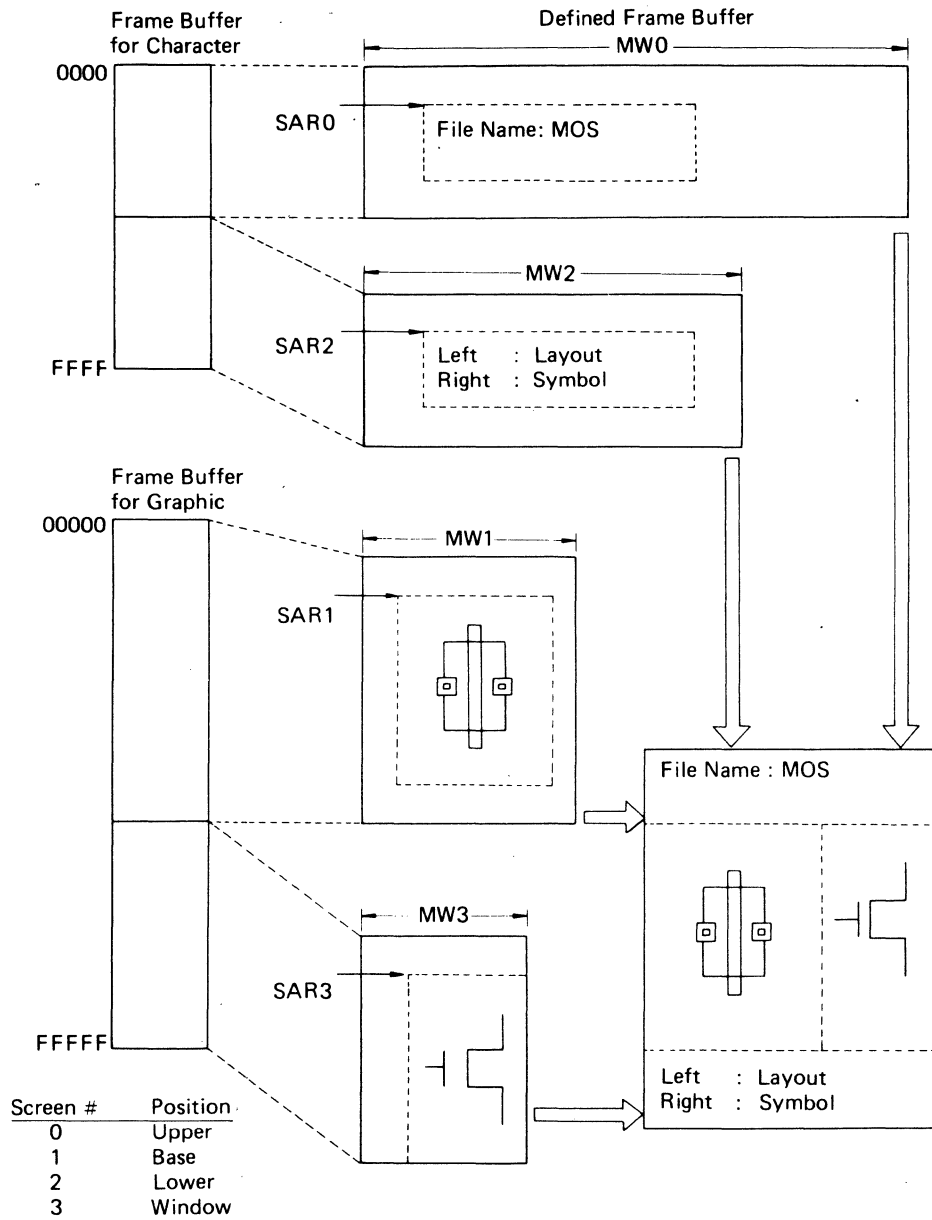


Figure 37 Display Screen Combination

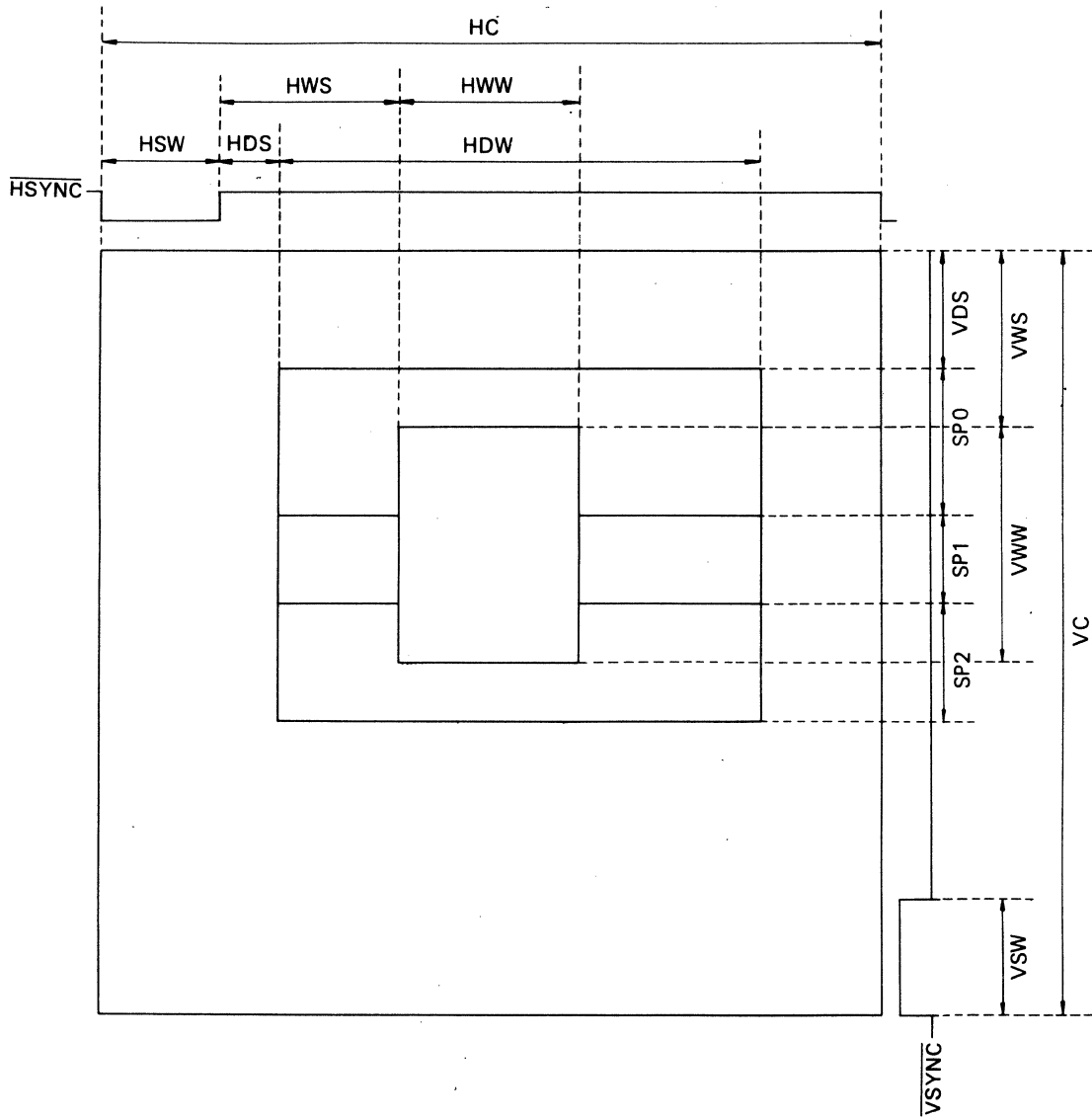
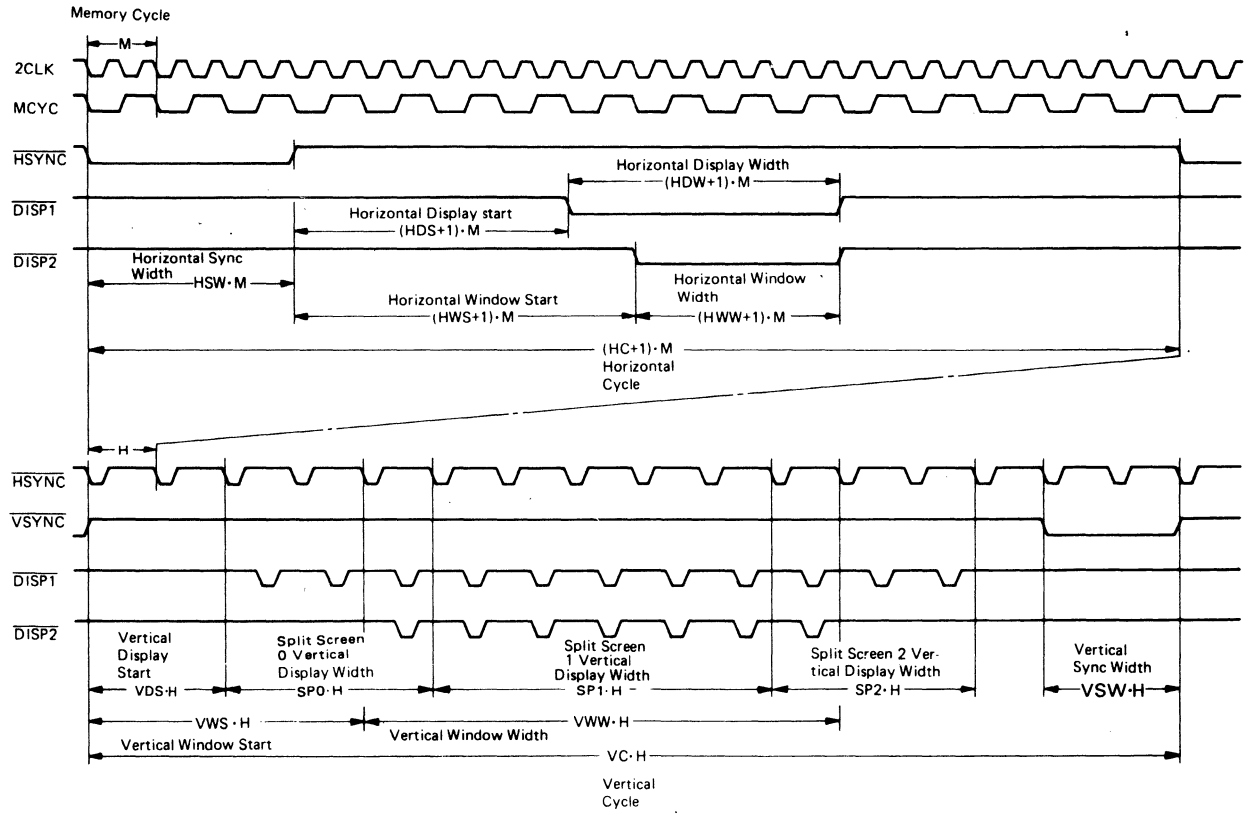


Figure 38 Display Screen Specification

Figure 39 Display Screen Timing



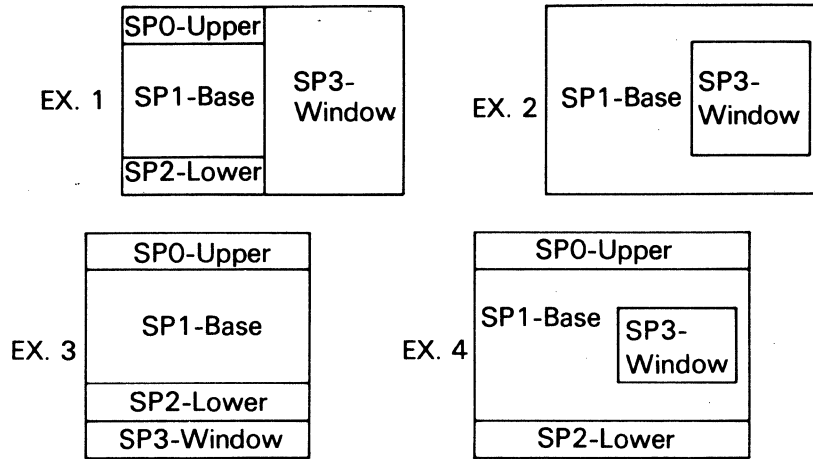


Figure 40 Example Screen Combinations

**GRAPHIC/CHARACTER ADDRESS SPACES**

The ACRTC controls two separate logical address spaces. The CHR pin allows external decoding if physically separate frame buffers are desired.

Each of the four logical screens (Upper, Base, Lower and Window) is programmed as residing in the Graphics address space or the Character address space.

ACRTC accesses to Graphics screens are treated as bit mapped using a 20 bit frame buffer address, with an address space of one megaword (1M by 16 bit).

ACRTC accesses to Character screens are treated as character generator mapped. In this case, a 64k word address space is used and 5 bits of raster address are output to an external character generator.

Multiple logical screens defined as Character can be externally

decoded to use separate character generators or different addresses within a combined character generator. Also, each Character screen may be defined with separate line spacing, separate cursors, etc.

**• CURSOR CONTROL**

The ACRTC has two Block Cursor Registers and a Graphics Cursor Register.

A Block cursor is used with Character screens. The cursor start and ending raster addresses are fully programmable. Also, the cursor width can be defined as one to eight memory cycles.

A Graphics cursor is defined by specifying the start/end memory in cycle the X dimension and the start/end raster in the Y dimension.

The Graphic cursor can output on character screens.

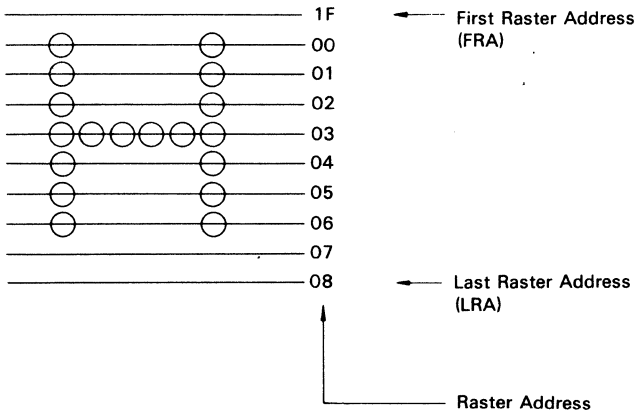


Figure 41 Character Screen Raster Addressing

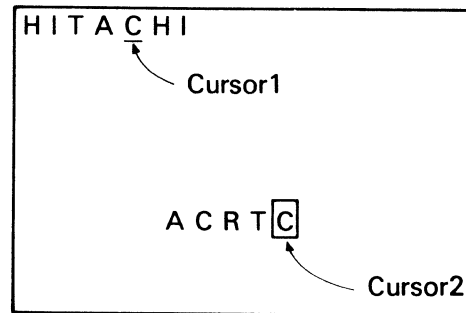


Figure 42 Two Separate Block Cursors

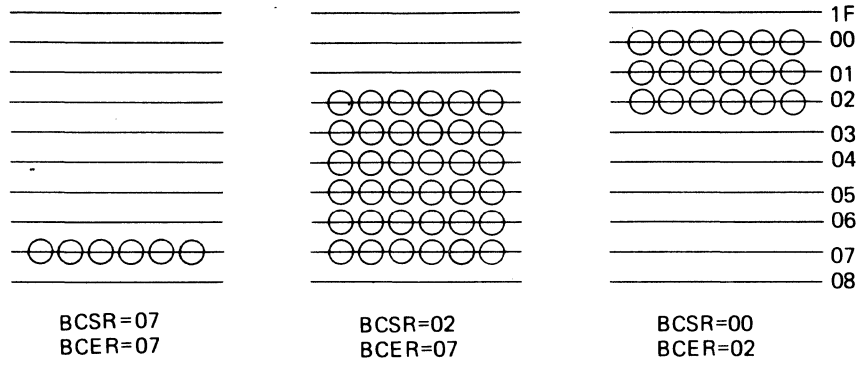


Figure 43 Block Cursor Examples

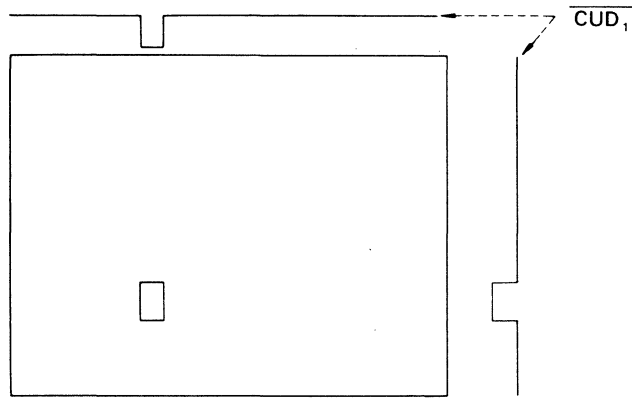


Figure 44 Graphic Cursor

The ACRTC provides two separate cursor outputs,  $\overline{CUD}_1$  and  $\overline{CUD}_2$ . These are combined with two character cursor registers and a graphics cursor register to provide three cursor modes.

**Block Mode**

Two Block cursors are output on  $\overline{CUD}_1$  and  $\overline{CUD}_2$  respectively.

**Graphic Mode**

The Graphic cursor is output on  $\overline{CUD}_1$ . Using an external cursor pattern memory allows a graphic cursor of various shapes. Two Block cursors are multiplexed on  $\overline{CUD}_2$ .

**Crosshair Mode**

The horizontal and vertical components of the Graphic cursor are output on  $\overline{CUD}_1$  and  $\overline{CUD}_2$  respectively. This allows simple generation of a crosshair cursor control signal.

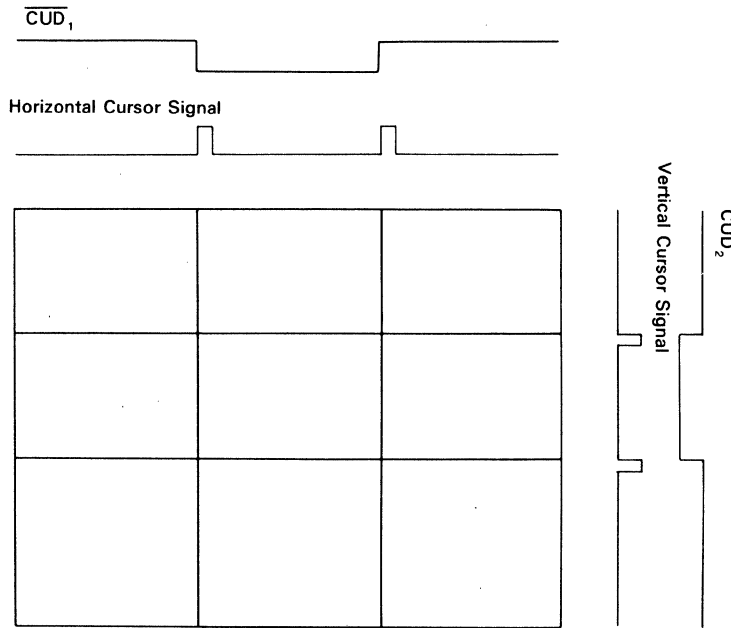


Figure 45 Crosshair Cursor

• **SCROLLING**

**VERTICAL SCROLL**

Each logical screen performs independent vertical scroll. On Character Screens, vertical smooth scroll is accomplished using the programmable Start Raster Address (SRA). Line by line scroll is accomplished by increasing or decreasing the screen start address by one unit of horizontal memory width.

On Graphics screens, vertical smooth scroll is accomplished by increasing or decreasing the screen start address by one unit of horizontal memory width.

**HORIZONTAL SCROLL**

Horizontal scroll can be performed in units of characters for Character screens and units of words (multi logical pixels) for Graphic screens by increasing or decreasing the screen start address by 1.

For smooth horizontal scroll, the ACRTC has dot shift video attributes which can be used with an external circuit which conditions shift register load/clocking.

Since this dot shift information is output each raster, horizontal smooth scroll is limited to either the Background screens or the Window screen at any given time. However, horizontal smooth scroll is independent for each of the Background screens (Upper, Base, Lower).



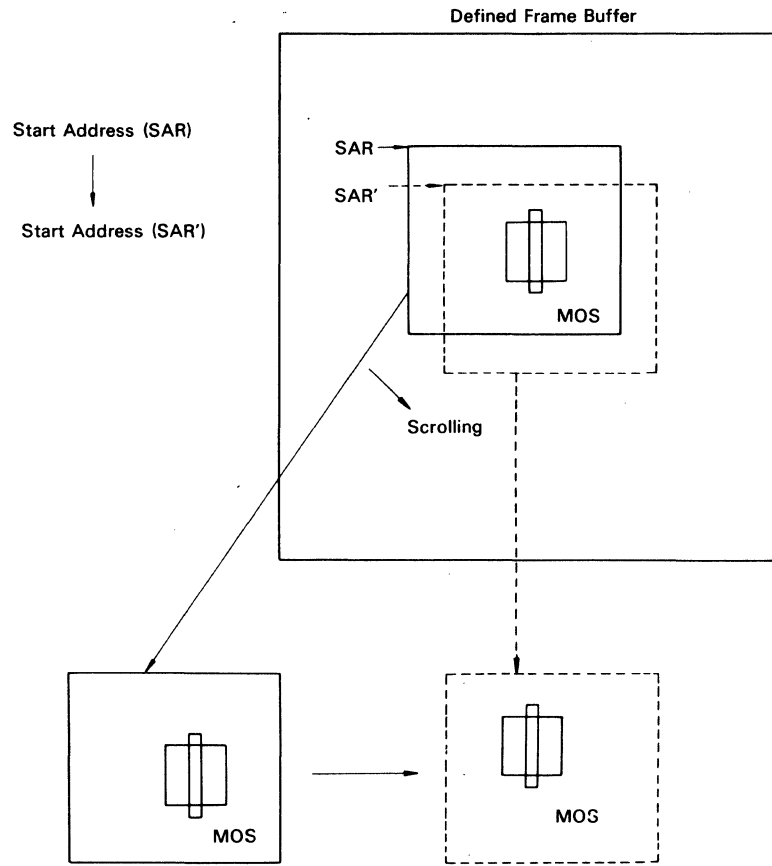


Figure 46 Scrolling By SAR (Start Address Register) Rewrite

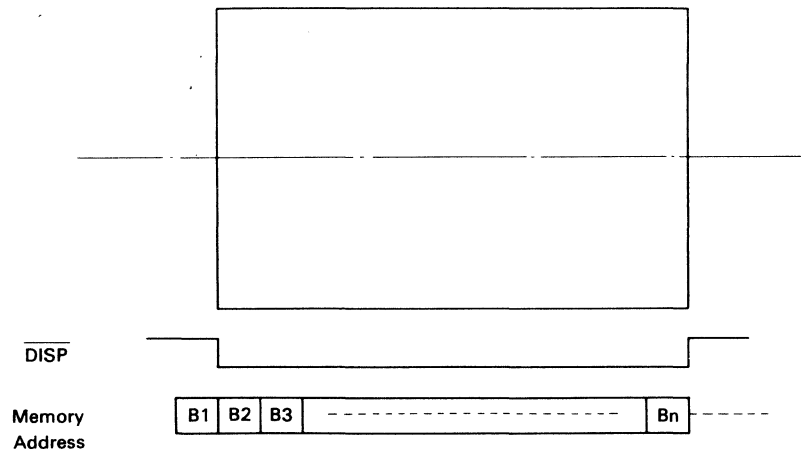


Figure 47 Horizontal Smooth Scroll – Base Screen

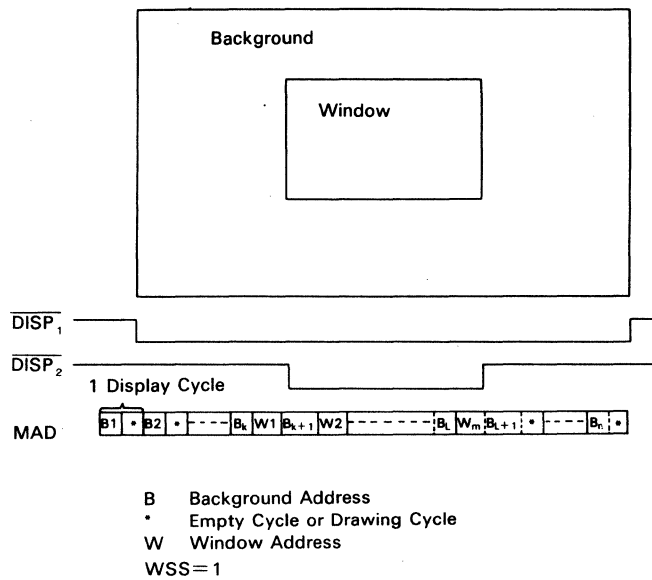


Figure 48 Horizontal Smooth Scroll – Window Screen

• **RASTER SCAN MODES**

The ACRTC has three software selectable raster scan modes – Non-Interlace, Interlace Sync and Interlace Sync & Video. In Non-Interlace mode a frame consists of one field. In the Interlace modes, a frame consists of two fields, the even and odd fields.

The Interlace modes allow increasing screen resolution while avoiding limits imposed by the CRT display device, such as maximum horizontal scan frequency or maximum video dot rate.

Interlace Sync mode simply repeats each raster address for both the even and odd fields. This is useful for increasing the

quality of a displayed figure when using an interlaced CRT device such as a Television Set with RF modulator.

Interlace Sync & Video mode displays alternate even and odd rasters on alternate even and odd fields. For a given number of rasters/character, this mode allows twice as many characters to be displayed in the vertical direction as Non-Interlace mode.

Note that for Interlace modes, the refresh frequency for a given dot on the screen is one-half that of the Non-Interlace mode. Interlace modes normally require the use of a CRT with a more persistent phosphor to avoid a flickering display.

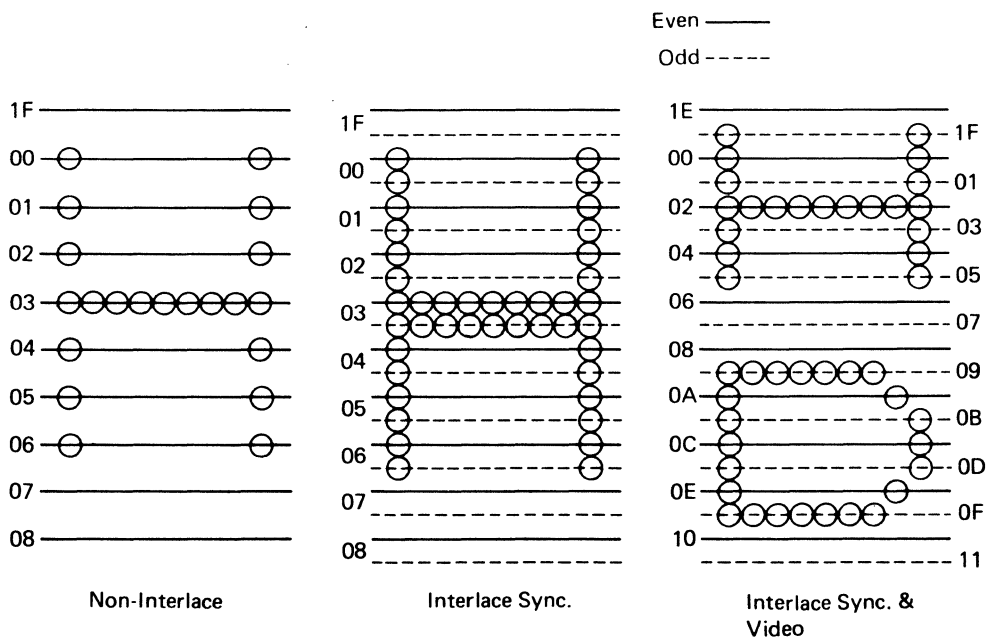


Figure 49 Raster Scan Modes

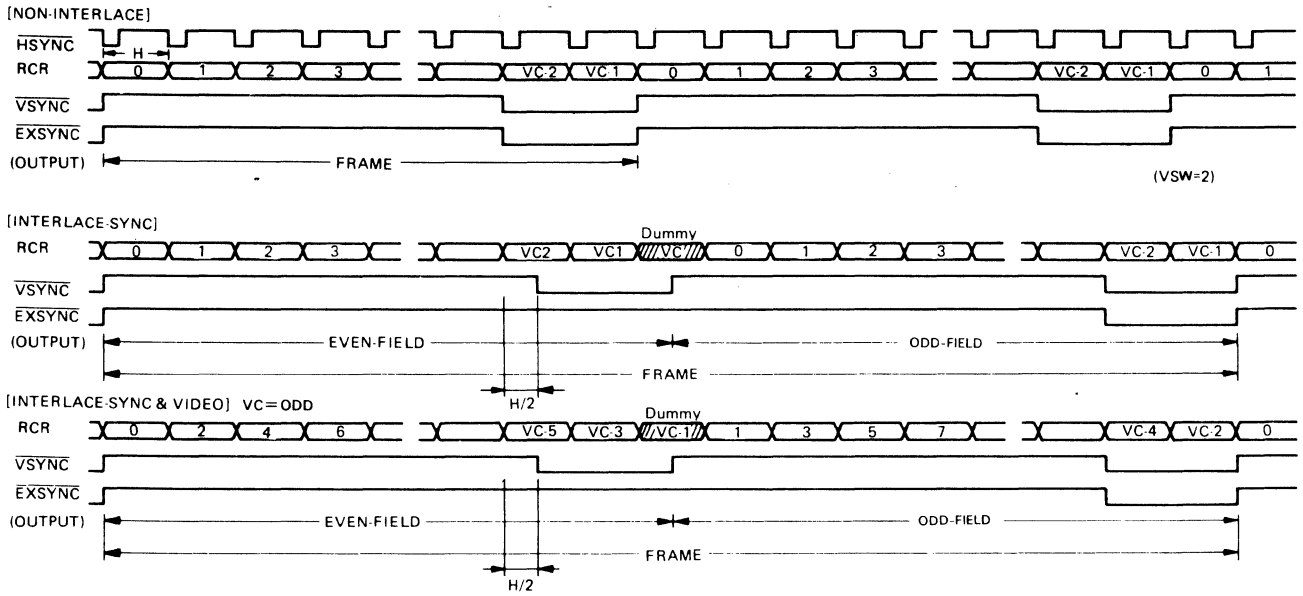


Figure 50 Raster Scan Timing

• ZOOMING

The Base screen (Screen 1) is supported by the ACRTC zooming function. Note that ACRTC zooming is performed by controlling the CRT timing signals. The contents of the frame buffer area being zoomed are not changed.

The ACRTC allows specification of a zoom factor (1 to 16) independently in the X and Y directions.

For horizontal zoom, the programmed zoom factor is output as video attributes. An external circuit uses this factor to condition the external shift register clock to accomplish horizontal zooming.

For vertical zoom, no external circuit is required. The ACRTC will scan a single raster multiple times to accomplish vertical zooming.

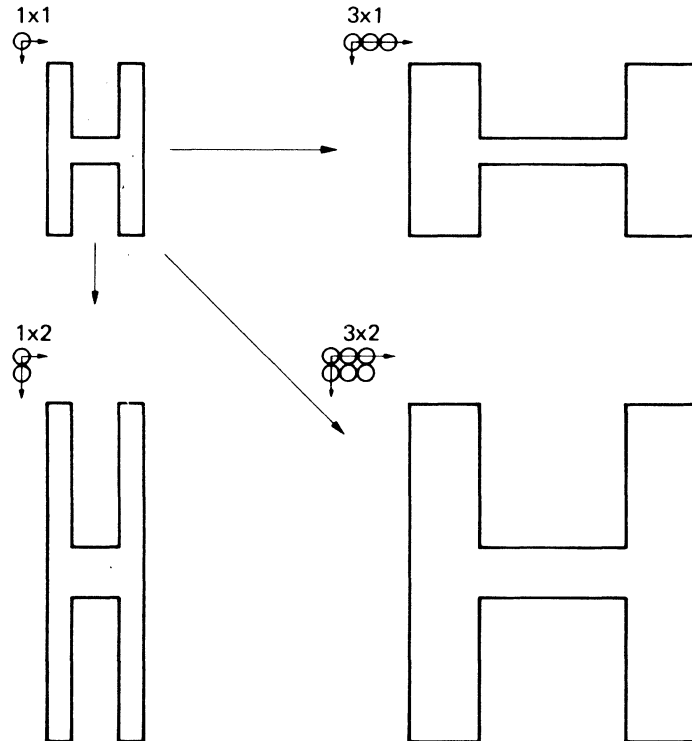


Figure 51 Zooming

● LIGHT PEN

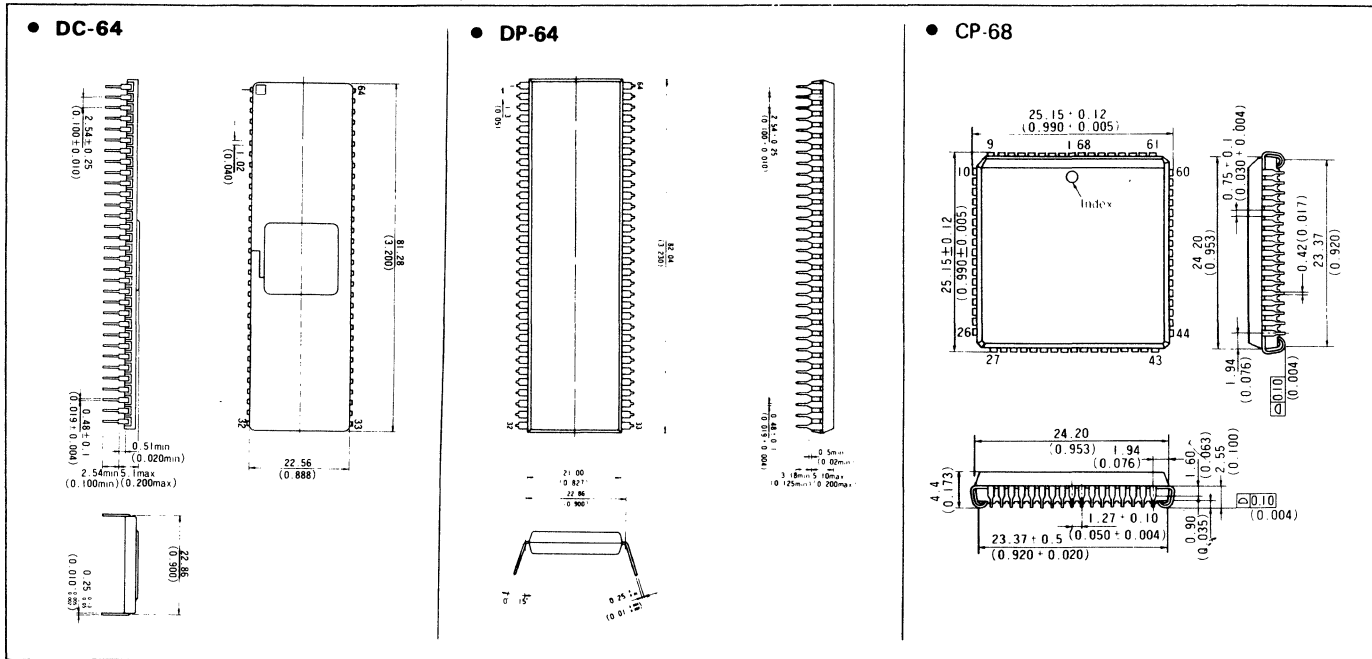
The ACRTC provides a 20 bit Light Pen Address Register and a Light Pen Strobe (LPSTB) input pin for connection with a light pen.

A light pen strobe pulse will occur when the CRT electron beam passes under the light pen during display refresh. When this pulse occurs, the contents of the ACRTC display refresh address counter will be latched into the Light Pen Address Register along with a logical screen (Character or Graphic screen) designator. Also, an ACRTC status flag indicating light pen activity is set, generating an optional (maskable) MPU interrupt. Note that for Superimposed access mode, when the light pen strobe occurs

in an area in which the Window overlaps a Background (Upper, Base or Lower) screen, the Background screen address will be latched. And even for all access mode, the Drawing address will be latched.

Various system and ACRTC delays will cause the latched address to differ slightly from the actual light pen position. The light pen address can be corrected using software, based upon system specific delays. Or, if the application does not require the highest light pen pointing resolution, software can 'bound' the light pen address by specifying a range of values associated with a given area of the screen.

■ PACKAGE DIMENSIONS



The information in this data sheet has been carefully checked; however, the contents of this data sheet may be changed and modified without notice. The company shall assume no responsibility for inaccuracies, any problem involving a patent caused when applying the descriptions in this data sheet.



Headquarter:

Hans-Pinsel-Str. 10A; D-8013 Haar b. München; Tel.: (089) 46 14-0; Telex: 5-22 593 hitc d; Telefax: (089) 46 31 51

Sales Offices:

North Germany/Benelux  
Central Germany  
South Germany  
Italy

Breslauer Str. 6; D-4040 Neuss 1; Tel.: (0 21 01) 15 00 27-29; Telex: 8-518 039 hiecd; Telefax: (0 21 01) 10 15 13  
Fabrikstr. 17; D-7024 Filderstadt 4; Tel.: (07 11) 77 20 11; Teletex: 7 111 410 HITECS; Telefax: (07 11) 7 77 51 16  
Hans-Pinsel-Str. 10A; D-8013 Haar b. München; Tel.: (089) 46 14-0; Telex: 5-22 593 hitc d; Telefax: (089) 46 31 51  
Via. B. Davanzati, 27; I-20158 Milano; Tel.: (02) 3 76 30 24; Telex: 323 377 hitec i; Telefax: (02) 68 37 30  
Hitachi c/o Multiservice; Via Cola di Rienzo 111; I-00193 Roma; Tel.: (06) 38 51 02; Telex: 680 450 gz i  
3, rue Antoine Étex; F-94000 Créteil; Tél.: (1) 43 39 45 00; Télex: 262 246, hitec f; Téléfax: (1) 43 39 84 93

France



# MOTOROLA SEMICONDUCTORS

2001 E. BROADWAY, CHICAGO, ILLINOIS 60611

## MC68153

### Advance Information

#### BUS INTERRUPTER MODULE

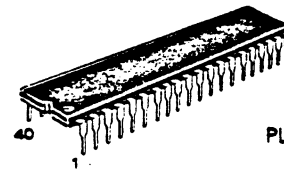
The bipolar LSI MC68153 Bus Interrupter interfaces a micro-computer system bus to multiple slave devices requiring interrupt capabilities. It handles up to 4 independent sources of interrupt requests and is fully programmable.

- VERSAbus/VMEbus Compatible
- MC68000 Compatible
- Handles 4 Independent Interrupt Sources
- 8 Programmable Read/Write Registers
- Programmable Interrupt Request Levels
- Programmable Interrupt Vectors
- Supports Interrupt Acknowledge Daisy Chain
- Control Registers Contain Flag Bits
- Single +5.0 Volt Supply
- Total Power Dissipation = 1.5 W Typical
- Temperature Range of 0°C to 70°C
- Chip Access Time = 200 ns Typical with 16 MHz Clock
- 40-Pin Dual-In-Line Package

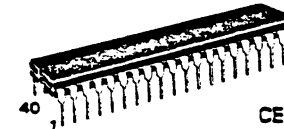
TTL

#### BUS INTERRUPTER MODULE

ADVANCED LOW POWER SCHOTTKY

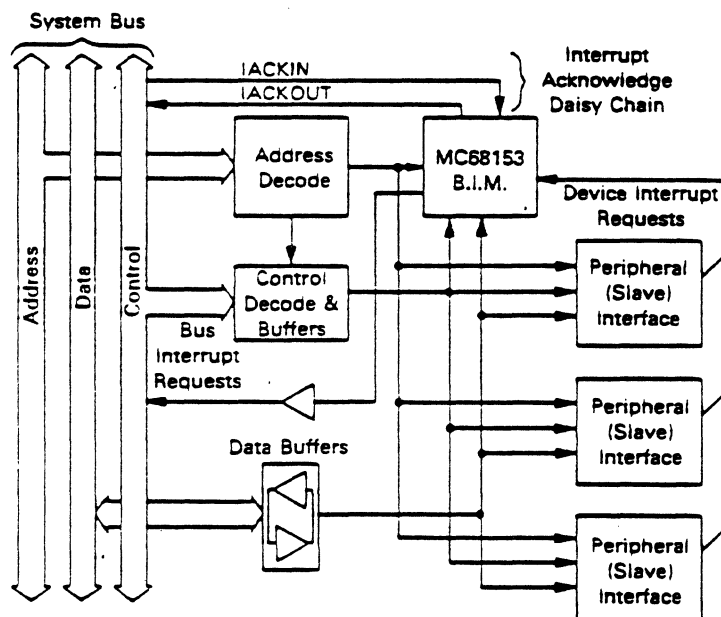


P SUFFIX  
PLASTIC PACKAGE  
CASE 711-03



L SUFFIX  
CERAMIC PACKAGE  
CASE 734-04

FIGURE 1 — MC68153 SYSTEM BLOCK DIAGRAM



VERSAbus is a trademark of Motorola.

#### PIN ASSIGNMENTS

VCC	1	40	A3
R/W	2	39	A2
CS	3	38	A1
DTACK	4	37	D7
IACK	5	36	D6
IACKIN	6	35	D5
IACKOUT	7	34	D4
IRQ1	8	33	D3
GND	9	32	D2
GND	10	31	GND
VCC	11	30	VCC
IRQ2	12	29	D1
IRQ3	13	28	D0
IRQ4	14	27	INTAE
IRQ5	15	26	INTAL1
IRQ6	16	25	INTALO
IRQ7	17	24	INT3
CLK	18	23	INT2
INT0	19	22	INT1
GND	20	21	VCC

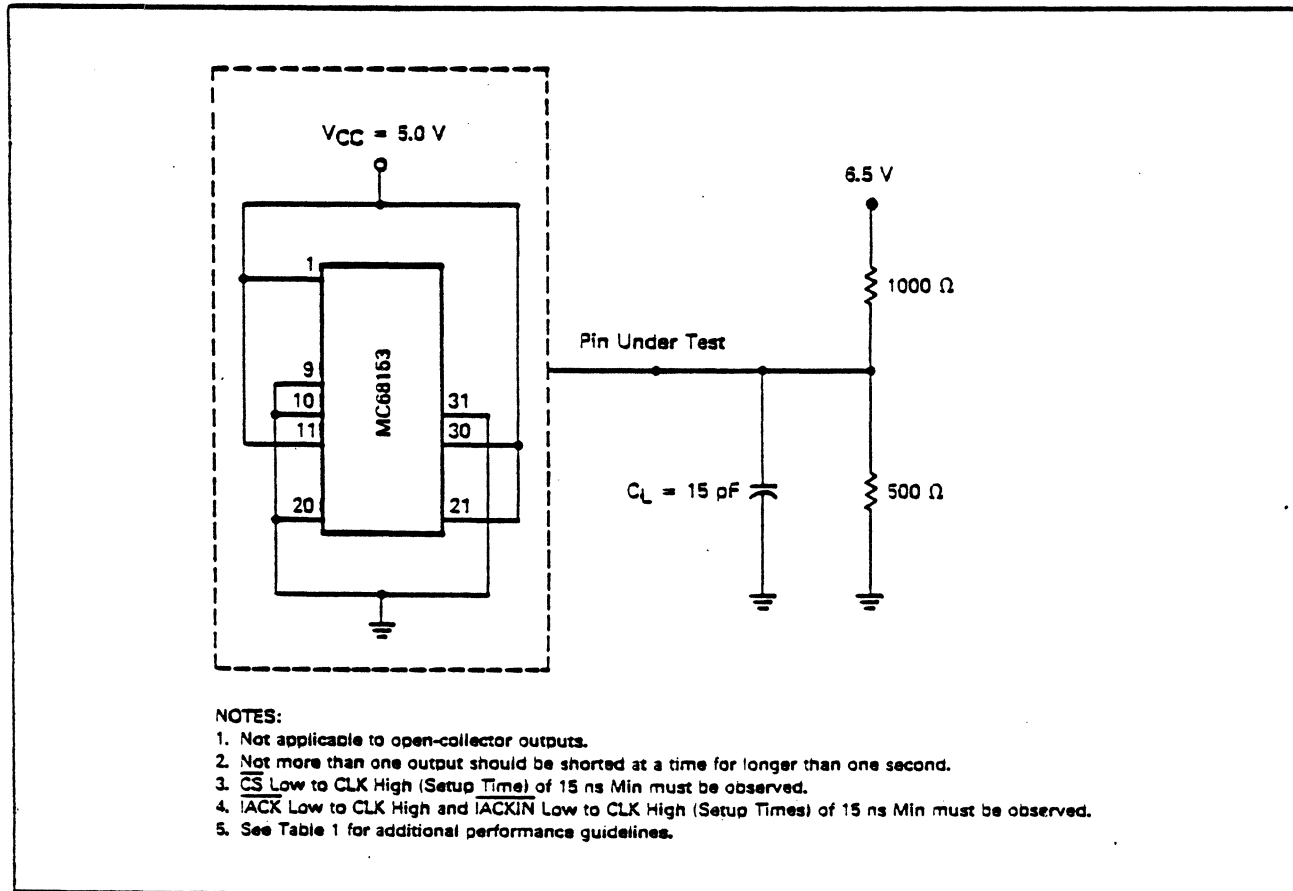
**ABSOLUTE MAXIMUM RATINGS** (Beyond which useful life may be impaired.)

Parameter	Symbol	Value	Unit
Supply Voltage	V <sub>CC</sub>	-0.5 to +7.0	V
Input Voltage	V <sub>in</sub>	-0.5 to +7.0	V
Input Current	I <sub>in</sub>	-30 to +5.0	mA
Output Voltage	V <sub>out</sub>	-0.5 to +5.5	V
Output Current	I <sub>OL</sub>	Twice Rated I <sub>OL</sub>	mA
Storage Temperature	T <sub>stg</sub>	-65 to +150	°C
Junction Operating Temperature	T <sub>J</sub>	-55 to +175	°C

**DC ELECTRICAL SPECIFICATIONS** (V<sub>CC</sub> = 5.0 V ±5%, T<sub>A</sub> = 0°C to 70°C)

Parameter	Symbol	Min	Max	Unit	Test Conditions
High Level Input Voltage	V <sub>IH</sub>	2.0	—	V	
Low Level Input Voltage	V <sub>IL</sub>	—	0.8	V	
Input Clamp Voltage	V <sub>IK</sub>	—	-1.5	V	V <sub>CC</sub> = MIN, I <sub>IN</sub> = -18 mA
High Level Output Voltage <sup>(1)</sup>	V <sub>OH</sub>	2.7	—	V	V <sub>CC</sub> = MIN, I <sub>OH</sub> = -400 μA
Low Level Output Voltage	V <sub>OL</sub>	—	0.4	V	V <sub>CC</sub> = MIN, I <sub>OL</sub> = 8.0 mA
Output Short Circuit Current <sup>(2)</sup>	I <sub>OS</sub>	-15	-130	mA	V <sub>CC</sub> = MAX, V <sub>OUT</sub> = 0 V
High Level Input Current	I <sub>IH</sub>	—	20	μA	V <sub>CC</sub> = MAX, V <sub>IN</sub> = 2.7 V
Low Level Input Current	I <sub>IL</sub>	—	-0.4	mA	V <sub>CC</sub> = MAX, V <sub>IN</sub> = 0.4 V
Supply Current	I <sub>CC</sub>	225	385	mA	V <sub>CC</sub> = MAX
Output Off Current (High)	I <sub>OZH</sub>	—	20	μA	V <sub>CC</sub> = MAX, V <sub>OUT</sub> = 2.4 V
Output Off Current (Low)	I <sub>OZL</sub>	—	-20	μA	V <sub>CC</sub> = MAX, V <sub>OUT</sub> = 0.4 V

**AC TEST CIRCUIT — AC Testing of All Outputs**



**AC ELECTRICAL SPECIFICATIONS (V<sub>CC</sub> = 5.0 V ±5%, T<sub>A</sub> = 0°C to 70°C)**

Parameter	Test Number(5)	Max (ns)
CLK High to Data Out Valid (Delay)(3)	1	55
CLK High to $\overline{DTACK}$ Low (Delay)(3)	2	40
CS High to $\overline{DTACK}$ High (Delay)	3	35
CLK High to Data Out Valid (Delay)(4)	4	55
CLK High to $\overline{INTAE}$ Low (Delay)(4)	5	40
$\overline{IACK}$ High to Data Out High Impedance (Delay)	6	60
$\overline{IACK}$ High to $\overline{DTACK}$ High (Delay)	7	45
CS High to Data Out High (Delay)	8	45
CS High to $\overline{IRQ}$ High (Delay)	9	60
$\overline{IACK}$ High to $\overline{INTAE}$ High (Delay)	10	35

**GENERAL DESCRIPTION**

The MC68153 Bus Interrupter Module (BIM) is designed to serve as an interrupt requester for peripheral devices in a microcomputer system. Up to 4 independent devices can be interfaced to the system bus by the MC68153. Intended for asynchronous master/slave bus operation, the BIM is compatible with VERSAbus, VMEbus, MC68000 device bus, and other system buses. Figure 1 shows a block diagram of a typical configuration. In this example, three peripheral devices (bus slaves) are connected to the system data bus. Each of these devices could be parallel I/O, serial I/O, or some other function. An interrupt request from any device is routed to the MC68153, and the BIM handles all interface to the system bus. It generates a bus interrupt request as a result of the device interrupt request. When the system interrupt handler or processor responds with an interrupt acknowledge cycle, the MC68153 can answer supplying an interrupt vector and handling all timing.

The functional block diagram of the MC68153 is shown in Figure 2. The device contains circuitry to accept four separate interrupt sources ( $\overline{INT0}$  -  $\overline{INT3}$ ). Interface to the system bus includes generation of bus interrupt requests ( $\overline{IRQ1}$  -  $\overline{IRQ7}$ ), response to a bus interrupt acknowledge cycle (either supplying a vector or passing on a daisy chain signal), and releasing the bus interrupt request signal at the proper time. The BIM has flexibility provided by eight programmable read/write registers. Four 8-bit vector registers (VR0 - VR3) contain status/address information and supply a byte vector in response to an interrupt acknowledge cycle for the corresponding interrupt source. Four other 8-bit control registers (CR0 - CR3) contain information that oversees operation of the interrupt circuitry. The control information is programmable and includes interrupt request level and interrupt enable and disable. Also contained in the control registers are flag-bits. These flags are useful for task coordination, resource management, and interprocessor communication.

**SIGNAL DESCRIPTION**

Throughout the data sheet, signals are presented using the terms asserted and negated independent of whether the signal is asserted in the high voltage or low voltage state. Active low signals are denoted by a superscript bar.

**BIDIRECTIONAL DATA BUS — D0 - D7**

Pins D0 - D7 form an 8-bit bidirectional data bus to/from the system bus. These are active high, 3-state pins.

**ADDRESS INPUTS — A1 - A3**

These active high inputs serve two functions. One function is to select one of the eight possible registers during a read or write cycle. Secondly, during an interrupt acknowledge A1 - A3 show the level of interrupt being acknowledged, and the BIM uses these to determine if a match exists with an internal level.

**CHIP SELECT —  $\overline{CS}$**

$\overline{CS}$  is an active low input used to select the BIM's registers for the current bus cycle. Address strobe, data strobe, and appropriate address bits must be included in the chip select equation.

**READ/WRITE —  $\overline{R/W}$**

The  $\overline{R/W}$  input is a signal from the system bus used to determine if the current bus cycle is a read (high) or write (low).

**DATA TRANSFER ACKNOWLEDGE —  $\overline{DTACK}$**

$\overline{DTACK}$  is an open-collector, active low output that signals the completion of a read, write, or interrupt acknowledge cycle. During read or interrupt acknowledge cycles,  $\overline{DTACK}$  is asserted by the MC68153 after data has been provided on the data bus; during write cycles it is asserted after data has been accepted from the data bus. A pullup resistor is required to maintain  $\overline{DTACK}$  high between bus cycles.



FIGURE 2 — MC68153 FUNCTIONAL BLOCK DIAGRAM

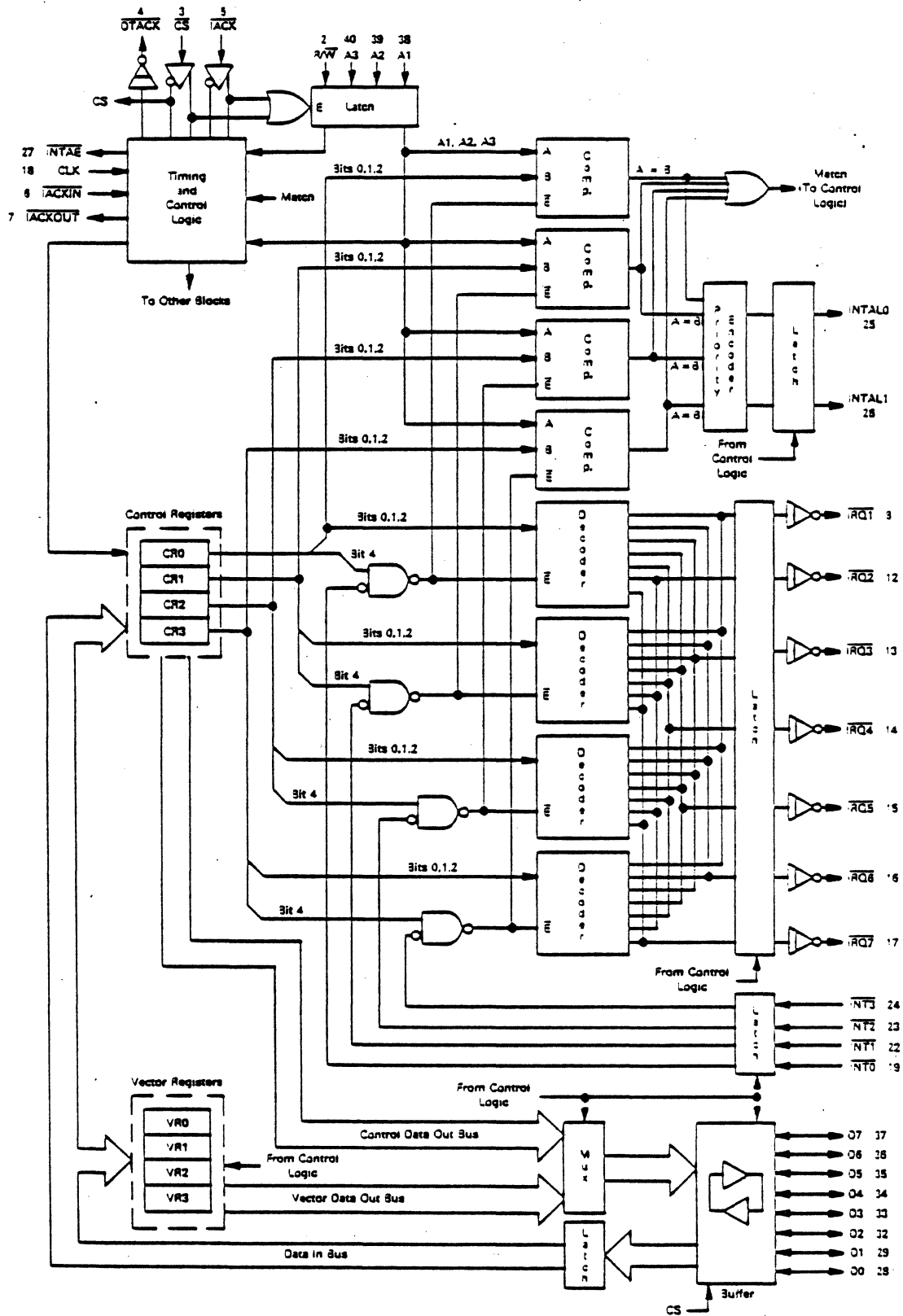
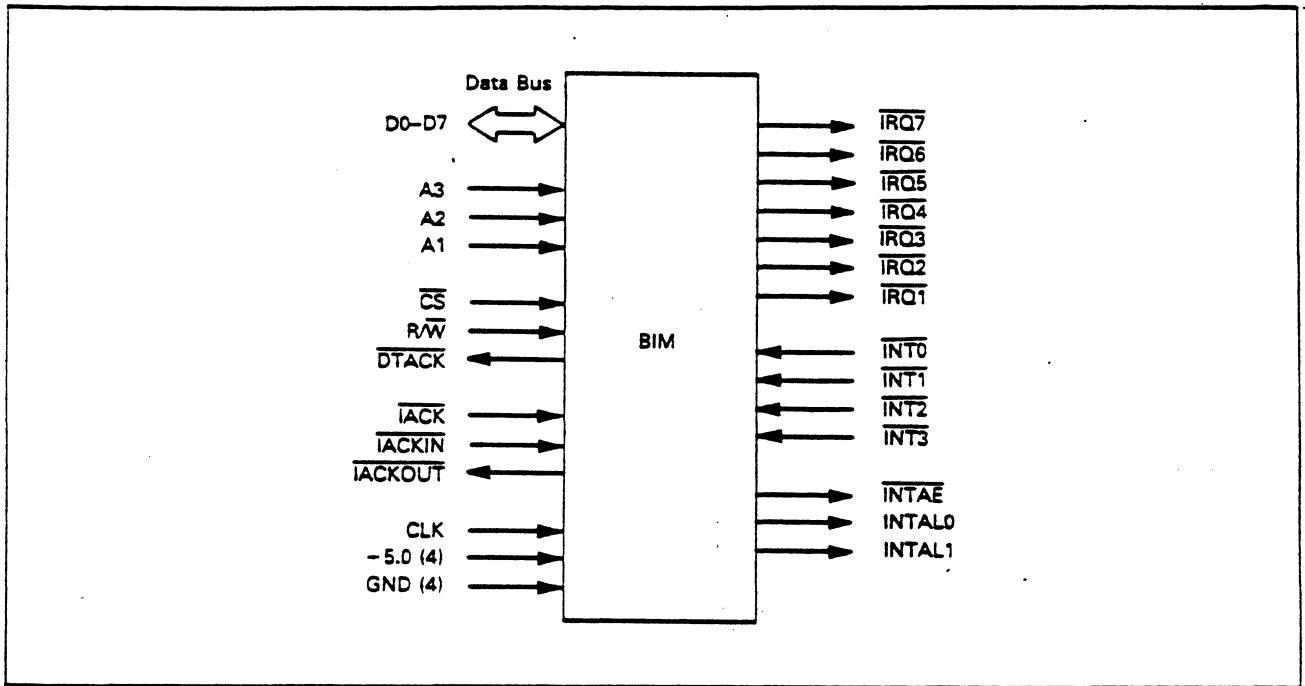




FIGURE 3 — LOGICAL PIN ASSIGNMENT



**INTERRUPT ACKNOWLEDGE SIGNALS —  $\overline{\text{IACK}}$ ,  $\overline{\text{IACKIN}}$ ,  $\overline{\text{IACKOUT}}$**

These three pins support the interrupt acknowledge cycle. A low level on the  $\overline{\text{IACK}}$  input indicates an interrupt acknowledge cycle has been initiated. This signal is conditioned externally with Address Strobe and the lower data strobe of an MC68000 type bus. After  $\overline{\text{IACK}}$  is asserted the BIM compares the interrupt level presented on address lines A1, A2, and A3 with the current levels generated internally and determines if a match exists. Then, if input  $\overline{\text{IACKIN}}$  is asserted (driven low), the BIM will either complete the interrupt acknowledge cycle if a match exists or assert output  $\overline{\text{IACKOUT}}$  if no match exists.

$\overline{\text{IACKIN}}$  and  $\overline{\text{IACKOUT}}$  form part of a prioritized interrupt acknowledge daisy chain. The daisy chain prioritizes interrupters and guarantees that two or more devices requesting an interrupt on the same level will not respond to the same cycle. The requesting device (or interrupter) must wait until  $\overline{\text{IACKIN}}$  is asserted and not pass the signal on (assert  $\overline{\text{IACKOUT}}$ ) if it is to complete the interrupt acknowledge cycle.

**BUS INTERRUPT REQUEST SIGNALS —  $\overline{\text{IRQ1}}$  -  $\overline{\text{IRQ7}}$**

These open-collector outputs are low when asserted, indicating a bus interrupt is requested at the corresponding level. An open-collector buffer is normally required for sufficient drive when interfacing to a system bus. A pullup resistor is required to maintain  $\overline{\text{IRQ1}}$  -  $\overline{\text{IRQ7}}$  high between interrupt requests.

**DEVICE INTERRUPT REQUEST SIGNALS —  $\overline{\text{INT0}}$  -  $\overline{\text{INT3}}$**

$\overline{\text{INT0}}$  -  $\overline{\text{INT3}}$  are active low inputs used to indicate to the BIM that a device wants a bus interrupt.

**INTERRUPT ACKNOWLEDGE ENABLE —  $\overline{\text{INTAE}}$**

During an interrupt acknowledge cycle, this output pin is asserted low to indicate that outputs INTAL0 and INTAL1 are valid. These two outputs contain an encoded number (x) corresponding to the interrupt ( $\overline{\text{INTx}}$ ) being acknowledged. This feature can be used to signal interrupting devices, which supply their own vector, when to respond to the interrupt acknowledge cycle with the vector and a DTACK signal.

**INTERRUPT ACKNOWLEDGE LEVEL — INTAL0, INTAL1**

These active high outputs contain an encoded number corresponding to the interrupt level being acknowledged. They are valid only when  $\overline{\text{INTAE}}$  is asserted low.

**CLOCK — CLK**

The CLK input is used to supply the clock for internal operations of the MC68153.

**RESET —  $\overline{\text{CS}}$ ,  $\overline{\text{IACK}}$**

Although a reset input is not supplied, an on-board reset is performed if  $\overline{\text{CS}}$  and  $\overline{\text{IACK}}$  are asserted simultaneously.



FIGURE 4 — MC68153 REGISTER MODEL

ADDRESS BIT			REGISTER BIT								REGISTER NAME
A3	A2	A1	FLAG	FLAG AUTO-CLEAR	EXTERNAL/INTERNAL	INTERRUPT ENABLE	INTERRUPT AUTO-CLEAR	INTERRUPT LEVEL			
0	0	0	F	FAC	X/IN	IRE	IRAC	L2	L1	L0	CONTROL REGISTER 0
0	0	1	F	FAC	X/IN	IRE	IRAC	L2	L1	L0	CONTROL REGISTER 1
0	1	0	F	FAC	X/IN	IRE	IRAC	L2	L1	L0	CONTROL REGISTER 2
1	1	1	F	FAC	X/IN	IRE	IRAC	L2	L1	L0	CONTROL REGISTER 3
1	0	0	V7	V6	V5	V4	V3	V2	V1	V0	VECTOR REGISTER 0
1	0	1	V7	V6	V5	V4	V3	V2	V1	V0	VECTOR REGISTER 1
1	1	0	V7	V6	V5	V4	V3	V2	V1	V0	VECTOR REGISTER 2
1	1	1	V7	V6	V5	V4	V3	V2	V1	V0	VECTOR REGISTER 3

**REGISTER DESCRIPTION**

The MC68153 contains 8 programmable read/write registers. There are four control registers (CR0 - CR3) that govern operation of the device. The other four (VR0 - VR3) are vector registers that contain the vector data used during an interrupt acknowledge cycle. Figure 4 illustrates the device register model.

**CONTROL REGISTERS**

There is a control register for each interrupt source, i.e., CR0 controls INT0, CR1 controls INT1, etc. The control registers are divided into several fields:

1. Interrupt level (L2, L1, L0) — The least significant 3-bit field of the register determines the level at which an interrupt will be generated:

L2	L1	L0	IRQ LEVEL
0	0	0	DISABLED
0	0	1	IRQ1
0	1	0	IRQ2
0	1	1	IRQ3
1	0	0	IRQ4
1	0	1	IRQ5
1	1	0	IRQ6
1	1	1	IRQ7

A value of zero in the field disables the interrupt.

2. Interrupt Enable (IRE) — This field (Bit 4) must be set (high level) to enable the bus interrupt request associated with the control register. Thus, if the INTX line is asserted and IRE is cleared, no interrupt request (IRQX) will be asserted.
3. Interrupt Auto-Clear (IRAC) — If the IRAC is set (Bit 3), IRE (Bit 4) is cleared during an interrupt acknowledge cycle responding to this request. This action of

clearing IRE disables the interrupt request. To re-enable the interrupt associated with this register, IRE must be set again by writing to the control register.

4. External/Internal (X/IN) — Bit 5 of the control register determines the response of the MC68153 during an interrupt acknowledge cycle. If the X/IN bit is clear (low level) the BIM will respond with vector data and a DTACK signal, i.e., an internal response. If X/IN is set, the vector is not supplied and no DTACK is given by the BIM, i.e., an external device should respond.
5. Flag (F) — Bit 7 is a flag that can be used in conjunction with the test and set instruction of the MC68000. It can be changed without affecting chip operation. It is useful for processor-to-processor communication and resource allocation.
6. Flag Auto-Clear (FAC) — If FAC (Bit 6) is set, the Flag bit is automatically cleared during an interrupt acknowledge cycle.

**VECTOR REGISTERS**

Each interrupt input has its own associated vector register. Each register is 8 bits wide and supplies a data byte during its interrupt acknowledge cycle if the associated External/Internal (X/IN) control register bit is clear. This data can be status, identification, or address information depending on system usage. The information is programmed by the system user.

**DEVICE RESET**

When the MC68153 is reset, the registers are set to a known condition. The control registers are set to all zeros (low). The vector registers are set to S0F. This value is the MC68000 vector for an uninitialized interrupt vector.



## FUNCTIONAL DESCRIPTION

### SYSTEM OVERVIEW

The MC68153 is compatible with many system buses, however, it is primarily intended for VMEbus, VERSAbus and MC68000 applications. Figure 5 shows a system configuration similar to VMEbus. In the figure only one system Data Transfer Bus (DTB) master is used. The Priority Interrupt structure provides a means for peripheral slave devices to ask for an interrupt of other processor (DTB master) activity and receive service from the processor. The MC68153 BIM acts as an interface device requesting and responding to interrupt acknowledge cycles for up to 4 independent slaves:

In Figure 5, functional modules are identified as Interrupters and an Interrupt Handler. An Interrupter (such as the MC68153) receives slave requests for an interrupt and handles all interface to the system bus required to ask for and respond to interrupt requests. The Interrupt Handler receives the bus interrupt requests, determines when an interrupt acknowledge will occur and at which level, and finally either performs the interrupt acknowledge (IACK) cycle or tells the DTB master to execute the IACK cycle.

The signal lines in the Priority Interrupt structure include (\* — indicates active low):

1. IRQ1\*–IRQ7\* — seven prioritized interrupt request lines.

2. IACK\* — signal line that indicates an interrupt acknowledge cycle is occurring.
3. IACKIN\*/IACKOUT\* — two signals that form part of a daisy chain that prioritizes interrupters.

In addition Data Transfer Bus control signals are involved in the IACK bus cycle:

1. AS\* — the Address Strobe asserted low indicates a valid address is on the bus.
2. DSO\* — the lower Data Strobe asserted low indicates a data transfer will occur on bus bits D00–D07.
3. WRITE\* — the Read/Write is negated indicating the data is to be read from the Interrupter.
4. A01–A03 — Address lines A01–A03 contain the encoded priority level of the IACK cycle.
5. D00–D07 — Data bus lines D00–D07 are used to pass the interrupt vector from the responding Interrupter to the Interrupt Handler.
6. DTACK\* — Data Transfer Acknowledge asserted low signals that the Interrupter has put the vector on the data bus.

FIGURE 5 — SIMPLE VMEbus CONFIGURATION

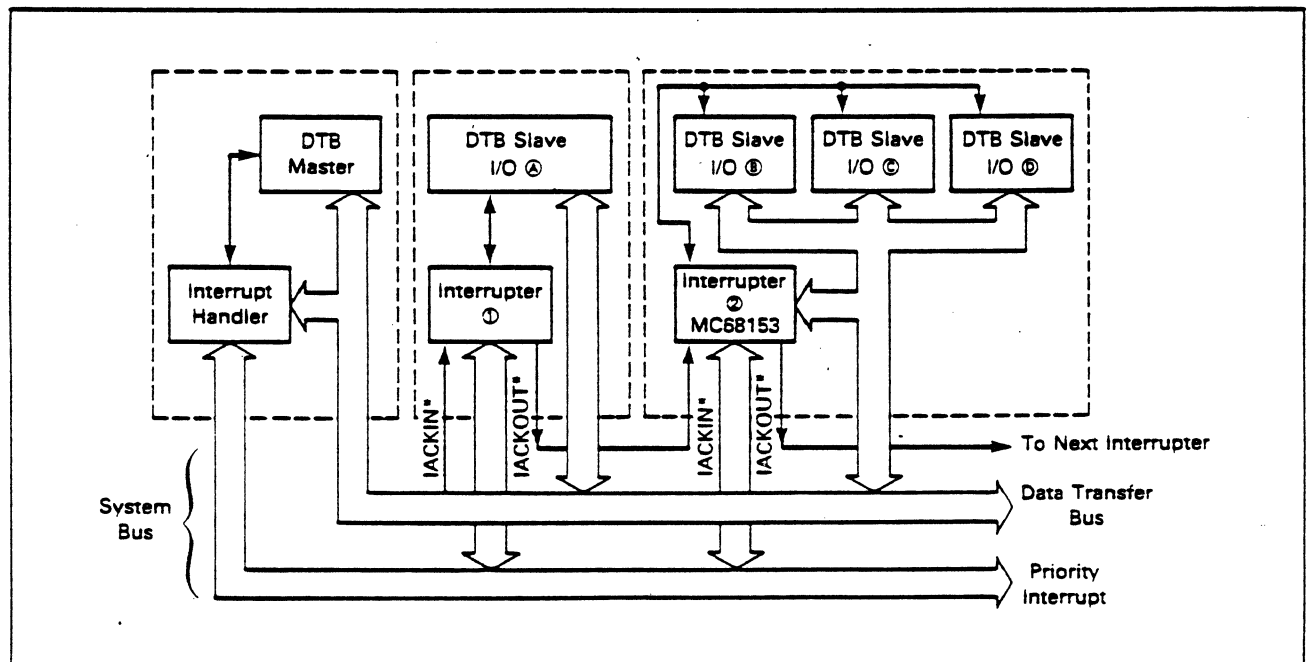
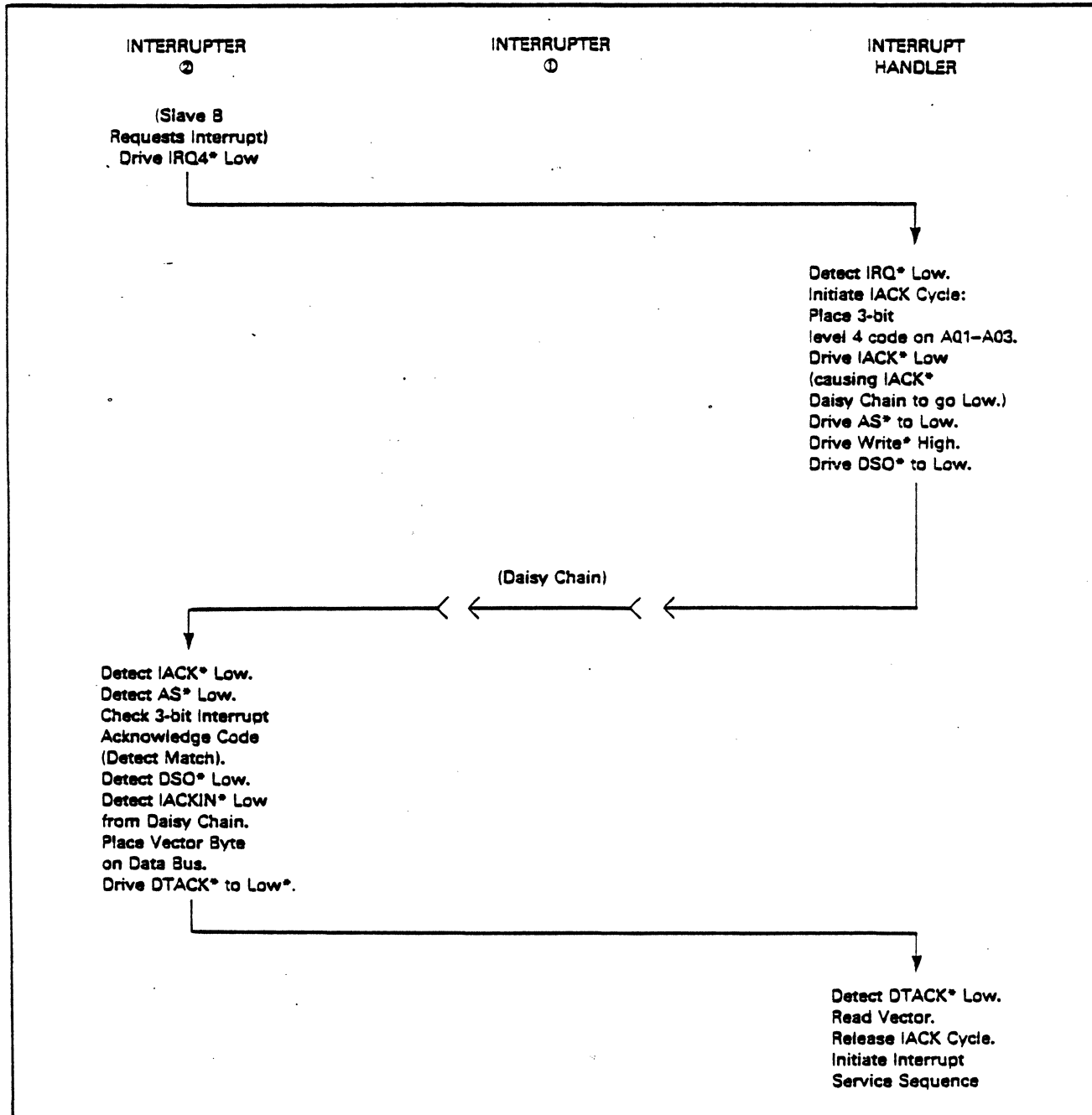


Figure 6 shows a flow diagram of a typical interrupt request and acknowledge operation. Briefly, the sequence of events is first, an Interrupter makes a request, next the Handler responds with an IACK cycle, then the Interrupter passes a vector to the Handler completing the IACK cycle, and finally the Handler uses the vector to determine additional action. Typically, an interrupt service routine is stored in software and the vector points to its starting address.

Note the daisy chain operation. If the IACK level (on A01-A03) does not match the Interrupter's request level or if no request is pending, the Interrupter passes the IACKIN\* signal on and asserts IACKOUT\*. This sequential action automatically prioritizes Requesters on the same level (first one in line with a request pending gets serviced) and prevents two or more Interrupters from responding simultaneously.

FIGURE 6 — INTERRUPT REQUEST AND ACKNOWLEDGE OPERATION FLOW DIAGRAM



This discussion is a very cursory look at the bus operation. For more details including situations with multiple bus masters, the user is directed to the VMEbus Specification MVMEBS or VERSAbus Specification M68KVBS. Also, the MC68153 can be used with other buses having similar interrupt structures.

### BIM BUS INTERFACE

Figure 7 shows a simplified block diagram of the MC68153 interface to VERSAbus or VMEbus. Address Decode and Control Decode are dependent on the application and must be designed to guarantee BIM ac specifications. It is possible in most cases that the decode logic can be shared with the slave devices. Buffers are provided where shown to comply with bus loading and drive specifications. It is also possible that buffers can be shared with the slave bus interface.

### READ/WRITE OPERATION

All eight BIM registers can be accessed from the sys-

tem bus in both read and write modes. The BIM has an asynchronous bus interface, primarily designed for MC68000-like buses. The following signals generate read and write cycles: Chip Select ( $\overline{CS}$ ), Read/Write ( $R/\overline{W}$ ), Address Inputs (A1-A3), Data Bus (D0-D7), and Data Transfer Acknowledge ( $\overline{DTACK}$ ). During read and write cycles the internal registers are selected by A1, A2, and A3 in compliance with the Figure 4 Truth Table.

Figure 8 shows the device timing for a read cycle.  $R/\overline{W}$  and A1-A3 are latched on the falling edge of  $\overline{CS}$  and must meet specified setup and hold times. Chip access time for valid data and  $\overline{DTACK}$  are dependent on the clock frequency as shown in the figure.

Figure 9 shows the device timing for a write cycle.  $R/\overline{W}$ , A1-A3, and D0-D7 are latched on the falling edge of  $\overline{CS}$  and must meet specified setup and hold times. Chip access time for  $\overline{DTACK}$  is dependent on the clock frequency as shown in the figure.

FIGURE 7 — VMEbus/VERSAbus INTERFACE BLOCK DIAGRAM

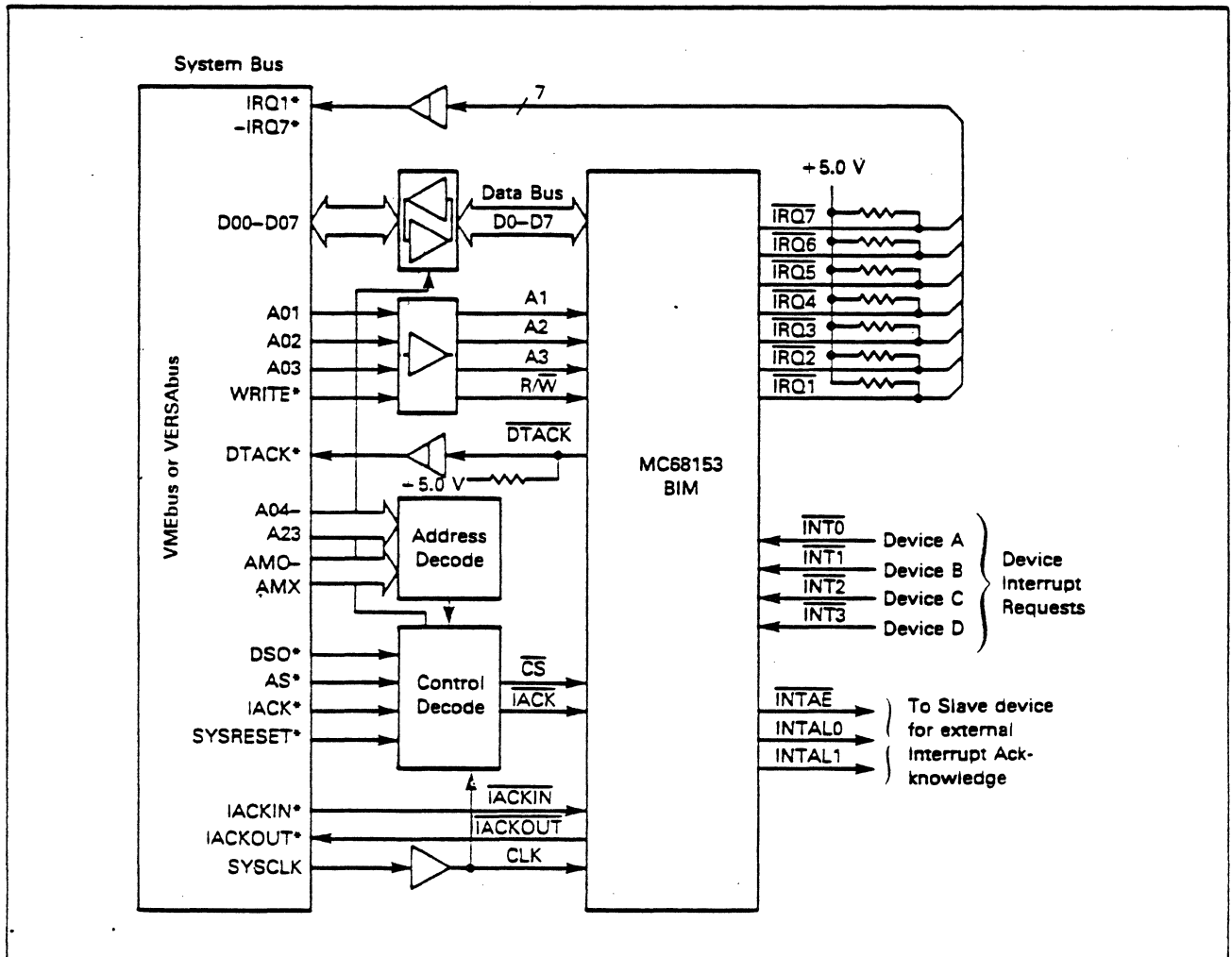


FIGURE 8 — READ CYCLE

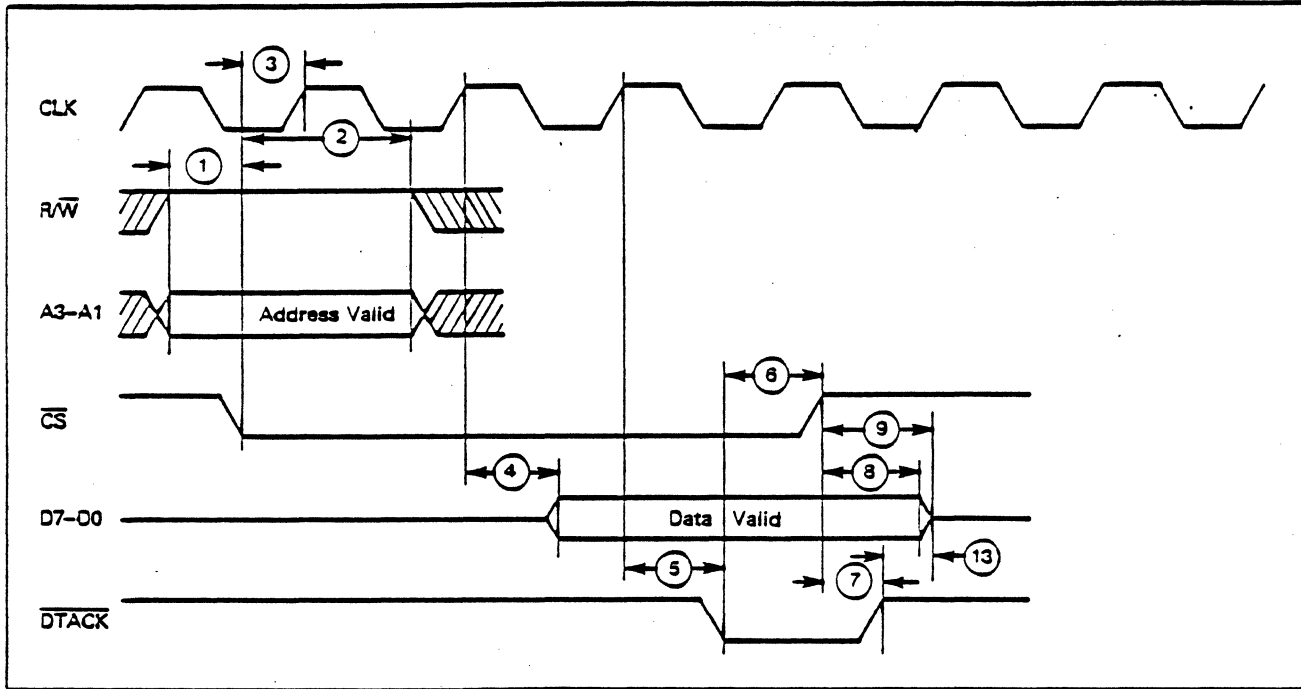
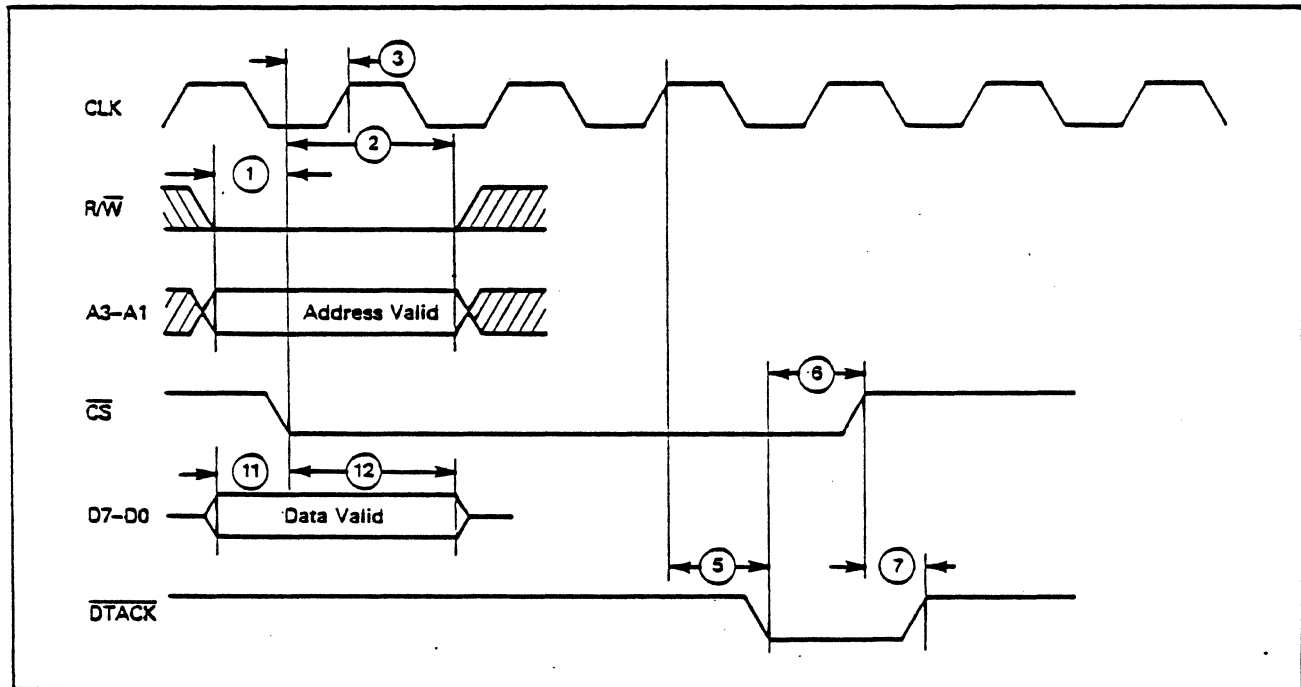


FIGURE 9 — WRITE CYCLE



## INTERRUPT REQUESTS

The MC68153 accepts device interrupt requests on inputs  $\overline{INT0}$ ,  $\overline{INT1}$ ,  $\overline{INT2}$ , and  $\overline{INT3}$ . Each input is regulated by Bit 4 (IRE) of the associated control register (CR0 controls  $\overline{INT0}$ , CR1 controls  $\overline{INT1}$ , etc). If IRE (Interrupt Enable) is set and a device input is asserted, an Interrupt Request open-collector output ( $\overline{IRQ1}$ – $\overline{IRQ7}$ ) is asserted. The asserted  $\overline{IROX}$  output is selected by the value programmed in Bits 0, 1, and 2 of the control register (L0, L1, and L2). This 3-bit field determines the interrupt request level as set by software.

Two or more interrupt sources can be programmed to the same request level. That  $\overline{IROX}$  output will remain asserted until multiple interrupt acknowledge cycles respond to all requests.

If the interrupt request level is set to zero, the interrupt is disabled because there is no corresponding IRQ output.

## INTERRUPT ACKNOWLEDGE

The response of an Interrupt Handler to a bus interrupt request is an interrupt acknowledge cycle. The IACK cycle is initiated in the MC68153 by receiving  $\overline{IACK}$  low.  $R/\overline{W}$ , A1, A2, A3 are latched, and the interrupt level on line A1–A3 is compared with any interrupt requests pending in the chip. Further activity can be one of four cases:

1. No further action required — This occurs if  $\overline{IACKIN}$  is not asserted. Asserting  $\overline{IACK}$  only starts the BIM activity. If the daisy chain signal never reaches the MC68153 ( $\overline{IACKIN}$  is not asserted), another Interrupter has responded to the  $\overline{IACK}$  cycle. The cycle will end, the chip  $\overline{IACK}$  is negated, and no additional action is required.
2. Pass on the interrupt acknowledge daisy chain — For this case,  $\overline{IACKIN}$  input is asserted by the preceding daisy chain interrupter, and  $\overline{IACKOUT}$  output is in turn asserted. The daisy chain signal is passed on when no interrupts are pending on a matching level or when any possible interrupts are disabled. The Interrupt Enable (IRE) bit of a control register can disable any interrupt requests, and in turn, any possible matches.
3. Respond internally — For this case,  $\overline{IACKIN}$  is asserted and a match is found. The MC68153 completes the IACK cycle by supplying an interrupt vector from the proper vector register followed by a  $\overline{DTACK}$  signal asserted.  $\overline{IACKOUT}$  is not asserted because the interrupt acknowledge cycle is completed by this device.

For the MC68153 to respond in this mode of operation, the EXTERNAL/INTERNAL control register bit ( $X/\overline{IN}$ ) must be zero. For each source of interrupt request, the associated control register determines the BIM response to an IACK cycle, and the  $X/\overline{IN}$

bit sets this response either internally ( $X/\overline{IN} = 0$ ) or externally ( $X/\overline{IN} = 1$ ).

4. Respond externally — For the final case,  $\overline{IACKIN}$  is also asserted, a match is found and the associated control register has  $X/\overline{IN}$  bit set to one. The MC68153 does not assert  $\overline{IACKOUT}$  and does assert  $\overline{INTAE}$  low.  $\overline{INTAE}$  signals that the requesting device must complete the IACK cycle (supplying a vector and  $\overline{DTACK}$ ) and that the 2-bit code contained on outputs INTAL0 and INTAL1 shows which interrupt source is being acknowledged.

These cases are discussed in more detail in the following paragraphs.

### Internal Interrupt Acknowledge

For an internal interrupt acknowledge to occur, the following conditions must be met:

1. One or more device interrupt inputs ( $\overline{INT0}$ – $\overline{INT3}$ ) has been asserted and corresponding control bit IRE value is one.
2.  $\overline{IACK}$  asserted.
3. A match exists between [A3, A2, A1] and the [L2, L1, L0] field of an enabled, requesting control register. If two or more devices are requesting at the same interrupt level, preference is given to the highest number requester, that is,  $\overline{INT3}$  has highest priority and  $\overline{INT0}$  has lowest.
4. Control register bit  $X/\overline{IN}$  of matching interrupt source must be zero.
5.  $\overline{IACKIN}$  asserted.

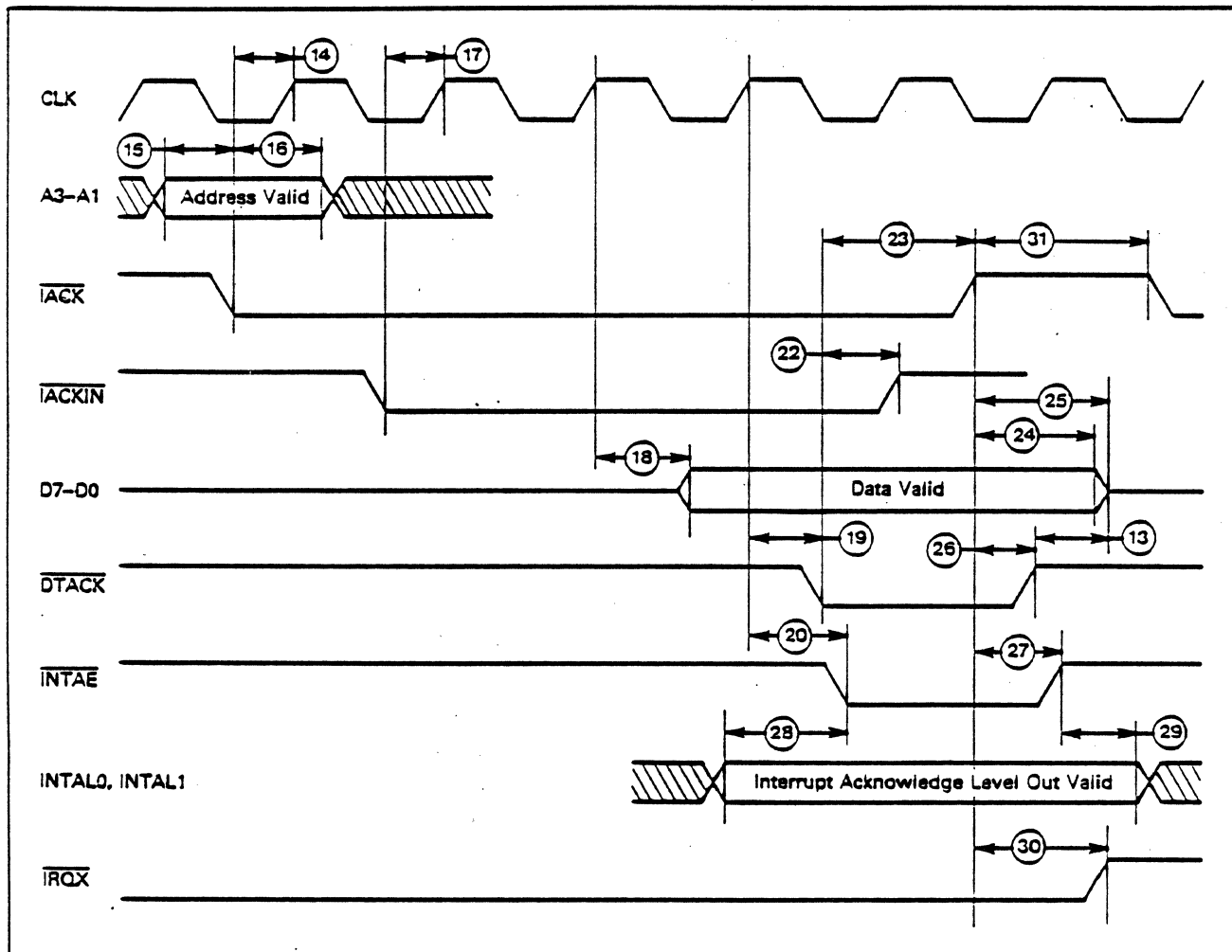
The internal interrupt acknowledge cycle timing is shown in Figure 10. The 8-bit interrupt acknowledge vector is presented to the data bus and  $\overline{DTACK}$  is asserted. Note also that INTAL0 and INTAL1 are valid and  $\overline{INTAE}$  is asserted during this cycle although they would normally not be used. The cycle is terminated (data and  $\overline{DTACK}$  released) after  $\overline{IACK}$  is negated.

During the IACK cycle, the INTERRUPT AUTO-CLEAR control bit (IRAC) comes into play. If the IRAC = one for the responding interrupt source, the INTERRUPT ENABLE (IRE) bit is automatically cleared during the IACK cycle, thus disabling the associated interrupt input and any  $\overline{IROX}$  output asserted due to this interrupt input. Before another interrupt can be requested from this source, IRE must be set to one by writing to the control register.

Note that  $\overline{IACKOUT}$  is not asserted because this device is responding to the IACK and does not pass the daisy chain signal on. Also, new device interrupt requests occurring on  $\overline{INT0}$ – $\overline{INT3}$  after  $\overline{IACK}$  is asserted are locked out to prevent any race conditions on the daisy chain.



FIGURE 10 — INTERRUPT ACKNOWLEDGE CYCLE — INTERNAL VECTOR



#### External Interrupt Acknowledge

For an external interrupt acknowledge, the same conditions as listed above are met with one exception. Control register bit  $X/\overline{IN}$  of matching interrupt source must be set to one. The timing is shown in Figure 11. For this cycle, the interrupt vector and  $\overline{DTACK}$  must be supplied by an external device.  $\overline{INTAE}$  is asserted indicating that  $INTAL0$  and  $INTAL1$  are valid. The external device can use these signals to enable the vector and  $\overline{DTACK}$ . The cycle is terminated after  $\overline{IACK}$  is negated.

The IRAC control bit acts in the external interrupt acknowledge the same as described for the internal response (see above). Also,  $\overline{IACKOUT}$  is not asserted and new device interrupts are disabled for reasons discussed above.

#### Pass On IACK Daisy Chain

If the MC68153 has no interrupt request pending at the same level as the interrupt acknowledge, the  $\overline{IACK}$  daisy chain signal is passed on to the next device if  $\overline{IACKIN}$  is asserted. The following conditions are thus met:

1.  $\overline{IACK}$  asserted.
2. No match exists between [A3, A2, A1] and the [L2, L1, L0] field of an enabled, requesting control register.
3.  $\overline{IACKIN}$  is asserted.

$\overline{IACKOUT}$  is asserted if these conditions are valid. This output drives  $\overline{IACKIN}$  of the next Interrupter on the daisy chain, passing the signal along. Figure 12 shows the timing for this case.  $\overline{IACKOUT}$  is negated after  $\overline{IACK}$  is negated.





FIGURE 11 — INTERRUPT ACKNOWLEDGE CYCLE — EXTERNAL VECTOR

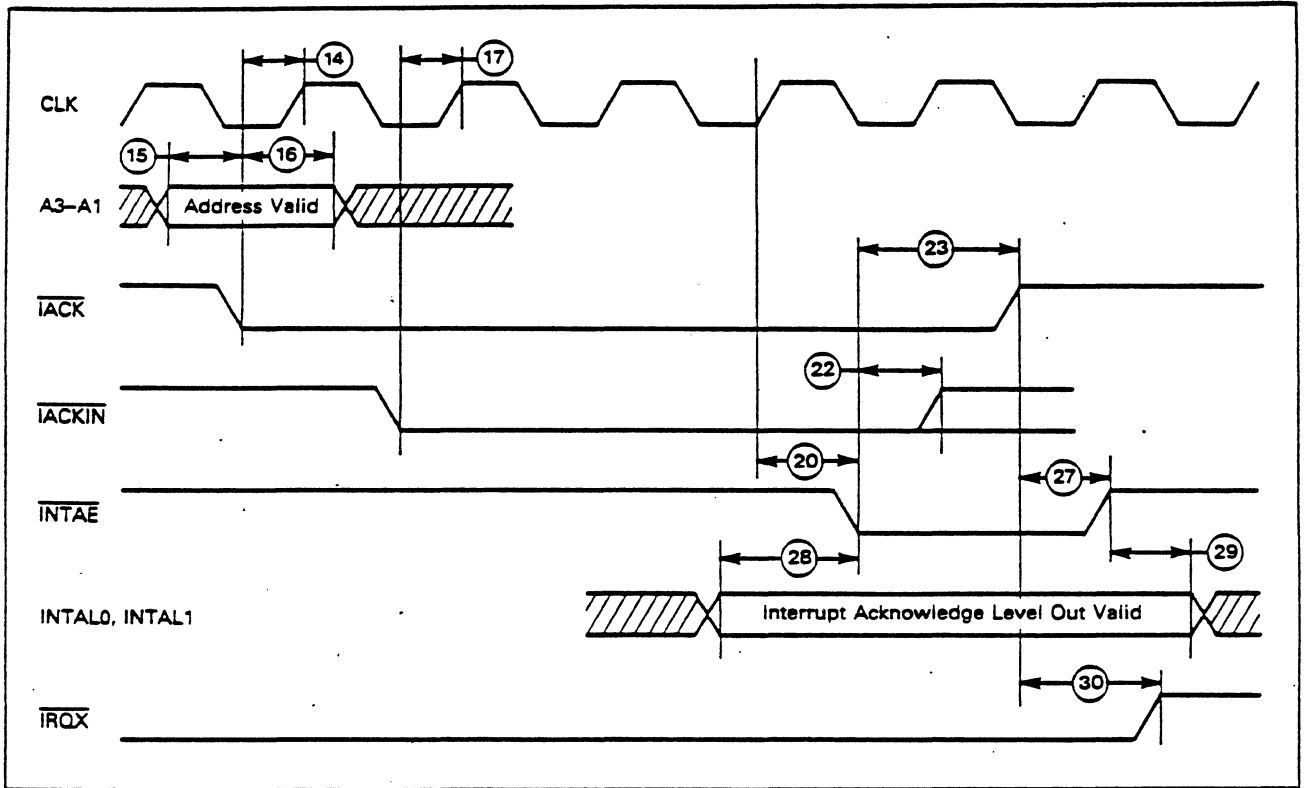
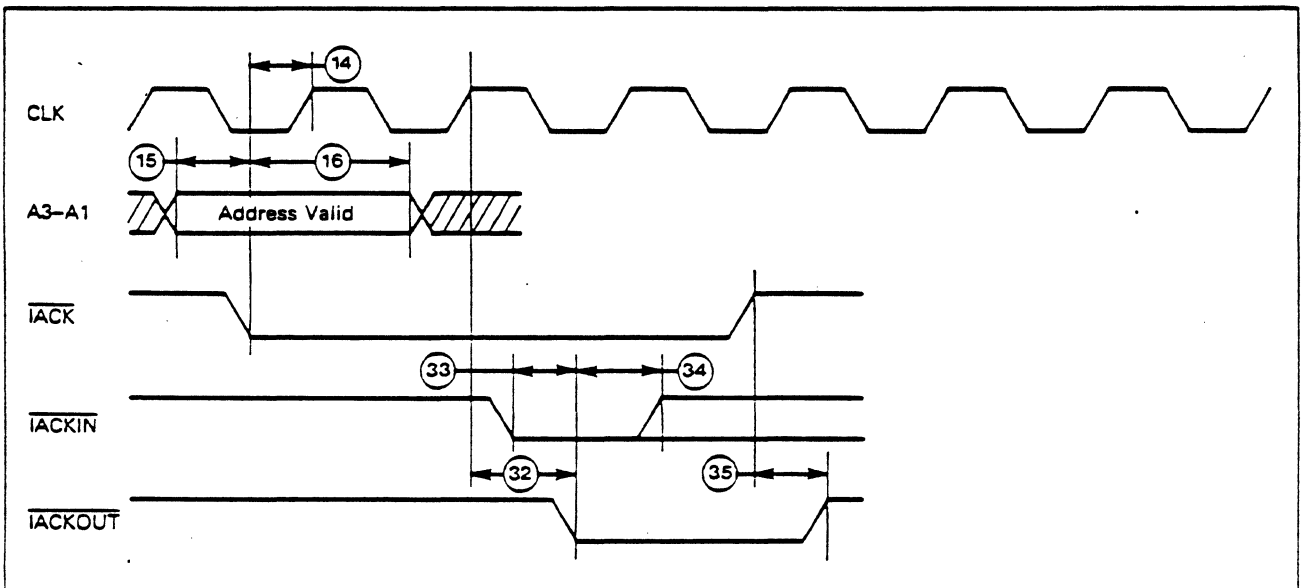


FIGURE 12 — INTERRUPT ACKNOWLEDGE CYCLE — IACKOUT



**CONTROL REGISTER FLAGS**

Each control register contains a Flag bit (F) and a Flag Auto-Clear bit (FAC). Both bits can be read or altered via a register write without affecting the interrupt operation of the device. The Flag is useful as a status indicator for resource management and as a semaphore in multitasking or multiprocessor systems. Flag (F) is located in bit position 7 and can be used with the MC68000 Test and Set (TAS) instruction.

The Flag Auto-Clear (FAC) is used to manipulate the Flag bit. If the Flag is set to one and the FAC is also one, an interrupt acknowledge cycle to the associated interrupt source clears the Flag bit. This feature is useful in determining the interrupt status and passing messages.

**RESET**

There is no reset input, however, a chip reset is activated by asserting both  $\overline{CS}$  and  $\overline{IACK}$  simultaneously (Figure 13). These inputs should be held low for a minimum of two clock cycles for a full reset function. The control registers are reset to all zeroes and the Vector Registers are set to a value of \$0F. This vector value is the uninitialized vector for the MC68000. See the MC68000 Users Manual for more details on this vector.

**CLOCK**

The chip clock is required for internal operation to occur. Typical frequency is 16 MHz in VMEbus and VERSAbus applications derived from the system clock. Any frequency can be used, however, up to 25 MHz (Figure 14).

FIGURE 13 — RESET

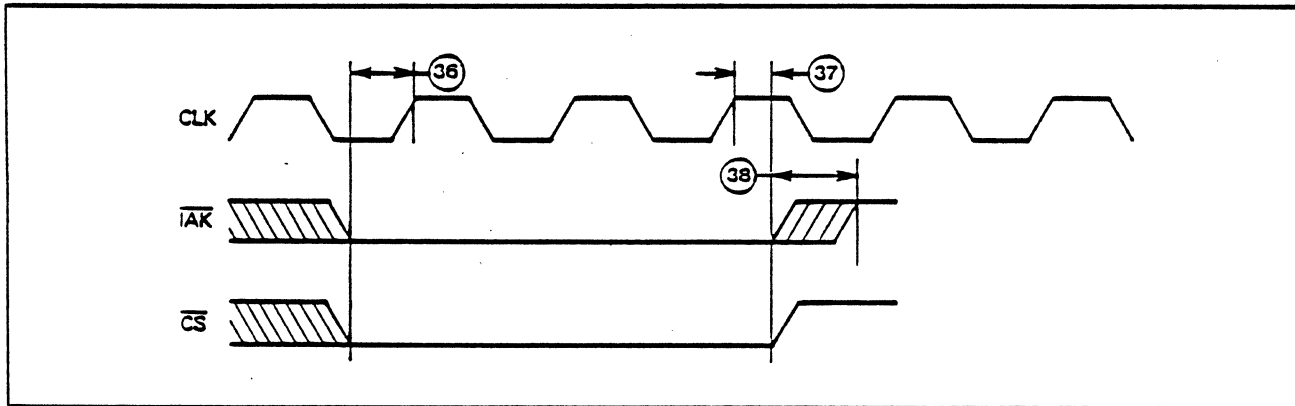
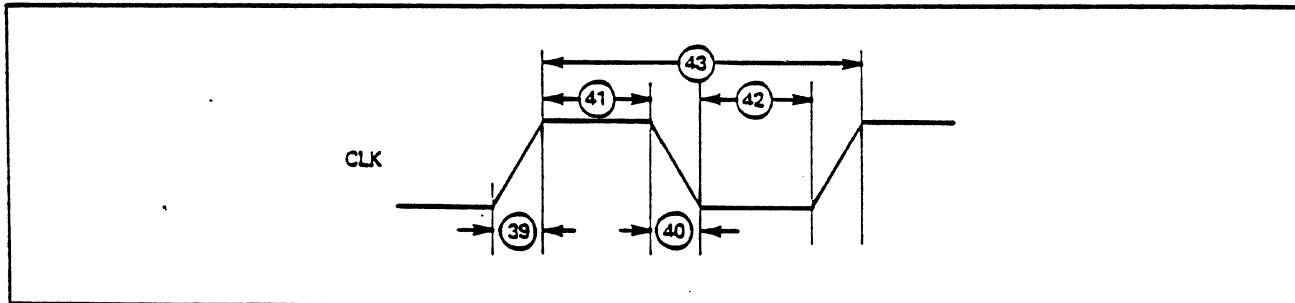


FIGURE 14 — CLOCK WAVEFORM



**TABLE 1**  
**AC PERFORMANCE SPECIFICATIONS**  
(VCC = 5.0 V ±5%, TA = 0°C to 70°C)

Number	Characteristic	Min	Max	Units	Notes
1	R/W, A1-A3 Valid to $\overline{CS}$ Low (Setup Time)	10	—	ns	
2	$\overline{CS}$ Low to R/W, A1-A3 Invalid (Hold Time)	5.0	—	ns	
3	$\overline{CS}$ Low to CLK High (Setup Time)	15	—	ns	1
4	CLK High to Data Out Valid (Delay)	—	55	ns	2
5	CLK High to $\overline{DTACK}$ Low (Delay)	—	40	ns	2
6	$\overline{DTACK}$ Low to $\overline{CS}$ High	0	—	ns	
7	$\overline{CS}$ High to $\overline{DTACK}$ High (Delay)	—	35	ns	10
8	$\overline{CS}$ High to Data Out Invalid (Hold Time)	0	—	ns	
9	$\overline{CS}$ High to Data Out High-Impedance (Hold Time)	—	50	ns	
10	$\overline{CS}$ High to $\overline{CS}$ or $\overline{IACK}$ Low	20	—	ns	
11	Data In Valid to $\overline{CS}$ Low (Setup Time)	10	—	ns	
12	$\overline{CS}$ Low to Data In Invalid (Hold Time)	5.0	—	ns	
13	$\overline{DTACK}$ High to Data Out High-Impedance	—	25	ns	10
14	$\overline{IACK}$ Low to CLK High (Setup Time)	15	—	ns	1
15	A1-A3 Valid to $\overline{IACK}$ Low (Setup Time)	10	—	ns	
16	$\overline{IACK}$ Low to A1-A3 Invalid (Hold Time)	5.0	—	ns	
17	$\overline{IACKIN}$ Low to CLK High (Setup Time)	15	—	ns	1, 8
18	CLK High to Data Out Valid (Delay)	—	55	ns	3
19	CLK High to $\overline{DTACK}$ Low (Delay)	—	40	ns	3
20	CLK High to $\overline{INTAE}$ Low (Delay)	—	40	ns	3
22	$\overline{DTACK}$ Low to $\overline{IACKIN}$ High	0	—	ns	8
23	$\overline{DTACK}$ Low to $\overline{IACK}$ High	0	—	ns	
24	$\overline{IACK}$ High to Data Out Invalid (Hold Time)	0	—	ns	
25	$\overline{IACK}$ High to Data Out High Impedance (Delay)	—	60	ns	
26	$\overline{IACK}$ High to $\overline{DTACK}$ High (Delay)	—	45	ns	10
27	$\overline{IACK}$ High to $\overline{INTAE}$ High (Delay)	—	35	ns	
28	$\overline{INTAL0}$ , $\overline{INTAL1}$ Valid to $\overline{INTAE}$ Low (Setup Time)	1.0	2.0	CLK Per	
29	$\overline{INTAE}$ High to $\overline{INTAL0}$ , $\overline{INTAL1}$ Invalid (Hold Time)	1.0	2.0	CLK Per	
30	$\overline{IACK}$ High to $\overline{IROx}$ High (Delay)	—	50	ns	7, 10
31	$\overline{IACK}$ High to $\overline{IACK}$ or $\overline{CS}$ Low	20	—	ns	
32	CLK High to $\overline{IACKOUT}$ Low (Delay)	—	40	ns	5
33	$\overline{IACKIN}$ Low to $\overline{IACKOUT}$ Low (Delay)	—	30	ns	4, 8
34	$\overline{IACKOUT}$ Low to $\overline{IACKIN}$ , $\overline{IACK}$ High	0	—	ns	8
35	$\overline{IACK}$ High to $\overline{IACKOUT}$ High (Delay)	—	35	ns	
36	$\overline{IACK}$ and $\overline{CS}$ both Low to CLK High (Setup Time)	15	—	ns	9
37	CLK High to $\overline{IACK}$ or $\overline{CS}$ High (Hold Time)	0	—	ns	
38	$\overline{IACK}$ or $\overline{CS}$ High to $\overline{IACK}$ and $\overline{CS}$ High (Skew)	—	1.0	CLK Per	6
39	Clock Rise Time	—	10	ns	
40	Clock Fall Time	—	10	ns	
41	Clock High Time	20	—	ns	
42	Clock Low Time	20	—	ns	
43	Clock Period	40	—	ns	

**NOTES:**

- This specification only applies if the VBIM had completed all operations initiated by the previous bus cycle when  $\overline{CS}$  or  $\overline{IACK}$  was asserted. Following a normal bus cycle, all operations are completed within 2 clock cycles after  $\overline{CS}$  or  $\overline{IACK}$  have been negated. If  $\overline{IACK}$  or  $\overline{CS}$  is asserted prior to completion of these operations, the new cycle, and hence,  $\overline{DTACK}$  is postponed.  
If the  $\overline{IACK}$ ,  $\overline{IACKIN}$  or  $\overline{CS}$  setup time is violated,  $\overline{DTACK}$  may be asserted as shown, or may be asserted one clock cycle later (i.e.  $\overline{IACK}$  will not be recognized until the next rising edge of the clock).
- Assumes that 3 has been met.
- Assumes that 14 and 17 have both been met.
- Assumes that 14 has been met. ( $\overline{IACKOUT}$  cannot go low prior to  $\overline{IACKIN}$  going low).
- Assumes that 14 has been met and  $\overline{IACKIN}$  has been low for at least the amount of time specified by 33.
- 38 is the minimum skew between the last moment when both  $\overline{IACK}$  and  $\overline{CS}$  are asserted to when both are negated, to insure that an access cycle is not unintentionally started.
- Assumes no other  $\overline{INTx}$  input is causing  $\overline{IROx}$  to be driven low.
- In non-daisy chain systems,  $\overline{IACKIN}$  may be tied low.
- Failure to meet this spec. causes RESET to be ignored for 1 clock period. It is then necessary to keep these signals low for 3 clock periods instead of 2.
- Delay time is specified from Input signal to Open-Collector Output pulled High thru 1.0 kΩ resistor to +6.5 V.



OUTLINE DIMENSIONS

DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	51.69	52.45	2.035	2.065
B	13.72	14.22	0.540	0.560
C	3.94	5.08	0.155	0.200
D	0.38	0.56	0.014	0.022
F	1.02	1.52	0.040	0.060
G	2.54 BSC		0.100 BSC	
N	1.65	2.16	0.065	0.085
J	3.20	3.58	0.125	0.141
K	2.92	3.43	0.115	0.135
L	15.24 BSC		0.600 BSC	
M	5.0	15.0	5.0	15.0
N	3.51	1.02	0.020	0.040

**NOTES:**

1. POSITIONAL TOLERANCE OF LEADS (D), SHALL BE WITHIN 0.25 mm (0.010) AT MAXIMUM MATERIAL CONDITION, IN RELATION TO SEATING PLANE AND EACH OTHER.
2. DIMENSION L TO CENTER OF LEADS WHEN FORMED PARALLEL.
3. DIMENSION S DOES NOT INCLUDE MOLD FLASH.

**CASE 711-03  
PLASTIC PACKAGE**

DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	51.31	53.24	2.020	2.096
B	12.70	15.49	0.500	0.610
C	4.06	5.84	0.160	0.230
D	0.38	0.56	0.015	0.022
F	1.27	1.65	0.050	0.065
G	2.54 BSC		0.100 BSC	
J	0.20	0.30	0.008	0.012
K	3.18	4.06	0.125	0.160
L	15.24 BSC		0.600 BSC	
M	5.0	15.0	5.0	15.0
N	3.51	1.27	0.020	0.050

**NOTES:**

1. DIM A - IS DATUM.
2. POSITIONAL TOLERANCE FOR LEADS:  $\pm 0.25(0.010) \text{ (I) TIA (D)}$
3.  $\text{SEATING PLANE}$
4. DIM L TO CENTER OF LEADS WHEN FORMED PARALLEL.
5. DIMENSIONS A AND B INCLUDE MENISCUS.
6. DIMENSIONING AND TOLERANCING PER ANSI Y14.5, 1973.

**CASE 734-04  
CERAMIC PACKAGE**

TYPICAL THERMAL CHARACTERISTICS

Package	$\theta_{JA}$ (Junction to Ambient) Still Air	Junction Temperature Still Air @ 70°C Ambient
L Suffix	40°C/W	147°C
P Suffix	35°C/W	137°C

Motorola reserves the right to make changes without further notice to any products herein to improve reliability, function or design. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. Motorola and  $\text{M}$  are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Employment Opportunity/Affirmative Action Employer.



**MOTOROLA Semiconductor Products Inc.**

BOX 20912 • PHOENIX, ARIZONA 85036 • A SUBSIDIARY OF MOTOROLA INC.

# Am8151

Graphics Color Palette (GCP)



Am8151

Advanced Micro Devices

October 1985

## DISTINCTIVE CHARACTERISTICS

- A 256 x 8 color map and a video DAC on a single chip
- Pixel rates up to 200 MHz
- DAC inputs for Overlay, Blank, Vsync, and Hsync
- Connects directly to 50  $\Omega$  and 75  $\Omega$  RS-343 monitor
- Three Am8151s in parallel allow 256 simultaneous colors out of a palette of over 16 million colors

## GENERAL DESCRIPTION

The Am8151 Graphics Color Palette (GCP) provides a color lookup table and a video DAC on a single chip for use in high performance graphics systems. The video DAC incorporates overlay, blank, and sync levels as well as a 256 level gray scale. Three Am8151s connected in parallel provide three outputs (Red, Green, and Blue) and allow simultaneous display of 256 colors from a palette of over 16 million colors.

The Am8151 has three levels of pipeline to allow a high pixel rate and to ease system timing requirements. The first level is prior to a 256 x 8 color lookup table in high-speed RAM. Eight bits of color address,  $A_0 - A_7$  (ECL or TTL) are latched and used to select one of 256 intensities to be fed to the DAC. At the second level, the intensity selected is latched at the output of the RAM prior to being decoded to select which of the 32 current sources will be turned on. An additional high-speed register between the DAC Decoder and the DAC ensures that all DAC inputs switch simultaneously, thereby reducing the maximum duration of the

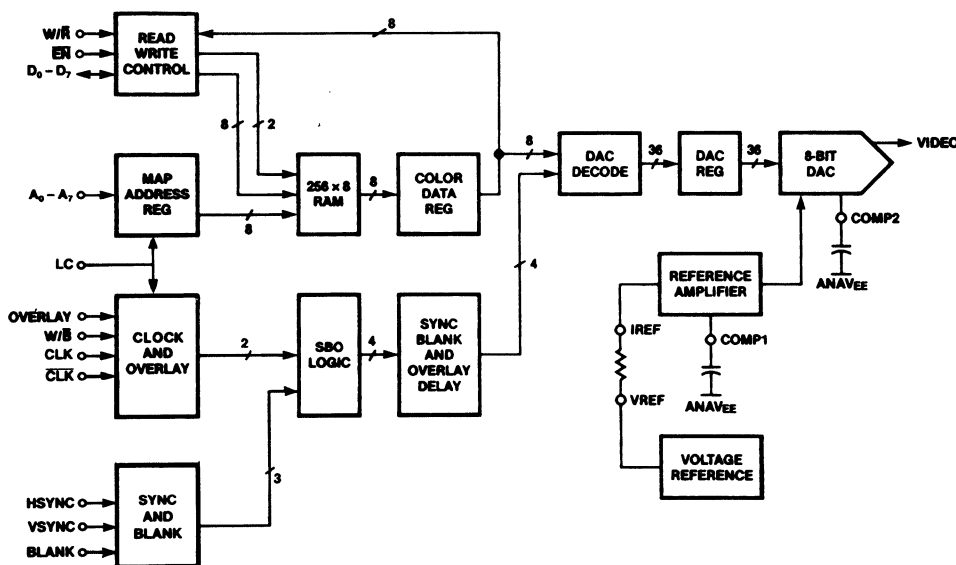
glitch at the output of the DAC. This additional register adds a third level of pipeline to the pixel data path.

The color lookup table is stored in RAM and may therefore be read and written by a graphics processor. In addition to the 8 address lines, and 8 control lines, an 8-bit data path  $D_0 - D_7$  (TTL), and two control signals  $\overline{EN}$  (TTL) and  $W/\overline{R}$  (TTL), are provided for this purpose.

The text capability of the AMD Alphanumeric Display Products or a generic overlay may be added with the OVERLAY (ECL or TTL) and  $W/\overline{B}$  (ECL or TTL) inputs. A HIGH on the OVERLAY input overrides the color pixel data and drives the DAC output to peak white or reference black depending on the state of the  $W/\overline{B}$  input. When three Am8151s are used in a system these pins can be connected to provide text or overlay capability in eight colors.

The Am8151 GCP is a part of the AMD Display Products Family which also includes the Am8150 Display Refresh Controller, the Am8157 Video Shift Register, the Am8177 Video Data Serializer and the Am8158 Video Timing Controller.

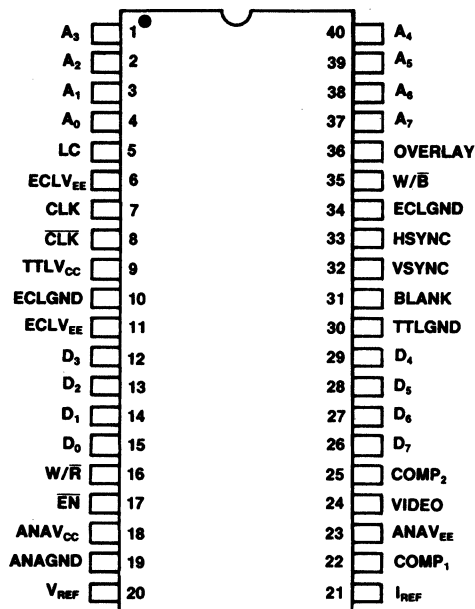
## BLOCK DIAGRAM



BD005032

Order # 04653D

### CONNECTION DIAGRAM Top View

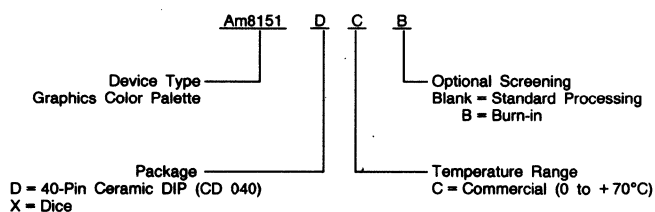


CD005972

Note: Pin 1 is marked for orientation

### ORDERING INFORMATION

AMD products are available in several packages and operating ranges. The order number is formed by a combination of the following: Device number, speed option (if applicable), package type, operating range and screening option (if desired).



Valid Combinations	
Am8151	DC, DCB, XC

#### Valid Combinations

Consult the local AMD sales office to confirm availability of specific valid combinations, to check on newly released valid combinations, and to obtain additional data on AMD's standard military grade products.

## PIN DESCRIPTION

<b>LC</b>	<p><b>LEVEL CONTROL (Input)</b> Level Control determines the logic compatibility of the twelve TTL/ECL input pins. If LC is tied to V<sub>CC</sub>, the logic levels are TTL. If LC is tied to ground, the logic levels are ECL.</p>	<b>W/<math>\bar{B}</math></b>	<p><b>WHITE/BLACK (Input, TTL/ECL)</b> The W/<math>\bar{B}</math> pin determines the level the OVERLAY pin will force the DAC output to. When W/<math>\bar{B}</math> is HIGH, a HIGH on the OVERLAY pin will force the DAC output to a peak white level. When W/<math>\bar{B}</math> is LOW, a HIGH on the OVERLAY pin will force the DAC output to a reference black level. The logic compatibility of this pin is determined by the LC pin.</p>
<b>A<sub>0</sub>-A<sub>7</sub></b>	<p><b>ADDRESS (Inputs, TTL/ECL)</b> These eight pins are used to address data stored in the color lookup table. The address on these pins is latched on the first rising CLK edge and decoded to select one of 256 intensities stored in the color lookup table. During a video refresh these pins should be connected to the color pixel data. During a color lookup table update these pins should be connected to the graphics processor's address bus. The logic compatibility of these pins is determined by the LC pin.</p>	<b>BLANK</b>	<p><b>BLANKING (Input, TTL)</b> The BLANK pin, when active, overrides the color pixel and overlay data to force the DAC output to a "blacker than black" blank level. A "blacker than black" level is required during the monitor's horizontal and vertical retrace. The blank signal is kept synchronized with the pixel data inside the Am8151 by delaying the blank signal the same number of clock cycles as the pixel data.</p>
<b>D<sub>0</sub>-D<sub>7</sub></b>	<p><b>DATA (Inputs/Outputs, TTL)</b> These eight pins are used to write data in to the color lookup table or to read data out of the color lookup table. The MSB is D<sub>7</sub>.</p>	<b>VSYNC</b> <b>HSYNC</b>	<p><b>VERTICAL SYNC (Input, TTL)</b> <b>HORIZONTAL SYNC (Input, TTL)</b> The HSYNC and VSYNC signals are internally x-ored to generate a composite sync signal. The composite sync signal, when HIGH, overrides the pixel, overlay, and blank signals to force the DAC output to a sync level. If both HSYNC and VSYNC are HIGH (composite sync is LOW) the DAC output is forced to a blank level. The composite signal is kept synchronized with the pixel data inside the Am8151 by delaying the sync signal the same number of clock cycles as the pixel data.</p>
<b>W/<math>\bar{R}</math></b>	<p><b>WRITE/READ (Input, TTL)</b> The W/<math>\bar{R}</math> controls the direction of color lookup table access by the system processor. When W/<math>\bar{R}</math> is HIGH and <math>\bar{E}N</math> is LOW, data is written in to the color lookup table. When W/<math>\bar{R}</math> is LOW and <math>\bar{E}N</math> is LOW, data is read from the color lookup table.</p>	<b>VIDEO</b>	<p><b>VIDEO (Output, Analog)</b> The VIDEO pin is the output of the DAC and is intended to directly drive most monitor inputs which are terminated into 50 or 75<math>\Omega</math>.</p>
<b><math>\bar{E}N</math></b>	<p><b>ENABLE (Input, TTL)</b> The <math>\bar{E}N</math> pin is used to enable color lookup table data onto the data bus D<sub>0</sub>-D<sub>7</sub> during a read operation and to enable a write into the color lookup table during a write operation. When <math>\bar{E}N</math> is HIGH, the eight data lines D<sub>0</sub>-D<sub>7</sub> are three-stated.</p>	<b>VREF</b>	<p><b>VOLTAGE REFERENCE (Output, Analog)</b> The VREF pin provides a precision reference voltage for use in setting the full scale current of the DAC. The reference input current (I<sub>REF</sub>) for the DAC, which determines the full scale current, may be generated from VREF by connecting an external resistor from VREF to IREF. The reference resistor value may be calculated by the relation <math>R_{REF} = 28.56/I_{FS}</math>.</p>
<b>CLK, <math>\bar{C}LK</math></b>	<p><b>CLOCK, <math>\bar{C}LK</math> (Inputs, TTL/ECL)</b> CLK and <math>\bar{C}LK</math> are the pixel clock inputs. In ECL mode these pins operate differentially. In TTL mode, <math>\bar{C}LK</math> must be tied to ground. The clock is used internally to latch the address pins, data at the output of the color lookup table, and the decoded DAC inputs. The logic compatibility of these pins is controlled by the LC pin.</p>	<b>IREF</b>	<p><b>REFERENCE CURRENT (Input, Analog)</b> A scaling current to the DAC should be provided at the IREF pin. The full scale current of the DAC can be determined from the relation <math>I_{FS} = 13.27 I_{REF}</math>.</p>
<b>OVERLAY</b>	<p><b>OVERLAY (Input, TTL/ECL)</b> The OVERLAY pin, when active, overrides the color pixel data to force the DAC output to a peak white or reference black level. The level the DAC output is forced to is set by the W/<math>\bar{B}</math> pin. The overlay signal is kept synchronized with the color pixel data inside the Am8151 by delaying the overlay signal the same number of clock cycles as the color pixel data. The logic compatibility of this pin is determined by the LC pin.</p>	<b>COMP<sub>1</sub>, COMP<sub>2</sub></b>	<p><b>COMPENSATION PINS</b> COMP<sub>1</sub> and COMP<sub>2</sub> are used to connect external compensation capacitors to the reference amp and DAC. It is recommended that a 0.1<math>\mu</math>F and a 100pF capacitor be connected from each pin to ANAV<sub>EE</sub>.</p>

<b>TTLVCC</b>	<b>TTL POSITIVE SUPPLY</b> Positive supply voltage for the TTL portions of the chip.	<b>ANAVEE</b>	<b>ANALOG NEGATIVE SUPPLY</b> Negative supply voltage for the analog portions of the chip.
<b>ANAVCC</b>	<b>ANALOG POSITIVE SUPPLY</b> Positive supply voltage for the analog portions of the chip.	<b>TTLGND</b>	<b>TTL GROUND</b> Ground for the TTL portions of the chip.
<b>ECLVEE</b> (2 pins)	<b>ECL NEGATIVE SUPPLY</b> Negative supply voltage for the ECL portions of the chip.	<b>ECLGND</b> (2 pins)	<b>ECL GROUND</b> Ground for the ECL portions of the chip.
		<b>ANAGND</b>	<b>ANALOG GROUND</b> Ground for the analog portions of the chip.

## FUNCTIONAL DESCRIPTION

The Am8151 is a Graphics Color Palette (GCP) providing a color lookup table and a video DAC for use in high performance graphics systems. The Am8151 is pipelined with digital color pixel data, overlay, blank, and sync inputs entering the pipeline, and an analog signal exiting 3 CLK cycles later. The three levels of pipeline are prior to the 256 x 8 RAM color lookup table, the DAC decoder, and the 36 current sources to generate an analog signal.

### Color Lookup Table

Eight lines of color pixel data are read through the pins  $A_0 - A_7$  and latched into the Address Registers on the first rising edge of the CLK.  $A_0 - A_7$  may be at either ECL or TTL logic levels. The LC pin is used to select the logic compatibility of these pins. These eight lines ( $A_0 - A_7$ ) are used as an address for the color lookup table and are decoded to select one of the 256 intensities stored in the color lookup table. Each intensity stored is 8 bits wide, with 255 corresponding to reference white and 0 corresponding to reference black. On the next rising CLK edge the intensity is latched into the color data registers to be decoded for the DAC. On a third rising CLK edge the decoded DAC inputs are latched and used to turn on or off the current sources making up the DAC.

### Color Lookup Table Update

The color lookup table may be loaded and read back by the graphics processor. For this purpose 8 bidirectional data lines ( $D_0 - D_7$ ) have been provided.  $D_7$  is the Most Significant Bit (MSB) and  $D_0$  is the Least Significant Bit (LSB) of data. In addition to the 8 data lines, 2 control lines are provided ( $\overline{EN}$  and  $W/\overline{R}$ ).  $\overline{EN}$  is an active LOW input that selects the chip for both write and read operations. When  $\overline{EN}$  is HIGH, the 8 bidirectional data lines are three-stated.  $W/\overline{R}$  controls the direction of the operation. If this pin is LOW, color lookup table data is read from the table and placed on the data lines  $D_0 - D_7$ . If  $W/\overline{R}$  is HIGH, data is read from the 8 data lines and written in to the color lookup table. For both the read and write operations the address of the data stored in the color lookup table is taken from the eight address lines ( $A_0 - A_7$ ). Because both the address inputs and the outputs of the color lookup table are latched, the clock must be left running during an update. Time must be allowed for the address to be latched before beginning a write cycle and for the address and then data to be latched before ending a read cycle. To insure the update cycle does not interfere with the screen refresh, modifications to the color lookup table should occur while blank is active. The ten additional lines ( $D_0 - D_7$ ,  $\overline{EN}$ , and  $W/\overline{R}$ ) used for a color lookup table update are all TTL-compatible.

### Overlay

Some graphics systems require a separate bit plane in addition to the color bit planes. An example of this might be separate hardware and a separate bit plane to handle text processing. The Am8151 provides two pins (OVERLAY and

$W/\overline{B}$ ) which override the color pixel data to provide an additional video source. If OVERLAY is HIGH, the pixel data on  $A_0 - A_7$  is overridden.  $W/\overline{B}$  selects the intensity of the overlay. If  $W/\overline{B}$  is HIGH, the DAC output will be at the Peak White level (10% brighter than Reference White). If  $W/\overline{B}$  is LOW, the DAC output will be at the Reference Black level. OVERLAY and  $W/\overline{B}$  may be at either TTL or ECL logic levels. The LC pin is used to select the logic compatibility of these pins. The OVERLAY and  $W/\overline{B}$  signals are delayed the same number of clock cycles as the color pixel data before being fed into the DAC Decoder to keep overlay and color pixel data synchronized.

### Blank

During horizontal and vertical retrace, pixel data should be ignored and the intensity output of the DAC should be driven to a "blacker than black" blank state. This is done by means of the BLANK input. BLANK is TTL-compatible and latched on the same clock as the pixel data. The BLANK signal is delayed the same number of clock cycles as the pixel data before being fed into the DAC Decoder to keep the BLANK signal and the pixel data synchronized. BLANK, when active, overrides the data and overlay inputs to drive the DAC output to the blank level.

### Composite Sync

In some systems, the monitor control signals HSYNC and VSYNC are mixed with the Red, Green, and Blue signals. These control signals synchronize the monitor sweep oscillators to the R, G and B signal information. The Am8151 provides the necessary circuitry to mix these signals with the pixel information. Two inputs are provided, HSYNC and VSYNC, which are combined to generate the composite sync. These inputs are TTL-compatible and are latched with the same clock as the pixel data. Internally to the chip, HSYNC and VSYNC are x-ored to generate a composite sync signal. The composite sync signal is generated in this manner to provide inverted HSYNC pulses during the much longer VSYNC pulse. This prevents the horizontal oscillator from losing synchronization during a vertical retrace and causes the horizontal oscillator to change phase by the width of HSYNC. The composite sync signal is delayed the same number of cycles as the pixel data and then, if active, overrides the data, OVERLAY, and BLANK signals to drive the output to the composite sync level.

### Voltage Reference

To aid in the generation of the scaling current, the Am8151 provides an on-chip precision voltage supply. The  $V_{REF}$  pin provides a temperature, supply voltage, and load invariant output voltage of typically 2.152 volts. The reference voltage has an initial accuracy of 1% and contains less than 0.1% noise. A scaling current is generated at  $I_{REF}$  by connecting an external resistor from  $V_{REF}$  to  $I_{REF}$ . The reference resistor value can be calculated by the relation  $R_{REF} = 28.56/IFS$ .



IFS is the Full Scale Output Current which is produced at a sync output level. When three Am8151s are used, full scale outputs may be matched by using a single voltage reference and connecting three identical reference resistors from this pin to the three I<sub>REF</sub> inputs.

### Reference Amp

The Am8151 provides an input to scale the VIDEO output. The input pin I<sub>REF</sub> provides a current scaling input. The relationship between I<sub>REF</sub> and IFS is IFS = 13.27 I<sub>REF</sub>. The current reference amp and the DAC both require external compensation capacitors. Two pins are provided for this purpose - COMP<sub>1</sub> and COMP<sub>2</sub>. COMP<sub>1</sub> and COMP<sub>2</sub> should be connected to two 0.1-μF and two 100-pF capacitors as shown in the typical connection diagram.

### DAC And DAC Decoder

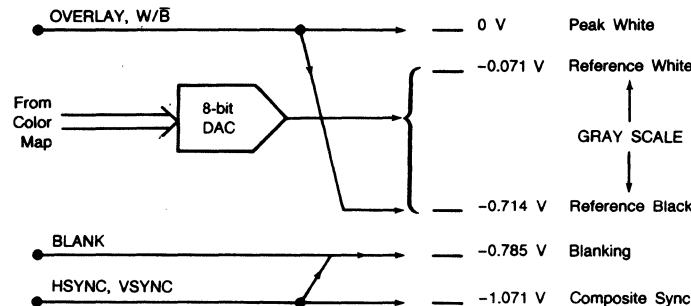
A major problem in high resolution graphics are glitches at the output of the DAC which occur when the digital input changes.

These glitches are caused by switch timing mismatches as one current source turns on and another turns off. The Am8151 avoids this problem by using 15 equal current sources for the four MSBs in the DAC. By using 15 current sources and decoding the four MSBs to select these, glitches caused by changes in these four bits are eliminated. The four LSBs are similarly decoded to avoid glitches. This limits glitches to only 16 transitions where both the four MSBs and the four LSBs change. The magnitude of the glitch is also limited by this to one-sixteenth of full scale. The length of the glitch is limited because every current switch input is individually registered after the DAC Decoder. In addition to these 30 current sources, six current sources are required to generate the Sync and Blank signal levels. The DAC Decoder, therefore, decodes the 8 bits of color pixel data, the overlay pixel data, the Blank, and the Sync signals to control which of the 36 current sources are on. The following truth table lists the nominal DAC output for all combinations of DAC Decoder inputs assuming a reference current of 1.076 mA.

TRUTH TABLE

W/ $\bar{B}$	OVERLAY	BLANK	HSYNC	VSYNC	Data	Current	Voltage into 75 Ω	Level
1	1	0	0	0	X	0 mA	0 mV	Peak White
X	0	0	0	0	255	.946 mA	-71 mV	Reference White
						•	255 Equal Steps	
						•		
X	0	0	0	0	0	9.520 mA	-714 mV	Reference Black
0	1	0	0	0	X	9.520 mA	-714 mV	Reference Black
X	X	1	0	0	X	10.466 mA	-785 mV	Blanking
X	X	X	1	1	X	10.466 mA	-785 mV	Blanking
X	X	X	0	1	X	14.280 mA	-1071 mV	Composite Sync
X	X	X	1	0	X	14.280 mA	-1071 mV	Composite Sync

### Am8151 DAC OUTPUT LEVELS



AF003661

### ABSOLUTE MAXIMUM RATINGS

Storage Temperature ..... -65 to +150°C  
 Ambient Temperature Under Bias ..... -55 to +125°C  
 Supply Voltage to Ground Potential  
 Continuous (TTL V<sub>CC</sub> and ANAV<sub>CC</sub>) ..... -0.5 to +7.0 V  
 Supply Voltage to Ground Potential  
 Continuous (ECL V<sub>EE</sub> and ANAV<sub>EE</sub>) ..... +0.5 to -7.0 V  
 DC Input Voltage (TTL) ..... -0.5 to +7.0 V  
 DC Input Current (TTL) ..... -30 to +5.0 mA  
 DC Input Voltage (ECL) ..... +0.5 to V<sub>EE</sub>

Stresses above those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.

### OPERATING RANGES

TTL Supply Voltage (TTLV<sub>CC</sub>) ..... +5 V ±5%  
 ECL Negative Supply Voltage (ECLV<sub>EE</sub>) ..... -5.2 V ±5%  
 Analog Positive Supply Voltage (ANAV<sub>CC</sub>) ..... +5 V ±5%  
 Analog Negative Supply Voltage (ANAV<sub>EE</sub>) ..... -5.2 V ±5%  
 Temperature (Note 6) ..... 0 to +70°C

Operating ranges define those limits over which the functionality of the device is guaranteed.

### DC CHARACTERISTICS over operating range (TTL)

Parameter Symbol	Parameter Description	Test Conditions (Note 1)	Min.	Typ. (Note 2)	Max.	Units
V <sub>IH</sub>	Input HIGH Level	Guaranteed Input Logical HIGH Voltage for All Inputs (Note 5)	2.0			Volts
V <sub>IL</sub>	Input LOW Level	Guaranteed Input Logical LOW Voltage for All Inputs (Note 5)			0.8	Volts
I <sub>IL</sub>	Input LOW Current	V <sub>CC</sub> = Max., V <sub>IN</sub> = 0.4 V			-0.4	mA
I <sub>IH</sub>	Input HIGH Current	V <sub>CC</sub> = Max., V <sub>IN</sub> = 2.7 V			40	μA
I <sub>I</sub>	Input HIGH Current	V <sub>IN</sub> = V <sub>CC</sub> = 5.25 V			0.1	mA
V <sub>OH</sub>	Output HIGH Voltage	V <sub>CC</sub> = Min., V <sub>IN</sub> = V <sub>IH</sub> = V <sub>IL</sub> I <sub>OH</sub> = -400 μA	2.4			Volts
V <sub>OL</sub>	Output LOW Voltage	V <sub>CC</sub> = Min., V <sub>IN</sub> = V <sub>IH</sub> = V <sub>IL</sub> I <sub>OL</sub> = 8 mA			0.5	Volts
I <sub>OS</sub>	Output Short Circuit Current (Note 3)	V <sub>CC</sub> = Max.	20		70	mA
TTL <sub>ICC</sub>	Power Supply Current	TTLV <sub>CC</sub> = Max.		40	60	mA
ANAL <sub>ICC</sub>	Power Supply Current	ANAV <sub>CC</sub> = Max. (Note 4)		5	10	mA
ECL <sub>IEE</sub>	Power Supply Current	ECLV <sub>EE</sub> = Max.		310	425	mA
ANAL <sub>IEE</sub>	Power Supply Current	ANAV <sub>EE</sub> = Max.		25	40	mA

### DC CHARACTERISTICS over operating range (ECL) (Note 6)

	Parameter Symbol	Test Conditions	0°C	25°C	70°C	Unit
ECL Inputs	V <sub>IH</sub> (Max.)	(Note 5)	-840	-780	-720	mV
	V <sub>IHA</sub> (Min.)		-1145	-1105	-1045	mV
	V <sub>ILA</sub> (Max.)	(Note 5)	-1490	-1475	-1450	mV
	V <sub>IL</sub> (Min.)		-1870	-1850	-1830	mV
	I <sub>IH</sub>	V <sub>EE</sub> = Max. V <sub>IN</sub> = V <sub>IH</sub> (Max.)	200	200	200	μA
	I <sub>IL</sub>	V <sub>EE</sub> = Max. V <sub>IN</sub> = V <sub>IL</sub> (Min.)	150	150	150	μA

- Notes: 1. For conditions shown as Min. or Max. use the appropriate value specified under recommended operating range.  
 2. Typical limits are at V<sub>CC</sub> = 5.0 V, V<sub>EE</sub> = -5.2 V, T<sub>A</sub> = 25°C  
 3. Not more than one output should be shorted at a time. Duration of short not to exceed one second.  
 4. ANAL<sub>ICC</sub> increases if additional devices are driven by V<sub>REF</sub>.  
 5. V<sub>IH</sub>, V<sub>IL</sub> are tested for each input at least once. Thereafter hard HIGH and LOW levels are used for all other tests.  
 6. A combination of skewing the limits and adjusting the pulse test ambient temperature is used to insure that the data sheet steady state limits are met at the ambient temperatures specified.

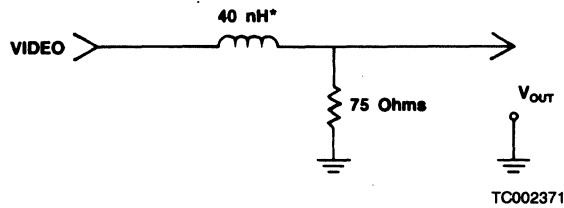


Figure 1.

\*An 80-nH inductor in series with the output results in a typical  $t_r$  and  $t_f$  of 2.5 ns with a 5% overshoot.

**DAC SPECIFICATIONS** over operating range unless otherwise specified

Parameter Symbol	Parameter Description	Test Conditions	Min.	Typ.	Max.	Units
	Resolution	Gray Scale (Note 7)	8	8	8	bits
	Linearity	Gray Scale (Note 9)			±.5	LSB
	Differential Linearity	Gray Scale (Note 9)			±.5	LSB
V <sub>OC</sub>	Output Compliance Voltage			±1.4		Volts
I <sub>ZS</sub>	Zero Scale Current			1		µA
I <sub>Max.</sub>	Output Maximum Current				22	mA
	Full Scale Temp. Current Coefficient			50		ppm/°C
PSS IFS±	Power Supply Sensitivity, Full Scale Positive	V <sub>CC</sub> = 5 V ±5% V <sub>EE</sub> = -5.2 V ±5%			0.25	% Gray Scale
	Glitch Energy	(Notes 11 & 13)		50		pV-sec
	Output Capacitance	(Note 11)		10		pF
I <sub>REF</sub>	Reference Current				1.66	mA
CLK	Clock Frequency	LC = GND, ECL Mode	D.C.		200	MHz
		LC = V <sub>CC</sub> , TTL Mode	D.C.		83	
t <sub>R</sub>	Risetime: 10 to 90%	(Notes 8 & 14)			3.5	ns
t <sub>F</sub>	Falltime: 90 to 10%	(Notes 8 & 14)			3.5	ns
t <sub>S</sub>	Settling Time (256 Level of Gray) (Note 12)	% Gray Scale (Notes 8, 10 & 11)	Bits Accuracy			ns
		±0.2	8	45		
		±0.4	7	30		
		±0.8	6	8		
		±1.6	5	6		
±3.2	4	4.5				

- Notes:
- Over full current range.
  - DAC settling,  $t_r$ ,  $t_f$  measurements are taken using Figure 1. Output loading and frequency may require capacitor addition or inductor value changes to compensate for loading effects (see Figures 2 and 3).
  - Gray scale is defined as output levels between reference white and reference black. DAC Linearity and Differential Linearity are measured at  $\leq 10$  MHz in production over operating temperature range at nominal supply voltages. Accuracy degrades with increasing frequency. DAC output terminated into 75-ohm load. I<sub>FS</sub> = 14.28 mA.
  - These numbers are measured using Tektronix sampling unit TYPE 3S2 and sampling sweep TYPE 3T77.
  - This typical value is provided for design reference and will not be tested in production.
  - Settling time is highly sensitive to PC board layout, decoupling and termination.
  - Am8151 produces a clock-related glitch on the video output pin occurring approximately 4 ns after both the rising and falling edges of the clock. The glitch is bipolar in shape and has a duration of approximately 5 ns. Typical amplitude is  $\pm 20$  mV around the DAC output level.

## EXPLANATION OF DAC SPECIFICATIONS

### Resolution:

Resolution refers to the number of discrete steps or levels which the DAC can provide, and is expressed as a number of bits. A DAC with n bits of resolution provides  $2^n$  discrete analog levels. Note that having n bits of resolution does not guarantee n bits of accuracy.

### Linearity:

Linearity is the maximum deviation of an actual output from an ideal output defined by a straight line drawn through the end points of the transfer function. A converter must be linear to within 1/2 a step to be accurate to its full resolution. In the Am8151, linearity is expressed as a fraction of the change in output caused by a change in the LSB.

### Differential Linearity:

Differential Linearity is the measure of the uniformity of step size. If the differential linearity is specified as 1/2 an LSB, the step size from one step to the next may be from 1/2 to 3/2 of an ideal step. In the Am8151, linearity is expressed as a fraction of the change in output caused by a change in the LSB.

### Output Compliance Voltage:

The Output Compliance Voltage is that voltage swing which may be impressed on the output current pin.

### Settling Time:

Settling Time is the time from when the output first changes until the output arrives at, and remains within, a certain error band around the final value. In the Am8151 this is specified to be the time from a 10% change in the output value, to within a certain percent of the gray scale.

### Zero Scale Current:

Zero Scale Current is the current produced at a Peak White output level.

### Output Maximum Current:

Full Scale Current is the current produced at a Sync output level. The Full Scale Current is determined by the relation  $IFS = 13.27 I_{REF}$ .

### Full Scale Current Temperature Coefficient:

The Full Scale Current Temperature Coefficient is the effect of temperature change, within the operating range, on the Full Scale Current. For the Am8151 this is given as the change in current per degree Centigrade.

### Power Supply Sensitivity:

The effect of the power supply voltage change on a full scale DAC output. In the Am8151, PSS IFS  $\pm$  is expressed as a percentage of the Gray Scale Range as reference black output varies while the voltage varies in the recommended operating range. This specification assumes the reference voltage is used.

### Glitch Energy:

Glitch Energy is an indication of the magnitude and duration of glitches. For the Am8151 this is specified in picovolt-seconds.

### Output Capacitance:

Output Capacitance is the internal capacitance of the VIDEO pin.

### Reference Current:

Reference Current is the range of acceptable reference currents at the  $I_{REF}$  pin.

## REFERENCE VOLTAGE OVER OPERATING RANGE

Parameter Symbol	Parameter Description	Test Conditions	Min.	Typ.	Max.	Units
$V_{REF}$	Reference Voltage	$I_{REF} = 1$ to 5 mA	2.127	2.152	2.174	Volts
	Line Regulation	$V_{CC} = 5 \pm 5\%$ $V_{EE} = -5.2 \pm 5\%$		.23	.4	% $V_{REF}$
	Load Regulation	$I_{REF} = 1$ to 5 mA		.1	.4	% $V_{REF}$
	Voltage Temp. Coefficient	(Note 11)		25		ppm/ $^{\circ}C$

## EXPLANATION OF REFERENCE VOLTAGE SPECIFICATIONS

### Reference Voltage:

Reference Voltage is the output voltage provided by the reference voltage source.

### Line Regulation:

Line Regulation is the effect of a change in the supply voltage on the reference voltage output. For the Am8151 this is specified as the percentage change in the reference voltage for changes in supply voltage within the Operating Range.

### Load Regulation:

Load Regulation is the effect of a change in the current sourced by the reference voltage on the reference voltage output. For the Am8151 this is specified as the percentage change in the reference voltage for changes in current as specified.

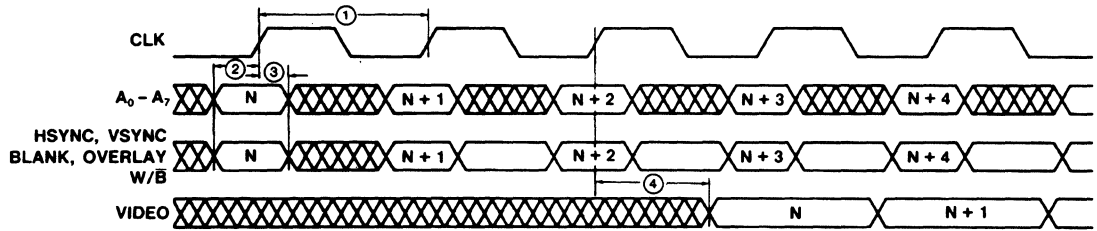
### Reference Voltage Temperature Coefficient:

Reference Voltage Temperature Coefficient is the effect of temperature change on the reference voltage. For the Am8151 this is given as the change in voltage per degree Centigrade.

AC SWITCHING CHARACTERISTICS over operating range							
No.	Parameter Symbol	Parameter Description	Test Conditions	Min.	Typ.	Max.	Units
1	t <sub>CLK</sub>	Clock Period (Note 14)	LC = GND, ECL Mode LC = V <sub>CC</sub> , TTL Mode	5 12			ns ns
2	t <sub>s</sub>	Address, OVERLAY, W/ $\bar{B}$ Setup before Clock $\uparrow$ HSYNC, VSYNC, BLANK Setup before Clock $\uparrow$ Address, OVERLAY, W/ $\bar{B}$ , HSYNC, VSYNC, BLANK Setup before Clock $\uparrow$ (Note 14)	LC = GND LC = GND LC = V <sub>CC</sub>	1.5 3.0 2.0			ns ns ns
3	t <sub>H</sub>	Address, OVERLAY, W/ $\bar{B}$ Hold after Clock $\uparrow$ HSYNC, VSYNC, BLANK Hold after Clock $\uparrow$ Address, OVERLAY, W/ $\bar{B}$ , HSYNC, VSYNC, BLANK after Clock $\uparrow$ (Note 14)	LC = GND LC = GND LC = V <sub>CC</sub>	2.0 2.0 2.0			ns ns ns
4	t <sub>PD</sub>	Clock $\uparrow$ to 10% VIDEO change (Notes 14 & 15)	LC = GND LC = V <sub>CC</sub>	3.0 4.0		7.0 11.0	ns ns
5	t <sub>PD</sub>	Clock $\uparrow$ to Data Valid (Read)				40	ns
6	t <sub>s</sub>	W/ $\bar{B}$ Setup before $\overline{EN}$ $\downarrow$ (Note 14)		20			ns
7	t <sub>H</sub>	W/ $\bar{B}$ Hold after $\overline{EN}$ $\uparrow$ (Note 14)		20			ns
8	t <sub>PD</sub>	$\overline{EN}$ $\downarrow$ to Data Active (Read)				40	ns
9	t <sub>PD</sub>	$\overline{EN}$ $\uparrow$ to Data Three-State (Read)				60	ns
10	t <sub>s</sub>	Address latched (Clock $\uparrow$ ) to $\overline{EN}$ $\downarrow$ Setup (Write) (Note 14)		10			ns
11	t <sub>s</sub>	Data (and Address) Setup before $\overline{EN}$ $\uparrow$ , Write Cycle Time (Write) (Note 14)		40			ns
12	t <sub>H</sub>	Data Hold after $\overline{EN}$ $\uparrow$ (Note 14)		10			ns

Note: 14. Not all tests are being performed in manufacturing. Tests are guaranteed by Engineering characterization with periodic manufacturing sampling.  
15. Clock  $\uparrow$  to 10% VIDEO change for any two Am8151s at the same temperature, voltage and load conditions typically does not differ by more than 0.5 ns for the ECL mode and 0.6 ns for the TTL mode.

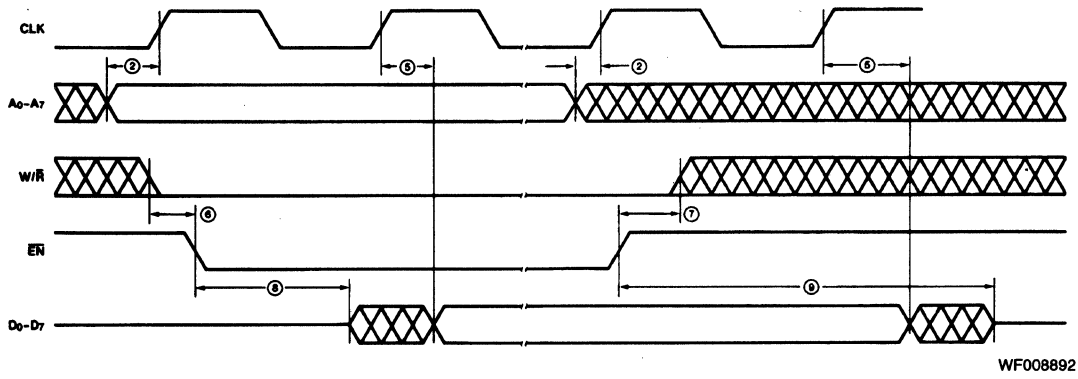
**SWITCHING WAVEFORMS**  
**VIDEO REFRESH TIMING**



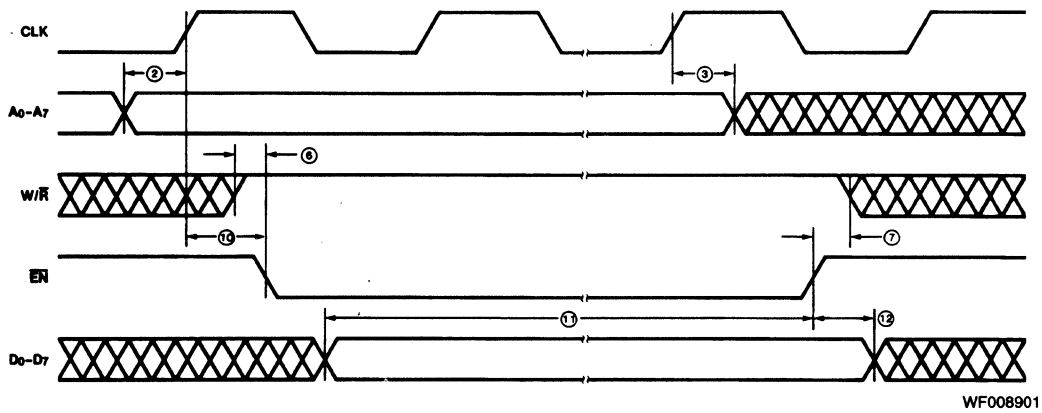
WF008882

### SWITCHING WAVEFORMS (Cont.)

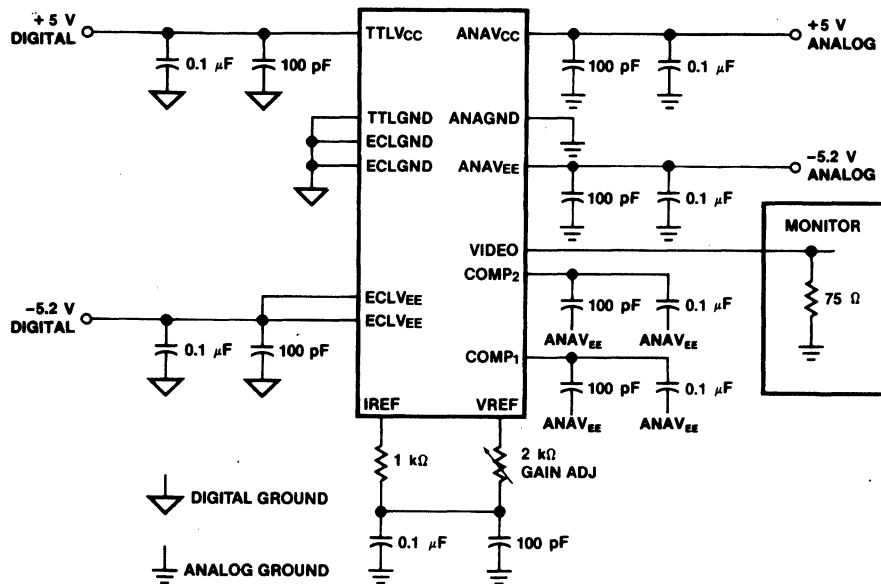
#### READ TIMING



#### WRITE TIMING

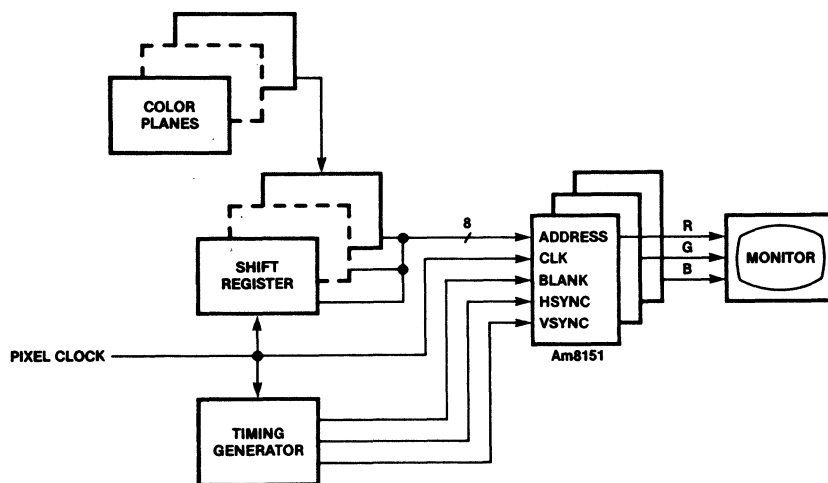


### TYPICAL CONNECTION DIAGRAM



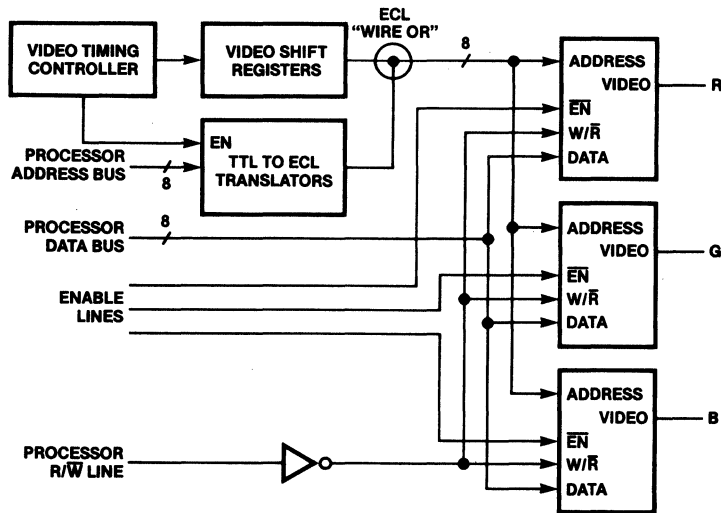
CD005692

### TYPICAL APPLICATION



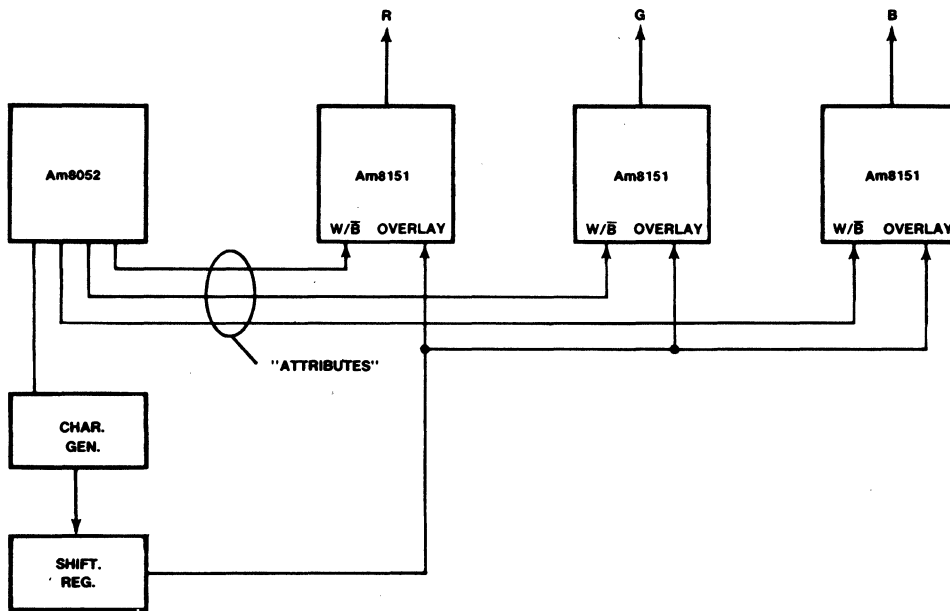
AF003410

### TYPICAL PROCESSOR CONNECTION



AF003420

### GENERATING 8 COLOR TEXT OVERLAY WITH W/B AND OVERLAY INPUTS



AF003720

### SINGLE Am8151 TRUTH TABLE

Overlay	W/B	Output
0	X	Graphics
1	1	Peak White
1	0	Ref. Black



**TRUTH TABLE FOR THREE Am8151s**

Overlay	W/B			Output
	(R)	(G)	(B)	
0	X	X	X	Graphics
1	0	0	0	Black
1	0	0	1	Blue
1	0	1	0	Green
1	0	1	1	Cyan
1	1	0	0	Red
1	1	0	1	Magenta
1	1	1	0	Yellow
1	1	1	1	White

**KEY TO SWITCHING WAVEFORMS**

WAVEFORM	INPUTS	OUTPUTS
	MUST BE STEADY	WILL BE STEADY
	MAY CHANGE FROM H TO L	WILL BE CHANGING FROM H TO L
	MAY CHANGE FROM L TO H	WILL BE CHANGING FROM L TO H
	DON'T CARE: ANY CHANGE PERMITTED	CHANGING: STATE UNKNOWN
	DOES NOT APPLY	CENTER LINE IS HIGH IMPEDANCE "OFF" STATE

KS000010

### TYPICAL PERFORMANCE CURVES

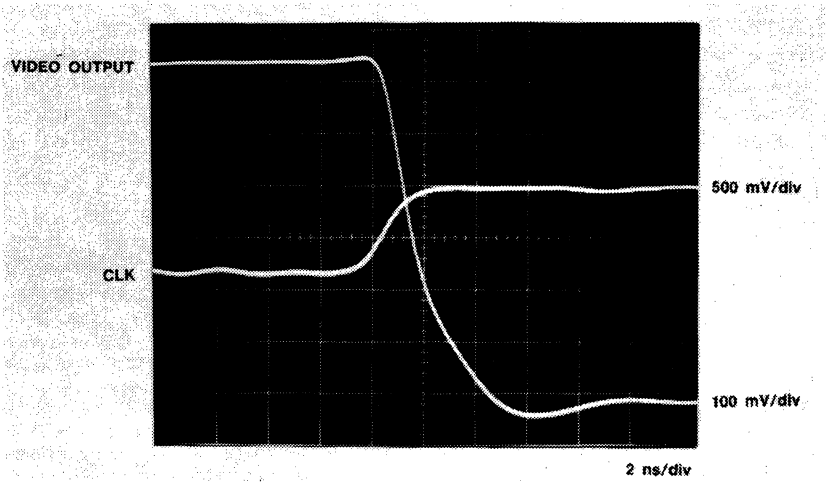


Figure 2. Video Output Falling Edge

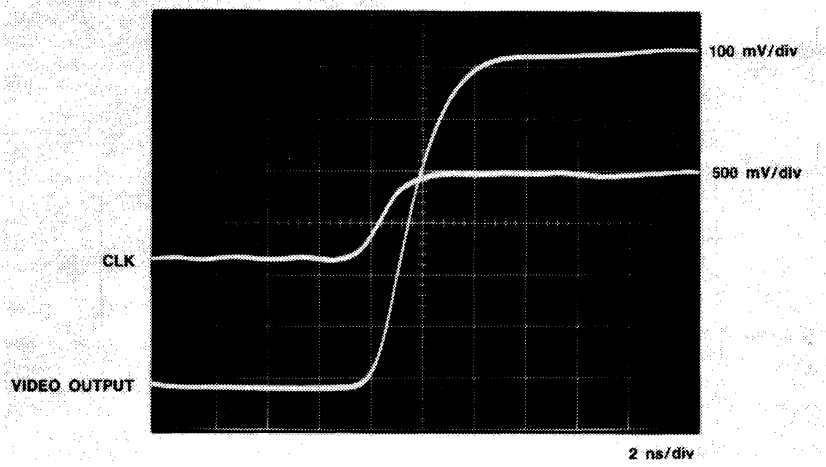
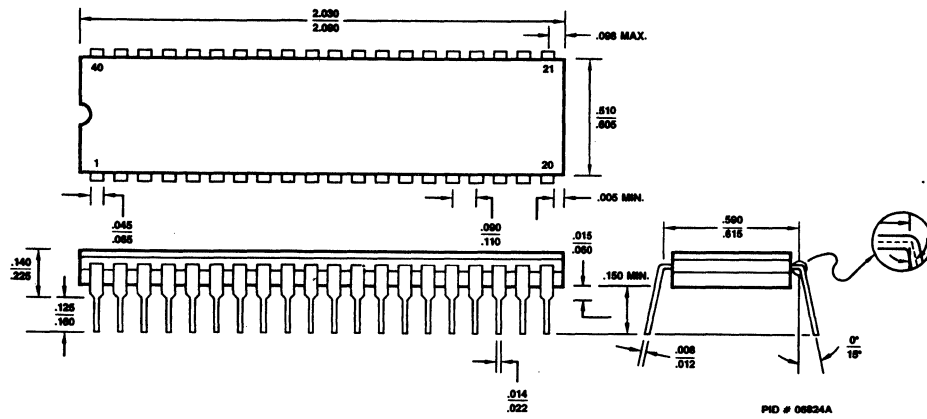


Figure 3. Video Output Rising Edge

# PHYSICAL DIMENSIONS

## CD 040



**ADVANCED MICRO DEVICES  
DOMESTIC SALES OFFICES**

ALABAMA .....	(205) 882-9122	MARYLAND .....	(301) 796-9310
ARIZONA,		MASSACHUSETTS .....	(617) 273-3970
Tempe .....	(602) 242-4400	MINNESOTA .....	(612) 938-0001
Tucson .....	(602) 792-1200	NEW JERSEY .....	(201) 299-0002
CALIFORNIA,		NEW YORK,	
El Segundo .....	(213) 640-3210	Liverpool .....	(315) 457-5400
Newport Beach .....	(714) 752-6262	Poughkeepsie (IBM only) .....	(914) 471-8180
San Diego .....	(619) 560-7030	Woodbury .....	(516) 364-8020
Sunnyvale .....	(408) 720-8811	NORTH CAROLINA,	
Woodland Hills .....	(818) 992-4155	Raleigh .....	(919) 847-8471
COLORADO .....	(303) 741-2900	OREGON .....	(503) 245-0080
CONNECTICUT,		OHIO,	
Southbury .....	(203) 264-7800	Columbus .....	(614) 891-6455
FLORIDA,		PENNSYLVANIA,	
Altamonte Springs .....	(305) 339-5022	Allentown (AT&T only) .....	(215) 398-8006
Clearwater .....	(813) 530-9971	Willow Grove .....	(215) 657-3101
Ft Lauderdale .....	(305) 484-8600	TEXAS,	
Melbourne .....	(305) 254-2915	Austin .....	(512) 346-7830
GEORGIA .....	(404) 449-7920	Dallas .....	(214) 934-9099
ILLINOIS .....	(312) 773-4422	Houston .....	(713) 785-9001
INDIANA .....	(317) 244-7207	WASHINGTON .....	(206) 455-3600
KANSAS .....	(913) 451-3115	WISCONSIN .....	(414) 782-7748

**INTERNATIONAL SALES OFFICES**

BELGIUM,		HONG KONG,	
Bruxelles .....	TEL: (02) 771 99 93	Kowloon .....	TEL: 3-695377
	FAX: (02) 762-3716		FAX: 1234276
	TLX: 61028		TLX: 50426
CANADA, Ontario,		ITALY, Milano .....	TEL: (02) 3390541
Kanata .....	TEL: (613) 592-0090		FAX: (02) 3498000
Willowdale .....	TEL: (416) 224-5193		TLX: 315286
	FAX: (416) 224-0056	JAPAN, Tokyo .....	TEL: (03) 345-8241
FRANCE,			FAX: 3425196
Paris .....	TEL: (01) 46 87 36 66		TLX: J24064AMDTKOJ
	FAX: (01) 46 86 21 85	LATIN AMERICA,	
	TLX: 202053F	Ft. Lauderdale .....	TEL: (305) 484-8600
GERMANY,			FAX: (305) 485-9736
Hannover area .....	TEL: (05143) 50 55		TLX: 5109554261 AMDFTL
	FAX: (05143) 55 53	SWEDEN, Stockholm .....	TEL: (08) 733 03 50
	TLX: 925287		FAX: (08) 733 22 85
München .....	TEL: (089) 41 14-0		TLX: 11602
	FAX: (089) 406490	UNITED KINGDOM,	
	TLX: 523883	Manchester area .....	TEL: (0925) 828008
Stuttgart .....	TEL: (0711) 62 33 77		FAX: (0925) 827693
	FAX: (0711) 625187		TLX: 628524
	TLX: 721882	London area .....	TEL: (04862) 22121
			FAX: (04862) 22179
			TLX: 859103

**NORTH AMERICAN REPRESENTATIVES**

CALIFORNIA		NEW JERSEY	
I <sup>2</sup> INC .....	OEM (408) 988-3400	TAI CORPORATION .....	(609) 933-2600
	DISTI (408) 496-6868	NEW MEXICO	
IDAHO		THORSON DESERT STATES .....	(505) 293-8555
INTERMOUNTAIN TECH MKGT .....	(203) 272-2963	NEW YORK	
INDIANA		NYCOM, INC .....	(315) 437-8343
SAI MARKETING CORP .....	(208) 322-5022	OHIO	
IOWA		Dayton	
LORENZ SALES .....	(317) 241-9276	DOLFUSS ROOT & CO .....	(513) 433-6776
MICHIGAN		Strongsville	
SAI MARKETING CORP .....	(319) 377-4666	DOLFUSS ROOT & CO .....	(216) 238-0300
NEBRASKA		PENNSYLVANIA	
LORENZ SALES .....	(313) 227-1786	DOLFUSS ROOT & CO .....	(412) 221-4420
		UTAH	
		R <sup>2</sup> MARKETING .....	(801) 595-0631

Advanced Micro Devices reserves the right to make changes in its product without notice in order to improve design or performance characteristics. The performance characteristics listed in this document are guaranteed by specific tests, guard banding, design and other practices common to the industry. For specific testing details, contact your local AMD sales representative. The company assumes no responsibility for the use of any circuits described herein.

# INITIALIZATION WITH MONI

FORCE COMPUTERS Inc./GmbH  
All Rights Reserved

This document shall not be duplicated, nor its contents used  
for any purpose, unless express permission has been granted.

Copyright by FORCE Computers®

## The Initialization Program MONI:

The SYS68K/AGC-1 manual includes a floppy disk which can be read under PDOS, the Real Time Multi-tasking Operating System.

After mounting of the floppy by typing in "SY0", the program can be executed by typing in:

```
MONI <CR>
```

The program displays the list of screen parameters and the default setting onto the terminal.

In this menu the user can input the parameters which specify the horizontal and vertical resolution of the display screen

Example:

To display 1024 x 800 Pixel with 50Hz non-interlaced and 4 bit/Pixel, the following sequence has to be typed in:

FR <Cr>	Specifies the FIELD RATE
50 <Cr>	Stands for 50Hz
RE <Cr>	Specifies the RASTER SCAN MODE
1 <Cr>	Stands for non-interlaced
GM <Cr>	Specifies the GRAPHICS BIT MODE
4 <Cr>	Stands for 4 bit/Pixel
XR <Cr>	Specifies the horizontal resolution
1024 <Cr>	Stands for 1024 Pixel
YR <Cr>	Specifies the vertical resolution
800 <Cr>	Stands for 800 lines

The menu for the horizontal display parameters can be entered if HO <CR> is typed in.

This menu allows to change the Horizontal SYNC width (HS), the Horizontal Back Porch (HB) and the Horizontal Front Porch (HF) in the same way as described above.

The values of these parameters depends on the timing specification of the used monitor.

To adapt the monitor to the AGC-1 board set the correct timing values (normally included in the User's Manual of the monitor) has to be entered.

The menu also shows the required Horizontal SCAN FREQUENCY, the Horizontal LINE PERIOD, the BLANK TIME and the active DISPLAY PERIOD.

The number of lines displayed in this menu include the lines required for the Vertical RETRACE PERIOD.

The command VT <Cr> selects the third menu which displays the calculated vertical display parameters.

In this menu the Vertical SYNC WIDTH and the Vertical BACK PORCH can be changed because the values of these parameters also depend on the electrical specifications of the used monitor. The procedure is the same as described for the first menu.

All SCREEN PARAMETER REGISTERS of the 63484 ACRCCT are shown on the terminal after typing CR <Cr>. These values are the values needed for the basic initialization of the SYS68K/AGC-1. Only the BASE SCREEN is initialized but the initialization of more screens does not change any of the SCREEN DISPLAY PARAMETERS which are important for the correct VIDEO timing needed by the used monitor.

To see if the monitor produces a stable picture, just type GI <Cr> and the SYS68K/AGC-1 will be initialized by MONI with the parameters entered before. If everything performs well, a stable frame on the monitor is shown but if the frame is not stable on the monitor, the monitor has to be adapted (modification of the horizontal oscillator) and finely tuned to the trigger frame. If this procedure is not successful or the frame does not appear in the middle of the monitor the screen parameters programmed before have to be modified.

This can be done by typing in the respective mnemonic (i.e. HS for the Horizontal SYNC WIDTH), entering the changed value and verifying the new adjustment by typing GI <Cr>. This procedure can be performed as long as the frame does not appear in the right way.

The MONI program can be left by typing in ESC <Cr> and normal PDOS commands can be typed in.

## SCREEN PARAMETERS

<FR>	FIELD RATE	(HZ):	60
<RS>	RASTER SCAN MODE		
	INTERLACED = 0 / NONINTERLACED = 1	:	1
<GM>	GRAPHIC BIT MODE (BIT / PER PIXEL):		4
<XR>	X - RESOLUTION (PIXEL):		1024
<YR>	Y - RESOLUTION (LINES):		800

---

\* HORIZONTAL MONITOR PARAMETERS \*

---

	ACTIVE DISPLAY PERIOD	(NSEC)	16000
<HS>	HORIZONTAL SYNCRON WIDTH	(NSEC)	750
<HB>	HORIZONTAL BACK PORCH	(NSEC)	2500
<HF>	HORIZONTAL FRONT PORCH	(NSEC)	1250
	HORIZONTAL LINE PERIOD	(NSEC)	20500
	HORIZONTAL BLANK TIME	(NSEC)	4500
	HORIZONTAL SCAN FREQUENCY (HERTZ)		48780
	MAXIMAL LINES (INCL. VERT. RETRACE)		812

---

\* VERTICAL MONITOR PARAMETERS

---

	MAXIMAL LINES (INCL. VERT. RETRACE)		812
<VS>	VERTICAL SYNCRON WIDTH	(LINES)	1
<VB>	VERTICAL BACK PORCH	(LINES)	10
	VERTICAL FRONT PORCH	(LINES)	1
	VERTICAL BLANK LINES	(LINES)	12
	ACTIVE DISPLAY LINES	(LINES)	800

---

\* A C R T C INITIALISATION PARAMETERS \*

---

HORIZONT SYNCRON	HSR	R82 :	HEX	\$00005103
HORIZONT DISPLAY	HDR	R84 :	HEX	\$0000093F
VERTICAL SYNCRON	VSR	R86 :	HEX	\$0000032C
VERTICAL DISPLAY	VDR	R88 :	HEX	\$00000901
SPLIT SCREEN WIDTH	SP1	R8A :	HEX	\$00000320
COMMAND CONTROL	CCR	R02 :	HEX	\$00000200
DISPLAY CONTROL	DCR	R06 :	HEX	\$0000C010
OPERATION MODE	OMR	R04 :	HEX	\$0000C038

\* These values correspond to the monitor : BARCO CDCT 6551



```

<FR> FIELD RATE (HZ): 50
<RS> RASTER SCAN MODE
      INTERLACED = 0 / NONINTERLACED = 1 : 1
<GM> GRAPHIC BIT MODE (BIT / PER PIXEL): 4
<XR> X - RESOLUTION (PIXEL): 1140
<YR> Y - RESOLUTION (LINES): 860

```

```

-----
* HORIZONTAL MONITOR PARAMETERS *
-----

```

```

ACTIVE DISPLAY PERIOD (NSEC) 17812
<HS> HORIZONTAL SYNCRON WIDTH (NSEC) 750
<HB> HORIZONTAL BACK PORCH (NSEC) 1250
<HF> HORIZONTAL FRONT PORCH (NSEC) 1190
HORIZONTAL LINE PERIOD (NSEC) 21000
HORIZONTAL BLANK TIME (NSEC) 3190
HORIZONTAL SCAN FREQUENCY (HERTZ) 47614
MAXIMAL LINES (INCL. VERT. RETRACE) 952

```

```

-----
* VERTICAL MONITOR PARAMETERS
-----

```

```

MAXIMAL LINES (INCL. VERT. RETRACE) 952
<VS> VERTICAL SYNCRON WIDTH (LINES) 2
<VB> VERTICAL BACK PORCH (LINES) 54
VERTICAL FRONT PORCH (LINES) 36
VERTICAL BLANK LINES (LINES) 92
ACTIVE DISPLAY LINES (LINES) 860

```

```

-----
* A C R T C INITIALISATION PARAMETERS *
-----

```

```

HORIZONT SYNCRON HSR R82 : HEX $00005303
HORIZONT DISPLAY HDR R84 : HEX $00000446
VERTICAL SYNCRON VSR R86 : HEX $000003B8
VERTICAL DISPLAY VDR R88 : HEX $00003502
SPLIT SCREEN WIDTH SPL R8A : HEX $0000035C
COMMAND CONTROL CCR R02 : HEX $00000200
DISPLAY CONTROL DCR R06 : HEX $0000C010
OPERATION MODE OMR R04 : HEX $0000C038

```

\* These values correspond to the monitor : BARCO CDCT 6551

```

<FR> FIELD RATE (HZ): 60
<RS> RASTER SCAN MODE
      INTERLACED = 0 / NONINTERLACED = 1 : 0
<GM> GRAPHIC BIT MODE (BIT / PER PIXEL): 4
<XR> X - RESOLUTION (PIXEL): 1280
<YR> Y - RESOLUTION (LINES): 1024

```

```

-----
* HORIZONTAL MONITOR PARAMETERS *
-----

```

```

ACTIVE DISPLAY PERIOD (NSEC) 20000
<HS> HORIZONTAL SYNCRON WIDTH (NSEC) 500
<HB> HORIZONTAL BACK PORCH (NSEC) 2000
<HF> HORIZONTAL FRONT PORCH (NSEC) 1500
HORIZONTAL LINE PERIOD (NSEC) 24000
HORIZONTAL BLANK TIME (NSEC) 4000
HORIZONTAL SCAN FREQUENCY (HERTZ) 41666
MAXIMAL LINES (INCL. VERT. RETRACE) 1388

```

```

-----
* VERTICAL MONITOR PARAMETERS
-----

```

```

MAXIMAL LINES (INCL. VERT. RETRACE) 1388
<VS> VERTICAL SYNCRON WIDTH (LINES) 1
<VB> VERTICAL BACK PORCH (LINES) 3
VERTICAL FRONT PORCH (LINES) 178
VERTICAL BLANK LINES (LINES) 182
ACTIVE DISPLAY LINES (LINES) 1024

```

```

-----
* A C R T C INITIALISATION PARAMETERS *
-----

```

```

HORIZONT SYNCRON HSR R82 : HEX $00005F02
HORIZONT DISPLAY HDR R84 : HEX $0000074F
VERTICAL SYNCRON VSR R86 : HEX $0000056C
VERTICAL DISPLAY VDR R88 : HEX $00000201
SPLIT SCREEN WIDTH SPL R8A : HEX $00000400
COMMAND CONTROL CCR R02 : HEX $00000200
DISPLAY CONTROL DCR R06 : HEX $0000C010
OPERATION MODE OMR R04 : HEX $0000C03B

```

\* These values correspond to the monitor : BARCO CDCT 6551

```

<FR> FIELD RATE (HZ): 50
<RS> RASTER SCAN MODE
      INTERLACED = 0 / NONINTERLACED = 1 : 0
<GM> GRAPHIC BIT MODE (BIT / PER PIXEL): 4
<XR> X - RESOLUTION (PIXEL): 1600
<YR> Y - RESOLUTION (LINES): 1200

```

```

-----
* HORIZONTAL MONITOR PARAMETERS *
-----

```

```

ACTIVE DISPLAY PERIOD (NSEC) 25000
<HS> HORIZONTAL SYNCRON WIDTH (NSEC) 2250
<HB> HORIZONTAL BACK PORCH (NSEC) 2000
<HF> HORIZONTAL FRONT PORCH (NSEC) 750
      HORIZONTAL LINE PERIOD (NSEC) 30000
      HORIZONTAL BLANK TIME (NSEC) 5000
      HORIZONTAL SCAN FREQUENCY (HERTZ) 33333
      MAXIMAL LINES (INCL. VERT. RETRACE) 1332

```

```

-----
* VERTICAL MONITOR PARAMETERS
-----

```

```

MAXIMAL LINES (INCL. VERT. RETRACE) 1332
<VS> VERTICAL SYNCRON WIDTH (LINES) 4
<VB> VERTICAL BACK PORCH (LINES) 60
      VERTICAL FRONT PORCH (LINES) 12
      VERTICAL BLANK LINES (LINES) 66
      ACTIVE DISPLAY LINES (LINES) 1200

```

```

-----
* A C R T C INITIALISATION PARAMETERS *
-----

```

```

HORIZONT SYNCRON HSR R82 : HEX $00007709
HORIZONT DISPLAY HDR R84 : HEX $00000763
VERTICAL SYNCRON VSR R86 : HEX $00000534
VERTICAL DISPLAY VDR R88 : HEX $00003C04
SPLIT SCREEN WIDTH SPL R8A : HEX $000004B0
COMMAND CONTROL CCR R02 : HEX $00000200
DISPLAY CONTROL DCR R06 : HEX $0000C010
OPERATION MODE OMR R04 : HEX $0000C03B

```

\* These values correspond to the monitor : CONRAC 7211

```

<FR> FIELD RATE (HZ): 60
<RS> RASTER SCAN MODE
      INTERLACED = 0 / NONINTERLACED = 1 : 1
<GM> GRAPHIC BIT MODE (BIT / PER PIXEL): 8
<XR> X - RESOLUTION (PIXEL): 720
<YR> Y - RESOLUTION (LINES): 560

```

```

-----
* HORIZONTAL MONITOR PARAMETERS *
-----

```

```

      ACTIVE DISPLAY PERIOD (NSEC) 22500
<HS> HORIZONTAL SYNCRON WIDTH (NSEC) 2500
<HB> HORIZONTAL BACK PORCH (NSEC) 2000
<HF> HORIZONTAL FRONT PORCH (NSEC) 1250
      HORIZONTAL LINE PERIOD (NSEC) 28250
      HORIZONTAL BLANK TIME (NSEC) 5750
      HORIZONTAL SCAN FREQUENCY (HERTZ) 35398
      MAXIMAL LINES (INCL. VERT. RETRACE) 589

```

```

-----
* VERTICAL MONITOR PARAMETERS
-----

```

```

      MAXIMAL LINES (INCL. VERT. RETRACE) 589
<VS> VERTICAL SYNCRON WIDTH (LINES) 2
<VB> VERTICAL BACK PORCH (LINES) 21
      VERTICAL FRONT PORCH (LINES) 6
      VERTICAL BLANK LINES (LINES) 29
      ACTIVE DISPLAY LINES (LINES) 560

```

```

-----
* A C R T C INITIALISATION PARAMETERS *
-----

```

```

HORIZONT SYNCRON HSR R82 : HEX $0000700A
HORIZONT DISPLAY HDR R84 : HEX $00000759
VERTICAL SYNCRON VSR R86 : HEX $0000024D
VERTICAL DISPLAY VDR R88 : HEX $00001402
SPLIT SCREEN WIDTH SPL R8A : HEX $00000230
COMMAND CONTROL CCR R02 : HEX $00000300
DISPLAY CONTROL DCR R06 : HEX $0000C000
OPERATION MODE OMR R04 : HEX $0000C038

```

\* These values correspond to the monitor : CONRAC 7211

```

<FR> FIELD RATE (HZ): 50
<RS> RASTER SCAN MODE
      INTERLACED = 0 / NONINTERLACED = 1 : 1
<GM> GRAPHIC BIT MODE (BIT / PER PIXEL): 8
<XR> X - RESOLUTION (PIXEL): 800
<YR> Y - RESOLUTION (LINES): 600

```

```

-----
* HORIZONTAL MONITOR PARAMETERS *
-----

```

```

      ACTIVE DISPLAY PERIOD (NSEC) 25000
<HS> HORIZONTAL SYNCRON WIDTH (NSEC) 2750
<HB> HORIZONTAL BACK PORCH (NSEC) 2500
<HF> HORIZONTAL FRONT PORCH (NSEC) 1750
      HORIZONTAL LINE PERIOD (NSEC) 32000
      HORIZONTAL BLANK TIME (NSEC) 7000
      HORIZONTAL SCAN FREQUENCY (HERTZ) 31250
      MAXIMAL LINES (INCL. VERT. RETRACE) 625

```

```

-----
* VERTICAL MONITOR PARAMETERS
-----

```

```

      MAXIMAL LINES (INCL. VERT. RETRACE) 625
<VS> VERTICAL SYNCRON WIDTH (LINES) 2
<VB> VERTICAL BACK PORCH (LINES) 17
      VERTICAL FRONT PORCH (LINES) 6
      VERTICAL BLANK LINES (LINES) 25
      ACTIVE DISPLAY LINES (LINES) 600

```

```

-----
* A C R T C INITIALISATION PARAMETERS *
-----

```

```

HORIZONT SYNCRON HSR R82 : HEX $00007F0B
HORIZONT DISPLAY HDR R84 : HEX $00000963
VERTICAL SYNCRON VSR R86 : HEX $00000271
VERTICAL DISPLAY VDR R88 : HEX $00001002
SPLIT SCREEN WIDTH SPI R8A : HEX $00000258
COMMAND CONTROL CCR R02 : HEX $00000300
DISPLAY CONTROL DCR R06 : HEX $0000C000
OPERATION MODE OMR R04 : HEX $0000C038

```

\* These values correspond to the monitor : CONRAC 7211

```

<FR> FIELD RATE (HZ): 60
<RS> RASTER SCAN MODE
      INTERLACED = 0 / NONINTERLACED = 1 : 0
<GM> GRAPHIC BIT MODE (BIT / PER PIXEL): 8
<XR> X - RESOLUTION (PIXEL): 1024
<YR> Y - RESOLUTION (LINES): 800

```

```

-----
* HORIZONTAL MONITOR PARAMETERS *
-----

```

```

      ACTIVE DISPLAY PERIOD (NSEC) 32000
<HS> HORIZONTAL SYNCRON WIDTH (NSEC) 2000
<HB> HORIZONTAL BACK PORCH (NSEC) 2000
<HF> HORIZONTAL FRONT PORCH (NSEC) 1000
      HORIZONTAL LINE PERIOD (NSEC) 37000
      HORIZONTAL BLANK TIME (NSEC) 5000
      HORIZONTAL SCAN FREQUENCY (HERTZ) 27027
      MAXIMAL LINES (INCL. VERT. RETRACE) 900

```

```

-----
* VERTICAL MONITOR PARAMETERS
-----

```

```

      MAXIMAL LINES (INCL. VERT. RETRACE) 900
<VS> VERTICAL SYNCRON WIDTH (LINES) 2
<VB> VERTICAL BACK PORCH (LINES) 33
      VERTICAL FRONT PORCH (LINES) 15
      VERTICAL BLANK LINES (LINES) 50
      ACTIVE DISPLAY LINES (LINES) 800

```

```

-----
* A C R T C INITIALISATION PARAMETERS *
-----

```

```

HORIZONT SYNCRON HSR R82 : HEX $00009308
HORIZONT DISPLAY HDR R84 : HEX $0000077F
VERTICAL SYNCRON VSR R86 : HEX $00000384
VERTICAL DISPLAY VDR R88 : HEX $00002002
SPLIT SCREEN WIDTH SP1 R8A : HEX $00000320
COMMAND CONTROL CCR R02 : HEX $00000300
DISPLAY CONTROL DCR R06 : HEX $0000C000
OPERATION MODE OMR R04 : HEX $0000C03B

```

\* These values correspond to the monitor : CONRAC 7211

```

<FR> FIELD RATE (HZ): 50
<RS> RASTER SCAN MODE
      INTERLACED = 0 / NONINTERLACED = 1 : 0
<GM> GRAPHIC BIT MODE (BIT / PER PIXEL): 8
<XR> X - RESOLUTION (PIXEL): 1160
<YR> Y - RESOLUTION (LINES): 870

```

```

-----
* HORIZONTAL MONITOR PARAMETERS *
-----

```

```

      ACTIVE DISPLAY PERIOD (NSEC) 36250
<HS> HORIZONTAL SYNCRON WIDTH (NSEC) 2000
<HB> HORIZONTAL BACK PORCH (NSEC) 2000
<HF> HORIZONTAL FRONT PORCH (NSEC) 1250
      HORIZONTAL LINE PERIOD (NSEC) 41500
      HORIZONTAL BLANK TIME (NSEC) 5250
      HORIZONTAL SCAN FREQUENCY (HERTZ) 24096
      MAXIMAL LINES (INCL. VERT. RETRACE) 962

```

```

-----
* VERTICAL MONITOR PARAMETERS
-----

```

```

      MAXIMAL LINES (INCL. VERT. RETRACE) 962
<VS> VERTICAL SYNCRON WIDTH (LINES) 2
<VB> VERTICAL BACK PORCH (LINES) 43
      VERTICAL FRONT PORCH (LINES) 1
      VERTICAL BLANK LINES (LINES) 46
      ACTIVE DISPLAY LINES (LINES) 870

```

```

-----
* A C R T C INITIALISATION PARAMETERS *
-----

```

```

HORIZONT SYNCRON HSR R82 : HEX $0000A508
HORIZONT DISPLAY HDR R84 : HEX $00000790
VERTICAL SYNCRON VSR R86 : HEX $000003C2
VERTICAL DISPLAY VDR R88 : HEX $00002A02
SPLIT SCREEN WIDTH SPL R8A : HEX $00000366
COMMAND CONTROL CCR R02 : HEX $00000300
DISPLAY CONTROL DCR R06 : HEX $0000C000
OPERATION MODE OMR R04 : HEX $0000C038

```

\* These values correspond to the monitor : CONRAC 7211

# PROGRAMMING

## EXAMPLES

FORCE COMPUTERS Inc./GmbH  
All Rights Reserved

This document shall not be duplicated, nor its contents used  
for any purpose, unless express permission has been granted.

Copyright by FORCE Computers®



```

*****
*
* The information in this document is subject to change
* without notice and should not be construed as a commitment
* by FORCE Computers.
*
* Neither FORCE Computers nor the authors assume any responsi-
* bility for the use or reliability of this document or the
* described software.
*
* Copyright (C) 1986, FORCE Computers
*
* General permission to copy or modify, but not for profit, is
* hereby granted, provided that the above copyright notice is
* included and reference made to the fact that reproduction
* privileges were granted by FORCE Computers.
*
*****

```

```

*
*   SYS68K/AGC-1   PROGRAMMING EXAMPLES
*   =====

```

```

*   THIS PROGRAM INCLUDES A COMPLETE SERIES OF SUBROUTINES
*   TO GIVE A BASIS FOR USER APPLICATIONS.
*
*   THE ROUTINES AT THE BEGINNING DEMONSTRATE HOW TO USE
*   THESE SUBROUTINES.
*
*   THE DEFAULT BOARD BASE ADDRESS IS SUGGESTED TO BE $B00000
*   IN THE STANDARD MEMORY AREA OF THE VMEBUS.

```

```

*   THE FOLLOW SUBROUTINES MUST BE CALLED BEFORE DRAWING OPERATIONS
*   CAN BE STARTED

```

```

BSR      BINIT                ;CALCULATE REGISTER ACCESS ADDRESSES
BSR      INIT                 ;INIT AGC-1
BSR      COLTAB               ;LOAD COLOR TABLES

```

```

*   FOR LOGICAL DRAWING OPERATIONS THE ORIGIN MUST BE SET FIRST
*   THIS ROUTINE LOCATES THE ORIGIN IN THE LOWER LEFT EDGE OF
*   THE VISIBLE SCREEN

```

```

MOVE.W  #1,D0                ;PUT SCREEN NUMBER
MOVE.W  #0,D1                ;PUT X - COORDINATE
MOVE.W  1024,D2              ;PUT Y - COORDINATE
BSR     OPOINT               ;SET ORIGIN POINT
BSR     SAVORG               ;SAVE ORIGIN ADDRESS

```

```

*   SET THE COLOR REGISTERS

```

```

MOVE.W  #3,D1                ;CHOOSE COLOR BLUE
BSR     SCOLO                ;SET COLOR REGISTER 0
MOVE.W  #1,D1                ;CHOOSE COLOR RED
BSR     SCOL1               ;SET COLOR REGISTER 1

```

```

*   THIS SUBROUTINE MOVES THE CURRENT POINTER ABSOLUT TO LOGICAL
*   SCREEN ADDRESS AS DEFINED IN CPA

```

```
LEA    CPA(PC),A2
BSR    ABMOVE
```

```
*      THIS SUBROUTINE MOVES THE CURRENT POINTER RELATIV TO LOGICAL
*      SCREEN ADDRESS AS DEFINED IN CPR.
*      AFTER THIS THE CURRENT POINTER IS IN THE MIDDLE OF THE SCREEN
```

```
LEA    CPR(PC),A2
BSR    REMOVE
```

```
*      THIS SUBROUTINE DRAWS A CIRCLE OF THE RADIUS AS DEFINED IN RAD1
*      PLACING THE CURRENT POINTER AT THE CENTER
```

```
LEA    RAD1(PC),A2
MOVE.W #1,D1                ;DRAWING DIRECTION CLOCKWISE
BSR    CIRCLE
```

```
*      PAINTS THE CIRCLE BY THE COLOR IN THE COLOR REGISTER USING
*      THE FIGURE PATTERN STORED IN THE PATTEN RAM
```

```
MOVE.W #1,DO                ;COLOR TO DO (RED)
BSR    SCOLO                 ;LOAD COLOR REG 0 WITH RED
BSR    SCOL1                 ;LOAD COLOR REG 1 WITH RED
MOVE.W #1,D1                ;SET EDGE MODE
BSR    PAINT
```

```
*      DRAWS 20 RECTANGLES WITH INCREASING SIZE
```

```
LEA    REMR(PC),A2
BSR    ABMOVE
MOVE.W #19,D6                ;NUMBER OF RECTANGLE
LEA    REC1(PC),A2           ;START POSITION
BSR    REMOVE                ;MOVE RELATIVE
RECS   LEA    REXY(PC),A2     ;LOAD SIZE OF RECTANGLE
MOVEA.L A2,A3                ;SAVE A2
BSR    RERECT                ;DRAW RELATIVE RECTANGLE
ADDI.W #20,(A3)
ADDI.W #20,2(A3)
LEA    REC2(PC),A2           ;NEXT START POSITION (RELATIVE)
BSR    REMOVE                ;MOVE
DBRA   D6,RECS               ;ALL RECTANGLES ?
```

```
*      DRAWS AN ABSOLUTE POLYLINE WITH THREE POINTS AS DEFINED
*      IN POLP BEGINNING FROM THE CURRENT POINTER
```

```
LEA    POIN(PC),A2
BSR    ABMOVE
LEA    POLP(PC),A2
BSR    ABPOLL
XEXT
```

```
*
```

```
CPA    DC.W    590,462
CPR    DC.W    50,50
RAD1   DC.W    200
REMR   DC.W    220,512
```

```

REC1  DC.W  -30,-15
REXY  DC.W  60,30
RECN  DC.W  -10,-10

```

```

POIN  DC.W  800,512

```

```

POLP  DC.W  3                ;NUMBER OF POINTS
      DC.W  800,200
      DC.W  1000,200
      DC.W  1000,512

```

```

*****
*****

```

```

*
*   EXAMPLE FOR INITIALISATION OF THE SYS68K/AGC-1
*

```

```

INIT  LEA    INITTAB(PC),A2
INI1  CMPI.W #FFFFFF,(A2)
      BEQ.S  INITEND
      MOVE.W (A2)+,D0          ;REGISTER NUMBER
      MOVE.W (A2)+,D1          ;REGISTER PARAMETER
      BSR    WRCONR            ;WRITE CONTROL REGISTER
      BRA.S  INI1
INITEND RTS

```

```

*   INITIALISATION TABLE

```

```

*   THE LEAST REGISTER WHICH SHOULD BE WRITTEN IS REGISTER
*   NUMBER $0004 THE OMR-REGISTER. THE START BIT (BIT 14)
*   SHOULD BE SET.

```

```

*   SCREEN RESOLUTION:
*   1280 X 1024 X 4 BIT/PIXEL WITH 60 HZ INTERLACE
*   (SEE CHAPTER 7 IN THE HARDWARE USERS MANUAL)

```

	R.NO /PARAMETER	COMMENT
INITTAB DC.W	\$0082,\$6308	;(HSR) HORIZONTAL SYNCRON REGISTER +
DC.W	\$0084,\$074F	;(HDR) HORIZONTAL DISPLAY REGISTER +
DC.W	\$0086,\$0535	;(VSR) VERTIKAL SYNCRON REGISTER +
DC.W	\$0088,\$632F	;(VDR) VERTIKAL DISPLAY REGISTER +
* DC.W	\$008C,\$....	;(SSW1) UPPER SCREEN WIDTH REGISTER
DC.W	\$008A,\$0400	;(SSW0) BASE SCREEN WIDTH REGISTER +
* DC.W	\$008E,\$....	;(SSW2) LOWER SCREEN WIDTH REGISTER
* .		
* .		
DC.W	\$00CA,\$0280	;(MWR1) BASE SCREEN MEMORY WIDTH +
DC.W	\$00CC,\$0003	;(SA1H) BASE SCREEN START ADDRESS +
DC.W	\$00CE,\$0020	;(SA1L) BASE SCREEN START ADDRESS +
* .		
* .		
DC.W	\$00EA,\$0000	;(ZFR) ZOOM FAKTOR REGISTER +
* .		
* .		
DC.W	\$0002,\$0200	;(CCR) COMMAND CONTROL REGISTER +
DC.W	\$0006,\$C010	;(DCR) DISPLAY CONTROL REGISTER +
DC.W	\$0004,\$C03B	;(OMR) OPERATION MODE REGISTER +
DC.W	\$FFFF	

```

*   +) THESE REGISTER MUST BE SET FOR BASIC INITIALISATION OF THE
*   SYS68K/AGC-1

```

\*\*\*\*\*

\* AGC-1:SR

\*\*\*\*\*  
\*\*\*\*\*

\* SYS68K/AGC-1  
\* DRIVER MODULE  
\* MAI 1986  
\*  
\*  
\*  
\*

\*\*\*\*\*  
\*\*\*\*\*

\* SYS68K/AGC-1 BOARD BASE ADDRESS  
\* =====

BBADR DC.L \$B00000 ;BOARD BASE ADDRESS

\* OFFSET TABLE FOR  
\* FOR BASE ADDRESS CALCULATION  
\* =====

INDEXT DC.L \$3C000 ;ACRTC ADDRESSREGISTER  
DC.L \$3C002 ;ACRTC DATENREGISTER  
DC.L \$36000 ;COLOR TABLE RED  
DC.L \$38000 ;COLOR TABLE GRUEN  
DC.L \$3A000 ;COLOR TABLE BLUE  
DC.L \$00000 ;BIM START ADDRESS

\* MAIN ROUTINE  
\* =====

BINIT LEA BBADR(PC),A0 ;GET BOARD BASE ADDRESS  
LEA INDEXT(PC),A2 ;GET OFFSET TABLE ADDRESS  
MOVE.W #5,D1 ;NUMBER OF OFFSETS  
BINI1 MOVE.L (A2),D0 ;PUT OFFSET IN D0  
BSR.S BASINIT ;CALCULATE  
ADDQ.L #4,A2 ;POINT TO NEXT OFFSET  
DBRA D1,BINI1 ;ALL ?  
RTS

\* INITIALISATION TO THE BASE ADDRESS  
\* AND REPLACE THEM TO THE OFFSET TABLE  
\* =====

BASINIT MOVEA.L (A0),A1  
ADDA.L D0,A1 ;ADD OFFSET TO THE BOARD BASE ADDRES  
MOVE.L A1,(A2) ;REPLACE TO OFFSET TABLE  
RTS

\*\*\*\*\*  
\* DEFINITION CONTROLLER REGISTER  
\*\*\*\*\*

FIFO EQU \$0000 ;FIFO  
CCR EQU \$0002 ;COMMAND CONTROL REGISTER  
OMR EQU \$0004 ;OPERATION MODE REGISTER  
DCR EQU \$0006 ;DISPLAY CONTROL REGISTER

```

RCR      EQU      $0080      ;RASTER COUNT REGISTER
HSR      EQU      $0082      ;HORIZONTAL SYNC.
HDR      EQU      $0084      ;HORIZONTAL DISPLAY
VSR      EQU      $0086      ;VERTICAL SYNC.
VDR      EQU      $0088      ;VERTICAL DISPLAY REGISTER
SSW1     EQU      $008A      ;SPLIT SCREEN WIDTH SP1
SSW2     EQU      $008C      ;SPLIT SCREEN WIDTH SP2
SSW3     EQU      $008E      ;SPLIT SCREEN WIDTH SP3
BCR      EQU      $0090      ;BLINK CONTROL REGISTER
HWR      EQU      $0092      ;HORIZONTAL WINDOW DISPLAY
VWRS     EQU      $0094      ;VERTICAL WINDOW DISPLAY START
VVRW     EQU      $0096      ;                               WIDTH

```

\* UPPER SCREEN

```

RAR0     EQU      $00C0      ;RASTER ADDRESS UPPER SCREEN
MWR0     EQU      $00C2      ;MEMORY WIDTH      "
SARHO    EQU      $00C4      ;START ADDRESS    HIGH WORD
SARLO    EQU      $00C6      ;START ADDRESS    LOW  WORD

```

\* BASE SCREEN

```

RAR1     EQU      $00C8      ;RASTER ADDRESS BASE    SCREEN
MWR1     EQU      $00CA      ;MEMORY WIDTH      "
SARH1    EQU      $00CC      ;START ADDRESS    HIGH WORD
SARL1    EQU      $00CE      ;START ADDRESS    LOW  WORD

```

\* LOWER SCREEN

```

RAR2     EQU      $00D0      ;RASTER ADDRESS LOWER SCREEN
MWR2     EQU      $00D2      ;MEMORY WIDTH      "
SARH2    EQU      $00D4      ;START ADDRESS    HIGH WORD
SARL2    EQU      $00D6      ;START ADDRESS    LOW  WORD

```

\* WINDOW SCREEN

```

RAR3     EQU      $00D8      ;RASTER ADDRESS WINDOW SCREEN
MWR3     EQU      $00DA      ;MEMORY WIDTH      "
SARH3    EQU      $00DC      ;START ADDRESS    HIGH WORD
SARL3    EQU      $00DE      ;START ADDRESS    LOW  WORD

```

\* ZOOM FACTOR

```

ZFR      EQU      $00EA      ;ZOOM FACTOR
PAGE

```

\*\*\*\*\*

\* CONTROLLER FUNKTIONS CODES \*

\*\*\*\*\*

```

*NAME          CODE          ;COMMENT          *
```

-----\*

\* \*

\* \*

\* REGISTER ACCESS COMMANDS \*

```

ORIG     EQU      $0400      ;ORIGN POINT AND CHOOSE SCREEN
WPR      EQU      $0800      ;WRITE PARAMETER REGISTER
RPR      EQU      $0C00      ;READ  PARAMETER REGISTER
WPTN     EQU      $1800      ;WRITE PATTERN RAM
RPTN     EQU      $1C00      ;READ  PATTERN RAM

```

\*\*\*\*\*

\* \*

\* DATA TRANSFER COMMANDS \*

```

DRD      EQU      $2400      ;DMA  READ
DWT      EQU      $2800      ;DMA  WRITE
DMOD     EQU      $2C00      ;DMA  MODIFY
RD       EQU      $4400      ;READ (ONE WORD FROM THE FRAME BUFFER)
WT       EQU      $4800      ;WRITE(ONE WORD TO  THE FRAME BUFFER)
MOD      EQU      $4C00      ;MODIFY

```

```

CLR      EQU      $5800      ;CLEAR (INITIALIZE FRAME BUFFER AERA)
SCLR     EQU      $5C00      ;SELECTIVE CLEAR
CPY      EQU      $6000      ;COPY
SCPY     EQU      $7000      ;SELECTIVE COPY

```

\*\*\*\*\*

\* GRAPHIC DRAWING COMMANDS \*

```

AMOVE    EQU      $8000      ;ABSOLUTE MOVE
RMOVE    EQU      $8400      ;RELATIVE MOVE
ALINE    EQU      $8800      ;ABSOLUTE LINE
RLINE    EQU      $8C00      ;RELATIVE LINE
ARCT     EQU      $9000      ;ABSOLUTE RECTANGLE
RRCT     EQU      $9400      ;RELATIVE RECTANGLE
APLL     EQU      $9800      ;ABSOLUTE POLYLINE
RPLL     EQU      $9C00      ;RELATIVE POLYLINE
APLG     EQU      $A000      ;ABSOLUTE POLYGON
RPLG     EQU      $A400      ;RELATIVE POLYGON
CRCL1    EQU      $A900      ;CIRCLE CLOCKWISE
CRCLO    EQU      $A800      ;CIRCLE COUNTER CLOCKWISE
ELPS1    EQU      $AD00      ;ELLIPSE CLOCKWISE
ELPS0    EQU      $AC00      ;ELLIPSE COUNTER CLOCKWISE
AARC1    EQU      $B100      ;ABSOLUTE ARC CLOCKWISE
AARCO    EQU      $B000      ;ABSOLUTE ARC COUNTER CLOCKWISE
RAR1     EQU      $B500      ;RELATIVE ARC CLOCKWISE
RARCO    EQU      $B400      ;RELATIVE ARC COUNTER CLOCKWISE
AEARC1   EQU      $B900      ;ABSOLUTE ELLIPSE ARC CLOCKWISE
AEARCO   EQU      $B800      ;ABSOLUTE ELLIPSE ARC COUN CLOCKWISE
REAR1    EQU      $BD00      ;RELATIVE ELLIPSE ARC CLOCKWISE
REARCO   EQU      $BC00      ;RELATIVE ELLIPSE ARC COUN CLOCKWISE
AFRCT    EQU      $C000      ;ABSOLUTE FILLED RECTANGLE
RFRCT    EQU      $C400      ;RELATIVE FILLED RECTANGLE
PAINT0   EQU      $C800      ;PAINT0 ---> E = 0
PAINT1   EQU      $C900      ;PAINT1 ---> E = 1
DOT       EQU      $CC00      ;DRAW DOT
PTN       EQU      $D000      ;PATTERN
AGCPY    EQU      $E000      ;ABSOLUTE GRAPHIC COPY
RGCPY    EQU      $F000      ;RELATIVE GRAPHIC COPY

```

PAGE

\*\*\*\*\*

\* DEFINITION DRAWING PARAMETER REGISTER \*

\*\*\*\*\*

```

COREGO   EQU      $0800      ;COLOR REGISTER 0
COREG1   EQU      $0801      ;COLOR REGISTER 1
CCOREG   EQU      $0802      ;COLOR COMPARSION REGISTER
ECOREG   EQU      $0803      ;EDGE COLOR REGISTER
RMASK    EQU      $0804      ;MASK REGHISTER

```

\* PATTERN RAM CONTROL REGISTER

```

PRC05    EQU      $0805      ;PATTERN POINT REGISTER
PRC06    EQU      $0806      ;PATTERN START POINT REGISTER
PRC07    EQU      $0807      ;PATTERN END POINT REGISTER

```

\* AERA DEFINITION REGISTER

```

DRXMIN   EQU      $0808      ;XMIN
DRYMIN   EQU      $0809      ;YMIN
DRXMAX   EQU      $080A      ;XMAX
DRYMAX   EQU      $080B      ;YMAX

```

\* READ WRITE POINTER REGISTER

```

DRWPH    EQU      $080C      ;READ WRITE POINTER HIGH WORD
DRWPL    EQU      $080D      ;" LOW WORD

```

\* DRAWING POINTER REGISTER

```

PDPH EQU $0810 ;DRAWING POINTER HIGH WORD
PDPL EQU $0811 ;" LOW WORD

* CURRENT POINTER REGISTER
PCPH EQU $0812 ;CURRENT POINTER HIGH WORD
PCPL EQU $0813 ;" LOW WORD
PAGE

```

```

*****
* DEFINITION DER CONTROLLER COLOR TABELLEN

```

```

COLTR EQU INDEXT+8 ;TABELLE ROT
COLTG EQU INDEXT+12 ;TABELLE GRUEN
COLTB EQU INDEXT+16 ;TABELLE BLAU

```

```

*****
* GRAPHIC FUNKTIONEN ACRTC
* FOR ALL FUNKTIONEN THE PARAMETER ADDRESS MUST BE GIVEN IN A2
**

```

```

MODI DC.W $0000
COMFILL DC.W $0000 ;X - COORDINATE
DC.W $0000 ;Y - COORDINATE
DC.W $0000 ;PP - PATTERN POINTER (PRO5)
SLSD DC.W $0000 ;SLANT AND SOURCE DIRECTION
SDSD DC.W $0000 ;SOURCE- AND DESTINATION SCAN
* ;DIRECTION
MM DC.W $0000 ;MODIFY MODE
PRA DC.W $0000 ;PATTERN RAM ADDRESS
*
*

```

```

*****
* SUBROUTINE: WRCOM
* FUNKTION: WRITE COMMANDCODE TO AGC
* COMMAND-
* EXTENSION: --

```

```

* INPUT: D0 :COMMANDCODE
*
* OUTPUT: COMMANDCODE TO ACRTC
*
* INTERNAL: D1 :STATUS
*
WRCOM: MOVEM D0-D1/A0/A1, -(A7) ;STORE D0
LEA INDEXT(PC), A0
LEA INDEXT+4(PC), A1
WRC1 MOVE.W (A0), D1 ;READ STATUSREGISTER
BTST #5, D1 ;FIFO READY
BEQ WRC1
MOVE.W #FIFO, (A0) ;FIFO IS DESTINATION
MOVE.W D0, (A1) ;WRITE COMMAND
MOVEM (A7)+, D0-D1/A0/A1 ;RESTORE D0
RTS ;RETURN
*
*

```

```

*****
* SUBROUTINE: WRPARA
* FUNKTION: WRITE COMMANDPARAMETER TO ACRTC
* COMMAND-
* EXTENSION: --
*

```

```

*      INPUT:   D1                :NUMBER OF PARAMETERS
*              A2                :PARAMETERADDRESS
*      OUTPUT:  PARMETER TO ACRTC
*
*      INTERNAL: D2
*
*
WRPARA: MOVEM   D1-D2/A0-A2,-(A7)          ;STORE D1 AND A2
        LEA    INDEXT(PC),A0
        LEA    INDEXT+4(PC),A1
        BRA.S  WRPA2
WRPA1  MOVE.W   (A0),D2                    ;READ STATUSREGISTER
        BTST  #1,D2                        ;FIFO READY
        BEQ   WRPA1
        MOVE.W (A2)+,(A1)                  ;WRITE PARAMETER
WRPA2  DBRA    D1,WRPA1                    ;DEKREMT NUMBER PARAMTERS
        MOVEM (A7)+,D1-D2/A0-A2          ;RESTORE DATA
        RTS
        PAGE

```

```

*****
* SCOLO:      SET COLORREGISTER  0
*   IN:       D1          COLOR NUMBER

```

```

SCOLO:  MOVEM   D0-D2,-(A7)
        MOVE.W  #COREGO,D0          ;SET COLOR REGISTER 0
        AND    #$000F,D1
        MOVE.W  D1,D2
        LSL.W  #4,D1
        ADD    D2,D1
        LSL.W  #4,D1
        ADD    D2,D1
        LSL   #4,D1
        ADD    D2,D1
        BSR    WRDPAR
        MOVEM  (A7)+,D0-D2
        RTS

```

```

*****
* SCOL1:     SET COLORREGISTER  1
*   IN:      D1          COLOR NUMBER

```

```

SCOL1:  MOVEM   D0-D2,-(A7)
        MOVE.W  #COREG1,D0         ;SET COLOR REGISTER 1
        AND    #$000F,D1
        MOVE.W  D1,D2
        LSL.W  #4,D1
        ADD    D2,D1
        LSL.W  #4,D1
        ADD    D2,D1
        LSL   #4,D1
        ADD    D2,D1
        BSR    WRDPAR
        MOVEM  (A7)+,D0-D2
        RTS

```

```

*****
* SECOL:     SET EDGE COLORREGISTER
*   IN:      D1          COLOR NUMBER

```

```

SECOL:  MOVEM   D0-D2,-(A7)

```



```

MOVE.W #ECOREG,D0      ;SET EDGE COLOR REGISTER
AND     #$000F,D1
MOVE.W  D1,D2
LSL.W   #4,D1
ADD     D2,D1
LSL.W   #4,D1
ADD     D2,D1
LSL     #4,D1
ADD     D2,D1
BSR     WRDPAR
MOVEM   (A7)+,D0-D2
RTS

```

\*

\*\*\*\*\*

```

* SUBROUTINE:   SSLANT
* FUNKTION:    SET SLANT IN SLSD WORT (PATTERN)
* COMMAND-
* EXTENSION:   --
*
* INPUT:       DO                      :SLANT
*
* OUTPUT:      DO IN SLSD(BIT 11)
*
* INTERNAL:    D1  A2
*
*
*

```

```

SSLANT: MOVEM   DO/D1/A2,-(A7)          ;STORE DO AND D1
        CLR     D1
        LSL     #8,D0                  ;SHIFT LEFT 8 BIT
        LSL     #3,D0
        MOVE.W  #$07FF,D1              ;SET MASK
        LEA     SLSD(PC),A2            ;LOAD SLSD ADRESS
        AND.W   D1,(A2)
        OR.W    DO,(A2)                ;SET SLANT MODE IN SLSD
        MOVEM   (A7)+,DO/D1/A2        ;RESTORE
        RTS

```

\*

\*

\*\*\*\*\*

```

* SUBROUTINE:   SESCOI
* FUNKTION:    SET SCAN DIRECTION IN SLSD (PATTERN)
* COMMAND-
* EXTENSION:   --
*
* INPUT:       DO                      :SCAN DIRECTION
*
* OUTPUT:      DO IN SLSD IN SLSD (BIT 8 -10)
*
* INTERNAL:    D1  A2
*
*
*

```

```

SESCDI: MOVEM   DO/D1/A2,-(A7)          ;STORE DO AND D1
        CLR     D1
        AND.W   #$0007,D0
        LSL     #8,D0                  ;SHIFT LEFT 8 BIT
        MOVE.W  #$0800,D1              ;SET MASK
        LEA     SLSD(PC),A2            ;LOAD MODI ADRESS
        AND.W   D1,(A2)
        OR.W    DO,(A2)                ;SET SCAN DIRECTION IN SLSD
        MOVEM   (A7)+,DO/D1/A2        ;RESTORE
        RTS

```

```

*
*****
* SUBROUTINE:   SETSSD
* FUNKTION:    SET SOURCE SCAN DIRECTION IN SDSD
* COMMAND-
* EXTENSION:   --
*
* INPUT:       DO                               :SOURCE SCAN DIRECTION
*
* OUTPUT:      DO IN SDSD (BIT 11)
*
* INTERNAL:    D1  A2
*
*
SETSSD: MOVEM  DO/D1/A2, -(A7)                ;STORE DO AND D1
        CLR    D1
        AND.W  #$0001, DO                    ;BLANKED BIT 15 - 1
        LSL   #8, DO                        ;SHIFT LEFT 11 BIT
        LSL   #3, DO
        MOVE.W #$0700, D1                    ;SET MASK
        LEA   SDSD(PC), A2                  ;LOAD SDSD ADDRESS
        AND.W  D1, (A2)
        OR.W   DO, (A2)                    ;SET SSD MODE IN SDSD
        MOVEM (A7)+, DO/D1/A2              ;RESTORE
        RTS                                  ;RETURN
*

```

```

*****
* SUBROUTINE:   SETDSD
* FUNKTION:    SET DESTINATION SCAN DIRECTION IN SDSD
* COMMAND-
* EXTENSION:   --
*
* INPUT:       DO                               :DESTINATION SCAN DIRECTION
*
* OUTPUT:      DO IN SDSD (BIT 8 -10)
*
* INTERNAL:    D1  A2
*
*
SETDSD: MOVEM  DO/D1/A2, -(A7)                ;STORE DO AND D1
        CLR    D1
        AND.W  #$0007, DO                    ;BLANKED BIT 15 - 3
        LSL   #8, DO                        ;SHIFT LEFT 8 BIT
        MOVE.W #$0800, D1                    ;SET MASK
        LEA   SDSD(PC), A2                  ;LOAD SDSD ADDRESS
        AND.W  D1, (A2)
        OR.W   DO, (A2)                    ;SET DSD MODE IN SDSD
        MOVEM (A7)+, DO/D1/A2              ;RESTORE
        RTS                                  ;RETURN
        PAGE
*

```

```

*****
* SUBROUTINE:   SETMFY
* FUNKTION:    SET MODIFY MODE IN MM
* COMMAND-
* EXTENSION:   --
*
* INPUT:       DO                               :MODIFY MODE
*
* OUTPUT:      DO IN MM (BIT 1-0)
*
* INTERNAL:    D1  A2
*

```

```

*
*
*
SETMFY: MOVEM    DO/D1/A2,-(A7)          ;STORE DO AND D1
        CLR      D1
        AND.W    #$0003,D0              ;BLANK OUT
        MOVE.W   #$FFFC,D1             ;SET MASK
        LEA     MM(PC),A2               ;LOAD MM  ADDRESS
        AND.W    D1,(A2)
        OR.W     DO,(A2)                ;SET MODIFY MODE IN MM
        MOVEM   (A7)+,DO/D1/A2         ;RESTORE
        RTS
        ;RETURN

```

```

*
*
*****

```

```

* SUBROUTINE:    SETPRA
* FUNKTION:     SET PATTERN RAM ADDRESS IN PRA
* COMMAND-
* EXTENSION:    --
*
* INPUT:        DO                      :PATTERN RAM ADDRESS
*
* OUTPUT:       DO IN PRA (BIT 3 - 0)
*
* INTERNAL:     D1  A2
*
*
*

```

```

SETPRA: MOVEM    DO/D1/A2,-(A7)          ;STORE DO AND D1
        CLR      D1
        AND.W    #$000F,D0              ;SET MASK
        MOVE.W   #$FFF0,D1             ;SET MASK
        LEA     PRA(PC),A2             ;LOAD PRA ADDRESS
        AND.W    D1,(A2)
        OR.W     DO,(A2)                ;SET PATTERN RAM ADDRESS IN PRA
        MOVEM   (A7)+,DO/D1/A2         ;RESTORE
        RTS
        ;RETURN
PAGE

```

```

*
*****

```

```

* SUBROUTINE:    SETAER
* FUNKTION:     SET AERAMODE IN MODI-BYTE
* COMMAND-
* EXTENSION:    --
*
* INPUT:        DO                      :AERA MODE
*
* OUTPUT:       DO IN MODI (BIT 7-5)
*
* INTERNAL:     D1  A2
*
*
*

```

```

SETAER: MOVEM    DO/D1/A2,-(A7)          ;STORE DO AND D1
        CLR      D1
        LSL     #5,D0                  ;SHIFT LEFT 5 BIT
        MOVE.W   #$001F,D1             ;SET MASK
        LEA     MODI(PC),A2            ;LOAD MODI ADDRESS
        AND.W    D1,(A2)
        OR.W     DO,(A2)                ;SET AERA MODE IN MODI
        MOVEM   (A7)+,DO/D1/A2         ;RESTORE
        RTS
        ;RETURN

```

```

*
*
*****

```

```

* SUBROUTINE:   SETCOL
* FUNKTION:    SET COLORMODE IN MODI-BYTE
* COMMAND-
* EXTENSION:   --
*
* INPUT:       D0                               :COLORMODE
*
* OUTPUT:      D0 IN MODI (BIT 4-3)
*
* INTERNAL:    D1
*
*
*
* SETCOL: MOVEM D0/D1/A2,-(A7)                   ;STORE D0 AND D1
*          CLR  D1
*          LSL  #3,D0                             ;SHIFT LEFT 3 BIT
*          MOVE.W #00E7,D1                       ;SET MASK
*          LEA  MODI(PC),A2                       ;LOAD MODI
*          AND.W D1,(A2)
*          OR.W D0,(A2)                           ;SET COLORMODE IN MODI
*          MOVEM (A7)+,D0/D1/A2                  ;RESTORE
*          RTS
*          PAGE

```

```

*
*****
* SUBROUTINE:   SETOPM
* FUNKTION:    SET OPERATIONMODE IN MODI-BYTE
* COMMAND-
* EXTENSION:   --
*
* INPUT:       D0                               :OPERATIONMODE
*
* OUTPUT:      D0 IN MODI (BIT 0-2)
*
* INTERNAL:    D1
*
*
*
* SETOPM: MOVEM D0/D1/A2,-(A7)                   ;STORE D0 AND D1
*          CLR  D1
*          MOVE.W #00F8,D1                       ;SET MASK
*          LEA  MODI(PC),A2                       ;LOAD MODI
*          AND.W D1,(A2)
*          OR.W D0,(A2)                           ;SET OPERATIONMODE IN MODI
*          MOVEM (A7)+,D0/D1/A2                  ;RESTORE
*          RTS
*          ;RETURN

```

```

*
*****
* SUBROUTINE:   SETMOD
* FUNKTION:    INSERTS AERA-, COLOR- AND OPERATIONMODE IN THE
*              COMMANDCODE
* COMMAND-
* EXTENSION:
*
* INPUT:       D0                               :COMMANDCODE
*
* OUTPUT:      D0                               :MODIFIERED COMMANDCODE
*
* INTERNAL:
*
*
* SETMOD: MOVEM A2,-(A7)                         ;STORE A2
*          LEA  MODI(PC),A2                       ;LOAD MODI
*          OR.W (A2),D0                           ;INSERT MODI IN COMMANDCODE
*          MOVEM (A7)+,A2                         ;RESTORE A2

```

RTS  
PAGE

```
*
*****
* SUBROUTINE: WRCONR
* FUNKTION: WRITE PARAMETER IN THE SELECTED CONTROL REGISTER
* COMMAND-
* EXTENSION: ---
*
* INPUT: DO :REGISTER NUMBER
* D1 :PARAMTER
* OUTPUT: REGISTER TO ACRTC ADRESSREGISTER
* PARAMETER ACRTC DATAREGISTER
* INTERNAL:
*
*
WRCONR: MOVEM DO-D1/A0/A1,-(A7)
LEA INDEXT(PC),A0
LEA INDEXT+4(PC),A1
MOVE.W DO,(A0) ;REGISTER IS DESTINATION
MOVE.W D1,(A1) ;WRITE PARAMETER
MOVEM (A7)+,DO-D1/A0/A1
RTS
*
*
```

```
*****
* SUBROUTINE: RECONR
* FUNKTION: READ THE PARAMETER FROM THE SELECTED CONTROL REGISTER
* COMMAND-
* EXTENSION: ---
*
* INPUT: DO :REGISTER NUMBER
*
* OUTPUT: D1 :PARAMETER
*
* INTERNAL: D1
*
*
RECONR: MOVEM DO/A0-A1,-(A7)
LEA INDEXT(PC),A0
LEA INDEXT+4(PC),A1
MOVE.W DO,(A0) ;DESTINATION REGISTER
MOVE.W (A1),D1 ;READ PARAMETER
MOVEM (A7)+,DO/A0-A1
RTS
PAGE
*
*
```

```
*****
* SUBROUTINE: WRDPAR
* FUNKTION: WRITE DRAWING PARAMETER INTO SELECTED DRAWING PARAMETER
* REGISTER
* COMMAND-
* EXTENSION: ---
*
* INPUT: DO :REGISTER NUMBER
* D1 :PARAMETER
* OUTPUT:
*
* INTERNAL:
*
*
WRDPAR: MOVEM DO-D2/A0-A1,-(A7)
LEA INDEXT(PC),A0
LEA INDEXT+4(PC),A1
WRDP1 MOVE.W (A0),D2 ;READ STATUSREGISTER
BTST #1,D2 ;WRITE FIFO READY
*
*
```

```

BEQ      WRDP1
MOVE.W  #FIFO,(A0)
MOVE.W  DO,(A1)           ;WRITE COMMAND
MOVE.W  D1,(A1)          ;WRITE PARAMETER
MOVEM   (A7)+,DO-D2/A0-A1
RTS

```

\*  
\*

\*\*\*\*\*

```

* SUBROUTINE:  REDPAR
* FUNKTION:   LOAD PARAMETER FROM SELECTED DRAWING PARAMETER REGISTER
*             INTO THE READ FIFO

```

```

* COMMAND-
* EXTENSION:  ---

```

```

* INPUT:     D0           :REGISTER NUMBER

```

```

* OUTPUT:    D1           :PARAMETER

```

```

* INTERNAL:  D2

```

\*

```

REDPAR: MOVEM  DO/D2/A0-A1,-(A7)
        LEA   INDEXT(PC),A0
        LEA   INDEXT+4(PC),A1
        BSR.S CLFIFO           ;CLEAR READ FIFO
        MOVE.W #FIFO,(A0)      ;FIFO IS DESTINATION
        MOVE.W DO,(A1)         ;WRITE REGISTER READ COMMAND
        MOVE.W (A1),D1         ;READ PARAMETER FROM FIFO
        MOVEM (A7)+,DO/D2/A0-A1
        RTS
        PAGE

```

\*

\*\*\*\*\*

```

* SUBROUTINE:  RFIPAR
* FUNKTION:   READ PARAMETERS FROM READ FIFO
*             (BE SURE )

```

```

* COMMAND-
* EXTENSION:  ---

```

```

* INPUT:     A2           :PARAMETER LIST

```

```

* OUTPUT:    WRITE THE CONTENTS OF READ FIFO INTO THE PARAMETERLIST

```

```

* INTERNAL:  D1

```

\*

```

RFIPAR: MOVEM  D1/A0-A2,-(A7)
        LEA   INDEXT(PC),A0
        LEA   INDEXT+4(PC),A1
RFST   MOVE.W  (A0),D1         ;READ STATUS REGISTER
        BTST  #2,D1           ;READ FIFO EMPTY
        BEQ.S RFEND
        MOVE.W (A1),(A2)+      ;
        BRA   RFST
RFEND  MOVEM   (A7)+,D1/A0-A2
        RTS

```

\*

\*

\*\*\*\*\*

```

* SUBROUTINE:  CLFIFO
* FUNKTION:   SET AND RESET THE ABORT-BIT IN THE COMMAND CONTROL REGISTER
*             AFTER EXECUTION THE READ FIFO AND THE WRITE FIFO WILL BE
*             CLEARED

```

```

* COMMAND-
* EXTENSION:  ---

```

```

* INPUT:  --
*
* OUTPUT:  --
*
* INTERNAL:  D0 D1 D2
*
*

```

```

CLFIFO: MOVEM  D0-D2,-(A7)
        MOVE.W #CCR,D0          ;LOAD REGISTER NUMBER
        BSR    RECONR          ;READ CONTROL REGISTER
        MOVE.W #8000,D2        ;LOAD MASK
        OR.W   D2,D1           ;SET ABORT BIT
        BSR    WRCONR          ;WRITE CONTROLREGISTER
        MOVE.W #7FFF,D2        ;LOAD MASK
        AND.W  D2,D1           ;RESET THE ABORT BIT
        BSR    WRCONR          ;WRITE CONTROL REGISTER
        MOVEM (A7)+,D0-D2
        RTS
        PAGE

```

```

*****

```

```

* SUBROUTINE:  WPARAM
* FUNKTION:    WRITE DATA TO THE PATTERN RAM
* COMMAND-
* EXTENSION:   PRA              :PATTERN RAM ADDRESS
*
* INPUT:       D0              :COMMAND CODE
*              D1              :NUMBER OF WORDS
*              A2              :PARAMETER ADDRESS
*
* OUTPUT:
*
* INTERNAL:
*
*

```

```

WRPLIS DS.W 16

```

```

WPARAM: MOVEM  D0-D3/A2/A3/A4,-(A7)
        LEA   PRA(PC),A3
        MOVE.W (A3),D2
        OR.W  D2,D0          ;INSERT COMMAND EXTENSION
        MOVE.W D1,D3         ;SAVE NUMBER OF WORDS
WPR1   LEA   WRPLIS(PC),A3   ;LOAD PATTERN LIST ADDRESS
        MOVEA.L A3,A4
        MOVE.W D3,(A3)+     ;WRITE ORIGN NUMBER OF WORDS
        MOVE.W D1,D3         ;LOOP COUNTER TO D3
        BRA.S WPLP2
WPLP1  MOVE.W (A2)+,(A3)+
WPLP2  DBRA D3,WPLP1        ;DEKREMENT AND JUMP
        MOVEA.L A4,A2       ;7777777777
        BSR    WRCOM        ;WRITE COMMAND
        ADD.W  #1,D1
        BSR    WRPARA       ;WRITE PARAMETER
        MOVEM (A7)+,D0-D3/A2/A3/A4
        RTS

```

```

*
*
*

```

```

*****

```

```

* SUBROUTINE:  RPARAM
* FUNKTION:    READ DATA FROM THE PATTERN RAM
* COMMAND-
* EXTENSION:   PRA
*
* INPUT:       D0              :COMMAND CODE
*              D1              :NUMBER OF WORDS TO BE READ

```

```

*      OUTPUT:      A2      :
*      INTERNAL:
*
LERAM   DC.W      $0000      ;PARAMETER ADDRESS
PRLIST  DS.W      16         ;RESERVE 16 WORD

```

```

RPARAM: MOVEM     DO-D2/A2/A3, -(A7)
        BSR       CLFIFO      ;CLEAR FIFO
        LEA       PRA(PC), A3  ;LOAD COMMAND EXTENSION ADDRESS
        MOVE.W    (A3), D2
        OR.W      D2, D0       ;INSERT COMMAND EXTENSION
        LEA       LERAM(PC), A2 ;LOAD PARAMETER ADDRESS
        MOVE.W    D1, (A2)     ;
        BSR       WRCOM       ;WRITE COMMAND
        MOVE.W    #1, D1       ;ONE PARAMETER TO BE WRITTEN
        BSR       WRPARA      ;WRITE PARAMETER
        LEA       PRLIST(PC), A2 ;LOAD PARAMETERLIST ADDRESS
        BSR       RFIPAR      ;READ OUT READ FIFO
        MOVEM     (A7)+, DO-D2/A2/A3
        RTS
        PAGE

```

```

*
*****

```

```

* SUBROUTINE:      CLPATT
*   FUNKTION:      CLEARED THE PATTERN
*   COMMAND-
*   EXTENSION:
*
*   INPUT:         --
*
*   OUTPUT:        --
*
*   INTERNAL:      A2 DO D1
*
*

```

```

CLPATT: MOVEM     DO/D1/A2, -(A7)
        LEA       PRAMO(PC), A2
        MOVE.W    #WPTN, DO
        MOVE.W    #16, D1
        BSR       WPARAM
        MOVEM     (A7)+, DO/D1/A2
        RTS
        PAGE

```

```

*
*****

```

```

* SUBROUTINE:      CPMOVE
*   FUNKTION:      MOVED CURRENTPOINTER ABSOLUT OR RELATIVE
*   COMMAND-
*   EXTENSION:      --
*
*   INPUT:         DO          :COMMANDCODE
*                  A2          :PARAMETERADDRESS
*   OUTPUT:        COMMANDCODE --> ACRTC
*                  PARAMETER  --> ACRTC
*   INTERNAL:      D1
*
*

```

```

CPMOVE: MOVEM     DO-D1/A2, -(A7)
        BSR       WRCOM       ;WRITE COMMAND
        MOVE.W    #2, D1      ;NUMBER OF PARAMETERS
        BSR       WRPARA      ;WRITE PARAMETERS
        MOVEM     (A7)+, DO-D1/A2

```





```
BSR      WRPARA      ;WRITE PARAMETER
MOVEM   (A7)+,DO-D1/A2 ;RESTORE
RTS
PAGE
```

\*

\*\*\*\*\*

```
* SUBROUTINE:  DELLIP
*   FUNKTION:  DRAW A ELLIPSE CLOCKWISE OR COUNTERCLOCKWISE
*   COMMAND-
*   EXTENSION: AERA - COL - OPM
*
*   INPUT:    DO      :COMMANDCODE
*             A2      :PARAMETERADDRESS
*   OUTPUT:   COMMANDCODE --> ACRTC FIFO
*             PARAMETER  --> ACRTC FIFO
*   INTERNAL:  D1
```

\*

\*

```
DELLIP: MOVEM   DO-D1/A2,-(A7)
        BSR     SETMOD      ;INSERT COMMANDEXTENSION
        BSR     WRCOM       ;WRITE COMMAND
        MOVE.W  #3,D1       ;NUMBER OF PARAMETERS
        BSR     WRPARA      ;WRITE PARAMETER
        MOVEM  (A7)+,DO-D1/A2 ;RESTORE
        RTS
```

\*

\*

\*\*\*\*\*

```
* SUBROUTINE:  RLARC
*   FUNKTION:  DRAW CLOCKWISE OR COUNTERCLOCKWISE AN ABSOLUTE OR RELATIVE
*               ARC
*   COMMAND-
*   EXTENSION: AERA - COL - OPM
*
*   INPUT:    DO      :COMMANDCODE
*             A2      :PARAMETERADDRESS
*   OUTPUT:   COMMAND  --> ACRTC FIFO
*             PARAMETER--> ACRTC FIFO
*   INTERNAL:  D1
```

\*

\*

```
RLARC:  MOVEM   DO-D1/A2,-(A7)
        BSR     SETMOD      ;INSERT COMMANDEXTENSION
        BSR     WRCOM       ;WRITE COMMAND
        MOVE.W  #4,D1       ;NUMBER OF PARAMETERS
        BSR     WRPARA      ;WRITE PARAMETER
        MOVEM  (A7)+,DO-D1/A2
        RTS
        PAGE
```

\*

\*\*\*\*\*

```
* SUBROUTINE:  ELLARC
*   FUNKTION:  DRAW CLOCKWISE OR COUNTERCLOCKWISE AN ABSOLUTE OR RELATIVE
*               ELLIPSE ARC
*   COMMAND-
*   EXTENSION: AERA - COL - OPM
*
*   INPUT:    DO      :COMMANDCODE
*             A2      :PARAMETERADDRESS
*   OUTPUT:   COMMAND  --> ACRTC FIFO
*             PARAMETER--> ACRTC FIFO
*   INTERNAL:  D1
```

\*

\*

```
ELLARC: MOVEM   DO-D1/A2,-(A7)
        BSR     SETMOD      ;INSERT COMMANDEXTENSION
```

```

BSR      WRCOM                ;WRITE COMMAND
MOVE.W  #6,D1                ;NUMBER OF PARAMETERS
BSR      WRPARA              ;WRITE PARAMETER
MOVEM   (A7)+,D0-D1/A2
RTS

*****
* SUBROUTINE:  FILREC
*   FUNKTION:  FILLS AN ABSOLUTE OR RELATIVE RECTANGULAR AREA SPECIFIED
*              WITH CP AND COMMAND PARAMETER WITH THE FIGURE PATTERN STORED
*              IN THE PATTERN RAM
*   COMMAND-
*   EXTENSION:  AERA - COL - OPM
*
*   INPUT:     D0                :COMMANDCODE
*              A2                :PARAMETERADDRESS
*   OUTPUT:    COMMAND  --> ACRTC FIFO
*              PARAMETER --> ACRTC FIFO
*   INTERNAL:  D1
*
FILREC:  MOVEM   D0-D1/A2,-(A7)
        BSR     SETMOD                ;INSERT COMMANDEXTENSION
        BSR     WRCOM                ;WRITE COMMAND
        MOVE.W  #2,D1                ;NUMBER OF PARAMETER
        BSR     WRPARA              ;WRITE PARAMETER
        MOVEM   (A7)+,D0-D1/A2
        RTS

*
*****
* SUBROUTINE:  FILL
*   FUNKTION:  FILLS A CLOSED AEREA SURROUNDED BY EDGE COLOR DEFINED IN
*              THE PARAMETERREGISTER (EDG) USING THE FIGURE PATTERN
*              SPECIFIED IN THE PATTERN RAM
*   COMMAND-
*   EXTENSION:  AERA      (COL-MODE MUST SPECIFIED 00)
*                (OPM-MODE MUST SPECIFIED 000)
*                E        EGDECOLOR INCLUDED IN THE COMMANDCODE
*   INPUT:     D0        COMMANDPARATER
*              A2        PARAMETERADDRESS
*   OUTPUT:    COMMAND  --> ACRTC FIFO
*              PARAMETER --> ACRTC FIFO
*   INTERNAL:
*
FILL:   MOVEM   D0-D2/A0/A2,-(A7)
        LEA    INDEXT(PC),A0          ;GET ADRESSREGISTER ADDRESS
        LEA    COMFILL(PC),A2        ;LOAD PARAMETERLIST FOR COMPLEX
*                                     ;FIGURE PAINTING
        BSR    SETMOD                ;INSERT COMMANDEXTENSION
        MOVE.W #$FFEO,D1            ;LOAD MASK
        AND.W  D1,D0                ;MASK COMMANDCODE
        BSR    CLFIFO              ;CLEAR FIFO
FILSTA  BSR    WRCOM                ;WRITE COMMAND
        CLR.L  D2                  ;CLEAR D3
RAGAIN  MOVE.W (A0),D2              ;READ STATUSREGISTER
        BTST  #2,D2                ;READ FIFO READY
        BNE.S REFIFO              ;YES --> READ FIFO
        BTST  #5,D2                ;COMMAND END
        BEQ   RAGAIN              ;NO --> READ AGAIN
        BRA.S FILEND              ;COMMAND END
REFIFO  BSR    RFIPAR              ;READ FIFO
        MOVE.W 4(A2),D1            ;LOAD PP --> D1
        MOVE.W #PRC05,D0          ;LOAD PATTERN RAM CONTR.REG (PR05)
        BSR    WRDPAR              ;SPECIFY THE PATTERN POINT
        MOVE.W #AMOVE,D0          ;LOAD ABSOLUTE MOVE

```

```

        BSR      CPMOVE          ;SPECIFY THE START POINT
        BRA      FILSTA         ;FILL THE AERA
FILEND  MOVEM   (A7)+,DO-D2/A0/A2
        RTS
        PAGE

```

```

*
*****

```

```

* SUBROUTINE:  SETDOT
* FUNKTION:   MARKS A DOT ON THE COORDINATE WHERE THE CP POINTS
* COMMAND-
* EXTENSION:  AERA - COL - OPM
*
* INPUT:      DO                :COMMAND CODE
*
* OUTPUT:
*
* INTERNAL:
*
*

```

```

SETDOT: MOVEM   DO,-(A7)
        BSR     SETMOD          ;INSERT COMMAND EXTENSION
        BSR     WRCOM           ;WRITE COMMAND
        MOVEM   (A7)+,DO
        RTS

```

```

*
*****

```

```

* SUBROUTINE:  PATDRA
* FUNKTION:   DRAW PATTERN ONTO THE RECTANGULAR AERA SPECIFIED BY THE
*             CURRENT POINTER AND BY THE PATTERN SIZE
* COMMAND-
* EXTENSION:  AERA - COL - OPM
*
* SL          :SLANT             (BIT 11 )
* SD          :SOURCE DIRECTION (BIT 10 - 8)
*
* INPUT:      DO                :COMMAND CODE
*             A2                :PARAMETERADDRESS
*
* OUTPUT:
*
* INTERNAL:
*
*

```

```

PATDRA: MOVEM   DO-D1/A2/A3,-(A7)
        MOVEA.L A2,A3          ;STORE PARAMETERADDRESS
        BSR     SETMOD          ;INSERT COMMAND EXTENSION
        LEA     SLSD(PC),A3    ;LOAD SL AND SD EXTENSION ADDRESS
        MOVE.W  (A3),D1
        OR.W    D1,DO          ;INSERT COMMAND EXTENSION EXCEPT
*                                     ;AERA - COL - OPM MODE
        BSR     WRCOM           ;WRITE COMMAND
        MOVE.W  #1,D1          ;NUMBER OF PARAMETER
        BSR     WRPARA         ;WRITE PARAMETER
        MOVEM   (A7)+,DO-D1/A2/A3
        RTS
        PAGE

```

```

*
*****

```

```

* SUBROUTINE:  GRACPY
* FUNKTION:   COPIES A RECTANGULAR AREA SPECIFIED BY THE ABSOLUTE OR
*             RELATIVE COORDINATES TO THE ADDRESS SPECIFIED BY THE
*             CURRENT POINTER (CP)
* COMMAND-
* EXTENSION:  AERA - COL - OPM
*
* S          :SOURCE SCAN DIRECTION (BIT 11)
* DSD        :DESTINATION SCAN DIRECTION (BIT 10-8)
*
* INPUT:      DO                :COMMAND CODE

```

```

*           A2                               :PARAMETERADDRESS
*   OUTPUT:
*
*   INTERNAL:
*
*
GRACPY:  MOVEM   DO-D1/A2/A3,-(A7)
          MOVEA.L A2,A3                       ;STORE PARAMETERADDRESS
          BSR     SETMOD                       ;INSERT COMMANDEXTENSION
          MOVE.W  #$FFC7,D1                   ;LOAD MASK
          AND.W   D1,D0                       ;MASK COMMAND CODE
          LEA     SDSD(PC),A3                 ;LOAD EXTENSION ADDRESS
          MOVE.W  (A3),D1                     ;
          OR.W    D1,D0                       ;INSERT EXTENSION S AND DSD
          BSR     WRCOM                       ;WRITE COMMAND
          MOVE.W  #4,D1                       ;NUMBER OF PARAMETER
          BSR     WRPARA                      ;WRITE PARAMETER
          MOVEM   (A7)+,DO-D1/A2/A3
          RTS
          PAGE
*

```

```

*****
*
*   ASSEMBLER INTERFACE DRIVER ACRTC
*
*****

```

```

*****
* SUBROUTINE:   OPOINT
* FUNKTION:    ASSOCIATES A LOGICAL X - Y SCREEN WITH PHYSICAL FRAME
*              BUFFER ADDRESS
*   INPUT:     DO                               :SCREEN NUMBER
*              D1                               :X - COORDINATE
*              D2                               :Y - COORDINATE
*   OUTPUT:
*
*   INTERNAL:
*
*****

```

```

OPOINT:  MOVEM   DO-D7/A2/A3,-(A7)
          LEA     BILDPA(PC),A2               ;LOAD DISPLAY PARAMETER
          MOVE.L  (A2)+,D3                   ;PIXEL PER LINE (FRAME)
          DIVU    #4,D3                      ;IN WORD PER LINE
          MOVE.L  (A2)+,D4                   ;NUMBER OF RASTERS (FRAME)
          MOVE.L  (A2),D5                    ;DISPLAY START ADDRESS
          MULU    D2,D3
          DIVU    #4,D1                      ;X COORDINATE IN WORDS
          AND.L   #$0000FFFF,D1             ;REST AUSBLENDEN
          ADD.L   D1,D3                      ;PHYSICAL ADDRESS
          MOVE.L  D3,D6
          DIVU    #4,D6
          LSR.L   #8,D6
          ;

```

```

LSR.L    #8,D6                ;PIXEL IN WORD (DPD)
AND.L    #8C,D6              ;MASKED
ADD.L    D5,D3                ;PLUS DISPLAY START OFFSET
LSL.L    #4,D3                ;
OR.L     D6,D3                ;INSERT DPD
AND.L    #00FFFFFF,D3
MOVE.L   #30,D7
LSL.L    D7,D0
OR.L     D0,D3                ;INSERT SCREEN NUMBER
LEA     ORGRWD(PC),A3
LEA     ORGPAR(PC),A2        ;NULL POINT
MOVE.W  D3,2(A3)
MOVE.W  D3,2(A2)            ;STORE DPL
SWAP    D3                    ;
MOVE.W  D3,(A3)
MOVE.W  D3,(A2)            ;STORE DPH
MOVE.W  #ORIG,D0
BSR     CPMOVE                ;SET ORIGN
MOVEM.L (A7)+,D0-D7/A2/A3
RTS

```

```

ORGPARG  DC.W    $0000        ;DPH
         DC.W    $0000        ;DPL

```

```

ORGRWD   DC.W    $0000        ;DGH
         DC.W    $0000        ,DPL

```

```

BILDPA   DC.L    1536         ;PIXEL PER LINE
         DC.L    1200         ;NUMBER OF RASTER
         DC.L    $12AC0       ;DISPLAY START ADDRESS

```

```

*****
*       SAVE ORIGN

```

```

         EVEN
ORIGSAV  DC.W    $0000
         DC.W    $0000

```

```

SAVORG:  MOVEM.L  A4/A3,-(A7)
         LEA     ORGPAR(PC),A4
         LEA     ORIGSAV(PC),A3
         MOVE.W (A4)+,(A3)+
         MOVE.W (A4),(A3)
         MOVEM.L (A7)+,A4/A3
         RTS

```

```

*****
*       CLEAR    SCREEN
DSCCLPA  DC.W    $0,$FF,$320

```

```

DSCLEAR  MOVEM.L  D0-D2/A2,-(A7)
         MOVE.L  #1,D0
         MOVE.L  #0,D1
         MOVE.L  #800,D2
         BSR     OPOINT
         LEA     ORGRWD(PC),A2
         MOVE.W  #DRWPH,D0
         MOVE.W  (A2)+,D1
         BSR     WRDPAR
         MOVE.W  #DRWPL,D0
         MOVE.W  (A2),D1
         BSR     WRDPAR
         MOVE.W  #CLR,D0
         BSR     WRCOM

```

```
MOVE.W #3,D1
LEA DSCLPA(PC),A2
BSR WRPARA

LEA ORIGSAV(PC),A2
MOVE.W #ORIG,DO
BSR CPMOVE
MOVEM.L (A7)+,DO-D2/A2
RTS
```

```
*****
* SUBROUTINE: ABMOVE
* FUNKTION: ABSOLUT MOVE
*
* INPUT: A2 :PARAMETER ADDRESS
*
* OUTPUT:
*
* INTERNAL:
*
*****
```

```
ABMOVE: MOVE.W #ABMOVE,DO
BSR CPMOVE
RTS
```

```
*****
* SUBROUTINE: REMOVE
* FUNKTION: RELATIVE MOVE
*
* INPUT: A2 :PARAMETER ADDRESS
*
* OUTPUT:
*
* INTERNAL:
*
*****
```

```
REMOVE: MOVE.W #REMOVE,DO
BSR CPMOVE
RTS
```

```
*****
* SUBROUTINE: ABLINE
* FUNKTION: DRAW AN ABSOLUT LINE
*
* INPUT: A2 :PARAMETER ADDRESS
*
* OUTPUT:
*
* INTERNAL:
*
*****
```

```
ABLINE: MOVE.W #ALINE,DO
BSR LINREC
```

RTS

```
*****  
* SUBROUTINE:  RELINE  
*   FUNKTION:  DRAW A RELATIVE LINE  
*  
*   INPUT:    A2                      :PARAMETER ADDRESS  
*  
*   OUTPUT:  
*  
*   INTERNAL:  
*  
*****
```

```
RELINE: MOVE.W  #RLINE,DO  
        BSR    LINREC  
        RTS
```

```
*****  
* SUBROUTINE:  ABRECT  
*   FUNKTION:  DRAW AN ABSOLUT RECTANGLE  
*  
*   INPUT:    A2                      :PARAMETER ADDRESS  
*  
*   OUTPUT:  
*  
*   INTERNAL:  
*  
*****
```

```
ABRECT: MOVE.W  #ARCT,DO  
        BSR    LINREC  
        RTS
```

```
*****  
* SUBROUTINE:  RERECT  
*   FUNKTION:  DRAW A RELATIVE RECTANGLE  
*  
*   INPUT:    A2                      :PARAMETER ADDRESS  
*  
*   OUTPUT:  
*  
*   INTERNAL:  
*  
*****
```

```
RERECT: MOVE.W  #RRCT,DO  
        BSR    LINREC
```



RTS

```
*****
* SUBROUTINE:  ABPOLL
* FUNKTION:   ABSOLUT POLYLINE
*
* INPUT:      A2                :PARAMETER ADDRESS
*
* OUTPUT:
*
* INTERNAL:
*
*****
```

```
ABPOLL: MOVE.W  #APLL,DO
        BSR    POLYLG
        RTS
```

```
*****
* SUBROUTINE:  REPOLL
* FUNKTION:   RELATIVE POLYLINE
*
* INPUT:      A2                :PARAMETER ADDRESS
*
* OUTPUT:
*
* INTERNAL:
*
*****
```

```
REPOLL: MOVE.W  #RPLL,DO
        BSR    POLYLG
        RTS
```

```
*****
* SUBROUTINE:  ABPOLG
* FUNKTION:   ABSOLUT POLYGON
*
* INPUT:      A2                :PARAMETER ADDRESS
*
* OUTPUT:
*
* INTERNAL:
*
*****
```

```
ABPOLG: MOVE.W  #APLG,DO
        BSR    POLYLG
```

RTS

```
*****
* SUBROUTINE:  REPOLG
* FUNKTION:   RELATIVE POLYGON
*
* INPUT:      A2                :PARAMETER ADDRESS
*
* OUTPUT:
*
* INTERNAL:
*
*****
```

```
REPOLG: MOVE.W #RPLG,DO
        BSR    POLYLG
        RTS
```

```
*****
* SUBROUTINE:  CIRCLE
* FUNKTION:   DRAW A CIRCLE
*
* INPUT:      A2                :PARAMETER ADDRESS
*             D1                :DRAWING DIRECTION
*             :D1 = 1  CLOCKWISE
*             :D1 = 0  COUNTERCLOCKWISE
*
* OUTPUT:
*
* INTERNAL:
*
*****
```

```
CIRCLE: CMP    #1,D1                ;COMPARE DRAWING DIRECTION
        BNE.S  CIR1
        MOVE.W #CRCL1,DO            ;CLOCKWISE
        BRA.S  CIR2
CIR1   MOVE.W #CRCL0,DO            ;COUNTERCLOCKWISE
CIR2   BSR    DCIRCL                ;DRAW
        RTS
```

```
*****
* SUBROUTINE:  ELLIPS
* FUNKTION:   DRAW AN ELLIPSE CLOCKWISE OR COUNTERCLOCKWISE
*
* INPUT:      A2                :PARAMETER ADDRESS
*             D1                :DRAWING DIRECTION
*             :D1 = 1  CLOCKWISE
*             :D1 = 0  COUNTERCLOCKWISE
*
* OUTPUT:
*
* INTERNAL:
*
*****
```

```

ELLIPS: CMP      #1,D1
        BNE.S    ELL1
        MOVE.W   #ELPS1,DO
        BRA.S    ELL2
ELL1    MOVE.W   #ELPS0,DO
ELL2    BSR      DELLIP
        RTS

```

```

*****
* SUBROUTINE:   ABCARC
* FUNKTION:    DRAW AN ABSOLUTE CIRCLE ARC CLOCKWISE OR COUNTERCLOCKWISE
*
* INPUT:       A2          :PARAMETER ADDRESS
*              D1          :DRAWING DIRECTION
*              :D1 = 1   CLOCKWISE
*              :D1 = 0   COUNTERCLOCKWISE
*
* OUTPUT:
*
* INTERNAL:
*
*****

```

```

ABCARC: CMP      #1,D1
        BNE.S    ACARC1
        MOVE.W   #AARC1,DO
        BRA.S    ACARC2
ACARC1  MOVE.W   #AARCO,DO
ACARC2  BSR      RLARC
        RTS

```

```

*****
* SUBROUTINE:   RECARC
* FUNKTION:    DRAW A RELATIVE CIRCLE ARC CLOCKWISE OR COUNTERCLOCKWISE
*
* INPUT:       A2          :PARAMETER ADDRESS
*              D1          :DRAWING DIRECTION
*              :D1 = 1   CLOCKWISE
*              :D1 = 0   COUNTERCLOCKWISE
*
* OUTPUT:
*
* INTERNAL:
*
*****

```

```

RECARC: CMP      #1,D1
        BNE.S    RCARC1
        MOVE.W   #RARCI,DO
        BRA.S    RCARC2
RCARC1  MOVE.W   #RARCO,DO
RCARC2  BSR      RLARC
        RTS

```

```

*****
* SUBROUTINE:   ABEARC
* FUNKTION:    DRAW AN ABSOLUTE ELLIPS ARC CLOCKWISE OR COUNTERCLOCKWISE
*
* INPUT:       A2          :PARAMETER ADDRESS

```

```
*           D1           :DRAWING DIRECTION
*                               :D1 = 1  CLOCKWISE
*                               :D1 = 0  COUNTERCLOCKWISE
```

```
*   OUTPUT:
```

```
*   INTERNAL:
```

```
*****
```

```
ABEARC:  CMP      #1,D1
          BNE.S   AECR1
          MOVE.W  #AEARC1,DO
          BRA.S   AECR2
AECR1    MOVE.W  #AEARCO,DO
AECR2    BSR     ELLARC
          RTS
```

```
*****
```

```
* SUBROUTINE:  REEARC
*   FUNKTION:  DRAW A RELATIVE ELLIPS ARC CLOCKWISE OR COUNTERCLOCKWISE
```

```
*           INPUT:      A2           :PARAMETER ADDRESS
*                               D1     :DRAWING DIRECTION
*                               :D1 = 1  CLOCKWISE
*                               :D1 = 0  COUNTERCLOCKWISE
```

```
*   OUTPUT:
```

```
*   INTERNAL:
```

```
*****
```

```
REEARC:  CMP      #1,D1
          BNE.S   RECR1
          MOVE.W  #REARC1,DO
          BRA.S   RECR2
RECR1    MOVE.W  #REARCO,DO
RECR2    BSR     RLARC
          RTS
```

```
*****
```

```
* SUBROUTINE:  ABFRCT
*   FUNKTION:  ABSOLUT FILLED RECTANGLE
```

```
*           INPUT:      A2           :PARAMETER ADDRESS
```

```
*   OUTPUT:
```

```
*   INTERNAL:
```

```
*****
```

```
ABFRCT:  MOVE.W  #AFRCT,DO
          BSR     FILREC
          RTS
```

```

*****
* SUBROUTINE:   REFRCT
*   FUNKTION:   RELATIVE FILLED RECTANGLE
*
*     INPUT:    A2                               :PARAMETER ADDRESS
*
*     OUTPUT:
*
*   INTERNAL:
*
*****

```

```

REFRCT:  MOVE.W  #RFRCT,DO
        BSR     FILREC
        RTS

```

```

*****
* SUBROUTINE:   PAINT
*   FUNKTION:   FILLED A CLOSED AERA
*
*     INPUT:    D1                               :D1 = 0  EDGECOLOR MODE
*                                           :D1 = 1
*
*     OUTPUT:
*
*   INTERNAL:
*
*****

```

```

PAINT:  CMP     #1,D1
        BNE.S  PAIN1
        MOVE.W #PAINT0,DO
        BRA.S  PAIN2
PAIN1  MOVE.W  #PAINT1,DO
PAIN2  BSR     FILL
        RTS

```

```

*****
* SUBROUTINE:   DRADOT
*   FUNKTION:   DRAW A DOT
*
*     INPUT:
*
*     OUTPUT:
*
*   INTERNAL:
*
*****

```

```

DRADOT:  MOVE.W  #DOT,DO
        BSR     SETDOT
        RTS

```

```

*****
* SUBROUTINE:  PTTERN
*   FUNKTION:  DRAW GRAPHIC PATTERN WHICH STORED IN PATTERN RAM
*
*   INPUT:    A2                      :PARAMETER ADDRESS
*                                     : (SZ: SZY, SZX)
*   OUTPUT:
*
*   INTERNAL:
*
*****

```

```

PTTERN: MOVE.W #PTN,DO
        BSR   PATDRA
        RTS

```

```

*****
* SUBROUTINE:  ABGCPY
*   FUNKTION:  ABSOLUT GRAPHIC COPY
*
*   INPUT:    A2                      :PARAMETER ADDRESS
*
*   OUTPUT:
*
*   INTERNAL:
*
*****

```

```

ABGCPY: MOVE.W #AGCPY,DO
        BSR   GRACPY
        RTS

```

```

*****
* SUBROUTINE:  REGCPY
*   FUNKTION:  RELATIVE GRAPHIC COPY
*
*   INPUT:    A2                      :PARAMETER ADDRESS
*
*   OUTPUT:
*
*   INTERNAL:
*
*****

```

```

REGCPY: MOVE.W #RGCPY,DO
        BSR   GRACPY

```

RTS

\*\*\*\*\*

\*\*\*\*\*

\* VERTIKAL-BLANK ABWARTEN

BLNKTST MOVE #RCR,DO  
BLNK1 BSR RECONR ; RASTER-COUNT-REGISTER LESEN  
CMP #860,D1  
BLS BLNK1  
RTS

\*\*\*\*\*

\* SUBROUTINE COLTAB  
\* FUNKTION LOAD THE COLORTABLES RED GREEN BLUE  
\* IN --  
\* OUT --  
\* INTERNAL A2 A3 DO

COLTAB MOVEM A2/A3/DO, -(A7)  
MOVEA.L #COLTR,A2  
LEA ROT(PC),A3  
MOVE.L #15,DO  
BU1 MOVE.W (A3)+,(A2)+  
DBRA DO,BU1

\*

MOVEA.L #COLTG,A2  
LEA GRUEN(PC),A3  
MOVE.L #15,DO  
BU2 MOVE.W (A3)+,(A2)+  
DBRA DO,BU2

\*

\*

MOVEA.L #COLTB,A2  
LEA BLAU(PC),A3  
MOVE.L #15,DO  
BU3 MOVE.W (A3)+,(A2)+  
DBRA DO,BU3  
MOVEM (A7)+,A2/A3/DO  
RTS

BLAU DC.W 0,0,0,255,0,255,255,255,0,0,127,0,127,127,127,0  
EVEN  
ROT DC.W 0,255,0,0,255,255,0,255,127,0,0,127,127,0,127,255  
EVEN  
GRUEN DC.W 0,0,255,0,255,0,255,255,0,127,0,127,0,127,127,136  
EVEN

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*

\*

PATTER RAM

\*

\*

PRAMO DC.W 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0  
EVEN

\*

\*\*\*\*\*

\*

VERSCHIEDENE PATTERN

\*

SCHRAFFUR

PATTER1 DC.W #8080,#4040,#2020,#1010,#0808,#0404,#0202,#0101

DC.W \$8080,\$4040,\$2020,\$1010,\$0808,\$0404,\$0202,\$0101

\* SCHRAFFUR1

PATTER2 DC.W \$8888,\$8888,\$8888,\$BBBB,\$BBBB,\$8888,\$8888,\$8888  
DC.W \$8888,\$8888,\$8888,\$BBBB,\$BBBB,\$8888,\$8888,\$8888

\* MOSAIK

PATTER3 DC.W \$0000,\$7FFE,\$4002,\$4002,\$4FF2,\$4812,\$4812,\$4992  
DC.W \$4992,\$4812,\$4812,\$4FF2,\$4002,\$4002,\$7FFE,\$0000

\* KLEEBLAT

PATTER4 DC.W \$C000,\$7E78,\$2188,\$6186,\$4182,\$4182,\$4182,\$3FFC  
DC.W \$3FFC,\$4182,\$4182,\$4182,\$6186,\$1188,\$1E78,\$0000

\* SYMBOL1

PATTER5 DC.W \$B005,\$500B,\$2424,\$13C8,\$0810,\$0420,\$0240,\$0420  
DC.W \$0810,\$1188,\$1248,\$1188,\$0810,\$6426,\$1242,\$0180

\* SYMBOL2

PATTER6 DC.W \$0000,\$0000,\$0000,\$0780,\$1840,\$1720,\$2110,\$472C  
DC.W \$4952,\$C126,\$7EEE,\$30C4,\$38E7,\$38E3,\$0000,\$0000

\* COURSOR

PATTER7 DC.W \$0000,\$1FEE,\$0FFE,\$07FE,\$03FE,\$01FE,\$01FE,\$03FE  
DC.W \$07FE,\$0F0E,\$1F0E,\$3E06,\$7C02,\$7800,\$3000,\$0000  
EVEN  
END



# LIBRARY FOR THE AGC-1

FORCE COMPUTERS Inc./GmbH  
All Rights Reserved

This document shall not be duplicated, nor its contents used  
for any purpose, unless express permission has been granted.

Copyright by FORCE Computers®

```
*****
*
*
* The information in this document is subject to change
* without notice and should not be construed as a commitment
* by FORCE Computers.
*
* Neither FORCE Computers nor the authors assume any responsi-
* bility for the use or reliability of this document or the
* described software.
*
*           Copyright (C) 1986, FORCE Computers
*
* General permission to copy or modify, but not for profit, is
* hereby granted, provided that the above copyright notice is
* included and reference made to the fact that reproduction
* privileges were granted by FORCE Computers.
*
*****
```

```
-----
|
| This library is used as driver for the AGC.
| The implemented commands can be linked to a user program
| as follows:
|
|           F77 EXAMPLE
|           F77L EXAMPLE,AGCLIB/L,XLIB/L
|
| File name:      AGCLIB:DOC
| Programmed by:  K. Nguyen-Thanh
| Date:          09/18/86
|
|-----
```

```
-----
|
| AGC Address register: $C3C000
| AGC Data register: $C3C002
|
|-----
```

```
*****
*
*           INITIALISATION COMMANDS
*
*****
```

```
-----|
| Subroutine:  RESET1 |
|           |         |
| Function:    reset AGC |
|           |         |
|-----|
```

```
    call RESET1
```

```
-----|
| Subroutine:  CLEAR1 |
|           |         |
| Function:    clear image memory |
|           |         |
| Ram Address : C00000 - C40000 |
|           and : C40000 - E3FFFE |
|-----|
```

```
    call CLEAR1
```



```

*****
*   The following 6 routines can be used to set the graphic   *
*   parameters:                                             *
*   - source scan direction      : call SETSSD(ssd)         *
*   - destination scan direction : call SETDSD(dsd)         *
*   - pattern RAM address        : call SETPRA(pra)         *
*   - graphic area               : call SETAER(aer)         *
*   - modify mode                : call SETMFY(mm)         *
*   - operation mode             : call SETOPM(op)         *
*
*   The default values of ssd,dsd,pra,aer,mm,op are 0.     *
*
*****

```

```

-----
| Subroutine: SETSSD                                         |
| Function:   set source SCAN direction in SDSD.            |
| Note:      This function set the default for the bit 11 for all |
|            the commands codes using the source scan direction. |
|            ssd = 0 or 1                                     |
|-----

```

```

integer ssd

    call SETSSD(ssd)

```

Example:

```

    call SETSSD(0)

```

```

-----
| Subroutine:  SETPRA                                        |
| Function:    set pattern RAM address.                      |
| Note:       This function sets the default for Bit (0-3) for all |
|            the commands codes using the pattern RAM.         |
|            A pattern contains 16 words (16x16 bytes).        |
|            The pattern RAM address (pra) ranges from 0 to 15. |
|-----

```

```

integer pra

    call SETPRA(pra)

```

Example:

```

    call SETPRA(1)

```

-----  
Subroutine: SETAER

Function: Set area-mode.

Note: This function sets the default for Bits (5-7) for all  
the commands codes using the aera.  
The aeras (aer) range from 0-7.

integer aer

call SETAER(aer)

Example:

call SETAER(1)

-----  
Subroutine: SETOPM

Function: Set operation mode.

Note: Bit (0-2) in the command code, which depended to the  
operation mode, will be set to opm.  
The operation mode ranges from 0 - 7

integer opm

call SETOPM(opm)

Example:

call SETOPM(1)

-----  
Subroutine: SETMFY

Function: Set modify mode.

Note: This function sets the default for Bits (0-1) for all  
the commands codes using the mofify mode.  
mm ranges from 0 to 3.  
mm = 0: replace frame buffer data with command parameter data  
mm = 1: OR frame buffer data with command parameter data.  
mm = 2: AND frame buffer data with command parameter data.  
mm = 3: EOR frame buffer data with command parameter data.  
and rewrite to frame buffer.

integer mm

call SETMFY(mm)

Example:

call SETMFY(1)

```
*****
*
*           THE FOLLOWING ARE GRAPHIC COMMANDS           *
*
*****
```

```
-----|
| Subroutine:  ABMOVE                                |
| Function:    Absolute move to the position (ay,ax) |
| Note:       --                                    |
|-----|
```

```
integer ax,ay
```

```
call ABMOVE(ay,ax)
```

```
Example:
```

```
call ABMOVE(100,200)
```

```
-----|
| Subroutine  REMOVE                                |
| Function:    Relative move to the position (ry,rx). |
| Note:       --                                    |
|-----|
```

```
integer ry,rx
```

```
call REMOVE(ry,rx)
```

```
Example:
```

```
call REMOVE(100,200)
```

-----  
Subroutine SCOLO

Function The color register 0 (CLO) is set to 'col'.

Note: This register defines the drawing color in addition to the color register (CL1).  
If the scanned pattern RAM is equal to '0' the color is in CLO.  
If the scanned pattern RAM is equal to '1' the color is in CL1.(See also SCOL1).

-----  
integer col

call SCOLO(col)

Example: set color white

call SCOL1(7)

-----  
Subroutine SCOL1

Function: The color register 1 (CL1) is set to 'col'.

Note: This register defines the drawing color in addition to the color register 0 (CLO).  
If the scanned pattern RAM is equal to '1' the color is in CL1.  
If the scanned pattern RAM is equal to '0' the color is in CLO.(see also SCOLO).

-----  
integer col

call SCOL1(col)

Example: set color white

call SCOL1(7)

-----  
Subroutine: SECOL

Function: Set edge color register.

Note: The paint command can change the edg color.

-----  
integer edcol

call SECOL(edcol)

Example:

call SECOL(5)



```
-----  
Subroutine:  ABRECT
```

```
Function:    Draw an absolute rectangle.  
             The rectangle is defined by the current  
             point (CP) and the absolute coordinate x,y.
```

```
Note:       --  
-----
```

```
integer x,y
```

```
    call ABRECT(y,x)
```

```
Example: draw a rectangle defined by absolute coordinate (100,200)  
         and (150,300).
```

```
    call ABMOVE(100,200)  
    call ABRECT(150,300)
```

```
-----  
Subroutine:  RERECT
```

```
Function:    Draw a relative rectangle.  
             The rectangle is defined by CP and the  
             relative coordinate dx,dy.
```

```
Note:       --  
-----
```

```
integer dx,xy
```

```
    call RERECT(dy,dx)
```

```
Example: If the current point(CP) is at (100,200), this  
         example will draw the rectangle between (100,200)  
         and (150,300).
```

```
    call RERECT(50,100)
```

```
-----  
Subroutine:  ABPOLL
```

```
Function:    Draw an absolute polyline.  
             Each line from polyline connects the absolute  
             coordinate with the previous coordinate.
```

```
Note:       abuf is an array of coordinates (x1,y1...xn,yn)  
             n    is the number of the coordinates +1.  
-----
```

```
integer abuf(512),n
```

```
    call ABPOLL(abuf,n)
```

```
Example: draw a polyline consists of 2 lines.
```

```
    abuf(1)=100  
    abuf(2)=100  
    abuf(3)=200  
    abuf(4)=-200  
    call ABMOVE(0,0)  
    call ABPOLL(abuf,5)
```

Subroutine: REPOLL

Function: Draw a relative polyline.  
Each line from polyline connects the relative coordinate with the previous coordinate.

Note: abuf is an array of coordinates (x1,y1...xn,yn).  
n is number of the coordinates +1.

integer abuf(512),n

call REPOLL(abuf,n)

Example: draw a polyline consists of 2 lines.

```
abuf(1)=100
abuf(2)=100
abuf(3)=100
abuf(4)=-300
call ABMOVE(0,0)
call REPOLL(abuf,5)
```

Subroutine: ABPOLG

Function: Draw a absolute polygon.  
The same as ABPOLL but an additioned line connects the last coordinate with the first one.  
And a line connects the last - with the first coordinate

Note: abuf is an array of coordinate (x1,y1...xn,yn)  
n is number of the coordinates +1

integer abuf(512),n

call ABPOLG(abuf,n)

Example: draw a rectangle with polygon.

```
abuf(1)=100
abuf(2)=0
abuf(3)=100
abuf(4)=100
abuf(5)=0
abuf(6)=100
call ABMOVE(0,0)
call ABPOLL(abuf,7)
```

Subroutine: REPOLG

Function: Draw a relative polygon.  
The same as REPOLL but an additioned line connects  
the last coordinate with the first one.

Note: abuf is an array of coordinates (x1,y1...xn,yn).  
n is number of the coordinates +1.

integer abuf(512),n

call REPOLG(abuf,n)

Example: Draw a rectangle with polygon. The same rectangle  
as with ABPOLG.

```
abuf(1)=100
abuf(2)=0
abuf(3)=0
abuf(4)=100
abuf(5)=-100
abuf(6)=0
call ABMOVE(0,0)
call REPOLG(abuf,7)
```

Subroutine: ABCARC

Function: Draw a circle arc from current point to an  
absolute end point.

Note: xe,ye: endpoint.  
xc,yc: center.  
dir: drawing direction.

integer xe,ye,xc,yc,dir

call ABCARC(ye,xe,yc,xc,dir)

Example: draw a circle arc as half circle.

```
call ABMOVE(150,150)
call ABCARC(200,150,100,150,1)
```

Subroutine: RECARC

Function: Draw a relative circle arc from current point to an absolute end point.

Note:        xe,ye : relative endpoint to CP.  
              xc,yc : relative center to CP.  
              dir:    drawing direction.

integer ye,xe,yc,xc,dir

call RECARC(ye,xe,yc,xc,dir)

Example: draw a relative circle arc as half circle

call RECARC(100,0,50,0,1)

Subroutine: ABEARC

Function: Draw an absolute ellipsis arc from current point to an absolute end point.

Note:        xe,ye : absolute endpoint.  
              xc,yc : absolute center.  
              a,b    : calculated from  $a / b = dX*dX / dY*dY$   
                      whereby dX = ellipsis axis in x  
                              dY = ellipsis axis in y  
              dir    : drawing direction.

integer a,b,xend,yend,xcenter,ycenter,dir

call ABEARC(yend,xend,ycenter,xcenter,b,a,dir)

Example:

call ABMOVE(100,100)  
call ABEARC(200,200,250,320,140,160,1)

Subroutine: REEARC

Function: Draw a relative ellipsis arc with direction.

Note:        xe,ye : relative endpoint  
              xc,yc : relative center  
              a,b    : calculated from  $a / b = dX*dX / dY*dY$   
                      whereby dX = ellipsis axis in x  
                              dY = ellipsis axis in y

integer a,b,xend,yend,xcenter,ycenter,dir

call REEARC(yend,xend,ycenter,xcenter,b,a,dir)

Example:

call REEARC(100,100,150,220,40,60,1)

Subroutine: ABGCPY

Function: copy a absolute rectangle with mirror to the destination defined by CP.  
The graphic copy is dependant to the source- and destination scan direction (ssd,dsd).  
dx,dy: define the angles of the rectangle to be copied.

Note: The ssd and dsd are preceded if the routines SETSSD and SETDSD were called.  
If dsd=0 and ssd=0 then no mirror.

integer dy,dx,ay,ax

call ABGCPY(dy,dx,ay,ax)

Example: copy the rectangle beginning at (200,200)  
with dx=40 and dy=40 to position (0,0).

call ABMOVE(0,0)  
call ABGCPY(200,200,40,40)

Subroutine: REGCPY

Function: Copy a relative rectangle with mirror to the destination defined by CP.  
The graphic copy is dependant to the source- and destination scan direction (ssd,dsd).  
dx,dy: define the angles of the rectangle to be copied.

Note: -- The ssd and dsd are preceded if the routines SETSSD and SETDSD were called.  
If dsd=0 and ssd=0 then no mirror.

integer dx,dy,rx,ry

call REGCPY(dy,dx,ry,rx)

Example: copy a relative rectangle at (CPx+100,CPy+100)  
to the position (0,0) with dx=40, dy=40

call ABMOVE(0,0)  
call REGCPY(100,100,40,40)

Subroutine: CIRCLE

Function: Draw a circle with radius and direction.

Note: The radius is spezified in units of pixels.

integer radius,dir

call CIRCLE(radius,dir)

Example:

call CIRCLE(50,1)

Subroutine: ELLIPS

Function: Draw a ellips with a,b ,dx and direction.

Note: The paramter a,b,dx are spezified in units of pixels

integer a,b,dx,dir

call ELLIPS(a,b,dx,1)

Example:

call ABMOVE(300,300)

call ELLIPS(30,10,100,1)

Subroutine: PAINT

Function: Paint the enclosed aera with the edg color  
define by bit (E=0 and E=1).

Note: The edgcolor must be set different to the  
background color and the CLO, CL1 (drawing color).  
Edge Color Definition:

	E = 1	E = 0
Dot is regarded as already painted and as edge color	COL = EDG or COL = CLO or COL = CL1 (satisfied if one of the condition is met)	COL <> EDG or COL = CLO or COL = CL1 (satisfied if one of the condition is met)
Dot is redarded as unpainted	COL <> EDG and COL <> CLO and COL <> CL1 (satisfied if ALL the conditions are met)	COL = EDG and COL <> CLO and COL <> CL1 (satisfied ALL the conditions are met)

integer edg

call PAINT(edg)

Example: fill a circle

call COLNEU(7)

call EDGCOL(1)

call ABMOVE(100,100)

call CIRCLE(50,1)

call PAINT(1)

Subroutine: WPARAM

Function: Write pattern RAM

Note: A pattern contains 16 words (16x16 bytes). The pattern RAM address is created by calling the routine SETPRA. Default address is 0.  
Bevor write pattern, the pattern RAM control register (set, start- and endpointer in register \$805,\$806,\$807 must be written).  
nr = Number of words (0 .. 15)  
patrn = pattern , 8 x long word = 16

integer nr,i  
integer patrn(8)  
character\*9 buffer(8)

call WPARAM(patrn,nr)

Example: write pattern 'R' to pattern RAM

```
DATA buffer /'$F01C781C','$3C1C1E1C','$0F1C079C',  
1 '$03DC3FFC','$7FFCFFFC','$E01CE01C','$E01CFFFC',  
1 '$7FFC1FFC'/
```

```
do 6 i=1,8  
CALL XCDB(buffer(i),patrn(i))  
6 continue
```

Subroutine: RPARAM

Function: Read pattern RAM in to buffer

Note: A pattern contains 16 words (16x16 bytes). The pattern RAM address is created by calling the routine SETPRA. Default address is 0.  
nr = Number of word (0 .. 15)  
patrn = pattern , 8 x long word = 16

integer nr,i  
integer patrn(8)  
character\*9 buffer(8)

call RPARAM(patrn,nr)

Example: read 16 words from pattern RAM

CALL RPARAM(patrn,16)

Subroutine PTTERN

Function: Draw a graphic pattern which stored in pattern RAM

Note: The pattern is created by calling the routine WPARAM.  
No parameter is needed.

call PTTERN

```
|-----|
| Subroutine DRADOT
| Function: Draw a dot
|
| Note: -
|-----|
```

```
call DRADOT
```

```
|-----|
| Subroutine ABFRCT
|
| Function: Fill the rectangle with the predefined color.
|
| Note: -
|-----|
```

```
integer ay,ax
```

```
call ABFRCT(ay,ax)
```

```
Example: fill the rectangle defined by CP(100,100)
and (150,150)
```

```
call ABMOVE(100,100)
```

```
call REFRCT(150,150)
```

```
|-----|
| Subroutine REFRCT
|
| Function: Fill rectangle with the predefined color.
|
| Note: --
|-----|
```

```
integer ry,rx
```

```
call REFRCT(ry,rx)
```

```
Example: fill the rectangle defined by CP(100,100)
and (150,150)
```

```
call ABMOVE(100,100)
```

```
call REFRCT(50,50)
```



Software-cursor as crosshair

Function: Blends a crosshair at CP.

Note: The cursor step, size and color can be changed.  
The cursor size must be multiple of 2.  
stepx,stepy are the step, which the cursor  
has to move. If stepx=0 and stepy=0 the cursor  
d'ont move.

integer dx,dy,step,size,col

call CURSOR(stepx,stepy,size,col)

Example:

call CURSOR(0,0,8,4)

Subroutine: CPXY

Function: Read current point

Note: --

integer apx,apy

call CPXY(apy,apx)

Example:

call CPXY(apy,apx)  
write(9,\*) 'absolute coordinate apy, apx =',apy,apx

Subroutine WRITPA

Function: Write paramter register with value

Note: --

integer val,reg

call WRITPA(val,reg)

Example: write pattern pointer register Pr05,Pr06,Pr07 before  
write pattern RAM.

call WRITPA(0,2053)  
call WRITPA(2,2054)  
call WRITPA(2,2055)

-----  
Subroutine READPA

Function: Read paramter register.

Note: before the subroutine is called  
the parameter must be written

-----  
integer val

call READPA(val)

Example:

call READPA(val)  
write(9,\*) 'Parameter = ',val

END

# USER NOTES

FORCE COMPUTERS Inc./GmbH  
All Rights Reserved

This document shall not be duplicated, nor its contents used  
for any purpose, unless express permission has been granted.

Copyright by FORCE Computers®

# MODIFICATIONS

FORCE COMPUTERS Inc./GmbH  
All Rights Reserved

This document shall not be duplicated, nor its contents used  
for any purpose, unless express permission has been granted.

Copyright by FORCE Computers®