HITACHI MICROCOMPUTER SYSTEM

HD63484 ACRTC

Advanced CRT Controller

USER'S MANUAL

HITACHI

680-1-31A

First Edition: October 1984 680-1-31
Second Edition: July 1985 680-1-31A

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

1. This manual may, wholly or partially, be subject to change without notice.
2. All rights reserved: No one is permitted to reproduce or duplicate, in any form, the whole or a part of this manual without Hitachi's permission.
3. Hitachi will not be responsible for any damage to the user that may result from accidents or any other reasons during operation of his unit according to this manual.
4. This manual neither ensures the enforcement of any industrial properties on other rights, nor sanctions the enforcement right thereof.

Abbreviations

Name	Description	Name	Description
AARC	Absolute Arc	CER	Command Error
ABT	Abort	CHR	Character
ACM	Access Mode	CLR	Clear
ACP	Access Priority	CL0	Color 0 Register
ADR	Area Definition Register	CL1	Color 1 Register
AEARC	Absolute Ellipse Arc	CM	Cursor Mode
AFRCT	Absolute Filled Rectangle	CCMP	Color Comparison Register
AGCPY	Absolute Graphic Copy	COFF1	Cursor Off 1
ALINE	Absolute Line	COFF2	Cursor Off 2
AMOVE	Absolute Move	CON1	Cursor On 1
APLG	Absolute Polygon	CON2	Cursor On 2
APLL	Absolute Poly Line	CP	Current Pointer
AR	Address Register	CPY	Copy
ARCT	Absolute Rectangle	CRCL	Circle
ARD	Area Detect	CRE	Command Error Interrupt Enable
ARE	Area Detect Interrupt Enable	CSK	Cursor Display Skew
AREA	Area Detect Mode	CXE	Cursor X End
ATC	Attribute Code	CXS	Cursor X Start
ATR	Attribute Control	CYE	Cursor Y End
BCA	Block Cursor Address	CYS	Cursor Y Start
BCA1	Block Cursor Address 1	DCR	Display Control Register
BCA2	Block Cursor Address 2	DDM	Data DMA Mode
BCER1	Block Cursor End Raster 1	DMOD	DMA Modify
BCER2	Block Cursor End Raster 2	DN	Display Number
BCR	Blink Control Register	DOT	Dot
BCSR1	Block Cursor Start Raster 1	DP	Drawing Pointer
BCSR2	Block Cursor Start Raster 2	DPAH	Drawing Pointer Address High
BCUR1	Block Cursor Register 1	DPAL	Drawing Pointer Address Low
BCUR2	Block Cursor Register 2	DPD	Drawing Pointer Dot
BCW1	Block Cursor Width 1	DPH	Drawing Pointer High Address Word
BCW2	Block Cursor Width 2	DPL	Drawing Pointer Low Address Word
BLINK1	Blink 1	DRC	DMA Request Control
BLINK2	Blink 2	DRD	DMA Read
BOFF1	Blink Off 1	DSD	Destination Scan Direction
BOFF2	Blink Off 2	DSK	DISP Skew
BON1	Blink On 1	DSP	DISP Signal Control
BON2	Blink On 2	DWT	DMA Write
CCR	Command Control Register	EDG	Edge Color Register
CDM	Command DMA Mode	ELPS	Ellipse
CDR	Cursor Definition Register	FE	FIFO Entry
CED	Command End	FRA	First Raster Address
CEE	Command End Interrupt Enable	FRA0	First Raster Address 0

Abbreviations

Name	Description
FRA1	First Raster Address 1
FRA2	First Raster Address 2
FRA3	First Raster Address 3
GAI	Graphic Address Increment Mode
GBM	Graphic Bit Mode
GCR	Graphic Cursor Register
HC	Horizontal Cycle
HDR	Horizontal Display Register
HDS	Horizontal Display Start
HDW	Horizontal Display Width
HSD	Horizontal Scroll Dot
HSR	Horizontal Sync Register
HSW	Horizontal Sync Width
HWR	Horizontal Window Display Register
HWS	Horizontal Window Start
HWW	Horizontal Window Width
HZ	Horizontal Zoom
HZF	Horizontal Zoom Factor
IE	Interrupt Enable
LPAH	Light Pen Address High
LPAL	Light Pen Address Low
LPAR	Light Pen Address Register
LPD	Light Pen Strobe Detect
LPE	Light Pen Strobe Interrupt Enable
LRA	Last Raster Address
LRA0	Last Raster Address 0
LRA1	Last Raster Address 1
LRA2	Last Raster Address 2
LRA3	Last Raster Address 3
M/S	Master/Slave
MM	Modify Mode
MOD	Modify
MASK	Mask Register
MW	Memory Width
MW0	Memory Width 0
MW1	Memory Width 1
MW2	Memory Width 2
MW3	Memory Width 3
MWR	Memory Width Register
MWR0	Memory Width Register 0
MWR1	Memory Width Register 1
MWR2	Memory Width Register 2
MWR3	Memory Width Register 3
OMR	Operation Mode Register
OPM	Operation Mode
ORG	Origin
PAINT	Paint
PE	Pattern End
PEX	Pattern End X
PEY	Pattern End Y
PP	Pattern Pointer
PPX	Pattern Pointer X
PPY	Pattern Pointer Y
PRA	Pattern RAM Address
PRC	Pattern RAM Control Register
PS	Pattern Start
PSE	Pause
PSX	Pattern Start X
PSY	Pattern Start Y
PTN	Pattern
PZCX	Pattern Zoom Count X
PZCY	Pattern Zoom Count Y
PZX	Pattern Zoom X
PZY	Pattern Zoom Y
RAM	RAM Mode
RARC	Relative Arc
RAR	Raster Address Register
RAR0	Raster Address Register 0
RAR1	Raster Address Register 1
RAR2	Raster Address Register 2
RAR3	Raster Address Register 3
RC	Raster Count
RCR	Raster Count Register
RD	Read
REARC	Relative Ellipse Arc
RFE	Read FIFO Full Interrupt Enable
RFF	Read FIFO Full
RFR	Read FIFO Ready
RFRCT	Relative Filled Rectangle
RGCPY	Relative Graphic Copy
RLINE	Relative Line
RMOVE	Relative Move
RN	Register Number
RPLG	Relative Polygon

Abbreviations

Name	Description
RPLL	Relative Poly Line
RPR	Read Parameter Register
RPTN	Read Pattern RAM
RRCT	Relative Rectangle
RRE	Read FIFO Ready Interrupt Enable
RSM	Raster Scan Mode
RWP	Read/Write Pointer
RWPH	Read/Write Pointer High
RWPL	Read/Write Pointer Low
S	Source Scan Direction
SAH	Start Address High
SAL	Start Address Low
SAR	Start Address Register
SAR0	Start Address Register 0
SAR1	Start Address Register 1
SAR2	Start Address Register 2
SAR3	Start Address Register 3
SCLR	Selective Clear
SCPY	Selective Copy
SD	Scan Direction
SDA	Start Dot Address
SE	Split Screen Enable
SE0	Split Screen 0 Enable
SE1	Split Screen 1 Enable
SE2	Split Screen 2 Enable
SE3	Split Screen 3 Enable
SL	Slant
SPL	Split
SP0	Split Screen 0 Width
SP1	Split Screen 1 Width
SP2	Split Screen 2 Width
SR	Status Register
SRA	Start Raster Address
SSW	Split Screen Width
STR	Start
VC	Vertical Cycle
VDR	Vertical Display Register
VDS	Vertical Display Start
VSR	Vertical Sync Register
VSW	Vertical Sync Width
VWR	Vertical Window Register
VWS	Vertical Window Start
VWW	Vertical Window Width
VZF	Vertical Zoom Factor
WEE	Write FIFO Empty Interrupt Enable
WFE	Write FIFO Empty
WFR	Write FIFO Ready
WPR	Write Parameter Register
WPTN	Write Pattern RAM
WRE	Write FIFO Ready Interrupt Enable
WSS	Window Smooth Scroll
WT	Write
XMAX	X Maximum
XMIN	X Minimum
YMAX	Y Maximum
YMIN	Y Minimum
ZFR	Zoom Factor Register

TABLE OF CONTENTS

HD63484 ACRTC
(Advanced CRT Controller)

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 (CRTCs) 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 Postition 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 limited 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.
  - Progreammable Cursor and Display Timing Skew.
  - Eight User Defineable Video Attributes.
  - Light Pen Detection.
* VLSI CMOS Process

1. ACRTC INTRODUCTION

1.1 Applications

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 OS or applcation software. At this layer, a number of popular standards have emerged including GKS, Core, NAPLP, GSX and others.

Figure 1.1 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.

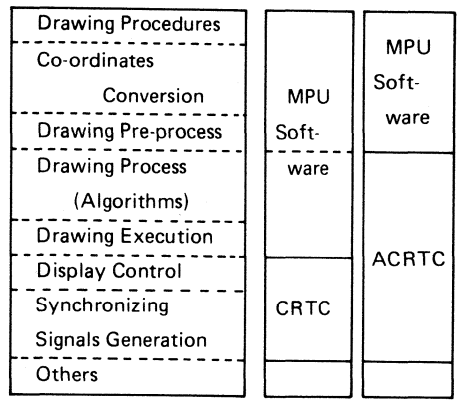


Figure 1.1 ACRTC vs. CRTC

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 1.2 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.

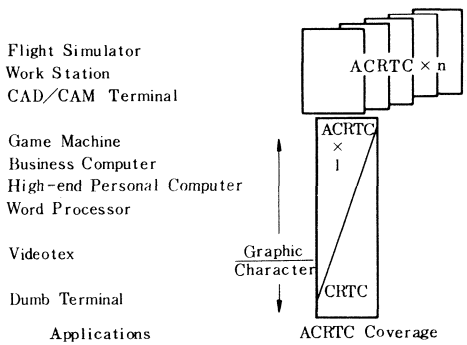


Figure 1.2 Application Spectrum

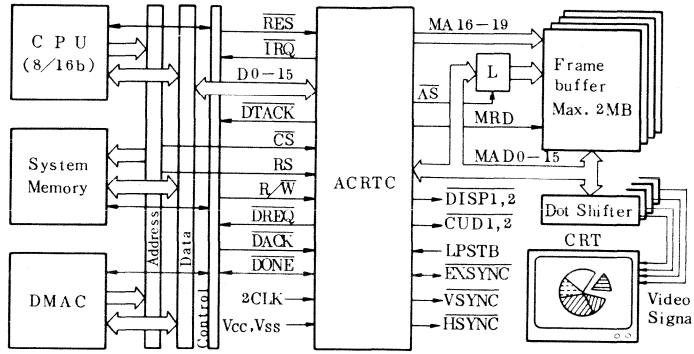


Figure 1.3 System Configuration

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/from 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.

1.3 Block Diagram

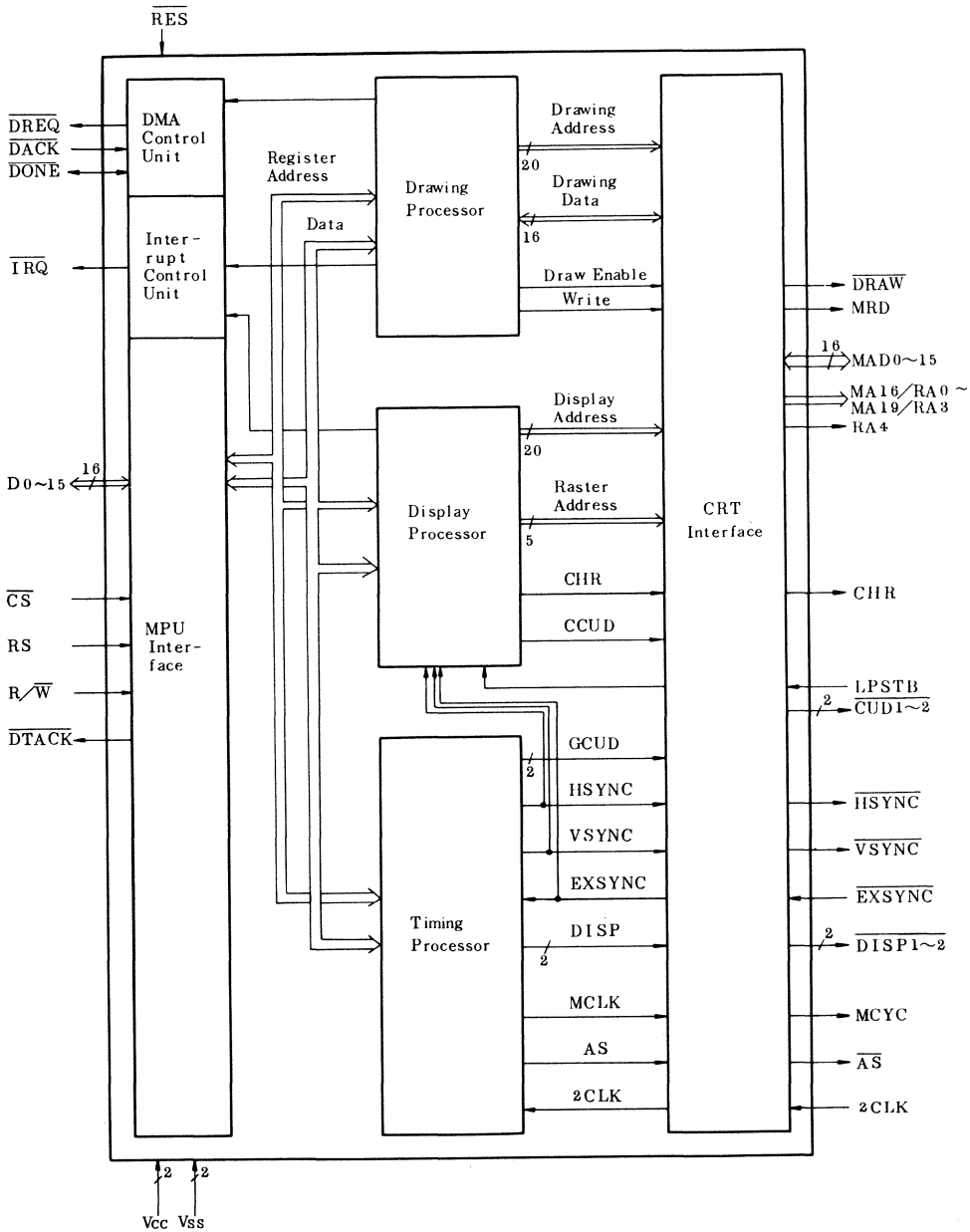


Figure 1.4 Block Diagram

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 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 control.
- 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.

1.4 Signal Description

Following is a brief description of the ACRTC pin functions organized as MPU Interface, DMAC Interface, CRT Interface and Power Supply. The detailed signal description is provided in section 4.

MPU Interface

$\overline{RES}$ – Input
Hardware reset input to the ACRTC.

D0 – D15 – Input/Output
The bidirectional data bus for communication with the host MPU or DMAC. In 8 bit data bus mode, DO-D7 are used.

R/$\overline{W}$ – Input
Controls the direction of host $\longleftrightarrow$ 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.

MAD0-15 – Input/Output

Multiplexed frame buffer address/data bus.

$\overline{AS}$ – Output

Address strobe for demultiplexing the frame buffer address/data bus (MAD0-15).

MA16/RA0-MA19/RA3 – Output

The high order address bits for graphic screens and the raster address outputs for character screens.

RA4 – Output

Provides the high order raster address bit (up to 32 rasters) for character screens.

CHR – Output

Indicates whether a graphic or character screen is being accessed.

MCYC – Output

Frame buffer memory access timing – one half the frequency of 2CLK.

MRD – Output

Frame Buffer data bus direction control.

$\overline{DRAW}$ – Output

Differentiates between drawing cycles and CRT display refresh cycles.

$\overline{DISP1}$, $\overline{DISP2}$ – Output

Programmable display enable timing used to selectively enable, disable and blank logical screens.

$\overline{CUD1}$, $\overline{CUD2}$ – Output

Provides cursor timing determined by ACRTC programmed parameters such as cursor definition, cursor mode, cursor address, etc.

$\overline{\text{VSYNC}}$ – Output
CRT device vertical synchronization pulse.

$\overline{\text{HSYNC}}$ – Output
CRT device horizontal synchronization pulse.

$\overline{\text{EXSYNC}}$ – Input/Output
For synchronization between multiple ACRTCs and other video signal generating devices.

LPSTB – Input
Connection to an external light pen.

1.5 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 1.5 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.

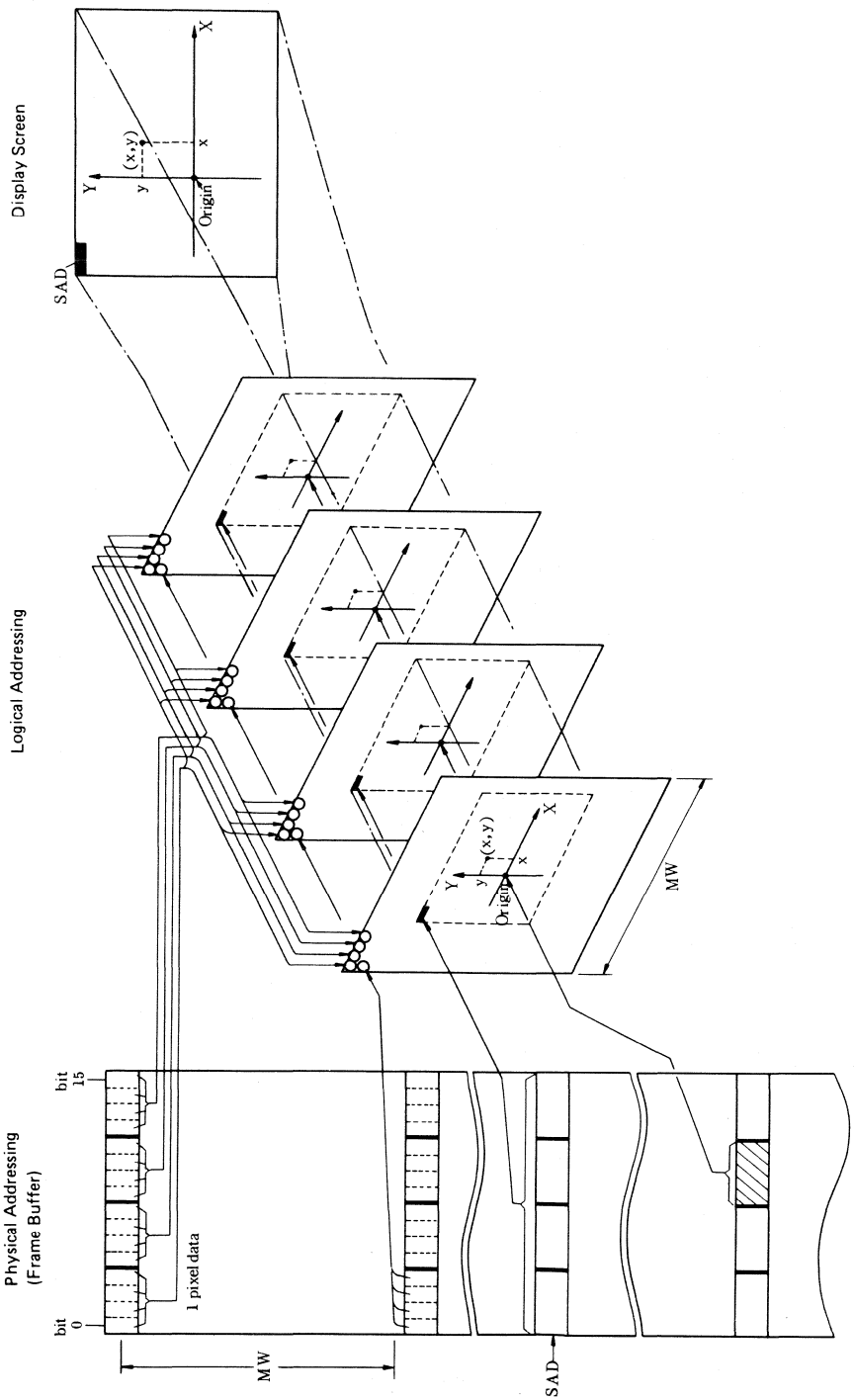


Figure 1.5 Logical/Physical Addressing

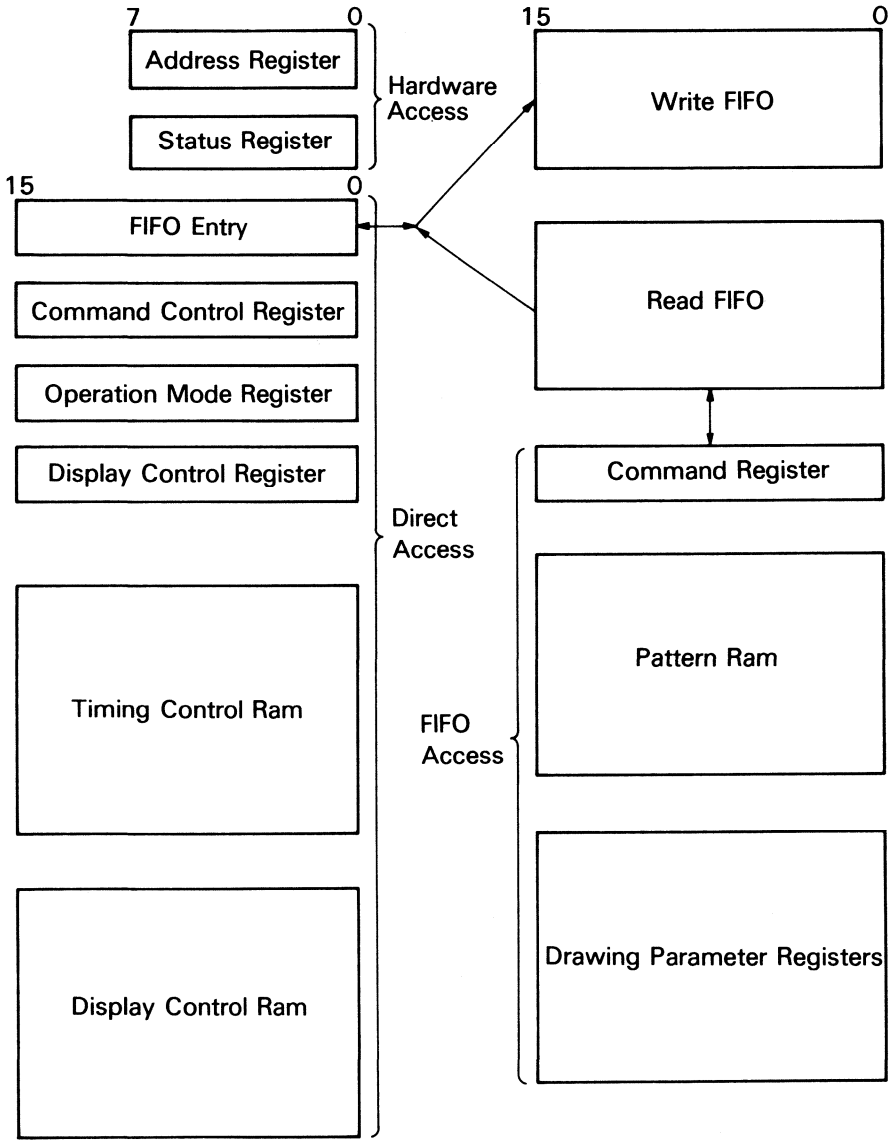


Figure 1.6 Accessible Registers

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. $\overline{\text{HSYNC}}$, $\overline{\text{VSYNC}}$), 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.

Type	Mnemonic	Function
Register Access Commands	ORG	Set Origin Point
	RPR,WPR	Read/Write Parameter Registers
	RPTN,WPTN	Read/Write Pattern RAM
Data Transfer Commands	DRD,DWT,DMOD	DMA Read/Write/Modify
	RD,WT,MOD	Read/Write/Modify
	CLR,SCLR	Clear
	CPY,SCPY	Copy
Graphic Drawing Commands	AMOVE,RMOVE	Move
	ALINE,RLINE	Line
	ARCT,RRCT	Rectangle
	APLL,RPLL	Polyline
	APLG,RPLG	Polygon
	CRCL	Circle
	ELPS	Ellipse
	AARC,RARC	Arc
	AEARC,REARC	Ellipse Arc
	AFRCT,RFRCT	Filled Rectangle
	PAINT	Paint
	DOT	Dot
	PTN	Pattern
	AGCPY,RGCPY	Graphic Copy

Figure 1.7 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.

1.8 Graphic Drawing

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 excample, 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.

2. SYSTEM INTERFACE

2.1 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.

2.2 CRT Interface

2.2.1 Frame Buffer Access

2.2.1.1 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)

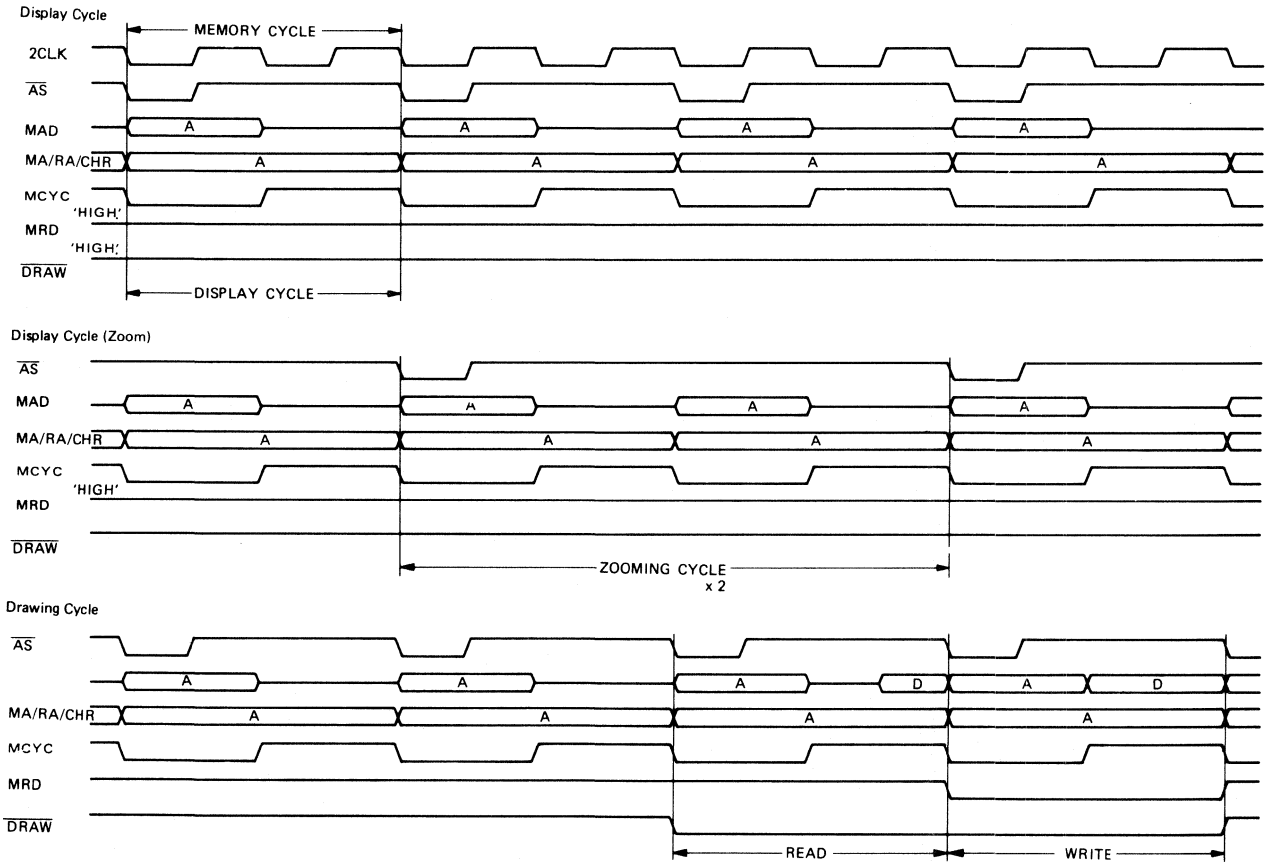


Figure 2.1(a) Access Mode Timing

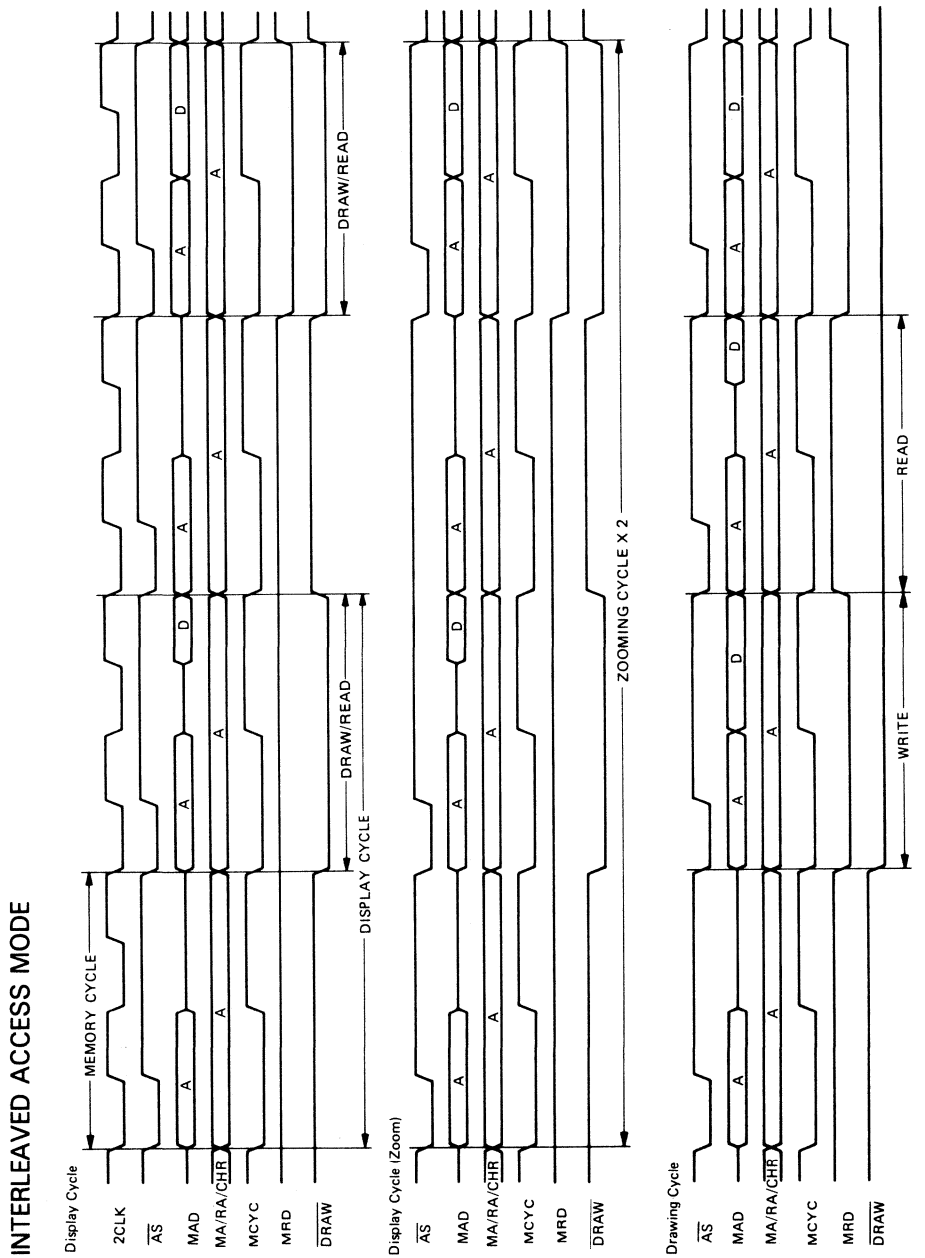


Figure 2.1(b) Access Mode Timing

SUPERIMPOSED ACCESS MODE

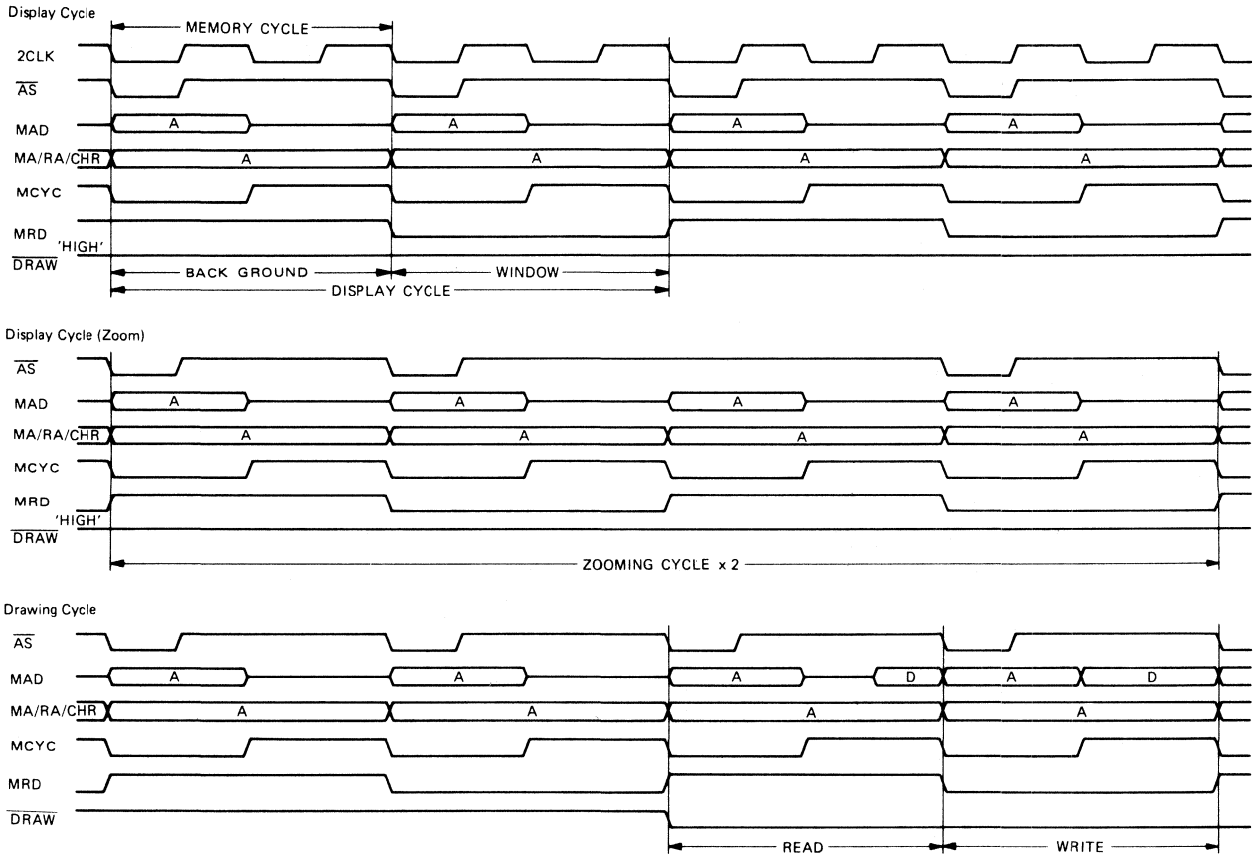


Figure 2.1(c) Access Mode Timing

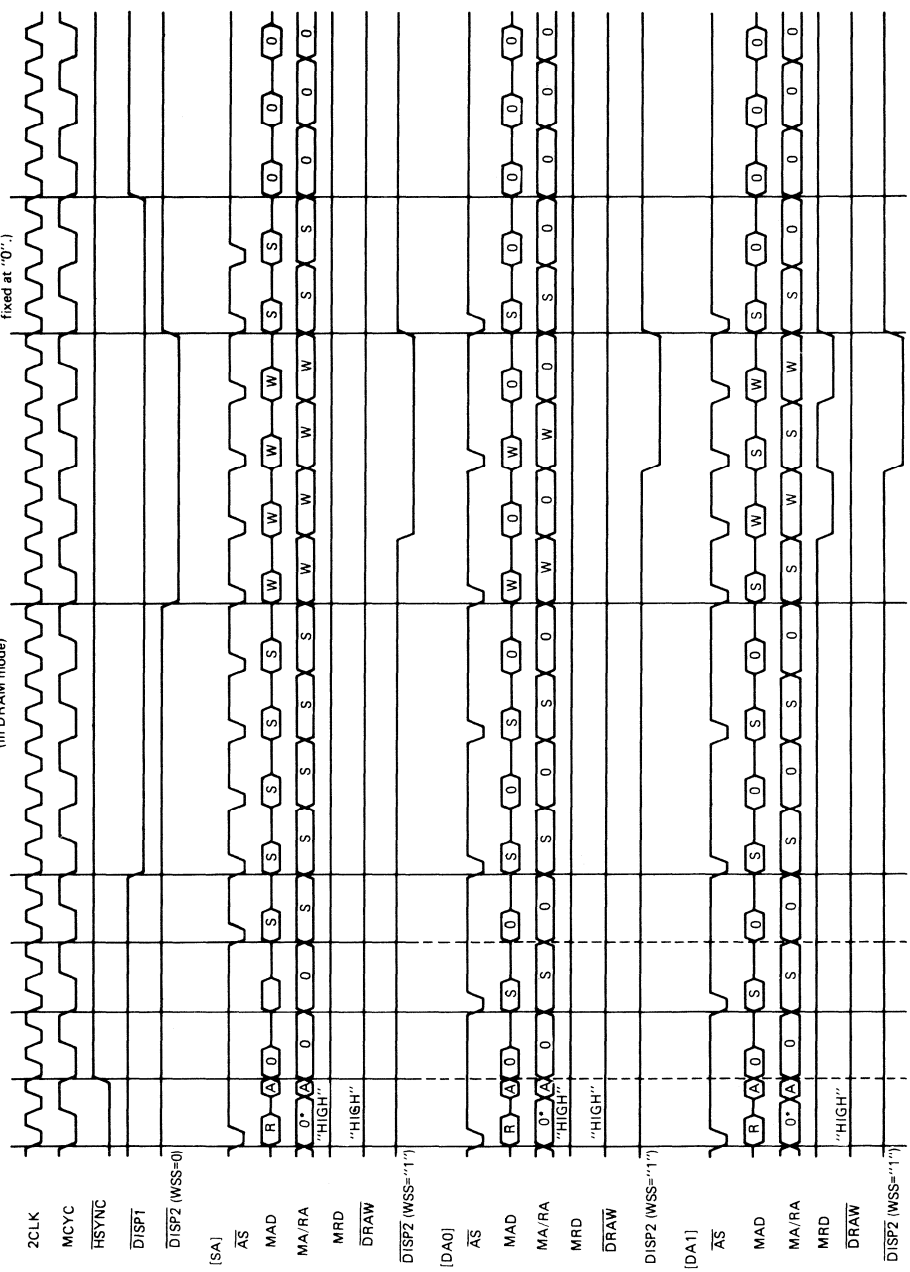


Figure 2.1(d) Access Mode Timing

2.2.1.2 Graphic Address Increment Mode

During display operation, the ACRTC can be programmed to control the graphic display address in six ways including increment by 1, 2, 4, 8 and 16 words, 1 word every two display cycles and no increment. (Note: +16 increment mode is not valid for R Mask Version.)

Setting GAI to increment by 2, 4 or 8 words per display cycle achieves linear increases in the video data rate i.e. for a given configuration setting GAI to 2, 4 or 8 words will achieve 2, 4 or 8 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.

Figure 2.2 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 × N × D)/(A × 16)

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

Note) R Mask Version don't offer +16 increment made.

Figure 2.2 Graphic Address Increment Modes

2.2.2 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 $\overline{\text{HSYNC}}$ 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 figure 2.3(a).

Address Increment Mode	Refresh Address Output Terminal
+1 (GAI=000)	MAD0–7
+2 (GAI=001)	MAD1–8
+4 (GAI=010)	MAD2–9
+8 (GAI=011)	MAD3–10
+16 (GAI=100)	MAD4–11
+0 (GAI=101)	MAD0–7
+1/2 (GAI=11X)	MAD0–7

Note) Only the S mask version can support +16 increment mode.

Figure 2.3(a) GAI and DRAM Refresh Addressing

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 $\overline{\text{HSYNC}}$ 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:

$$HSW \geqq N / (Tr \times Fh)$$

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.

$$HSW \geqq 128 / (2 \times 15.75) = 4.06$$

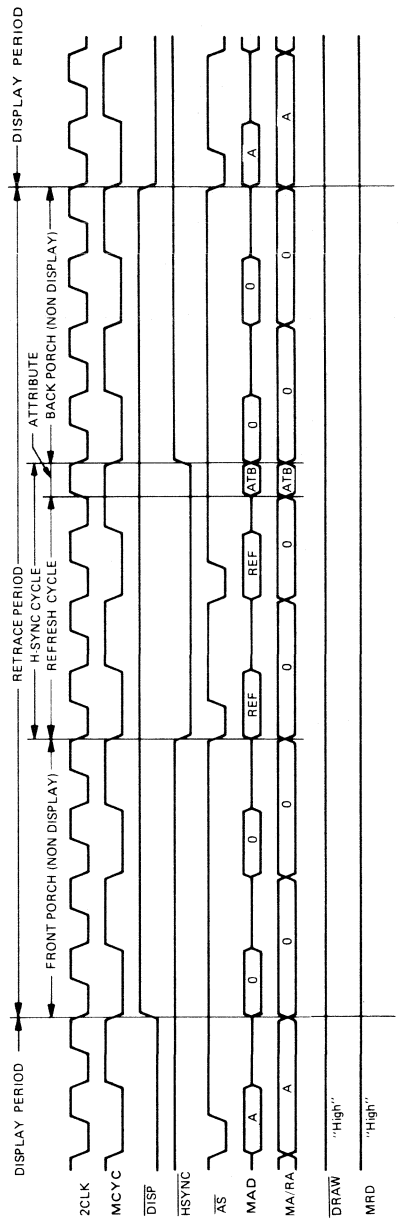


Figure 2.3(b) DRAM Refresh Timing

2.2.3 External Synchronization

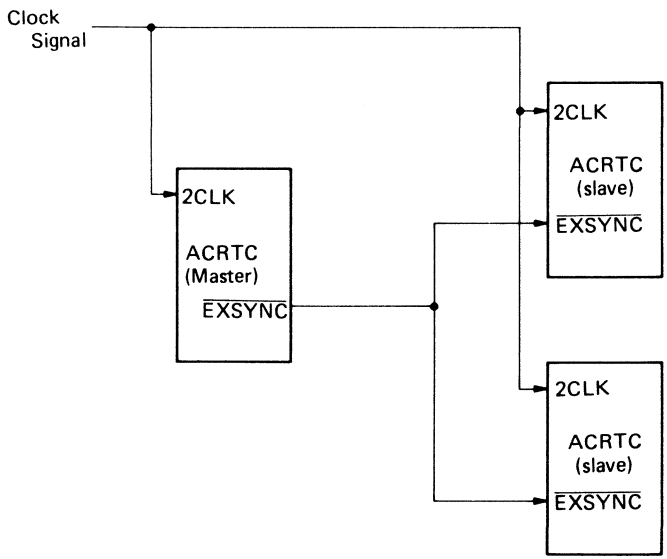
The ACRTC $\overline{\text{EXSYNC}}$ pin allows synchronization of multiple ACRTCs or other video signal generators. The ACRTC may be programmed as a single Master device, or as one of a number of Slave devices.

To synchronize multiple ACRTCs, simply connect all the $\overline{\text{EXSYNC}}$ pins together.

For synchronizing to other video signals, the connection scheme depends on the raster scan mode. In Non-Interlace mode, $\overline{\text{EXSYNC}}$ corresponds to $\overline{\text{VSYNC}}$. In Interlace modes, $\overline{\text{EXSYNC}}$ corresponds to $\overline{\text{VSYNC}}$ of the odd field.

Note) 1. The ACRTC performs the synchronization everytime it accepts the pulse input from $\overline{\text{EXSYNC}}$ in the slave mode.
It is recommended that the synchronous pulse should be input from $\overline{\text{EXSYNC}}$ only when the synchronization gap between the synchronous signal of the master device and that of the ACRTC in the slave mode ($\overline{\text{HSYNC}}$ and $\overline{\text{VSYNC}}$ are output also in the slave mode.).

2. The ACRTC needs to be controlled not to execute the drawing operation during $\overline{\text{EXSYNC}}$ input.



3. In the slave mode, when the ACRTC accepts the pulse input of EXSYNC, the inner counters and control circuits for cursor blink and Attribute blink are resetted everytime. In this case, the ACRTC in the slave mode performs Cursor blink and Attribute blink correctly.

Figure 2.4(a) External Synchronization

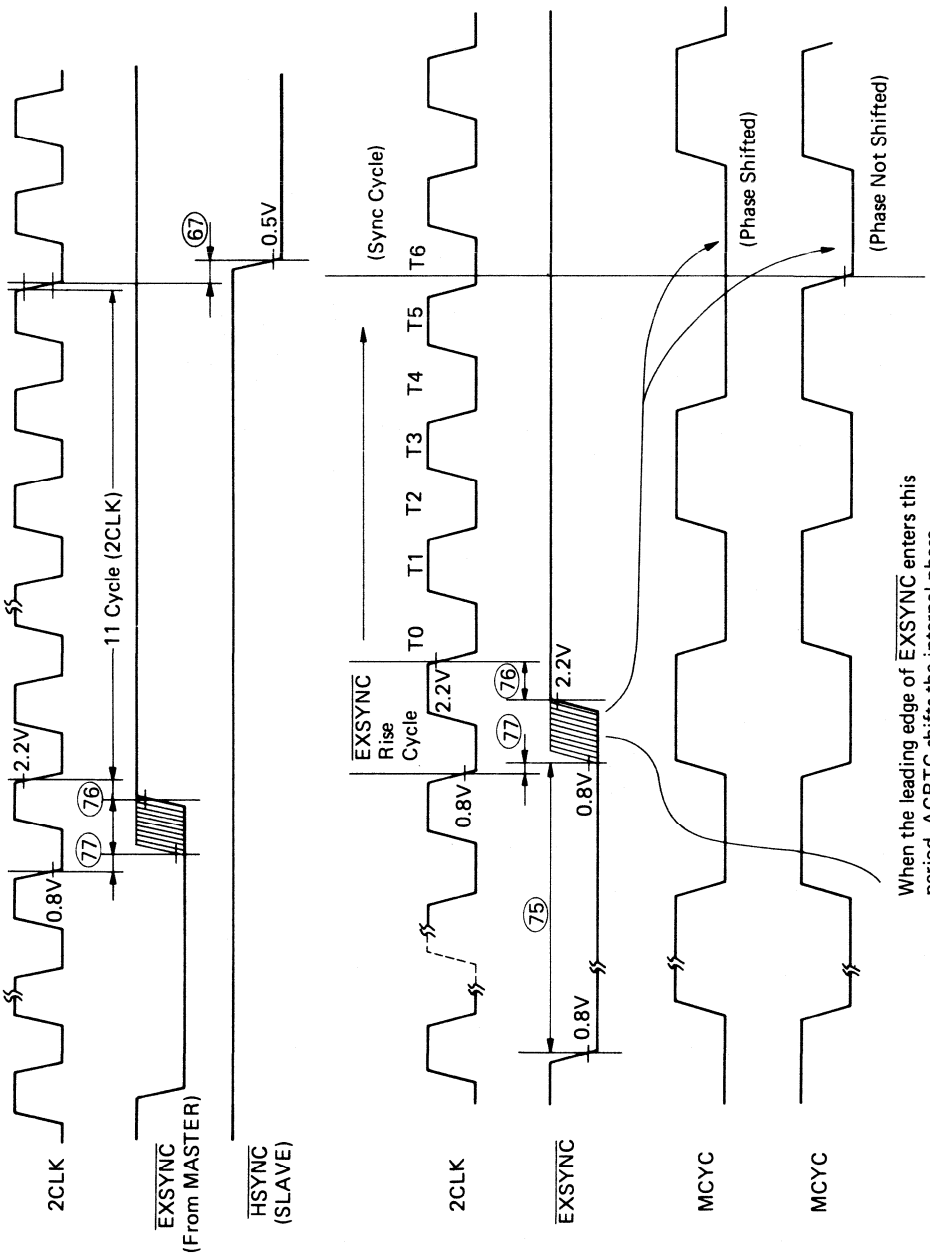


Figure 2.4(b) EXSYNC Timing

2.3 MPU Interface

2.3.1 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 $\overline{RES}$.

An MPU bus cycle is initiated when $\overline{CS}$ is asserted (following the assertion of RS and R/$\overline{W}$). The ACRTC responds to $\overline{CS}$ low by asserting $\overline{DTACK}$ low to complete the data transfer. $\overline{DTACK}$ 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 $\overline{CS}$ "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, $\overline{DTACK}$ can simply be left open assuming the system design guarantees that WAIT states cannot occur as described above. If WAIT states may occur, $\overline{DTACK}$ can be used with external logic to synthesize a READY signal.

2.3.2 DMA Transfer

The ACRTC can interface with an external DMA controller using three handshake signals, DMA Request ($\overline{DREQ}$), DMA Acknowledge ($\overline{DACK}$) and DMA Done ($\overline{DONE}$).

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.

2.3.2.1 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 ($\overline{DREQ}$ 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 $\overline{DONE}$ 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.

2.3.2.2 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 $\overline{\text{DREQ}}$ is a low level control output to the DMAC which allows multiple data transfers during each acquisition of the MPU bus.

2.3.3 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.

3. DISPLAY FUNCTION

3.1 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.

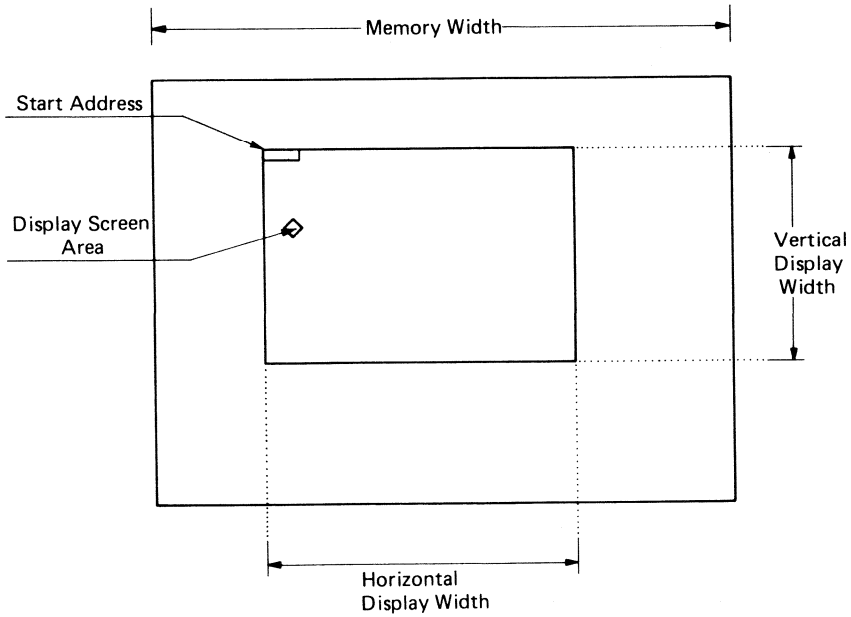


Figure 3.1 Display Screen/Frame Buffer Relationship

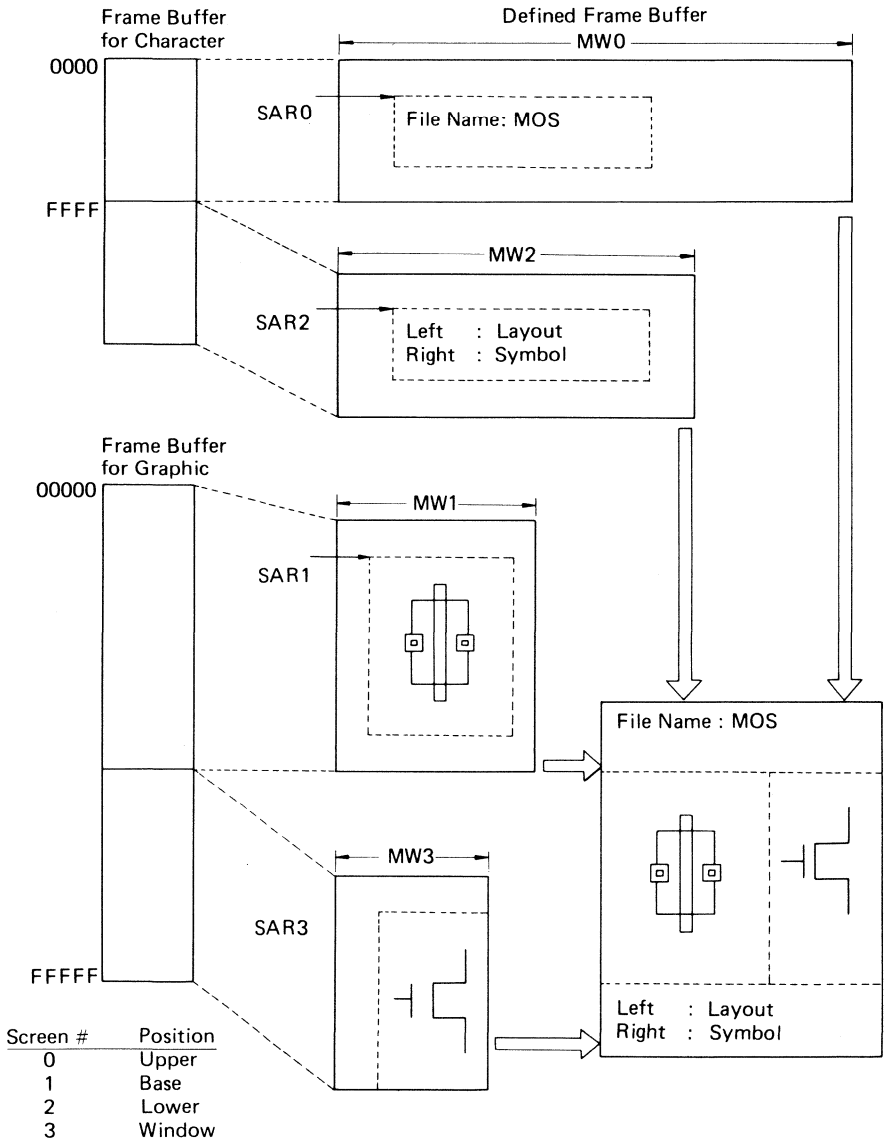


Figure 3.2 Display Screen Combination

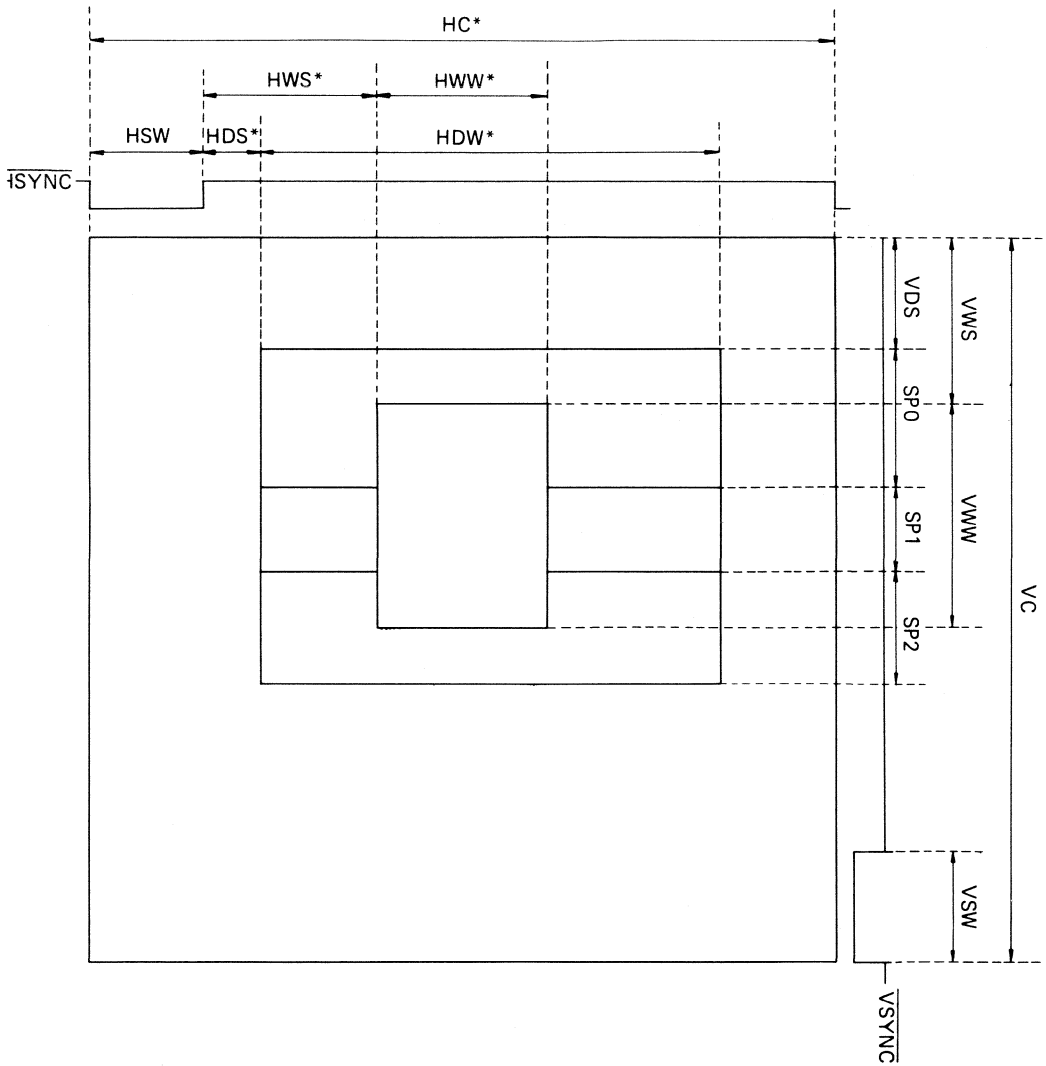


Figure 3.3 Display Screen Specification

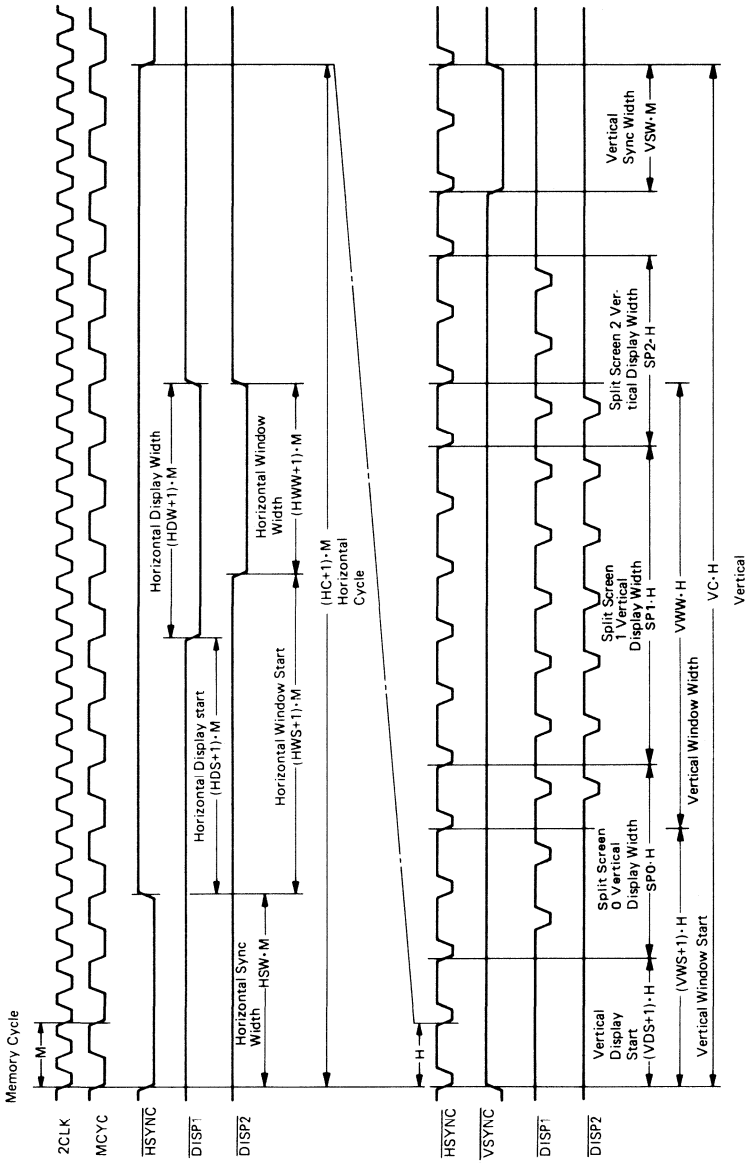


Figure 3.4 Display Screen Timing

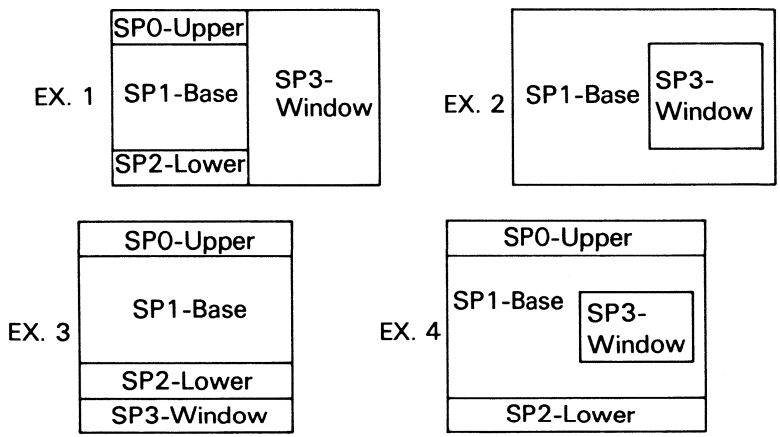


Figure 3.5 Example Screen Combinations

3.1.1 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.

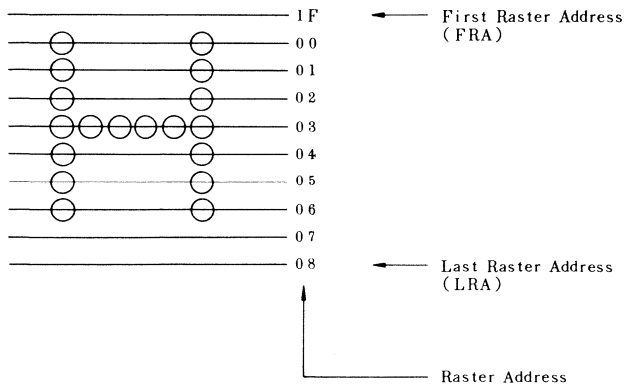


Figure 3.6 Character Screen Raster Addressing

3.2 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. The Block cursor don't output on Graphic screens.

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.

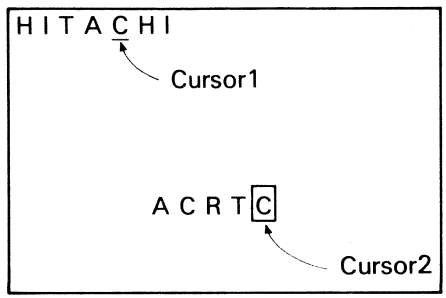


Figure 3.7(a) Two Separate Block Cursors

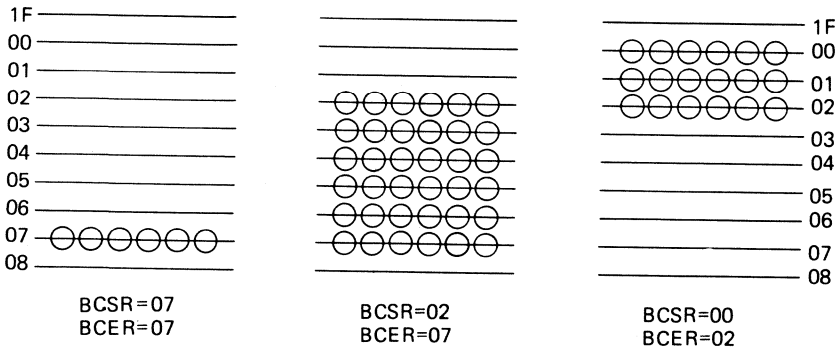


Figure 3.7(b) Block Cursor Examples

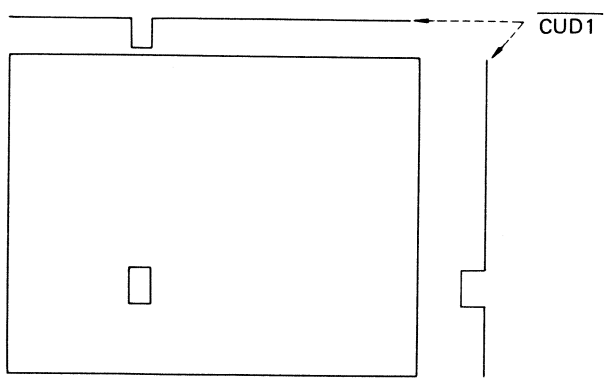


Figure 3.8 Graphic Cursor

The ACRTC provides two separate cursor outputs, $\overline{CUD1}$ and $\overline{CUD2}$. These are combined with two character cursor registers and a graphics cursor register to provide three cursor modes.

3.2.1 Block Mode

Two Block cursors are output on $\overline{CUD1}$ and $\overline{CUD2}$ respectively.

3.2.2 Graphic Mode

The Graphic cursor is output on $\overline{CUD1}$. Using an external cursor pattern memory allows a graphic cursor of various shapes. Two Block cursors are multiplexed on $\overline{CUD2}$.

3.2.3 Crosshair Mode

The horizontal and vertical components of the Graphic cursor are output on $\overline{CUD1}$ and $\overline{CUD2}$ respectively. This allows simple generation of a crosshair cursor control signal.

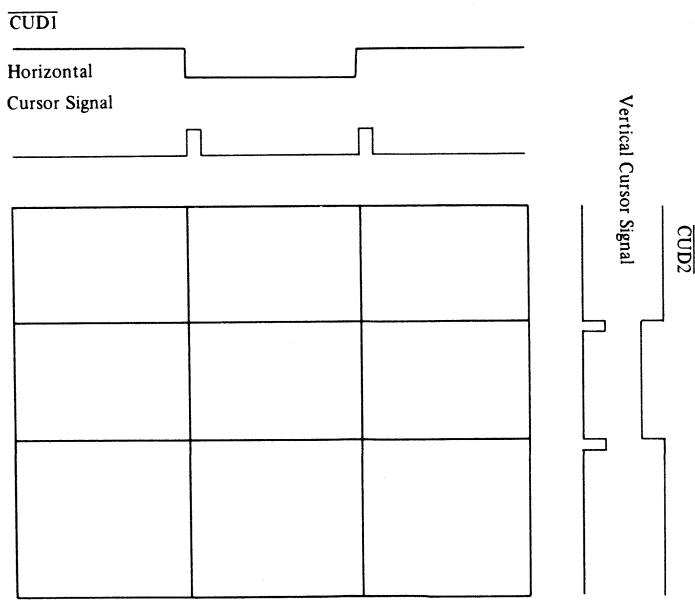


Figure 3.9 Crosshair Cursor

3.3 Scrolling

3.3.1 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.

3.3.2 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).

Note) The Background screens and the Window screen can be horizontally scrolled at a time in the dual access mode 1 by using the attribute code (ATC) etc.

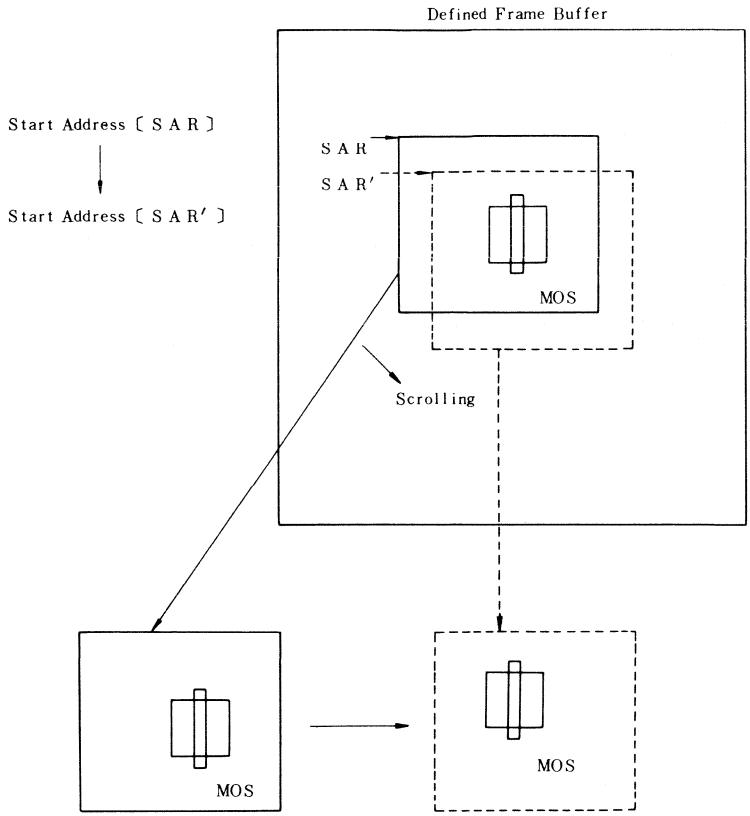


Figure 3.10 Scrolling By SAR (Start Address Register) Rewrite

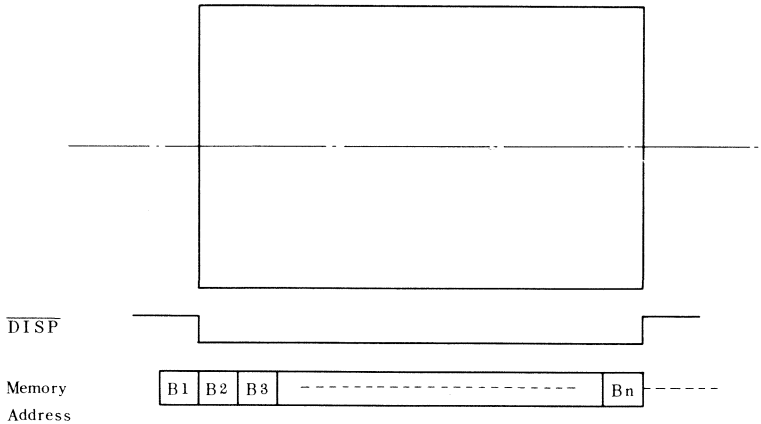


Figure 3.11 Horizontal Smooth Scroll — Base Screen

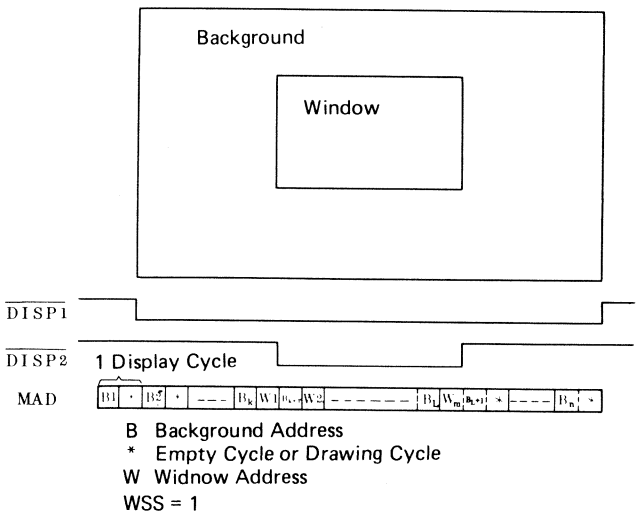


Figure 3.12 Horizontal Smooth Scroll — Window Screen

3.4 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.

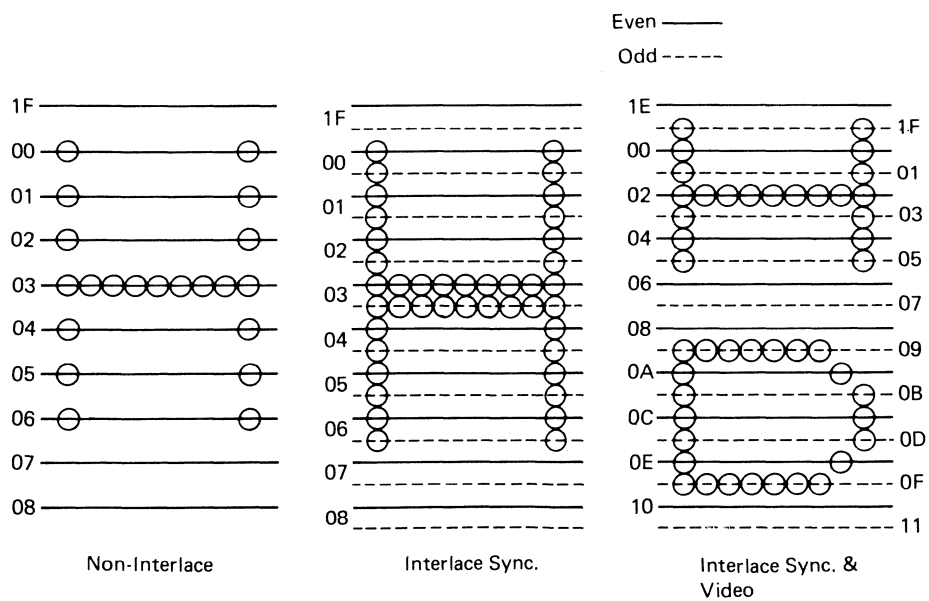


Figure 3.13 Raster Scan Modes

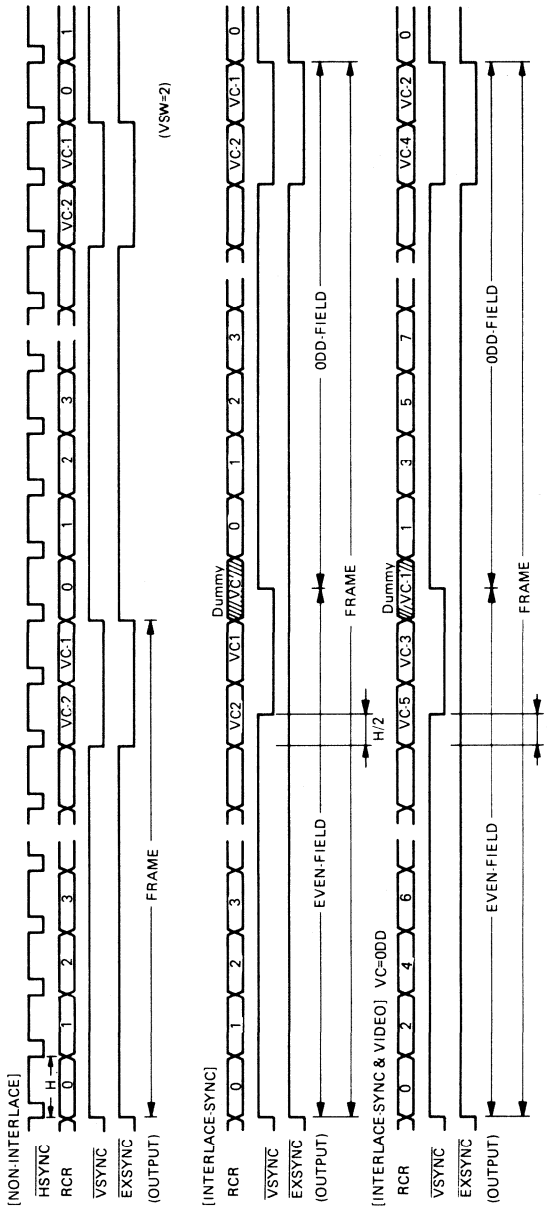


Figure 3.14 Raster Scan Timing

3.5 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 in unit of integer) 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.

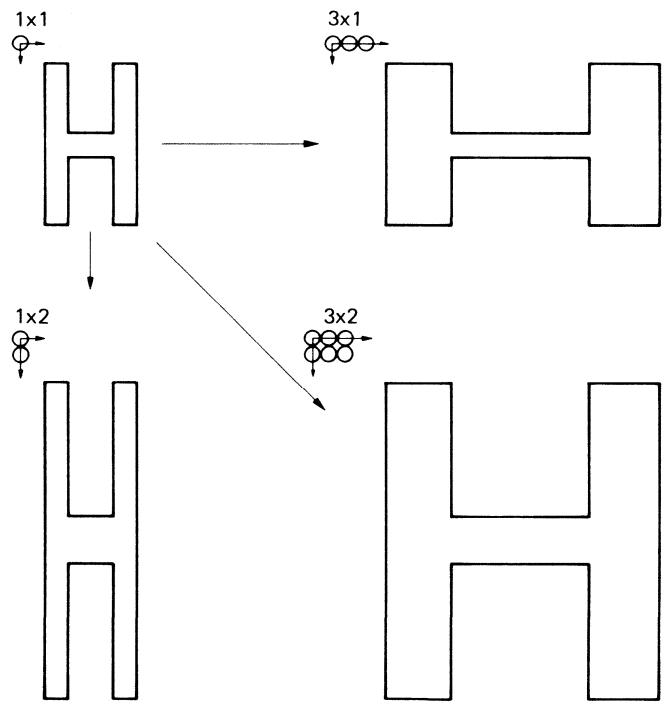


Figure 3.15 Zooming

3.6 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.

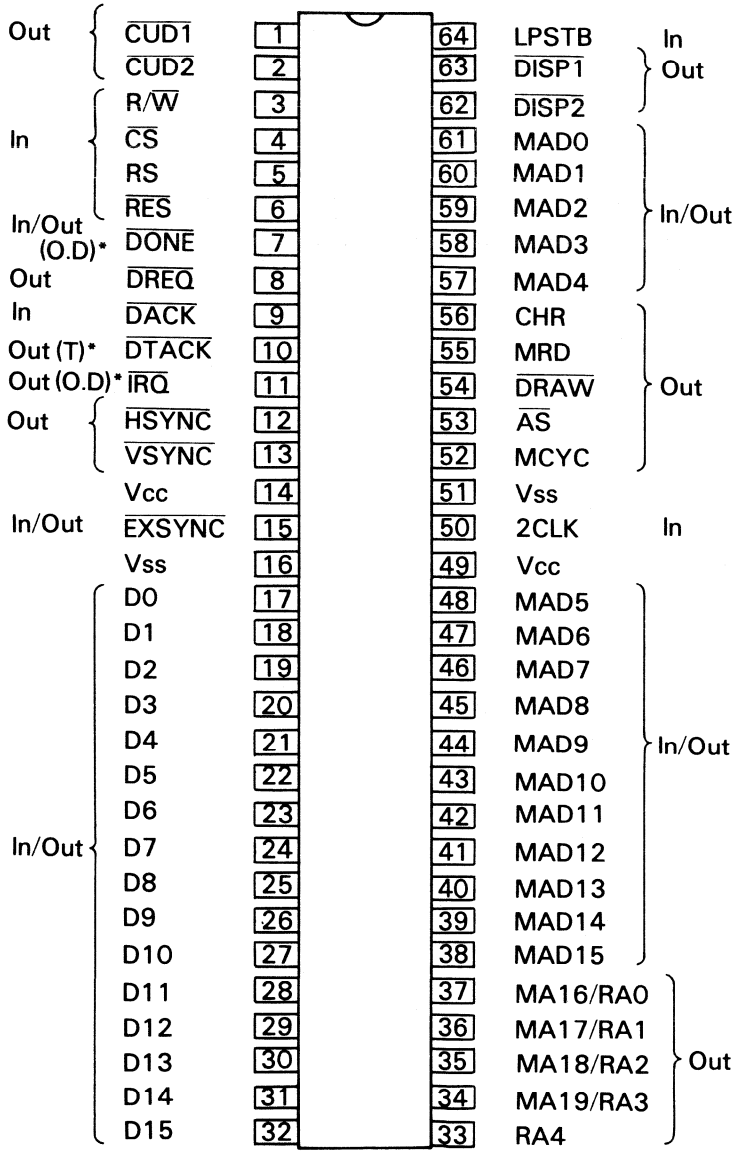
A light pen strobe pulse will occur when the CRT electron beam passes under the light pen during display refresh. When this pulse occurs, the contents of the ACRTC display refresh address counter will be latched into the Light Pen Address Register along with a logical screen (Character or Graphic screen) 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 is not 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.

4. SIGNAL DESCRIPTION

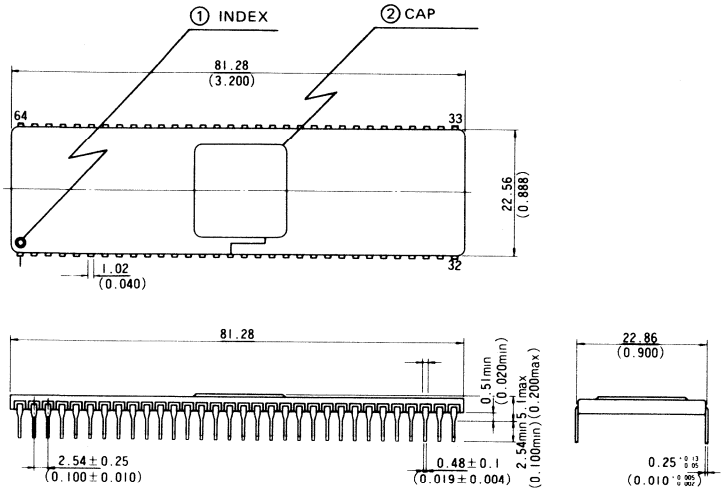
4.1 Pin Arrangement



* O.D : Open Drain
T : Three-state

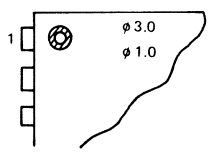
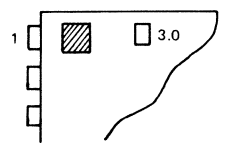
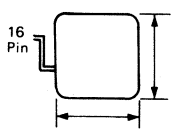
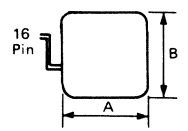
Figure 4.1 Pin Arrangement

4.1.1 PACKAGE Dimension [UNIT: mm (inch)]



(DC-64)

4.1.2 Package Difference Between R Mask and S Mask

	R mask	S mask
① IND		
② CAP		 A = 16.4±0.3 B = 18.4±0.3

4.2 Signal Functions

The ACRTC signal functions are grouped into 4 functional categories, MPU Interface, DMAC Interface, CRT Interface and Power Supply. All signals are TTL compatible.

4.2.1 MPU Interface

4.2.1.1 Reset ($\overline{RES}$:INPUT)

A low level on the $\overline{RES}$ input forces the ACRTC into the following state.

(a) Drawing and Display operation is stopped.

(b) ACRTC registers are initialized as follows.

- ○ Status register (SR) – CED, WFR and WFE bits are set to 1, all other bits reset to 0. ($FF23)
- ○ Command Control Register (CCR) – The ABT bit is set to 1. All other bits are reset to 0. ($8000)
- ○ Operation Mode Register (OMR) – The M/S and STR bits are reset to 0. All other bits are unaffected.
  All other ACRTC registers are unaffected by $\overline{RES}$.
- ○ FIFO Entry (FE)
  The pointer is cleared, and the written command/parameter and the read data are lost.

(c) The DRAM refresh address is placed on the MAD lines determined by the graphic address increment (GAI) mode. This remains the case until the start bit (STR) in the Operation Mode Register (OMR) is set to 1. $\overline{HSYNC}$ is also held low during the period from $\overline{RES}$ until the start bit in OMR is set to 1 by the host. The DRAM refresh address is not updated during $\overline{RES}$ input as for R mask version.

(d) $\overline{RES}$ need to be "Low" when it is supplied power.

4.2.1.2 Bi-directional Host System Data Bus (D0-D15:INPUT/OUTPUT:3-STATE)

These lines are used for data transfer between the ACRTC and the host system data bus (MPU and/or DMAC). D0-D15 outputs are three state buffers and remain in the high impedance state except during host reads of ACRTC registers.

At the rising edge of $\overline{RES}$, depending on the state of the $\overline{DACK}$ input, the ACRTC can be configured for an 8 bit data bus using D0-D7. In this case, D8-D15 should be left open.

Refer to 4.2.2.2 DMA Acknowledge ($\overline{DACK}$) as for the 8/16 bit bus selection.

4.2.1.3 Read/Write (R/$\overline{W}$:INPUT)

R/$\overline{W}$ controls the direction of transfer between the host system bus and the ACRTC. During non-DMA transfers, when R/$\overline{W}$ is high, data is transferred from the ACRTC to the host, and when low, data is transferred from the host to the ACRTC.

When the ACRTC executes a DMA transfer using an external DMAC, the polarity of R/$\overline{W}$ is reversed. In this case, when R/$\overline{W}$ is high, data is transferred from the host to the ACRTC, and when low, data is transferred from the ACRTC to the

host.

4.2.1.4 Chip Select ($\overline{CS}$:INPUT)

The Chip Select, when low, enables access of the ACRTC by the host MPU. Note that Chip Select must not be low during DMA transfers ($\overline{DACK}$ = low). RS and R/$\overline{W}$ must be valid when $\overline{CS}$ is asserted and write data must be valid prior to the trailing (rising) edge of $\overline{CS}$.

When the ACRTC host data bus mode is 16 bit data bus, 8 bit data transfers are not allowed.

Note) The ACRTC operates erroneously when both $\overline{CS}$ and $\overline{DACK}$ are asserted to low during the same period.

4.2.1.5 Register Select (RS: INPUT)

RS is used to select ACRTC hardware accessed registers. When RS is low, reads (R/$\overline{W}$ = high) access the Status register and writes (R/$\overline{W}$ = low) access the Address register. When RS is high, reads and writes access the particular ACRTC Control register with address defined in the previous write to the Address register. If the accessed register is in the range of r80 – rFF, the address register will automatically be incremented to allow access to the next sequential register address. This allows high speed initialization of registers in the address range of r80 – rFF without requiring the MPU to reload the address register for each sequential access. Note that the address increment is +1 for 8 bit host interface mode and +2 for 16 bit host interface mode.

Normally, RS is connected to the least significant bit of the MPU address bus.

During the DMA data transfer, ACRTC does not use RS because the FIFO is automatically selected. So RS need to be held either high or low level. (However as for the R mask version, the RS must be held "high" during DMA data transfer.)

4.2.1.6 Data Transfer Acknowledge ($\overline{DTACK}$:OUTPUT:OPEN DRAIN)

The ACRTC will drive $\overline{DTACK}$ low to indicate completion of a data transfer cycle. $\overline{DTACK}$ is compatible with asynchronous bus interface hosts including the HD68000 MPU and HD68450 DMAC.

4.2.1.7 Interrupt Request ($\overline{IRQ}$:OUTPUT:OPEN DRAIN)

This open drain output is driven low when the ACRTC requires interrupt service. In order to generate an $\overline{IRQ}$, the interrupting condition must be enabled in the Command Control Register (CCR).

The action required to clear the interrupting condition is specified in the Status Register description (section 5.3).

Note) In order to negate $\overline{IRQ}$, the interrupting condition flag in the status register need to be cleared.

4.2.2 DMAC Interface

Three DMA handshaking lines allow the ACRTC to use an external DMA controller. The DMA protocol is directly compatible with HD68450 DMAC single address mode transfers.

4.2.2.1 DMA Request ($\overline{\text{DREQ}}$:OUTPUT)

During DMA transfer mode, $\overline{\text{DREQ}}$ is used to request data transfer service from the host bus DMAC. $\overline{\text{DREQ}}$ is asserted to active low level by ACRTC execution of a Data DMA transfer command (when the Data DMA Mode bit (DDM) in CCR is set to 1) or by setting the Command/Parameter DMA transfer mode bit (CDM) in the CCR to 1. Data DMA can be programmed as burst or cycle steal, while Command/Parameter DMA can only be cycle steal mode.

4.2.2.2 DMA Acknowledge ($\overline{\text{DACK}}$:INPUT)

$\overline{\text{DACK}}$ is an answer back signal from the DMAC to which $\overline{\text{DREQ}}$ has been issued and indicates that the host bus has been acquired, and data transfer can occur. Note that when $\overline{\text{DACK}}$ is asserted low, $\overline{\text{CS}}$ must not be low and the R/$\overline{\text{W}}$ signal polarity is reversed.

RS and R/$\overline{\text{W}}$ must be valid prior to $\overline{\text{DACK}}$ assertion and data written to the ACRTC must be valid prior to the trailing (rising) edge of $\overline{\text{DACK}}$.

$\overline{\text{DACK}}$ is also used to define whether an 8 or 16 bit host data bus is used. During reset, if $\overline{\text{DACK}}$ is low, 8 bit mode is used and if $\overline{\text{DACK}}$ is high, 16 bit mode is used. In the case of 8 bit mode, host-ACRTC communication occurs on the D0-D7 portion of the data bus, white D8-D15 are disabled and driven high. When 8 bit bus mode is selected the automatic increment mode for the Address Register is set to '+1' (alternating even and odd register addresses), while 16 bit bus mode sets it to '+2' (even addresses only).

When 16 bit host data bus mode is used, 8 bit transfers are not allowed. When DMA is not used, $\overline{\text{DACK}}$ should be pulled up to a high level. In 8 bit interface mode, $\overline{\text{DACK}}$ must be held low level by the external circuit except $\overline{\text{RES}}$ input period.

4.2.2.3 Done (DONE:INPUT/OUTPUT:OPEN DRAIN)

$\overline{\text{DONE}}$ is used to terminate DMA transfers. During Data DMA transfers, $\overline{\text{DONE}}$ is an output and when asserted low indicates DMA termination to the external DMAC. During Command/Parameter DMA transfers, $\overline{\text{DONE}}$ is an input asserted low by the external DMAC to terminate DMA. Note that Data DMA cannot be terminated by externally forcing $\overline{\text{DONE}}$ low.

$\overline{\text{DONE}}$ is open drain when in the output state, and should be pulled up to high level when not used.

4.2.3 CRT Interface

4.2.3.1 Clock (2CLK:INPUT)

This is the basic operating clock for the ACRTC which is derived from the external dot clock. The ACRTC internally divides 2CLK by 2 to generate the MCYC Memory Cycle Clock. Thus, 2CLK is twice the frequency of the frame buffer memory cycle time. 2CLK must be a continuous clock input.

4.2.3.2 Vertical Synchronization ($\overline{\text{VSYNC}}$:OUTPUT)

$\overline{\text{VSYNC}}$ is used to output the active low Vertical Synchronization timing signal required by the CRT display device.
Note) $\overline{\text{VSYNC}}$ is output even in the slave mode.

4.2.3.3 Horizontal Synchronization ($\overline{\text{HSYNC}}$:OUTPUT)

$\overline{\text{HSYNC}}$ is used to output the active low Horizontal Synchronization timing signal required by the CRT display device.

$\overline{\text{HSYNC}}$ is also used when the ACRTC performs DRAM refresh addressing of the frame buffer. In the case that the STR start bit or the RAM bit in the Operation Mode Register (OMR) are set to 0, $\overline{\text{HSYNC}}$ asserted low indicates that the DRAM refresh addresses are present on MAD frame buffer address/data bus.
Note) $\overline{\text{HSYNC}}$ is output even in the slave mode.

4.2.3.4 External Synchronization ($\overline{\text{EXSYNC}}$:INPUT/OUTPUT)

$\overline{\text{EXSYNC}}$ is an active low input and output signal used for synchronizing multiple ACRTCs or synchronizing the ACRTC with other video generating devices. The ACRTC is programmed as a master or slave by the state of the M/S bit in the Operation Mode Register (OMR). When the ACRTC is master, $\overline{\text{EXSYNC}}$ is an output which may be used to drive a slave video generating devices $\overline{\text{VSYNC}}$ input or a slave ACRTCs $\overline{\text{EXSYNC}}$ input. When the ACRTC is slave, $\overline{\text{EXSYNC}}$ is an input which receives the masters $\overline{\text{EXSYNC}}$ (master is another ACRTC) or $\overline{\text{VSYNC}}$ (master is another video generating device).

In both master and slave configurations, the timing of $\overline{\text{EXSYNC}}$ depends on the interlace mode. For example, in interlaced modes, $\overline{\text{EXSYNC}}$ timing corresponds to $\overline{\text{VSYNC}}$ of the odd field.
Note) The ACRTC should be programmed as a master in a single operation.

4.2.3.5 Light Pen Strobe (LPSTB:INPUT)

LPSTB input accepts a positive strobe pulse generated by an external light pen. When asserted high, the current frame buffer display refresh address is latched into the Light Pen Address Register and the LPD (Light Pen Detect) bit in the Status Register is set to 1, generating an interrupt if enabled to do so by the MPU. The stored address will be different from the actual address due to the following delays.

(a) ACRTC address output delay
(b) Address output to video signal output delay
(c) Light pen detection to LPSTB delay
(d) LPSTB to internal recognition delay

The actual address should be calculated by adjusting the stored address considering the above delays. Also note that, for Superimposed access mode, when the light pen strobe occurs in the Window screen, the overlapped Background (Upper, Base, Lower) screen address is latched. Besides the Drawing address is not latched.

When LPSTB is input during the horizontal synchronizaton period, horizontal back porch period, and vertical retrace period, correct address cannot be stored.

4.2.3.6 Memory Cycle (MCYC:OUTPUT)

MCYC frequency is one-half that of thr ACRTC 2CLK input and is output continuously. MCYC determines frame buffer memory access timing. MCYC low indicates the address portion of the memory access while MCYC high indicates the data portion of the memory access.

4.2.3.7 Address Strobe ($\overline{AS}$:OUTPUT)

$\overline{AS}$ output is used to latch the frame buffer address. When $\overline{AS}$ is low, the MAD outputs contain the frame buffer address. $\overline{AS}$ is also used to load the external parallel to serial (shift register) converter with the data from frame buffer during the display cycle. $\overline{AS}$ output is asserted to Low only when the valid address for drawing or displaying (including refresh) is output.

4.2.3.8 Memory Read (MRD:OUTPUT)

During a frame buffer access, MRD indicates the direction of data transfer between the ACRTC and the frame buffer. When MRD is high, a frame buffer read cycle occurs, and when MRD is low, a frame buffer write cycle occurs. In superimposed mode, MRD low indicates the display cycle of window screen data (second phase).

4.2.3.9 Draw ($\overline{DRAW}$:OUTPUT)

The $\overline{DRAW}$ signal differentiates between ACRTC drawing and CRT display refresh cycles. When $\overline{DRAW}$ is low, the MAD outputs contain multiplexed drawing address and data information. When $\overline{DRAW}$ is high, the MAD outputs contain a display refresh address during the address portion of the cycle, and are high impedance during the data portion of the cycle.

Note) $\overline{DRAW}$ is asserted to low level only during the drawing cycle.

4.2.3.10 Frame Buffer Memory Address/Data (MAD0-MAD15:INPUT/OUTPUT:3-STATE)

MAD0-MAD15 are the time multiplexed, bi-directional frame buffer memory address and data bus. When $\overline{AS}$ is low, MAD contains the lower 16 bits of the drawing or display address. When $\overline{AS}$ is high and $\overline{DRAW}$ is low, MAD transfers the drawing data to and from the frame buffer.

When no frame buffer access is occurring, the MAD bus is "Low".

When the RAM or STR bit in the Operation Mode Register (OMR) is set to 0, the 8 bit DRAM refresh address is output on MAD during $\overline{HSYNC}$ low. The particular bits of MAD used for this 8 bit refresh address depend on the programmed Graphic Address Increment (GAI) mode.

Note) The address cycle and the data cycle in MAD are separable through MCYC.

4.2.3.11 Memory Address/Raster Address (MA16/RA0-MA19/RA3:OUTPUT)

These lines output either the 4 most significant bits of the frame buffer address (MA16-MA19) or the 4 least significant bits of the raster address (RA0-RA3). In Character mode (CHR = high), these lines are used as a raster address for connection to an external character generator. In Graphic mode (CHR = low) these lines are used with MAD0-MAD15 to provide a 20 bit linear frame buffer address.

4.2.3.12 Raster Address 4 (RA4:OUTPUT)

In Character mode (CHR = high), RA4 output the most significant bit of the raster address. Thus, 5 bits (RA0-RA4) provide up to 32 rasters per character.

In Graphic mode (CHR = low), the state of this output is undefined.

4.2.3.13 Character (CHR:OUTPUT)

CHR is an output indicating whether the current frame buffer address on MAD has been defined as corresponding to character (CHR = high) or graphic (CHR = low). When high, MAD0-MAD15 contains a 16 bit frame buffer address, while other MAD lines contain raster address information. When low, MAD0-MAD15, MA16-MA19 contains a 20 bit linear frame buffer address. CHR can be used to enable an external character generator. Also, CHR can be used to enable the appropriate memory bank in the case that character and graphic memory are separated.

4.2.3.14 Display Timing ($\overline{DISP1}$,$\overline{DISP2}$:OUTPUT)

These active low outputs indicate the active display period of the screen. They can be used in one of two ways.

(a) Background screen/window screen display timing signal
(b) Vertical/horizontal display timing signal

4.2.3.15 Cursor Display ($\overline{CUD1}$,$\overline{CUD2}$:OUTPUT)

These outputs are externally logically combined with the video signal to produce the cursor display on the screen. Three modes of cursor display are selectable by setting the cursor mode (CM) bits in the Cursor Definition Register (CDR).

Cursor Mode	Description	$\overline{CUD1}$	$\overline{CUD2}$
BLOCK	The separate display of two BLOCK cursors	Block cursor 1	Block cursor 2
GRAPHIC	The display of a GRAPHIC cursor and two multiplexed BLOCK cursors	Graphic cursor	Block cursor 1&2
CROSSHAIR	The X and Y portions of a CROSSHAIR cursor	X portion	Y portion

4.2.4 Power Supply

4.2.4.1 Vcc, Vss

These pins supply power to the ACRTC. V_{CC} is specified as 5V ± 5% (4.75V~5.25V).

Note) The ACRTC has two pairs of V_{CC} and V_{SS}.
These four pines need to be certainly connected to the power supply or the ground.

4.2.5 Video Attributes

The ACRTC outputs 20 bits of video attributes on MAD0-MAD15 and MA16/RA0-MA19/RA3. These attributes are output 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.

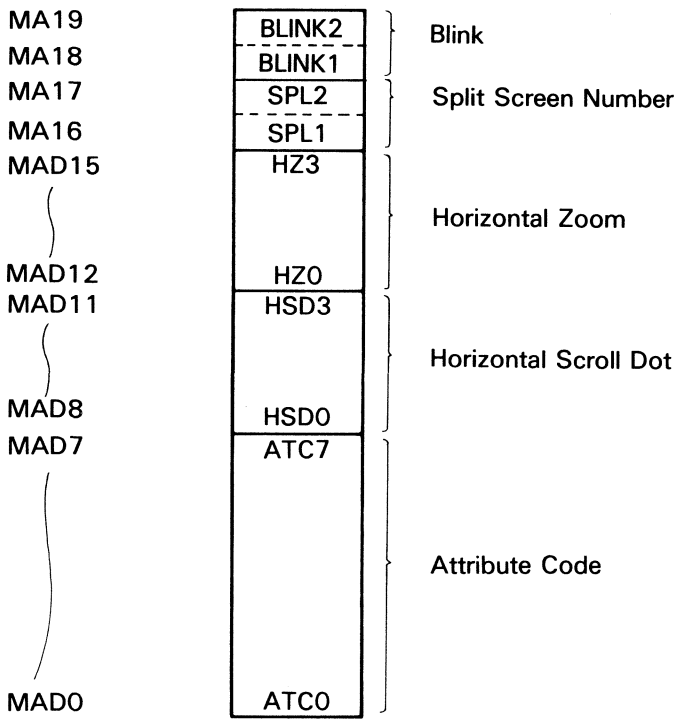


Figure 4.2 Video Attributes

4.2.5.1 Attribute Code (ATC0-ATC7:MAD0-MAD7)

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 signl like lamps used in the system.
(9) Code strage (max. 8 bit) or selection signal etc.

4.2.5.2 Horizontal Scroll Dot (HSD0-HSD3:MAD8-MAD11)

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 (WSS) 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.

4.2.5.3 Horizontal Zoom Factor (HZ0-HZ3:MAD12-MAD15)

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. Don't rewrite HZF while the base screen is displayed, because Hz is renewed by the raster. In the Single Access mode and the Interleave Mode (Dual Access 0), don't Display the Window Screen on the zoomed Base Screen.

4.2.5.4 Split Position (SPL1-SPL2:MA16-MA17)

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.

4.2.5.5 Blink (BLINK1-BLINK2:MA18-MA19)

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 $\overline{\text{VSYNC}}$ pulses. These lines are used to implement character and screen blink.

5. REGISTER DESCRIPTION

5.1 Internal Register Access

The ACRTC incorporates more than 200 bytes of internal Control registers and Control RAM which are accessible by the host MPU. The programming model is shown in figure 5.1.

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

Hexadecimal numbers are denoted by a leading $ i.e. $1234, $FF, etc.

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 ZFRs register address is shown as 'rEA'.

For FIFO accessible Drawing Parameter Registers, the register address is shown as 'PrNN'. For example, the Color Comparison Register is addressed as parameter register 2 hex, so the CMP register address is shown as 'Pr02'.

Bit subfields within the register are denoted using decimal bit numbers in which bit 0 is the least significant bit and bit 15 the most significant bit.

When the register diagram is shown, unused bits will be shaded. Unless stated otherwise, unused bits may be freely written with any value, and that value will be returned on subsequent reads of the register.

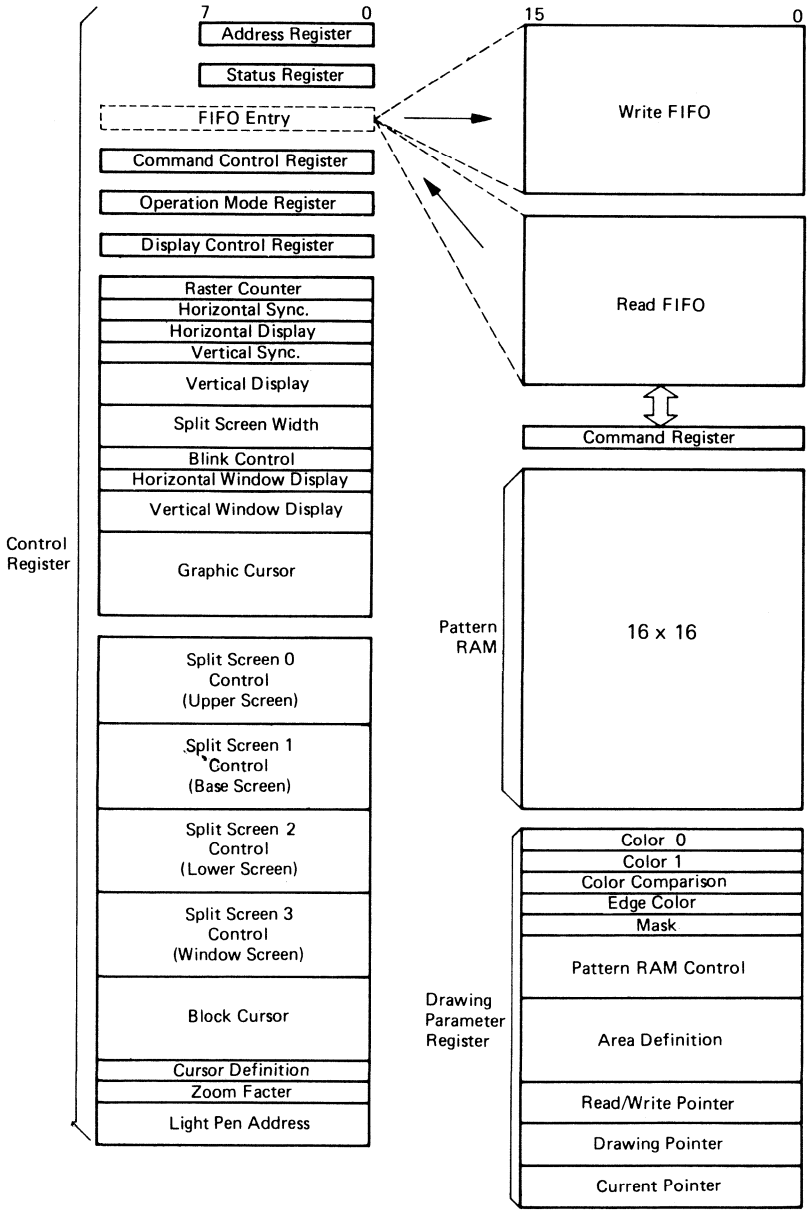


Figure 5.1 Programming Model

CS	RS	R/W	Reg. No.	Register Name	Abbr.	DATA (H) 15	14	13	12	11	10	9	8	DATA (L) 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	r00	FIFO Entry	FE									FE							
		1/0	r02	Command Control	CCR	ABT	PSE	DDM	CDM	DRC		GBM		CRE	ARE	CEE	LPE	RFE	RRE	WRE	WEE
		1/0	r04	Operation Mode	OMR	M/S	STR	ACP	WSS	CSK		DSK		RAM		GAI			ACM		RSM
		1/0	r06	Display Control	DCR	DSP	SE1		SE0		SE2		SE3				ATR				
		–	r08 | r7E																		
		1	r80	Raster Count	RCR				—								RC				
		1/0	r82	Horizontal Sync.	HSR					HC						—			HSW		
		1/0	r84	Horizontal Display	HDR					HDS							HDW				
		1/0	r86	Vertical Sync.	VSR				—								VC				
		1/0	r88	Vertical Display	VDR					VDS						—			VSW		
0	1	1/0	r8A						—								SP1				
		1/0	r8C	Split Screen Width	SSW				—								SP0				
		1/0	r8E						—								SP2				
		1/0	r90	Blink Control	BCR			BON1				BOFF1				BON2			BOFF2		
		1/0	r92	Horizontal Window Display	HWR					HWS							HWW				
		1/0	r94	Vertical Window Display	VWR				—								VWS				
		1/0	r96						—								VWW				
		1/0	r98	Graphic Cursor	GCR					CXE							CXS				
		1/0	r9A						—							CYS					
		1/0	r9C						—							CYE					
		–	r9E	ACRTC Work Area	–					—											
		–	rA0 | rBE	—																	
		1/0	rC0	UPPER Raster Address 0	RAR0			—				LRA0				—			FRA0		
		1/0	rC2	(Back Memory Width 0	MWR0	CHR		—									MW0				
		1/0	rC4	Ground) Start Address 0	SAR0				—			SDA0				—			SAOH/SAR0		
		1/0	rC6														SA0L				
		1/0	rC8	BASE Raster Address 1	RAR1			—				LRA1				—			FRA1		
		1/0	rCA	(Back Memory Width 1	MWR1	CHR		—									MW1				
		1/0	rCC	Ground) Start Address 1	SAR1				—			SDA1				—			SA1H/SRA1		
		1/0	rCE														SA1L				
		1/0	rD0	LOWER Raster Address 2	RAR2			—				LRA2				—			FRA2		
		1/0	rD2	(back Memory Width 2	MWR2	CHR		—									MW2				
		1/0	rD4	Ground) Start Address 2	SAR2				—			SDA2							SA2H/SRA2		
		1/0	rD6	Star 3													SA2L				
		1/0	rD8	Window Raster Address 3	RAR3			—				LRA3				—			FRA3		
		1/0	rDA	Memory Width 3	MWR3	CHR		—									MW3				
		1/0	rDC	Start Address 3	SAR3				—			SDA3				—			SA3H/SRA3		
		1/0	rDE														SA3L				
		1/0	rE0	Block Cursor 1	BCUR1			BCW1				BCSR1				—			BCER1		
		1/0	rE2														BCA1				
		1/0	rE4	Block Cursor 2	BCUR2			BCW2				BCSR2				—			BCER2		
		1/0	rE6														BCA2				
		1/0	rE8	Cursor Definition	CDR		CM			CON1			COFF1				—		CON2		COFF2
		1/0	rEA	Zoom Factor	ZFR			HZF				VZF					—				
		1	rEC	Light Pen Address	LPAR				—					CHR		—			LPAH		
		1	rEE														LPAL				
		–	rF0 | rFE	—	–																

Note) Read data from the unused field and the undefined area is invalid. As for writing, "0" needs to be written into the unused field Don't write into the undefined area.

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

Figure 5.1 (cont.) Programming Model

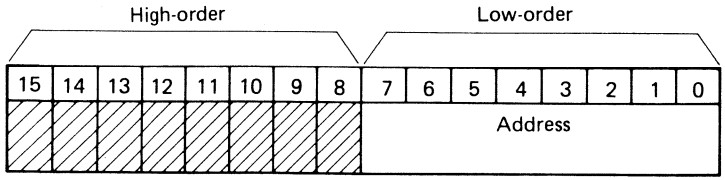


Figure 5.2 Address Register (AR)

AR is a write only register used to specify the address (0-$FF) of the ACRTC control register to be accessed. AR is written during MPU write cycles in which $\overline{CS}$ and RS are both low.

In the 16 bit mode, the least significant bit of AR is always recognized as 0, and thus the AR provides a word register address. In the 8 bit bus mode, if AR is even, the most significant byte of the control register is accessed. If odd, the least significant byte of the control register is accessed. Independent of 8 or 16 bit bus mode, AR should be loaded with 0 to access the read and write FIFOs.

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 initialization/access of these registers, reads and writes to the register address space r80-rFF result in automatic incrementing of AR. Thus, the programmer need not explicitly address each register for sequential access. AR is incremented by +1 in 8 bit bus mode, and by +2 in 16 bit bus mode.

AR is not incremented for accesses of r00-r7F.

Note) Because of the AR automatic increment for the control register of r80-rFF, the address must be written again into AR for accessing the same control register.

In the 16 bit mode, the upper-byte registers must always be written "$00", and the access to only upper or lower-byte registers is not provided.

5.3 Status Register (SR: $\overline{CS}$, $\overline{RS}$ = low, R/$\overline{W}$ = high)

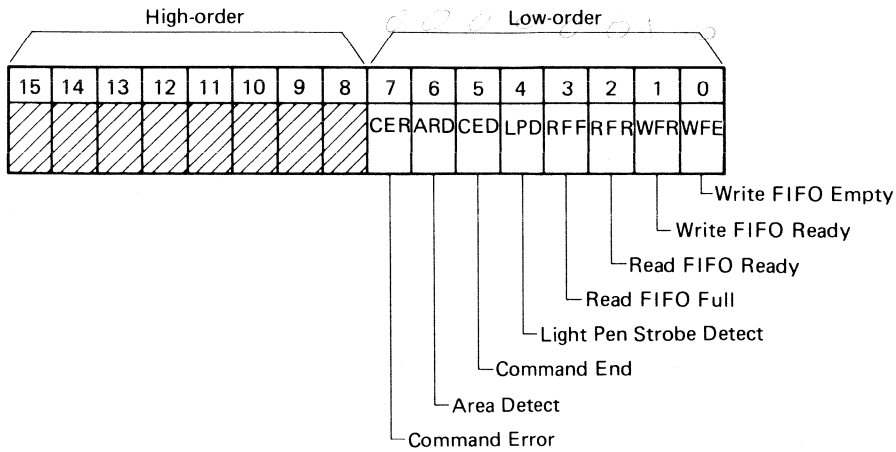


Figure 5.3 Status Register (SR)

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 ($\overline{IRQ}$) to be generated.

When hardware $\overline{RES}$ is asserted, the CED, WFE and WFR bits are set to 1 and all other bits are reset to 0.

Note) In the 16 bit bus mode, the upper-byte registers must always be written "$FF". The status is "$FF23" after the $\overline{RES}$ input or the ABT bit set.

○ Command Error Flag (CER: bit 7)

CER set to 1 indicates that the ACRTC has detected an undefined command.

Note 1) ○ A command is recognized to be undefined when the data firstly written into the write FIFO → except during the DMA data transfer and the command takes the values as shown in the table below.

A command, therefore, is not recognized to be undefined if the specified value in the command/parameter except the code of the below table.

○ When the undefined command is detected, the data in the FIFO must be cleared by the ABT setting after suspending the command/parameter transfer into the write FIFO and recognizing the absence of the data in the read FIFO. The command/parameter

writing is resumed after these operations.

○ Code regarded as the undefined command

0 0 0 0	0 0 x x	x x x x	x x x x
0 0 0 1	0 x x x	x x x x	x x x x
0 0 1 0	0 0 x x	x x x x	x x x x
0 0 1 1	x x x x	x x x x	x x x x
0 1 0 0	0 0 x x	x x x x	x x x x
0 1 0 1	0 x x x	x x x x	x x x x

x : Indicates "0" or "1"

CER is cleared by inputting $\overline{RES}$ or setting the ABT bit (in 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 MPU to detect whether the ACRTC has performed clipping or hitting during graphic drawing.

ARD is cleared by inputting $\overline{RES}$ or executing the RPR (Read Parameter Register) command or setting the ABT bit (in CCR) = 1.

○ Command End (CED: bit 5)

CED set to 1 indicates that the ACRTC is able to accept a new command.

Note 2) The CED flag is immediately reset if a next command is written in the write FIFO.

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 (LPSTB) has occurred and the Light Pen Address Register contains the latched address.

Note 3) In the R mask version, the proper address is not stored in the light pen register, but the flag can be set by the light pen strobe input.

LPD is cleared by inputting $\overline{RES}$ or reading the Light Pen Address Register or setting the ABT bit (in CCR) = 1.

○ Read FIFO Full (RFF: bit 3)

RFF set to 1 indicates that the read FIFO is full (contains 8 words/16 bytes of data).

Note 4) The command execution may be paused when RFF is set by the paint command, eg. Read out the data in the read FIFO immediately when RFF is set.

RFF is cleared by inputting $\overline{RES}$ or reading at least one 16 bit word from the read FIFO or setting the ABT bit (in CCR) = 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 inputting $\overline{RES}$ or setting the ABT bit (in CCR) = 1 or reading all data from the read FIFO.

Note 5) It indicates that the data is set in the read FIFO in every command

execution. The ACRTC operates erroneously when the read FIFO data is read before the RFR set. Make sure that RFR is already set before reading the read FIFO data (except during the DMA command execution).

○ Write FIFO Ready (WFR: bit 1)

WFR set to 1 indicates that the write FIFO is not full, and MPU writes can occur. WRF 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/16 bytes of data.

Note 6) It indicates that the write FIFO has more than 1 word of "space" for writing the command/parameter or the data. The command/parameter or the data is not taken into the ACRTC properly when writing the data into the write FIFO before the WFR set. Make sure that WFR is already set before writing the data into the write FIFO (except during the DMA command or the command DMA execution).

○ 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.

Note 7) It indicates that the write FIFO is completely empty. Up to 8 words of the command/parameter can be written here.

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

Table 5.1 Setting and Resetting of Status Register

Bit	Status Register	Set	Reset
7	CER (Command Error)	· An undefined command has been detected	· Abort · $\overline{\text{RES}}$ input
6	ARD (Area Detect)	· An area has been detected according to the AREA mode command	· Execute a Read Parameter Register (RPR) command · Abort · $\overline{\text{RES}}$ input
5	CED (Command End)	· A command has been executed · Abort · $\overline{\text{RES}}$ input	· Write a command to the write FIFO
4	LPD (Light Pen Strobe Detect)	· LPSTB has occurred, and contains the latched display address	· Read the Light Pen Address Register after reading the Status Register · Abort · $\overline{\text{RES}}$ input
3	RFF (Read FIFO Full)	· The read FIFO is full	· Read data from the read FIFO · Abort · $\overline{\text{RES}}$ input
2	RFR (Read FIFO Ready)	· The read FIFO contains data	· Read all data from the read FIFO · Abort · $\overline{\text{RES}}$ input
1	WFR (Write FIFO Ready)	· The write FIFO is not full · Abort · $\overline{\text{RES}}$ input	· The write FIFO is full
0	WFE (Write FIFO Empty)	· The write FIFO is empty · Abort · $\overline{\text{RES}}$ input	· Write data into the write FIFO

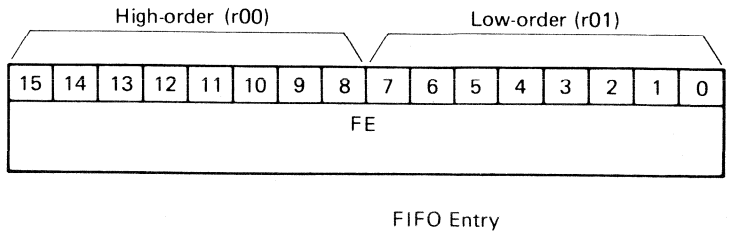


Figure 5.4 FIFO Entry (FE)

When the AR contains the FIFO Entry address (r00), reads and writes to the ACRTC ($\overline{CS}$ = low, RS = high) utilize the corresponding 16 byte read or write FIFOs.

In 16 bit bus mode, 16 bit words are written and read to/from the appropriate FIFO. In 8 bit bus mode, FIFO writes are in the order of high byte-low byte, while FIFO reads are in the order of high byte-low byte.

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

Note) The data should be written or read to/from the FIFO, after recognizing the status register flag (except the DMA processing group).
The Write FIFO and Read FIFO are completely separated.

5.5 Command Control Register (CCR: r02-r03)

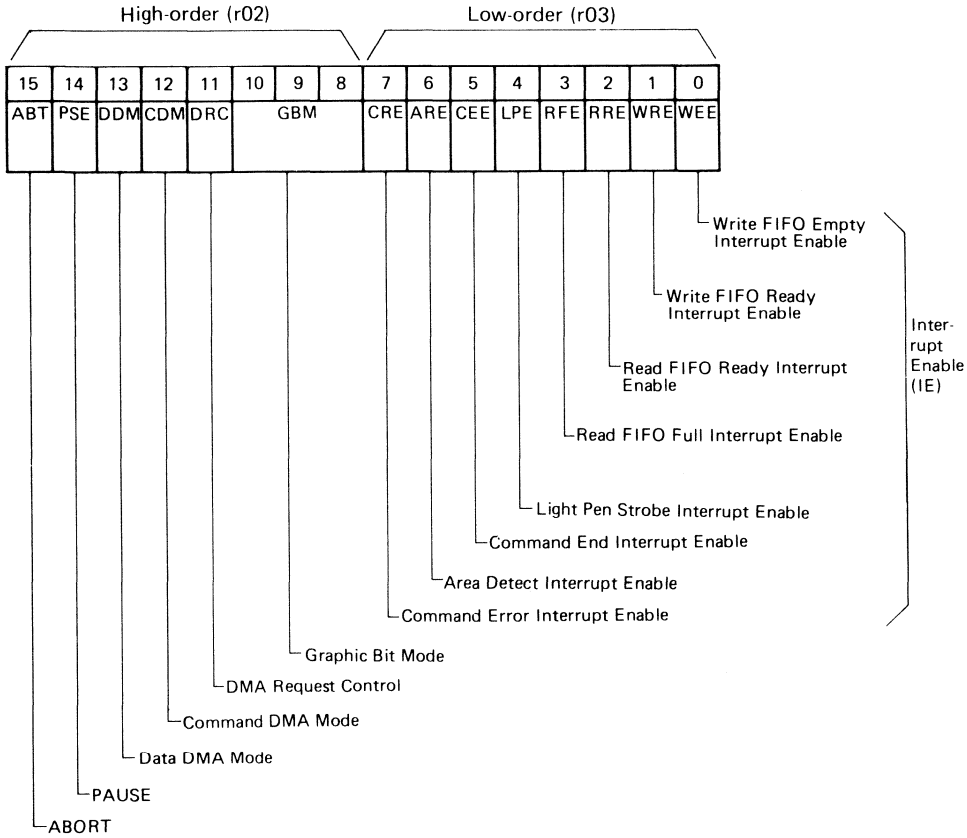


Figure 5.5 Command Control Register

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

When $\overline{RES}$ is asserted, the ABT (abort) bit is initialized to 1 and all other CCR bits are initialized to 0. ($8000)

○ Abort (ABT: bit 15)

ABT	Functions
0	ACRTC command execution is enabled. When ABT is changed from 0 to 1, the ACRTC cannot access the FIFOs.
1	ACRTC command execution is aborted and the read/write FIFOs are cleared. The status register (SR) is set to $23. But ACRTC display operation continue.

Note) When the ABT bit is set to "1" during command execution, the origin point may be shift from the initial point. In this case, after the ABT bit is reset to "0", The ORG Command should be issued again.

○ Pause (PSE: bit 14)

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

Note) As for the R mask version, "0" must always be specified because PSE doesn't function here.

○ Data DMA Mode (DDM: bit 13)

DDM	Functions
0	Data DMA transfer mode is disabled. $\overline{\text{DREQ}}$ is not asserted even if the MPU issues DMA transfer commands.
1	Data DMA transfer mode is enabled. Whether DMA is burst or cycle steal mode is determined by DRC (bit 11 this register). DDM must be set before DMA data transfer commands are issued.

Note) MPU must not access ACRTC FIFOs during cycle steal transfer.
DDM is only for specifying $\overline{\text{DREQ}}$ output inhibition/permission. DDM set to 0 does not change the poolarity and the meaning of the external signal or the internal processing procedure etc.

○ Command DMA Mode (CDM: bit 12)

CDM	Functions
0	Command/Parameter DMA transfer mode is disabled. Commands and parameters are issued under MPU program control.
1	Command/Parameter DMA transfer mode is enabled. Command/Parameter DMA mode is terminated when (a) the MPU resets CDM to 0 or (b) the $\overline{\text{DONE}}$ input is asserted (which also resets CDM to 0).

Note) (a) Command/Parameter DMA transfers use cycle stealing DMA regardless of the state of DRC (bit 11 this register).

(b) Data DMA transfer commands cannot be issued using Command/Parameter DMA. Insure that the Commands and Parameters transferred by Command/Parameter DMA do not include DMA data transfer commands.

(c) In the ACRTC R Mask and S Mask version, it doesn't use the Command/Parameter DMA. So CDM should be set to "0".

○ DMA Request Control (DRC: bit 11)

DRC	Functions
0	Burst Mode: $\overline{\text{DREQ}}$ is designated as a level signal (burst mode). DRC = 0 is only valid for data DMA transfer commands. Burst mode can only be used for Data DMA. $\overline{\text{DREQ}}$ is output at every RFF set in reading, and at every WFR set in writing. $\overline{\text{DREQ}}$ output is suspended when the reading data is emptied in the read FIFO or the written data is filled in the write FIFO.
1	Cycle Steal Mode: $\overline{\text{DREQ}}$ is designated as a pulse signal (cycle steal mode). $\overline{\text{DREQ}}$ is output once for each word (16 bit data bus mode) or once for each byte (8 bit data bus mode) transfer. $\overline{\text{DREQ}}$ is output at RFR set in reading, and at WFR in writing. MPU can access the ACRTC, when the DMAC timing and the ACRTC timing differ or the reading data is emptied in the read FIFO or the written data is filled in the write FIFO.

○ Graphic Bit Mode (GBM: bit 10 - bit 8)

GBM defines the number of physical bits of frame buffer memory associated with a logical pixel. 1 bit per pixel is monochrome, while 16 bits per pixel allows a logical pixel to assume 1 of 64K possible colors or tones.

GBM			Mode	Number of colors displayed per pixel	Pixels/16 bit word
10	9	8			
0	0	0	1 bit/pixel	1	16
0	0	1	2 bits/pixel	4	8
0	1	0	4 bits/pixel	16	4
0	1	1	8 bits/pixel	256	2
1	0	0	16 bits/pixel	64K	1
	•		←----------INVALID----------→		
	•				
	•				
1	1	1			

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

An $\overline{IRQ}$ 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 to enable 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	LPSTB 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

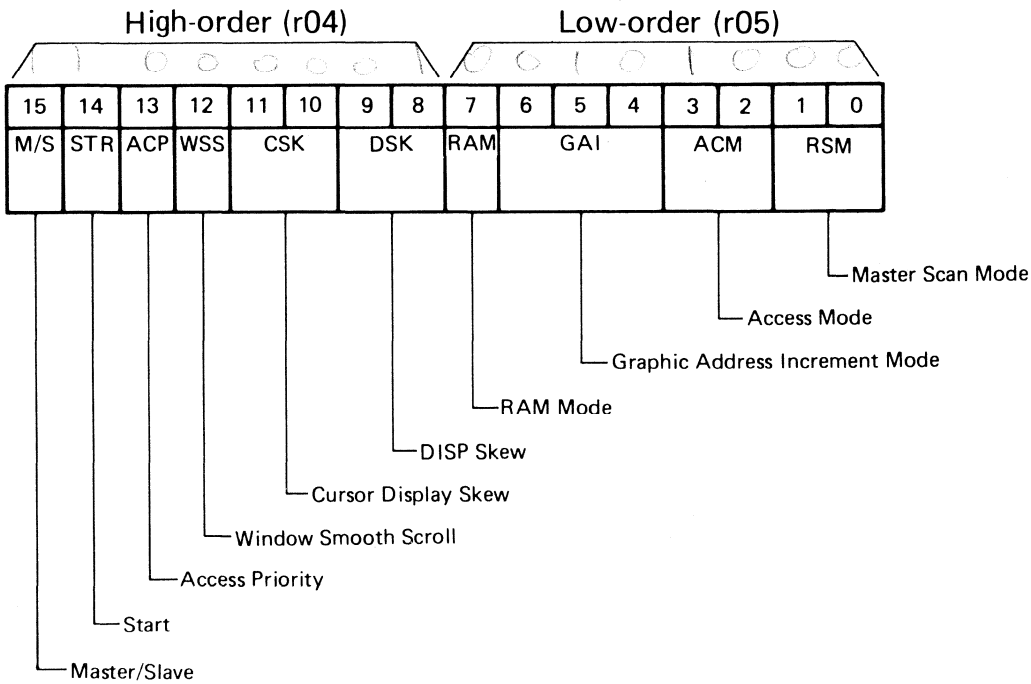


Figure 5.6 Operation Mode Register (OMR)

OMR determines major operating parameters and modes of the ACRTC. The 2 most significant bits (M/S and STR) are reset to 0 and all other bits are unaffected by $\overline{RES}$.

○ 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 is reset to 0 during $\overline{RES}$. When a single ACRTC is used, M/S should be set to 1 and the $\overline{EXSYNC}$ pin left open.

M/S	Functions
0	Slave Mode: EXSYNC is defined as an input. ACRTC internal operations are reset on the rising edge of the $\overline{EXSYNC}$ input. For non-interlace modes, the masters $\overline{VSYNC}$ should be connected to the $\overline{EXSYNC}$ input. For interlaced modes, the $\overline{VSYNC}$ of the masters odd field should be connected to the $\overline{EXSYNC}$ input. In the specific case of multiple ACRTC synchronization, the master and all slaves ACRTCs $\overline{EXSYNC}$ pins should be connected independent of interlace mode.
1	Master Mode: $\overline{EXSYNC}$ is defined as an output. For non-interlace modes, the $\overline{EXSYNC}$ output timing is the same as $\overline{VSYNC}$ output timing. For interlace modes, the $\overline{EXSYNC}$ output timing is generated by the $\overline{VSYNC}$ output for the odd field.

Note) $\overline{HSYNC}$ and $\overline{VSYNC}$ are always outputs regardless of the state of the M/S bit.
In the slave mode, when $\overline{EXSYNC}$ is input at the every frame, Cursor Blink and Attribute Blink are operated correctly. In this case, these should be blinked by the external circuit or the ACRTC in the master mode.

○ Start (STR: bit 14)

The STR bit is used to start and stop ACRTC operation. STR is reset to 0 by ACRTC hardware $\overline{RES}$. Initializing of registers which control basic ACRTC operation should only be performed when STR is reset to 0.

STR	Functions
0	ACRTC display control and drawing operations are halted. $\overline{DISP}$, $\overline{CUD}$, $\overline{VSYNC}$, etc. go to the inactive high level. $\overline{HSYNC}$ is set to low level, and the DRAM refresh address is output on the MAD lines regardless of the state of the RAM mode bit (bit 7 of this register). 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.

○ Drawing Access Priority (ACP: bit 13)
ACP determines whether or not the ACRTC executes drawing operations on the frame buffer during the display refresh period.

ACP	Functions
0	Display priority mode: During the display period, the ACRTC halts drawing operations. thus, flashing due to simultaneous display and drawing access of the frame buffer is eliminated. Drawing operations are performed during horizontal and vertical retrace. If DRAM refresh mode is enabled (RAM bit is reset to 0) drawing is inhibited during the DRAM refresh period. In Interleaved Access Mode drawing can occur simultaneously with display, without 'flashing', since drawing and display access to the frame buffer is interleaved. In Superimposed Access Mode, flashless Background screen drawing may occur during idle Window display cycles.
1	Drawing priority mode: Drawing is performed during the display period. To reduce the 'flashing' effect caused by drawing-display contention the ACRTC may be programmed to drive the $\overline{DISP}$ signals to the inactive high level during drawing operations. If the RAM bit is reset to 0 (DRAM refresh mode), drawing is inhibited during the DRAM refresh period. If the RAM bit is set to 1 (Static RAM mode), drawing is also performed during the $\overline{HSYNC}$ low time.

Note) Since the last cycle of $\overline{HSYNC}$ low time is used as a video attribute output period, this cycle is never used for drawing regardless of the state of ACP and RAM bits.

○ Window Smooth Scroll (WSS: bit 12)

WSS determines whether horizontal smooth scroll is applied to the Window screen. Window smooth scroll is only available in the Superimposed access mode. Therefore, if the Window screen is disabled, or the access mode is Single or Interleaved, WSS must be reset to 0. The horizontal smooth scroll is implemented by using four bits of SDA (Start Dot Address) programmed in the Window Start Address Register (SAR3). These bits are output on MAD12-MAD15 during the video attribute output period (last cycle of $\overline{\text{HSYNC}}$ low) and are used to control an external circuit which modifies the parallel to serial converter (shift register) timing.

WSS	Functions
0	Horizontal smooth scroll is not performed for the Window screen. One cycle Window screen prefetch does not occur.
1	Horizontal smooth scroll is performed for the Window screen. The Window display refresh cycle starts one cycle earlier than programmed in the Horizontal Window Register (HWR) Horizontal Display Start (HDS) field.

○ Cursor Display Skew (CSK: bit 11 - bit 10)

CSK defines the delay time for $\overline{\text{CUD1}}$ and $\overline{\text{CUD2}}$ in units of memory cycle independent of frame buffer access mode (i.e. Single, Interleaved or Superimposed). The $\overline{\text{CUD1}}$ and $\overline{\text{CUD2}}$ skew allows compensating for delays due to frame buffer memory, character generator or other external logic access time.
In the Crosshair cursor mode, CSK = 00 should not be used.

CSK		Functions
11	10	
0	0	No skew. $\overline{\text{CUD2}}$ output is always high.
0	1	$\overline{\text{CUD1}}$, $\overline{\text{CUD2}}$ are skewed by one memory cycle.
1	0	$\overline{\text{CUD1}}$, $\overline{\text{CUD2}}$ are skewed by two memory cycles.
1	1	$\overline{\text{CUD1}}$, $\overline{\text{CUD2}}$ are skewed by three memory cycles.

○ DISP Skew (DSK: bit 9 - bit 8)

DSK defines the $\overline{\text{DISP1}}$, $\overline{\text{DISP2}}$ delay in units of memory cycle independent frame buffer access mode.

DSK		Functions
9	8	
0	0	No skew.
0	1	$\overline{\text{DISP1}}$, $\overline{\text{DISP2}}$ are skewed by one memory cycle.
1	0	$\overline{\text{DISP1}}$, $\overline{\text{DISP2}}$ are skewed by two memory cycles.
1	1	$\overline{\text{DISP1}}$, $\overline{\text{DISP2}}$ are skewed by three memory cycles.

○ RAM Mode (RAM: bit 7)

The RAM bit determines whether or not the ACRTC will place an 8 bit DRAM refresh address on the MAD outputs during $\overline{\text{HSYNC}}$ low. In this context, $\overline{\text{HSYNC}}$ low time is also referred to as the 'DRAM refresh period' except for the last cycle of $\overline{\text{HSYNC}}$ low, which is referred to as the 'Attribute output period'. The refresh addressing mechanism is compatible with standard 16K, 64K and 256K bit DRAMs.

RAM	Functions
0	Dynamic RAM mode: During the DRAM refresh period, the ACRTC outputs the 8 bit refresh address on MAD. Note that the particular MAD lines used for the DRAM refresh address are determined by the Gaphic Address Increment (GAI) mode. The DRAM refresh address is decremented by 1 every refresh cycle.
1	Static RAM mode: No DRAM refresh address is placed on MAD. Drawing is performed during the DRAM refresh period ($\overline{\text{HSYNC}}$ low - except the attribute output period) regardless of the Access Priority (ACP) definition.

Note) In the R Mask version, DRAM refresh address is not output during the Reset period.

○ Graphic Address Increment mode (GAI: bit 6 - bit 4)

As described earlier, using the Gaphic Bit Mode field in the Command Control Register (GBM in CCR), the number of physical frame buffer bits associated with a logical pixel can be selected as 1, 2, 4, 8 or 16.

However, when the frame buffer organization is fixed as 16 bit words, if 1 bit per pixel GBM is specified, each word contains 16 logical pixels. If 4 bits per pixel GBM is specified, each word contains only 4 logical pixels. thus, a '16 color' display compared to a monochrome display will require a 2CLK input which is 4 times faster to achieve the same logical pixel resolution.

A simple technique for solving this problem is to increase the number of frame buffer bits output for each display cycle. In the above example, if 4 words (64 bits) of frame buffer are accessed each display refresh cycle, the 'color' system 2CLK input is the same frequency as the 'monochrome' system which has equivalent logical pixel resolution.

GAI accomodates this technique and other special cases by modifying the frame buffer address increment used for each successive graphic screen display access.

GAI allows the display address increment to be 1, 2, 4, 8 or 16 words (16-256 bits), 0 increment (display constant pattern) and increment every two display cycles (used when superimposing screens character and graphic screens).

Note 1) GAI applies only to graphic screen display accesses. Graphic screen drawing accesses and character screen accesses used a fixed increment of 1 word.

GAI			Functions
6	5	4	
0	0	0	Graphic screen display address incremented by 1 every display cycle.
0	0	1	Graphic screen display address incremented by 2 every display cycle.
0	1	0	Graphic screen display address incremented by 4 every display cycle.
0	1	1	Graphic screen display address incremented by 8 every display cycle.
1	0	0	Graphic screen display address incremented by 16 every display cycle. Note 1)
1	0	1	Graphic screen display address not incremented.
1	1	0	Graphic screen display address incremented by 1 every two display cycles.
1	1	1	

Note 1) It is not available for the R mask version.
In the R mask version, when GAI is set to 100, graphic screen display address is not incremented.

○ Access Mode (ACM: bit 3 - bit 2)
The ACRTC provides three frame buffer access modes — Single, Interleaved and Superimposed.

ACM		Functions
3	2	
0	x	Single Access Mode: The frame buffer is accessed once every display cycle. The Window screen access has higher priority than overlapped Background screen accesses. When ACP = 0 (display priority mode), drawing is not performed during the display period.
1	0	Interleaved Access Mode (Dual Access Mode 0): The frame buffer is accessed twice every display cycle. Display and drawing cycles are interleaved during each phase of the display cycle. Even if ACP = 0 (Display priority mode), 'flashless' drawing will occur during display period. The Window screen has highest priority as in Single Access Mode.
1	1	Superimposed Access Mode (Dual Access Mode 1): The frame buffer is accessed twice every display cycle. The first phase accesses the Background screen, the second phase accesses the Window screen. In this case the Background and Window screens have equal priority, and are superimposed. Drawing is performed during the second phase in which the Window screen is not being displayed even when ACP = 0.

x = Don't care

Note) 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.

RSM		Functions
1	0	
0	0	Non-Interlace Mode
0	1	
1	0	Interlace Sync Mode
1	1	Interlace Sync & Video Mode

5.7 Display Control Register (DCR: r06-r07)

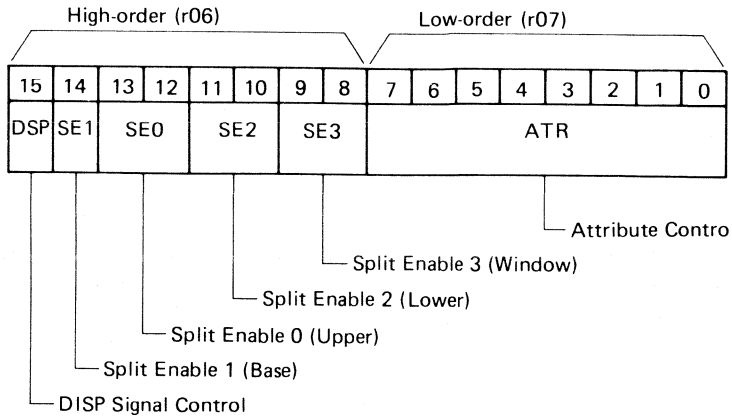


Figure 5.7 Display Control Register (DCR)

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

Logically, the ACRTC has a Background screen (Upper, Base and Lower split screens) and a Window screen. When overlapping occurs, either the Window screen has priority (Single Access Mode, Interleaved Access Mode) or the Window screen and the Background screen have equal priority (Superimposed Access Mode).

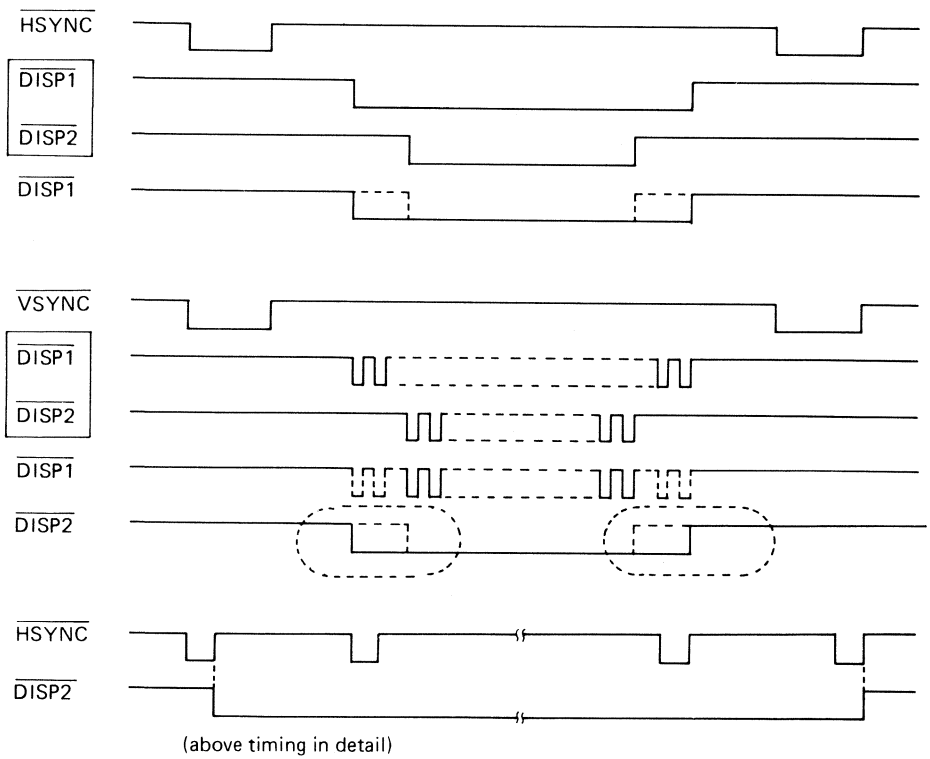
DCR allows screens to be enabled, disabled and blanked. If the Upper, Lower and Window screens are disabled, they need not be defined. The Base screen must always be defined. When screens are blanked ($\overline{\text{DISP}}$ timing output held inactive high), the display address is also inhibited. The ACRTC uses the idle frame buffer bus (MAD0-15 and MA16-19) for drawing operations.

○ $\overline{\text{DISP}}$ Signal Control (DSP: bit 15)

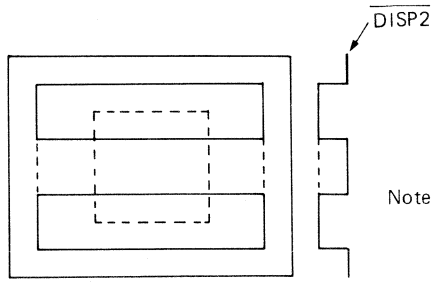
DSP defines the output mode of the $\overline{\text{DISP1}}$ and $\overline{\text{DISP2}}$ display timing signals.

DSP	Functions
0	$\overline{\text{DISP1}}$ is driven active low during the display period of the Background screen (combined horizontal and vertical display). $\overline{\text{DISP2}}$ is controlled similarly for the Window screen.
1	$\overline{\text{DISP1}}$ is driven active low during the horizontal display of both the Background and Window screens. $\overline{\text{DISP2}}$ is driven active low during the vertical display period of both the Background and Window screens. Thus, $\overline{\text{DISP2}}$ is high during vertical retrace. This allows another device which shares direct access to the frame buffer with the ACRTC to determine when the frame buffer is available.

When specified "DSP = 0", the signals within [ ] are output.



(above timing in detail)



Note: This is the $\overline{DISP2}$ output example, when the split screen 1 display is inhibited. The output is not negated while the window is displayed.

○ Split Enable 1 (SE1: bit 14)

SE1 allows the Base screen (screen 1) to be blanked. Drawing can occur when the Base screen is blanked since frame buffer display access is suppressed. Note that the Base screen parameters must be defined, even if the Base screen is always blanked.

SE1	Functions
0	The ACRTC inhibits the display enable timing ($\overline{DISP1}$ and/or $\overline{DISP2}$) and 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 (screen 0) to be enabled, disabled and blanked. If always disabled, the Upper screen parameters need not be defined. When the Upper screen is blanked, drawing may occur since frame buffer display access is suppressed.

Note) Specify the memory width register (RC2, RC3) for the split screen 0 in spite of the SE0 specification when specifying the screen number 0 for ORG.

SE0		Functions
13	12	
0	x	The ACRTC disables the Upper screen. Therefore, the Background screen contains two parts maximum — the Base and Lower screens. The Base screen is moved upward by the number of rasters in the disabled Upper screen.
1	0	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 CRT screen.
1	1	The ACRTC outputs display enable timing and display addresses for the Upper screen.

x = Don't care

○ Split Enable 2 (SE2: bit 11 - bit 10)

SE2 allows the Lower split screen (screen 2) to be enabled, disabled and blanked. If always disabled, the Lower screen parameters need not be defined. When the Lower screen is blanked, drawing may occur since frame buffer display access is suppressed.

Note) Specify the memory width register (RD2, RD3) for the split screen 2 in spite of the SE2 specification when specifying the screen number 2 for ORG.

SE2		Functions
11	10	
0	x	The ACRTC disables the Lower screen. Therefore, the Background screen contains two parts maximum — the Base and Upper screens.
1	0	The display enable timing and display address outputs are inhibited for the Lower screen. The area of the Lower screen, though blanked, remains on the CRT screen.
1	1	The ACRTC outputs display enable timing and display addresses for the Lower screen.

x = Don't care

○ Split Enable 3 (SE3: bit 9 - bit 8)

SE3 allows enabling, disabling and blanking of the Window screen (screen 3). When disabled or blanked, the overlapped Background screens are displayed.

SE3		Functions
9	8	
0	x	The ACRTC disables the Window screen and overlapped Background screens (as defined by SE0, SE1 and SE2) are displayed. If always disabled, the Window screen parameters need not be defined. For Superimposed access mode the second (Window) phase of the display cycle is not used. The ACRTC may execute drawing operations during this second phase.
1	0	The ACRTC disables the display enable timing and display address outputs for the Window screen. The area of the Window screen, though blanked, remains on the CRT. However, Window screen parameters must be defined. For superimposed access modes, the overlapped Background screens are displayed. For Single and Interleaved access modes, the ACRTC may perform drawing during the display time for the blanked Window screen.
1	1	The ACRTC outputs the display enable timing and display addresses for the Window screen.

x = Don't care

○ Attribute Control (ATR: bit 7 - bit 0)

These 8 bits can be freely programmed as user defined video attributes. These bits are output on MAD7 – MAD0 prior to the rising edge of $\overline{\text{HSYNC}}$.

When programmed dynamically, ATR allows video attributes to be controlled on a raster by raster basis.

5.8 Timing Control RAM (r80-9F)

These registers are used to define the overall screen and CRT timing signal characteristics, and parameters associated with the Base, Upper, Lower and Window screens.

Raster Count Register (RCR)
Horizontal Sync Register (HSR)
Horizontal Display Register (HDR)
Horizontal Window Register (HWR)
Vertical Sync Register (VSR)
Vertical Display Register (VDR)
Split Screen Width Register (SSW)
Vertical Window Display Register (VWR)
Blink Control Register (BCR)
Graphic Cursor Register (GCR)

5.8.1 Raster Count Register (RCR: r80-r81)

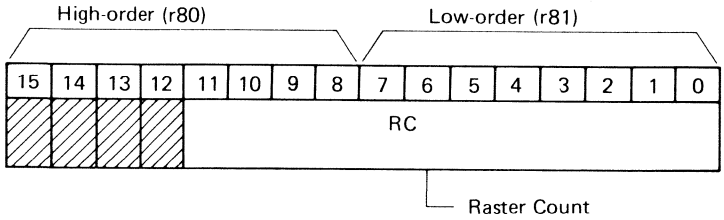


Figure 5.8 Raster Count Register (RCR)

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 $\overline{RES}$ is undefined. If RCR read operation is desired, the HSW (Horizontal Sync Width) should be set greater than or equal to 3. RCR should only be read when $\overline{HSYNC}$ is high.

The high order 4 bits of RCR are always 0.

RCR is updated depending on the ACRTC raster scan modes as shown.

Scan Mode	Functions
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 odd fields. Because a dummy raster is added to the even field, the maximum raster number for the even field is one greater than that for the odd field.
Interlace Sync and Video	RCR starts counting at 0 in the even field and at 1 in the odd field, and increments 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 the Interlace Sync Mode.

5.8.2 Horizontal Sync Register (HSR: r82-r83)

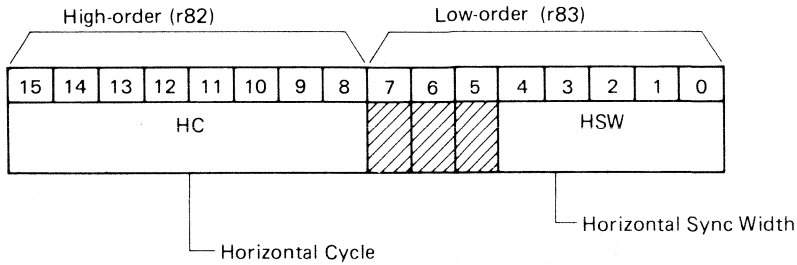


Figure 5.9 Horizontal Sync Register (HSR)

HSR defines the Horizontal Cycle (HC) and 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. 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.

H C MSB LSB	Display (Memory cycle)
0 0 0 0 0 0 0 0	1
0 0 0 0 0 0 0 1	2
(	(
)	)
1 1 1 1 1 1 1 0	255
1 1 1 1 1 1 1 1	256

○ Horizontal Sync Width (HSW: bit 4 - bit 0)

HSW specifies the $\overline{\text{HSYNC}}$ active low time in unis 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. When the ACRTC DRAM refresh feature is used, DRAM refresh timing should be factored into the choice of HSW.

H S W MSB LSB	Pulse width (Memory cycle)
0 0 0 0 0	*1
0 0 0 0 1	*2
0 0 0 1 0	2
0 0 0 1 1	3
⟨	⟨
1 1 1 1 0	30
1 1 1 1 1	31

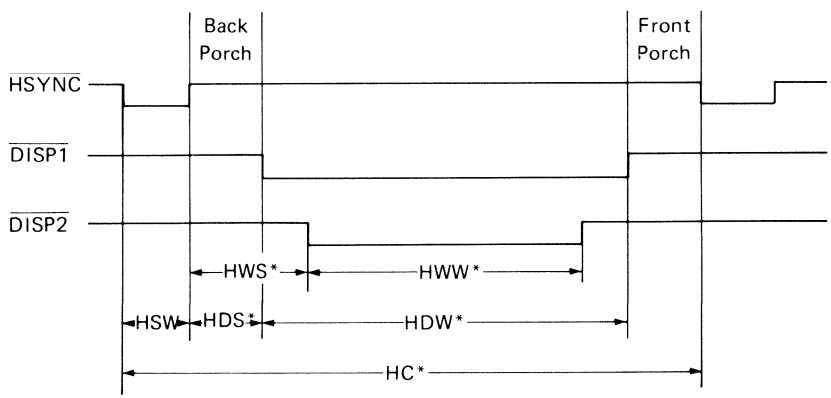
*1 Not used.

*2 Two memory cycles are assummed.

NOTES ON THE CONTROL REGISTER SETUP

Horizontal Synchronization Signal (r82~r85, r92~r93)

Note) *Setup value = Specified value − 1



Horizontal Front Porch

Raster Scan Mode	Horizontal Front Porch
Non-Interlace, Interlace Sync.	5 or more
Interlace Sync. & Video	6 or more

(Unit: Memory Cycle)

Horizontal Back Porch + Horizontal Synchronization Pulse Width (HSW + HDS*)

Raster Scan Mode	Zooming-up Display	Screen	Horizontal Synchronization Pulse Width + Horizontal Back Porch
Non-Interlace, Interlace Sync.	Not Performed	Graphic	6 or more
		Character	7 or more
	Performed	Graphic	7 or more
		Character	8 or more
Interlace Sync. & Video	Not Performed	Graphic	8 or more
		Character	9 or more
	Perfomede	Graphic	9 or more
		Character	10 or more

(Unit: Memory Cycle)

Note) In dual access mode (Interleave, Superimpose mode), the above value needs to be incremented by + 1.

Register Name	Specified Value		Unit
H C*	1~256	Note 2), 8)	Memory Cycle
HSW	2~31	Note 2), 3)	Memory Cycle
HDS*	2~256	Note 2), 4), 5)	Memory Cycle
HDW*	2~256	Note 2), 4), 6)	Memory Cycle
HWS*	HDS*~HDS* + HDW*	Note 5)	Memory Cycle
HWW*	2~HDW*	Note 6), 7)	Memory Cycle

Note 1) In *-marked registers,

Setup value = Specified value − 1

Note 2) HC* ≥ HSW + HDS* + HDW* + 5
(At the non-interlace/interlace mode setup)
HC* ≥ HSW + HDS* + HDW* + 6
(At the interface sync. & video mode setup)

Note 3) 1 cannot be set to HSW. When the raster counter (RCR) is used, the setup value is to be 3 or more.

Note 4) 1 cannot be set to HDS*, HDW*.

Note 5) When setting the dual access mode, the following rule must be kept.

HDS* is even.	→ HWS* is to be even.
HDS* is odd.	→ HWS* is to be odd.

Note 6) When setting the dual access mode, an even number must be specified.

Note 7) HWW* ≤ HDS* + HDW* − HWS*

Note 8) HC* > HSW × 2 + 1

5.8.3 Horizontal Display Register (HDR: r84-r85)
Horizontal Window Display Register (HWR: r92-r93)

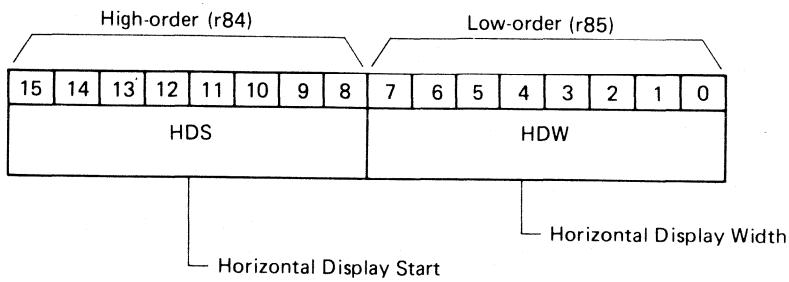


Figure 5.10 Horizontal Display Register (HDR)

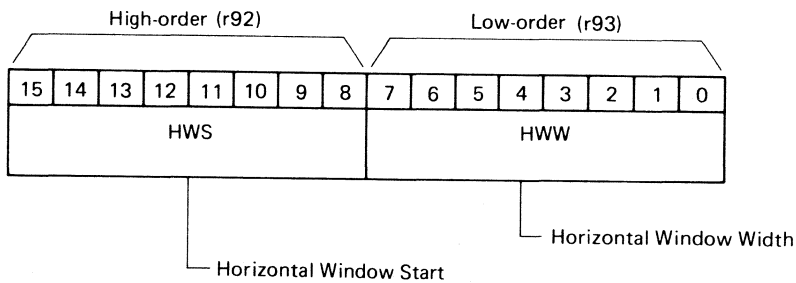


Figure 5.11 Horizontal Window Display Register (HWR)

HDR specifies the horizontal display start position and horizontal display width in units of memory cycles.

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 $\overline{\text{HSYNC}}$ (Horizontal Front Porch) 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 $\overline{\text{HSYNC}}$ and the horizontal Window display starting point in units of memory cycles. If the Horizontal Window Start is HS memory cycles, HWS should be set to HS-1.

HDS/HWS MSB LSB	Display width (Memory cycle)
0 0 0 0 0 0 0 0	*
0 0 0 0 0 0 0 1	2
⎛	⎛
⎝	⎝
1 1 1 1 1 1 1 0	255
1 1 1 1 1 1 1 1	256

* Not used

* Not used

Note) The address for the split screen is output from one display cycle earlier than the setted HDS, (during the background screen display).

○ 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)
HWW defines the Window display period for one raster in units of memory cycles. If the Horizontal Window Width is HW memory cycles, HWW should be set to HW-1.

HDW/HWW MSB LSB	Display width (Memory cycle)
0 0 0 0 0 0 0 0	*
0 0 0 0 0 0 0 1	2
⎛	⎛
⎝	⎝
1 1 1 1 1 1 1 0	255
1 1 1 1 1 1 1 1	256

* Not used

5.8.4 Vertical Sync Register (VSR: r86-r87)

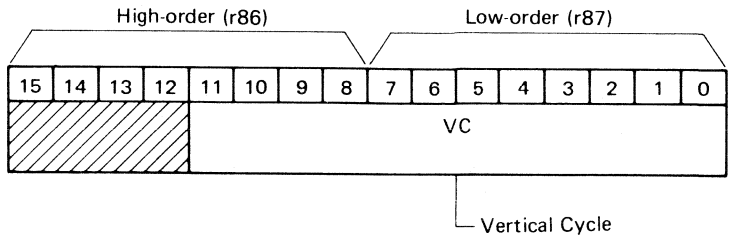


Figure 5.12 Vertical Sync Register (VSR)

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 in one frame is V, VC is set to V.
- Interlace Sync Mode
  When the number of rasters in one field (even or odd) 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.

V C MSB LSB	Vertical cycle (Number of rasters)
0 0 0 0 0 0 0 0 0 0 0 0	*
0 0 0 0 0 0 0 0 0 0 0 1	1
0 0 0 0 0 0 0 0 0 0 1 0	2
⎛ ⎞	⎛ ⎞
1 1 1 1 1 1 1 1 1 1 1 0	4094
1 1 1 1 1 1 1 1 1 1 1 1	4095

* Not used

5.8.5 Vertical Display Register (VDR: r88-r89)

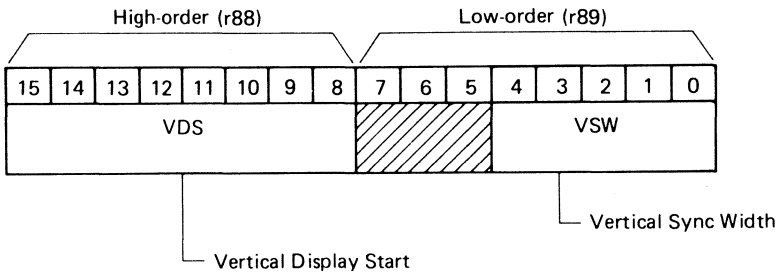


Figure 5.13 Vertical Display Register (VDR)

VDR defines vertical sync width ($\overline{\text{VSYNC}}$ low period) and vertical display start and width in units of rasters.

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

VSW defines $\overline{\text{VSYNC}}$ low pulse width in units of rasters. VSW is set depending on the CRT display device specification. VSW should be set to a non-zero value.

V S W MSB LSB	Pulse width (Number of raster)
0 0 0 0 0	*
0 0 0 0 1	1
0 0 0 1 0	2
⟨	⟨
1 1 1 1 0	30
1 1 1 1 1	31

* Not used

○ Vertical Display Start (VDS: r88)

VDS defines the period from the rising edge of $\overline{\text{VSYNC}}$ 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).

V D S MSB LSB	Display start (Number of rasters)
0 0 0 0 0 0 0 0	*
0 0 0 0 0 0 0 1	1
⟨	⟨
1 1 1 1 1 1 1 0	254
1 1 1 1 1 1 1 1	255

* Not used

5.8.6 Vertical Window Display Register (VWR: r94-r97)

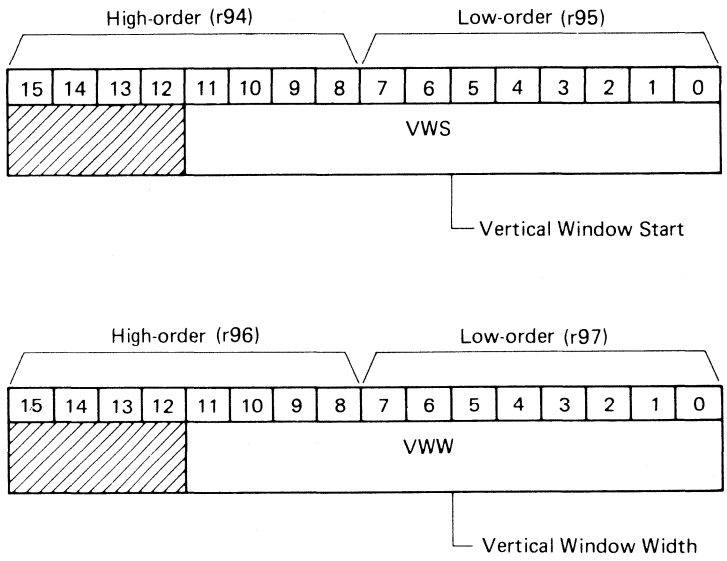


Figure 5.14 Vertical Window Display Register (VWR)

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 $\overline{\text{VSYNC}}$ 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 than or equal to VDS.

V W S MSB LSB	Display start position (Number of rasters)
0 0 0 0 0 0 0 0 0 0 0 0	*
0 0 0 0 0 0 0 0 0 0 0 1	1
(	(
)	)
1 1 1 1 1 1 1 1 1 1 1 0	4094
1 1 1 1 1 1 1 1 1 1 1 1	4095

* Not used

○ 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.

V W W MSB LSB	Display width (Number of rasters)
0 0 0 0 0 0 0 0 0 0 0 0	*
0 0 0 0 0 0 0 0 0 0 0 1	1
0 0 0 0 0 0 0 0 0 0 1 0	2
( )	( )
1 1 1 1 1 1 1 1 1 1 1 0	4094
1 1 1 1 1 1 1 1 1 1 1 1	4095

* Not used

5.8.7 Split Screen Width Register (SSW: r8A-r8F)

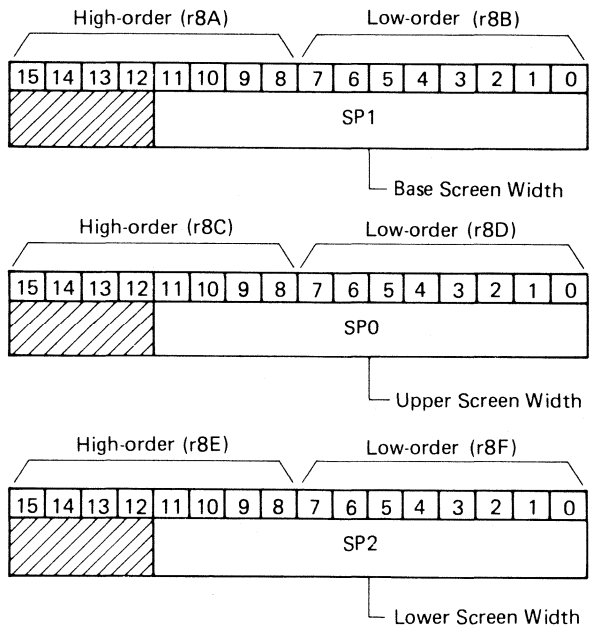


Figure 5.15 Split Screen Width Register (SSW)

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

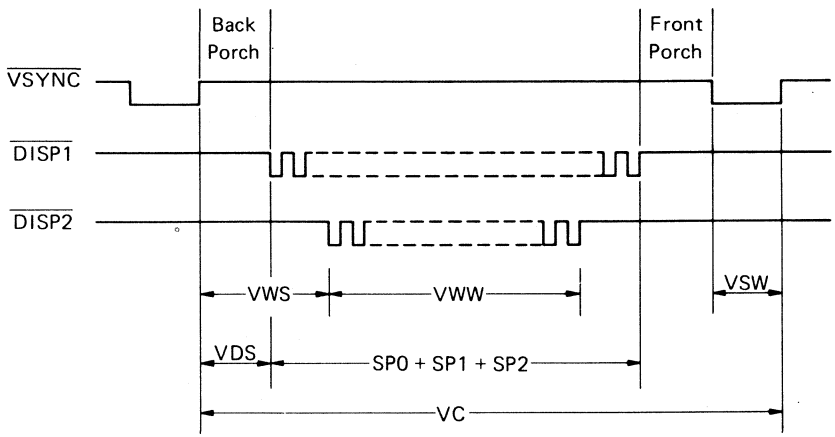
○ Split Screen Width (SP0: r8C-r8D bit 11 - bit 0)
(SP1: r8A-r8B bit 11 - bit 0)
(SP2: r8E-r8F bit 11 - bit 0)

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.

SP0/SP1/SP2 MSB LSB	Display width (Number of rasters)
0 0 0 0 0 0 0 0 0 0 0 0	*
0 0 0 0 0 0 0 0 0 0 0 1	1
0 0 0 0 0 0 0 0 0 0 1 0	2
⟨	⟨
1 1 1 1 1 1 1 1 1 1 1 0	4094
1 1 1 1 1 1 1 1 1 1 1 1	4095

* Not used

Vertical Synchronization Signal (r86~r8F, r94~r97)



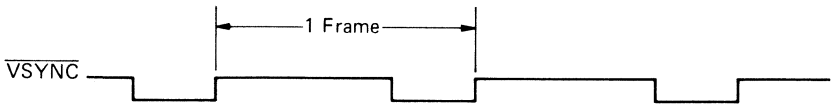
Register Name	Specified Value			Unit
VC	1~4095		Note 2)	Raster
VSW	1~31		Note 3)	Raster
VDS Note 1)	Non-Interlace/ Interlace Sync. Mode	1~255	Note 4)	Raster
	Interlace Sync. & Video Mode	4~255	Note 4)	
SP1 Note 1)	Non-Interlace/ Interlace Sync. Mode	2~4095	Note 5), 6)	Raster
	Interlace Sync. & Video Mode	4~4095	Note 5), 6)	
VDW Note 1)	Non-Interlace/ Interlace Sync. Mode	2~4095	Note 5)	Raster
	Interlace Sync. & Video Mode	4~4095	Note 5)	
SP2 Note 1)	Non-Interlace/ Interlace Sync. Mode	2~4095	Note 5), 6)	Raster
	Interlace Sync. & Video Mode	4~4095	Note 5), 6)	
VWS Note 1)	VDS ~ SP0 + SP1 + SP2		Note 4)	Raster
VWW Note 1)	Non-Interlace/ Interlace Sync. Mode	2~SP0 +SP1 + SP2	Note 5), 7)	Raster
	Interlace Sync. & Video Mode	4~SP0 +SP1 + SP2	Note 5), 7)	

Note 1) 0 cannot be specified.

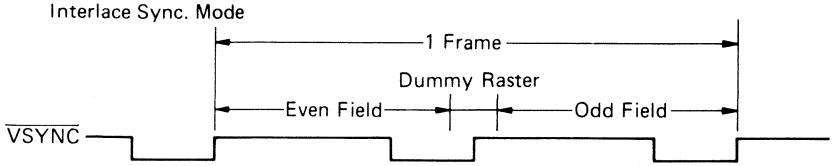
Note 2) 0 cannot be specified. The setup value varies as shown below according to the raster scan mode.

- Total raster number scanned in 1 frame [Non-Interlace Mode]
- Total raster number scanned in 1 field [Interlace Sync. Mode]
  (The dummy rasters between the odd/even number fields are not included.)
- Total raster number scanned in 1 frame [Interlace Sync. & Video Mode]
  (The dummy rasters between the odd/even number field are included, and become odd numbers.)

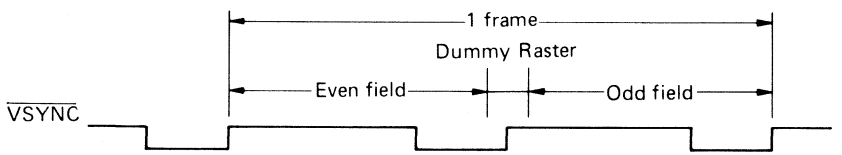
Non-Interlace Mode



Interlace Sync. Mode



Interlace Sync. & Video Mode



$VC \geq VDS + SP0 + SP1 + SP2 + VSW$

(Non-interlace/Interlace sync. mode)

$VC \geq VDS + SP0 + SP1 + SP2 + 2 \times VSW + 1$

(Odd-number) (Interlace sync. & Video mode)

Note 3) 0 cannot be specified.

Note 4) The specified value must be more than 1 under non-interlace mode or interlace sync. mode, and more than 4 under interlace sync. & video mode.
The specified value must be the raster counts per 1 frame under non-interlace mode, the raster counts per 1 field under interlace sync. mode, and the total raster counts of even field and odd field under interlace sync & video mode.

Note 5) The specified value must be more than 2 under non-interlace or interlace sync. mode, and more than 4 under interlace sync. & video mode. The specification under each mode depends on the same restriction as described in Note 4).

Note 6) The specified value is regarded as 0, if the split screen is disabled.

Note 7) $VWW \leq VDS + SP0 + SP1 + SP2 - VWS$

5.8.8 Blink Control Register (BCR: r90-r91)

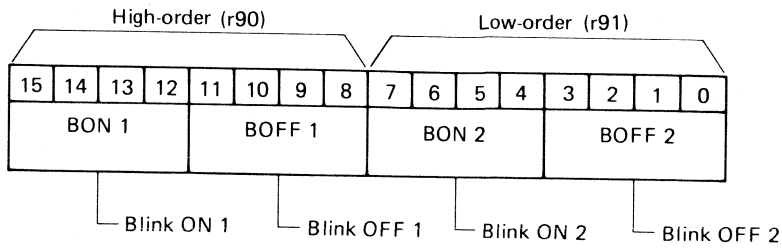


Figure 5.16 Blink Control Register (BCR)

BCR defines the blink on and off period for the BLINK1 and BLINK2 video attributes. BLINK1 and BLINK2 are output from MA18 and MA19 during each rasters video attribute output cycle.

○ **Blink ON (BON1: r90 bit 15 - bit 12)**
(BON2: 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.

BON1				Blink "High" level (Field)
15	14	13	12	
0	0	0	0	*
0	0	0	1	8
0	0	1	0	12
0	0	1	1	16
0	1	0	0	20
0	1	0	1	24
0	1	1	0	28
0	1	1	1	32
1	0	0	0	36
1	0	0	1	40
1	0	1	0	44
1	0	1	1	48
1	1	0	0	52
1	1	0	1	56
1	1	1	0	60
1	1	1	1	64

BON2				Blink "High" level (Field)
7	6	5	4	
0	0	0	0	*
0	0	0	1	8
0	0	1	0	12
0	0	1	1	16
0	1	0	0	20
0	1	0	1	24
0	1	1	0	28
0	1	1	1	32
1	0	0	0	36
1	0	0	1	40
1	0	1	0	44
1	0	1	1	48
1	1	0	0	52
1	1	0	1	56
1	1	1	0	60
1	1	1	1	64

* BLINK is always "Low".

○ Blink OFF (BOFF1: r90 bit 11 - bit 8)
(BOFF2: 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 and BOFF(1/2) = 0 are programmed.

BOFF1				Blink "Low" level (Field)
11	10	9	8	
0	0	0	0	*
0	0	0	1	8
0	0	1	0	12
0	0	1	1	16
0	1	0	0	20
0	1	0	1	24
0	1	1	0	28
0	1	1	1	32
1	0	0	0	36
1	0	0	1	40
1	0	1	0	44
1	0	1	1	48
1	1	0	0	52
1	1	0	1	56
1	1	1	0	60
1	1	1	1	64

BOFF2				Blink "Low" level (Field)
3	2	1	0	
0	0	0	0	*
0	0	0	1	8
0	0	1	0	12
0	0	1	1	16
0	1	0	0	20
0	1	0	1	24
0	1	1	0	28
0	1	1	1	32
1	0	0	0	36
1	0	0	1	40
1	0	1	0	44
1	0	1	1	48
1	1	0	0	52
1	1	0	1	56
1	1	1	0	60
1	1	1	1	64

* In the case of BON(1/2) ≠ 0, BLINK(1/2) will always become "HIGH" level.

5.8.9 Graphic Cursor Register (GCR: r98-r9D)

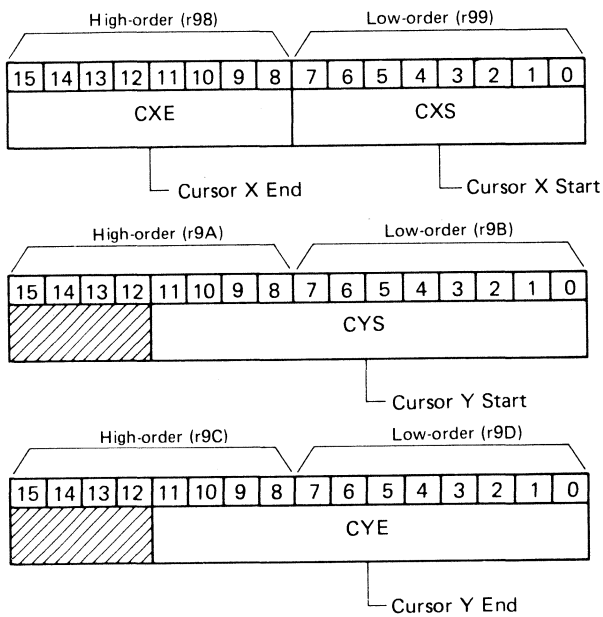
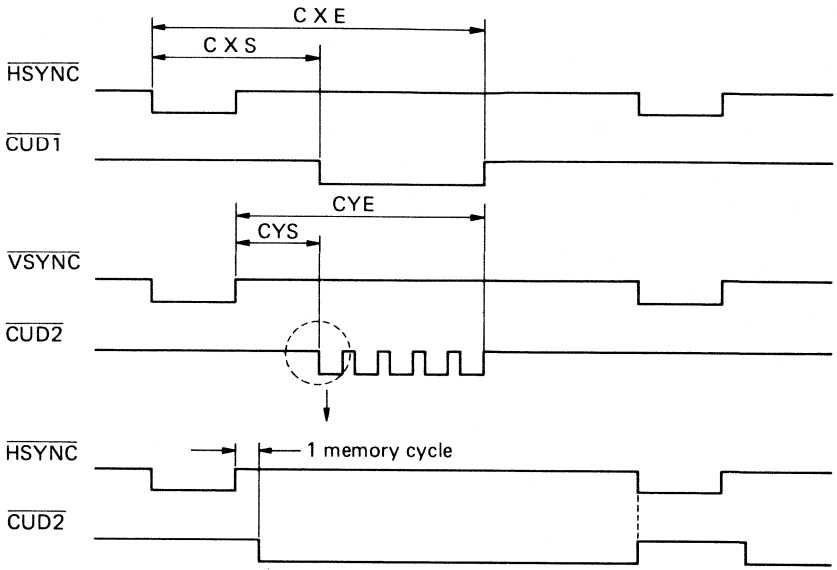


Figure 5.17 Graphic Cursor Register (GCR)

GCR defines the horizontal and vertical start and end positions for displaying a graphic cursor.

- **Cursor X Start (CXS: r99)**
  CXS defines the horizontal cursor start position from the falling edge of $\overline{\text{HSYNC}}$ in units of memory cycles.
- **Cursor X End (CXE: r98)**
  CXE defines the horizontal cursor end position from the falling edge of $\overline{\text{HSYNC}}$ in units of memory cycles.
- **Cursor Y Start (CYS: r9A, r9B bit 11 - bit 0)**
  CYS defines the vertical cursor start position from the rising edge of $\overline{\text{VSYNC}}$ in units of rasters.
- **Cursor Y End (CYE: r9C, r9D bit 11 - bit 0)**
  CYE defines the vertical cursor end position from the rising edge of $\overline{\text{VSYNC}}$ in units of rasters.

Graphic Cursor Signal (r98~r9D)



Note) This is an example timing under the crosshair cursor output mode. the output only from the $\overline{CUD1}$ is available under the block cursor output mode.

Register Name	Specified Value		Unit
CXS	HSW + 1~255	Note 2)	Memory Cycle
CXE*	CXS + 1~256	Note 3)	Memory Cycle
CYS	0~VC − VSW − 1	Note 4)	Raster
CYE*	CYS + 1~VC − VSW	Note 5)	Raster

Note 1) Setup value = Specified value − 1, as for the *-marked registers.

Note 2) The specified value must be more than 3. The cursor (X-direction component) is not output, if it is specified as below 3.

Note 3) The specified value must be more than 3. The cursor (X-direction component) is not output, if it is specified as "CXE* < CXS".

Note 4) The cursor (Y-direction component) is not output, if it is specified during the vertical synchronization period.

Note 5) The cursor (Y-direction component) is not output, if it is specified during the vertical synchronization period, and also if it is specified as CYE* < CYS.

5.8.10 ACRTC Working Register (r9E-9F)

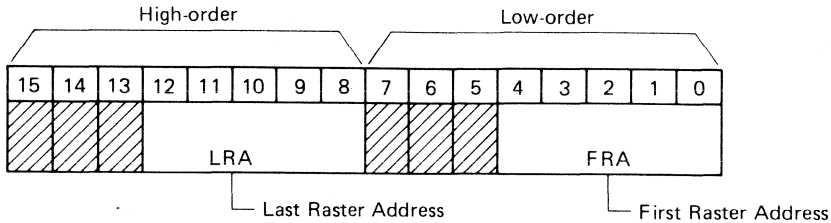
Internal ACRTC work area. The host MPU must never access this register.

5.9 Display Control RAM (rC0-rEF)

The Display Control RAM are registers containing parameters used by the ACRTC address generation logic. There are four sets of Raster Address, Memory Width and Start Address registers providing independent control for each of the four logical screens (Upper, Base, Lower and Window). Also, the Cursor Definition Register contains information for two separate cursors.

Raster Address Registers (RAR0-RAR3)
Memory Width Registers (MWR0-MWR3)
Start Address Registers (SAR0-SAR3)
Block Cursor Register (BCR)
Cursor Definition Register (CDR)
Zoom Factor Register (ZFR)
Light Pen Address Register (LPAR)

5.9.1 Raster Address Register
(RAR0: rC0-rC1) (RAR1: rC8-rC9)
(RAR2: rD0-rD1) (RAR3: rD8-rD9)



Raster address register 0 : LRA0 (rC0), FRA0 (rC1) Upper Screen
Raster address register 1 : LRA1 (rC8), FRA1 (rC9) Base Screen
Raster address register 2 : LRA2 (rD0), FRA2 (rD1) Lower Screen
Raster address register 3 : LRA3 (rD8), FRA3 (rD9) Window

Figure 5.18 Raster Address Register (RAR)

RAR specifies the raster addressing per character row (including line spacing) for character screens (CHR = high). RAR0-3 apply to screens 0-3, the Upper, Base, Lower and Window screens respectively.

○ First Raster Address (FRA: bit 4 - bit 0)
FRA determines the first raster line address of the character row, and can be set to any value between 0 and 31.

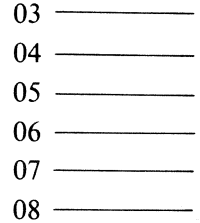
FRA					Raster address
4	3	2	1	0	
0	0	0	0	0	0
0	0	0	0	1	1
		⟨			⟨
1	1	1	1	0	30
1	1	1	1	1	31

○ Last Raster Address (LRA: bit 12 - bit 8)
LRA determines the last raster line address of the character row, and can be set to any value between 0 and 31.

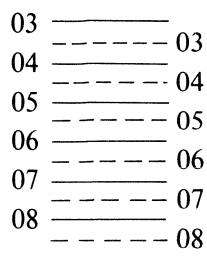
LRA					Raster address
12	11	10	9	8	
0	0	0	0	0	0
0	0	0	0	1	1
		⟨			⟨
1	1	1	1	0	30
1	1	1	1	1	31

The number of raster lines per character row is determined by the relation between FRA and LRA and also depends on the raster scan mode. In the following examples, FRA (=3) represents the first raster address in the even field.
Note that the relation between FRA and LRA is not restricted. FRA can be less, equal or greater than LRA as shown below. In this example, non-interlace mode is used.

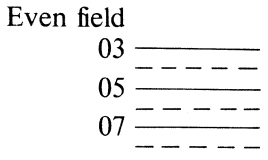
i) Non-interlace mode

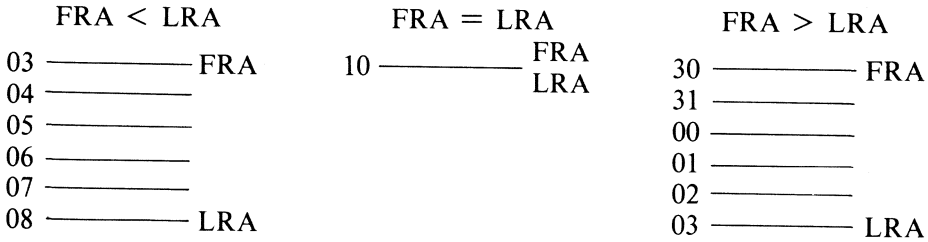


ii) Interlace sync mode

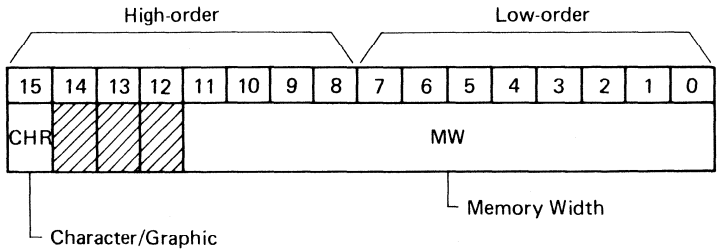


iii) Interlace sync & Video mode





5.9.2 Memory Width Register
(MWR0: rC2-rC3) (MWR1: rCA-rCB)
(MWR2: rD2-rD3) (MWR3: rDA-rDB)



Memory Width Register 0 rC2 rC3 : Upper Screen
Memory Width Register 1 rCA rCB : Base Screen
Memory Width Register 2 rD2 rD3 : Lower Screen
Memory Width Register 3 rDA rDB : Window Screen

Figure 5.19 Memory Width Register (MWR)

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 also determines whether the defined screen is a Character (CHR = high) or Graphic (CHR = low) screen. MWR0-3 apply to screens 0-3, the Upper, Base, Lower and Window screen respectively.

MWR should be greater than or equal to Horizontal Display Width (HDW - r85). MWR must be greater than HDW to perform horizontal smooth scroll. MWR maximum value is 4096.

Note) Never fail to set up the memory width of the screen specified by ORG, whether the screen setup is valid or invalid.

○ Character/Graphic (CHR: bit 15)

CHR	Functions
0	The screen is defined as GRAPHIC
1	The screen is defined as CHARACTER

Note) In the ACRTC, the memory for the graphic screen is called the frame buffer, and that for the character screen is called the refresh memory. "Frame buffer" sometimes indicates the frame buffer and the refresh memory together.

○ Memory Width (MW: bit 11 - bit 0)

MSB M W LSB	Memory width (Number of words)
0 0 0 0 0 0 0 0 0 0 0 0	0
0 0 0 0 0 0 0 0 0 0 0 1	1
⎛ ⎞	⎛ ⎞
⎝ ⎠	⎝ ⎠
1 1 1 1 1 1 1 1 1 1 1 0	4094
1 1 1 1 1 1 1 1 1 1 1 1	4095

5.9.3 Start Address Register
(SAR0: rC4-rC7) (SAR1: rCC-rCF)
(SAR2: rD4-rD7) (SAR3: rDC-rDF)

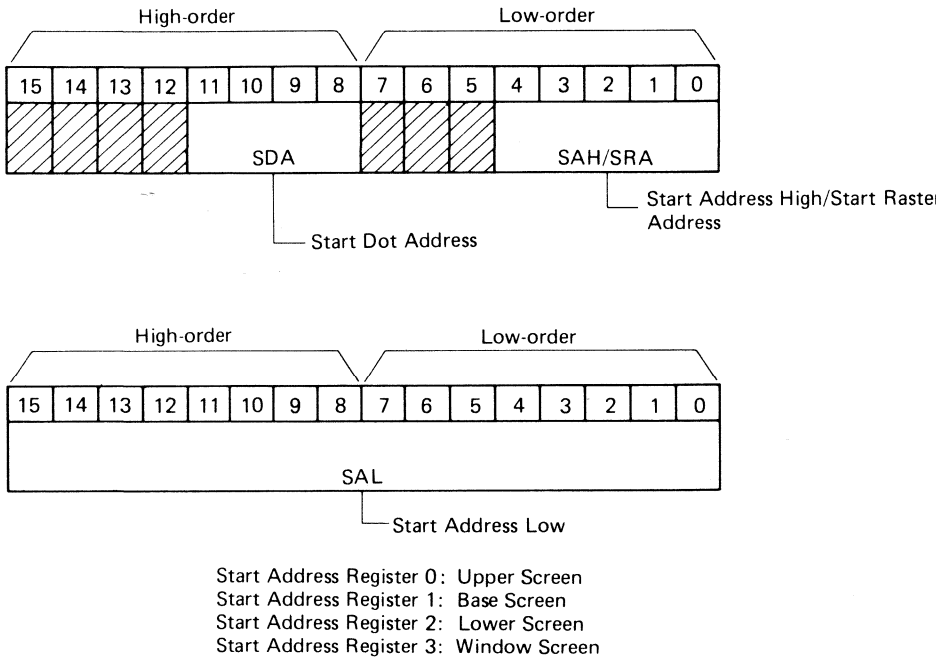


Figure 5.20 Start Address Register (SAR)

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.

Screens defined as Character have a 64K by 16 bit physical address space. Screens defined as Graphic have a 1M by 16 bit physical address space. In either case, SAR can take on any address. Frame Buffer addresses will 'wraparound' to 0 when the physical address space limit is reached independent of split screen position.

○ Start Address Low (SAL: bit 15 - bit 0)
For Character screens, SAL contains the 16 bit start address. For Graphic screens, SAL contains the least significant 16 bits of the 20 bit start address.

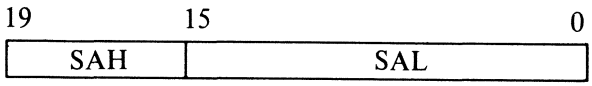
○ Start Address High (SAH: bit 3 - bit 0)
Start Raster Address (SRA: bit 4 - bit 0)
For Character screens, SRA provides the 5 bit (0-31) start raster address.

i) Character Screen

04 _____ SRA
05 _____
06 _____
07 _____ LRA
02 _____ FRA
03 _____
04 _____
05 _____
06 _____
07 _____ LRA
02 _____ FRA

For Graphic screens, SAH provides the most significant 4 bits of the 20 bit start address.

ii) Graphic Screen



Increment or decrement of SRA provides vertical smooth scroll with no additional external hardware.

Note) The different timings are to be referred to SAR in each screen. The screen to be setup-changed requires the SAR change during the timing except the display start raster of each screen, because values are loaded from the internal register to the controller during the horizontal back porch of the display start raster of each screen in the ACRTC.

The timing of reference to SAR of each screen is different.

○ Start Dot Address (SDA: bit 11 - bit 8)
SDA is used to define a start dot horizontal offset (0-15). the contents of SDA are output on HSD0-3 (MAD8-11) during the video attribute output cycle of each horizontal scan. External circuitry which controls the parallel-serial converter (shift register) load and clock based on SDA and the corresponding HSD outputs allows horizontal smooth scroll for both Character and Graphic screens.

5.9.4 Block Cursor Register (BCUR: rE0-rE7)

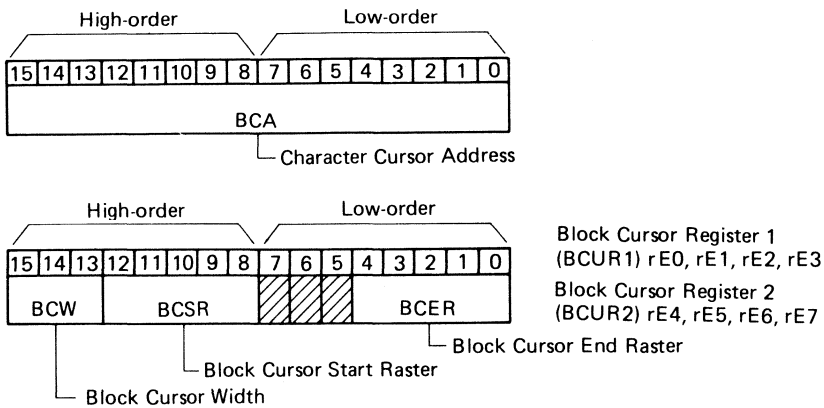


Figure 5.21 Block Cursor Register (BCUR)

BCUR defines the block cursor location (refresh memory physical memory address), start and end raster and block cursor length for two independent cursors. Depending on cursor mode, the ACRTC $\overline{CUD1}$ and $\overline{CUD2}$ lines can support the simultaneous display of both block cursors.

Should two (or more) screens be defined to contain the same refresh memory address, if the block cursor is located at that address, it will be displayed on both screens.

○ Block Cursor Address (BCA1: rE2-rE3) (BCA2: rE6-rE7)
BCA defines the 16 bit address for the block cursor. Note that the block cursor is only enabled for Character screens (CHR = high).

○ Block Cursor Start Raster (BCSR: bit 12 - bit 8)
BCSR determines the 5 bit block cursor start raster address (0-31).

○ Block Cursor Width (BCW: bit 15 - bit 13)
BCW defines the block cursor width (1-8) in units of memory cycles.

B C W			Cursor width (Memory cycle)
15	14	13	
0	0	0	1
0	0	1	2
	⟨		⟨
1	1	0	7
1	1	1	8

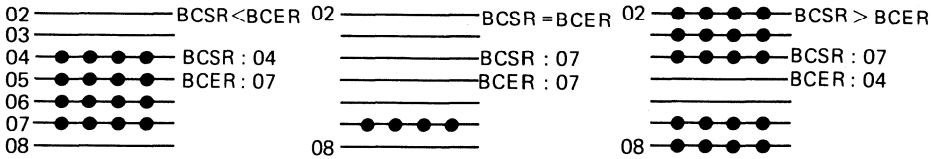
B C S R					Raster address
12	11	10	9	8	
0	0	0	0	0	0
0	0	0	0	1	1
		⟨			⟨
1	1	1	1	0	30
1	1	1	1	1	31

○ Block Cursor End Raster (BCER: bit 4 - bit 0)
BCER determines the 5 bit block cursor end raster address (0-31).

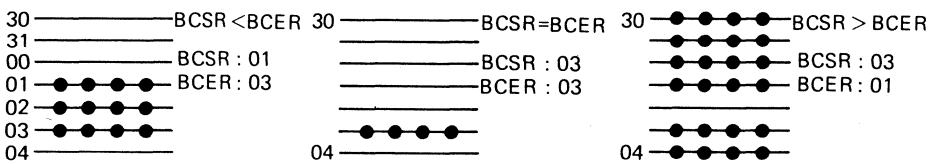
B C E R					Raster address
4	3	2	1	0	
0	0	0	0	0	0
0	0	0	0	1	1
		⟨			⟨
1	1	1	1	0	30
1	1	1	1	1	31

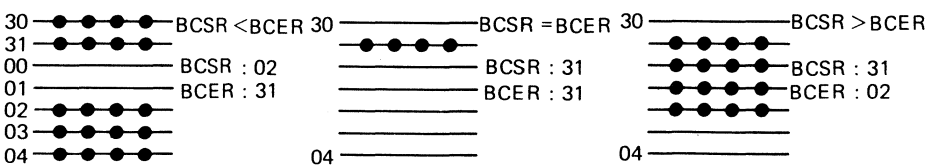
Based on FRA, LRA, BCSR and BCER, the block cursor can take on a number of different configurations as shown below.

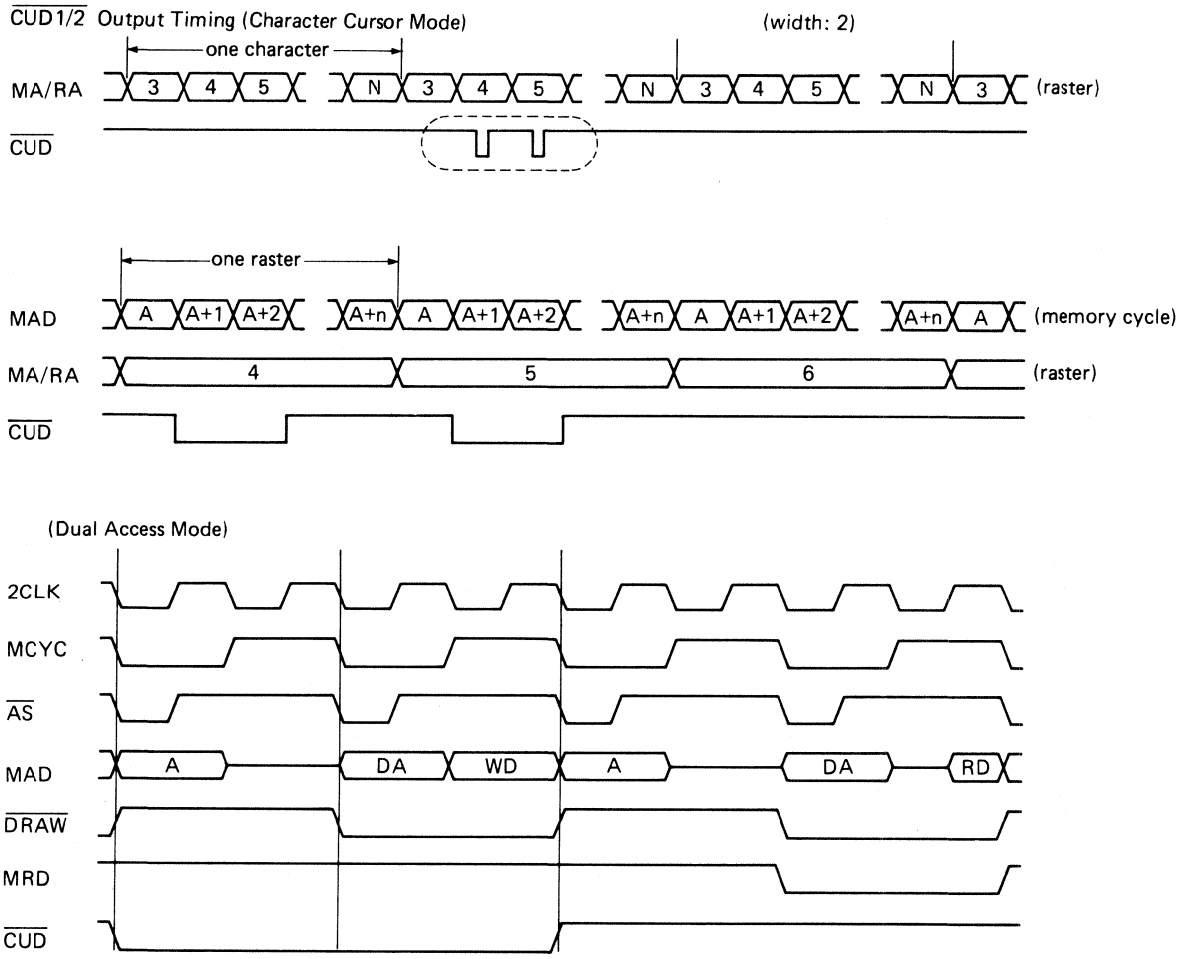
FRA ≤ LRA (FRA:02, LRA:08)



FRA > LRA (FRA:30, LRA:04)






CUD1/2 Output Timing (Character Cursor Mode)
one character
(width: 2)
MA/RA
3
4
5
N
3
4
5
N
3
4
5
N
3
(raster)
CUD
one raster
MAD
A
A+1
A+2
A+n
A
A+1
A+2
A+n
A
A+1
A+2
A+n
A
(memory cycle)
MA/RA
4
5
6
(raster)
CUD
(Dual Access Mode)
2CLK
MCYC
AS
MAD
A
DA
WD
A
DA
RD
DRAW
MRD
CUD

5.9.5 Cursor Definition Register (CDR: rE8-rE9)

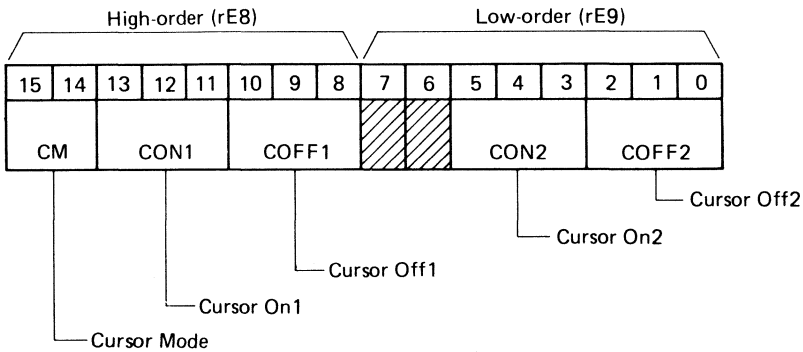


Figure 5.22 Cursor Definition Register

CDR defines the cursor types and the way in which the $\overline{CUD1}$ and $\overline{CUD2}$ outputs are controlled. Depending on CDR, up to three cursors may be simultaneously displayed. Cursor types are defined as follows.

BLOCK – The standard 'block' type (including underline) cursor typically used on alphanumeric displays.

GRAPHIC – The ACRTC may generate a rectangular cursor area of arbitrary X and Y dimension. Normally, this is used to enable an external cursor bit map circuit. In this case, the cursor may take on any user defined graphic shape.

CROSSHAIR – The ACRTC can display a crosshair cursor. The X and Y (horizontal and vertical) dimensions are independently programmable.

○ Cursor Mode (CM: rE8 bit 15 - bit 14)
CM defines the type of cursor(s) to be displayed and the way in which the $\overline{CUD1}$ and $\overline{CUD2}$ outputs are interpreted.

CM		Functions
15	14	
0	x	Block Cursor Mode: Block cursor 1 (defined in BCR1) is output on $\overline{CUD1}$. Block cursor 2 (defined in BCR2) is output on $\overline{CUD2}$. The Graphic cursor (defined in GCR) is not used.
1	0	Graphic Cursor Mode: Graphic Cursor (GCR) is output on $\overline{CUD1}$. Block cursor 1 and 2 are combined and output on $\overline{CUD2}$.
1	1	Crosshair Cursor Mode: The horizontal element is output on $\overline{CUD1}$. The vertical element is output on $\overline{CUD2}$. The Block cursor (BCR) is not used.

x = Don't care.

Note) The cursor skew must be more than 1 in using the output on $\overline{CUD2}$.

○ Cursor ON (CON1: rE8 bit 13 - bit 11)
(CON2: rE9 bit 5 - bit 3)
Cursor OFF (COFF1: rE8 bit 10 - bit 8)
(COFF2: rE9 bit 2 - bit 0)

CON and COFF determine the cursor blink timing. CON1/COFF1 apply to $\overline{CUD1}$ and CON2/COFF2 apply to $\overline{CUD2}$. The unit time is 4 field periods. In Crosshair Cursor Mode, CON1/COFF1 is used for blink timing and CON2/COFF2 are not used.

CON1			Blink "Low" level (Field period)
13	12	11	
0	0	0	*
0	0	1	8
0	1	0	12
0	1	1	16
1	0	0	20
1	0	1	24
1	1	0	28
1	1	1	32

CON2			Blink "Low" level (Field period)
5	4	3	
0	0	0	*
0	0	1	8
0	1	0	12
0	1	1	16
1	0	0	20
1	0	0	24
1	1	0	28
1	1	1	32

* Cursor is output at "High" level.

COFF1			Blink "High" level (Field period)
10	9	8	
0	0	0	*
0	0	1	8
0	1	0	12
0	1	1	16
1	0	0	20
1	0	1	24
1	1	0	28
1	1	1	32

COFF2			Blink "High" level (Field period)
2	1	0	
0	0	0	*
0	0	1	8
0	1	0	12
0	1	1	16
1	0	0	20
1	1	0	24
1	1	0	28
1	1	1	32

* Unless "CON=000" is set, cursor is output at "Low" level.

5.9.6 Zoom Factor Register (ZFR: rEA)

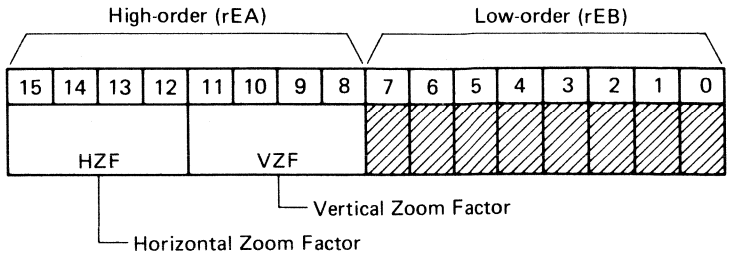


Figure 5.23 Zoom Factor Register (ZFR)

ZFR determines the horizontal (memory cycle) and vertical (raster) multipliers (1 to 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 $F for 16 times zoom.
Note) Set "REB=$00" in 16 bit interface mode.

○ Horizontal Zoom Factor (HZF: bit 15 - bit 12)

HZF defines the horizontal zoom factor in units of memory cycles. The ACRTC will output a same display address by HZF times. HZF is output as video attributes on MAD12-15 lines for use by an external circuit which controls shift clock timing.

H Z F				Factor of zooming up in the horizontal director (Magnitude)
15	14	13	12	
0	0	0	0	1
0	0	0	1	2
		⟨		⟨
		⟩		⟩
1	1	1	0	15
1	1	1	1	16

○ Vertical Zoom Factor (VZF: bit 11 - bit 8)

VZF defines the vertical zoom factor. The ACRTC performs the vertical zoom by modifying its frame buffer address (Graphic screens) or raster address (Character screens) so that multiples of the same raster data are displayed.

V Z F				Factor of zooming up in the vertical director (Magnitude)
11	10	9	8	
0	0	0	0	1
0	0	0	1	2
		⟨		⟨
		⟩		⟩
1	1	1	0	15
1	1	1	1	16

5.9.7 Light Pen Address Register (LPAR: rEC-rEF)

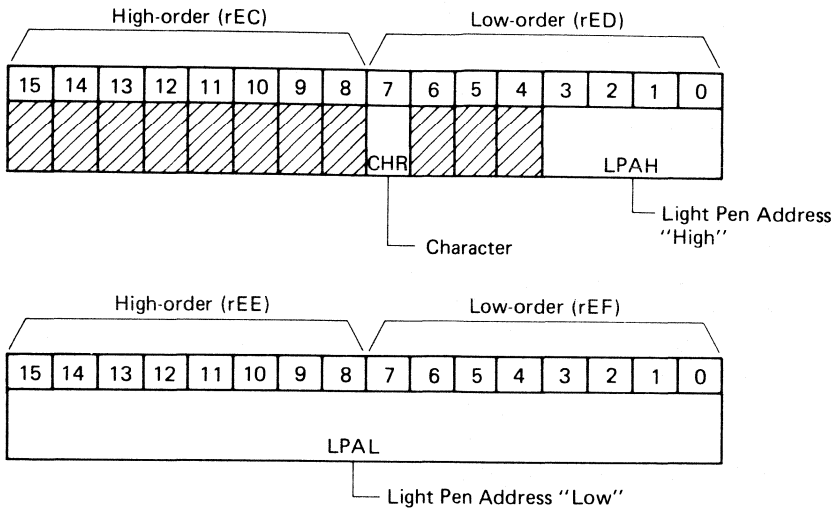


Figure 5.24 Light Pen Address Register (LPAR)

LPAR is a read only register. When the ACRTC LPSTB input is asserted, the current display address is latched into 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 MPU software depending on system configuration. In Superimposed access mode, light pen strobes which occur within a superimposed Window/Background display cause the Background address to be latched.

○ Character/Graphic (CHR: rED bit 7)
CHR indicates whether the latched display address corresponds to a screen defined as Character or Graphic.

CHR	Functions
0	LPAR contains Graphic screen address
1	LPAR contains Character screen address

○ Light Pen Address High (LPAH: rED bit 3 - bit 0)
LPAH is only valid if CHR = 0 and contains the most significant 4 bits of the 20 bit Graphic screen display address.
Note) Bit 15~bit 8 values are indefinite in the graphic screen mode, and bit 6~bit 4 values are in the character screen mode.

○ Light Pen Address Low (LPAL: rEE-rEF)
If CHR = 0, LPAL contains the least significant 16 bits of the 20 bit Graphic screen display address. If CHR = 1, LPAL contains the 16 bit Character screen display address.

5.10 Drawing Control Registers

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

a) Pattern RAM

b) Drawing Parameter Registers
   - Color 0 Register (CL0)
   - Color 1 Register (CL1)
   - Color Comparison Register (CCMP)
   - Edge Color 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. The Drawing Parameter Registers are accessed using the Read and Write Parameter Register (RPR, WPR) commands.

5.10.1 Pattern RAM

The ACRTC includes 32 byte pattern RAM. The Pattern RAM is used for predefining 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 color 'palette' corresponding to 0 and 1 data values is defined using the Color 0 (CL0) and Color 1 (CL1) registers.

To store color 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 color coded contents of the Pattern RAM are directly written to the frame buffer. 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.

5.10.2 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	CL0	CL0															
Pr01	R/W	Color 1	CL1	CL1															
Pr02	R/W	Color Comparison	CCMP	CCMP															
Pr03	R/W	Edge Color	EDG	EDG															
Pr04	R/W	Mask	MASK	MASK															
Pr05	R/W	Pattern RAM Control	PRC	PPY								PZCY				PPX			PZCX
↓				PSY												PSX			
Pr07				PEY								PZY				PEX			PZX
Pr08	R/W	Area Definition **	ADR	XMIN															
↓				YMIN															
				XMAX															
Pr0B				YMAX															
Pr0C	R/W	Read Write Pointer	RWP	DN								RWPH							
Pr0D				RWPL															
Pr0E	—	.....	—	—															
Pr0F				—															
Pr10	R	Drawing Pointer	DP	DN								DPAH							
Pr11				DPAL														DPD	
Pr12	R	Current Pointer	CP	X															
Pr13				Y															
Pr14	—	.....	—	—															
Pr15				—															

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

Figure 5.25 Drawing Parameter Registers

Note 1) Never fail to set CL0 and CL1 regardless of the color mode.
Note 2) CCMP → need not be set unless "OPM = 100 or 101" is set.
Note 3) EDG → need not be set when the paint command is not used.
Note 4) MASK is available only for DMOD, MOD, SCLR and SCPY commands.
Note 5) The area definition register need not be set when the AREA mode is not used ("000" or "100" is set).
Note 6) The drawing pointer or the current pointer can be rewritten by using ORG and graphic commands.
Note 7) The current pointer is defined by binary complements for negative values of X and Y axis.

5.10.2.1 Color 0 Register (CL0: Pr00)

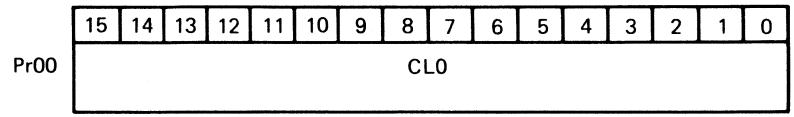


Figure 5.26 Color Register 0 (CL0)

When logical drawing data = 0 in the pattern RAM, the contents of CL0 are stored in the frame buffer.

Note) The data setup differs depending on the graphic bit mode. The same data (color) should be set every 2, 4 or 8 bits in 2, 4 or 8 bits/pixel mode, respectively.

5.10.2.2 Color 1 Register (CL1: Pr01)

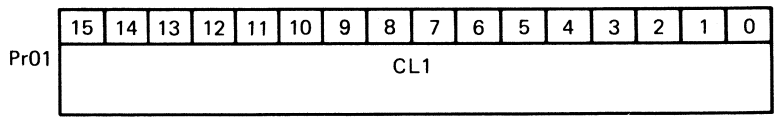


Figure 5.27 Color Register 1 (CL1)

When logical drawing data = 1 in the pattern RAM, the contents of CL1 are stored in the frame buffer.

Note) The same note as in CL0 should be required.

5.10.2.3 Color Comparison Register (CCMP: Pr02)

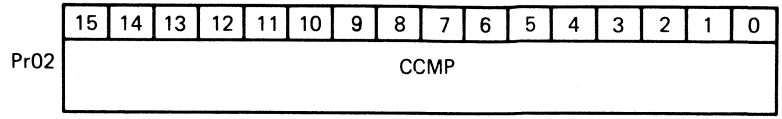


Figure 5.28 Color Comparison Register (CCMP)

CCMP defines a comparison color for use with conditional drawing operations. Conditional drawing applies various logical comparisons between the drawing data and CCMP to determine if drawing should occur.
Note) The same note as in CL0 should be required.

5.10.2.4 Edge Color Register (EDG: Pr03)

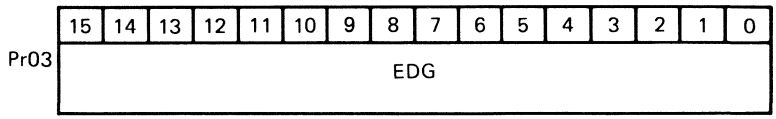


Figure 5.29 Edge Color Register (EDG)

EDG defines the boundary edge color for use by the PAINT command. In one mode, the edge is defined as the color contained in EDG. In another mode, the edge is defined as any color except the color contained in EDG.
Note) The same note as in CL0 should be required.

5.10.2.5 Mask Register (MASK: Pr04)

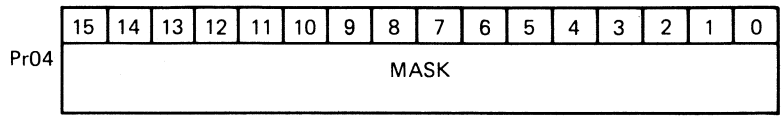


Figure 5.30 Mask Register (MASK)

When performing data transfer and drawing of the frame buffer, MASK is used to mask bits upon which drawing and other logical operations should not be performed. If MASK bit is 0, the corresponding frame buffer bit is excluded from logic operation.

Note) Only DMOD, MOD, SCLR and SCPY command can use the MASK Register. The graphic drawing and the data transfer group commands without MM cannot use the MASK register.

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

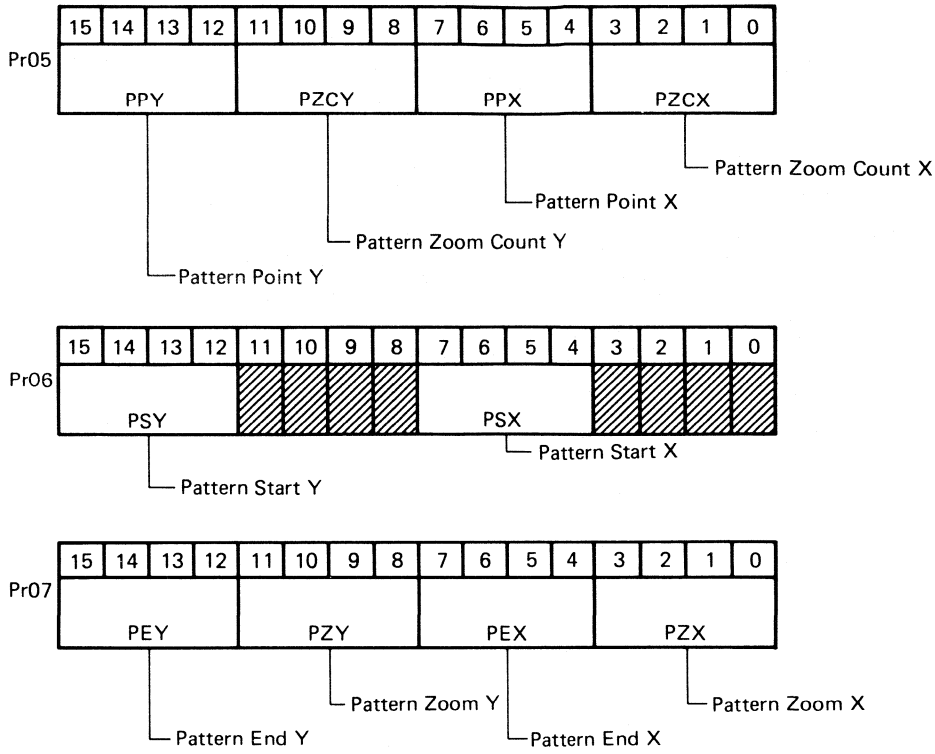


Figure 5.31 Pattern RAM Control Register (PRC)

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 the 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 to between 0-15 for Color Register indirect drawing and between 0-3 for Pattern RAM direct drawing.
Note) The color mode "11" is called the Pattern RAM direct drawing mode, and others are the Color Register indirect drawing mode.

○ 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 to between 0-15 for Color Register indirect drawing and between 0-3 for Pattern RAM direct drawing.
Note) Set "PSX $\leq$ PEX" and "PSY $\leq$ PEY".

○ 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 by 1 magnification (no magnification) while PZX, PZY = $F specifies by 16 magnification.
Note) Set PZX and PZY to 0 not in zooming.

○ 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 before zooming.

○ Pattern Pointer X (PPX: Pr05 bit 7 - bit 4)
Pattern Pointer Y (PPY: Pr05 bit 15 - bit 8)
The current reference point within the Pattern RAM is specified by PPX and PPY. When using PPX, PPY to define a pattern scan starting point, the relationship PSX $\leq$ PPX $\leq$ PEX and PSY $\leq$ PPY $\leq$ PEY must be maintained.
Note) Normally, PPX and PPY should be set before starting drawing.

5.10.2.7 Area Definition Register (ADR: Pr08 - Pr0B)

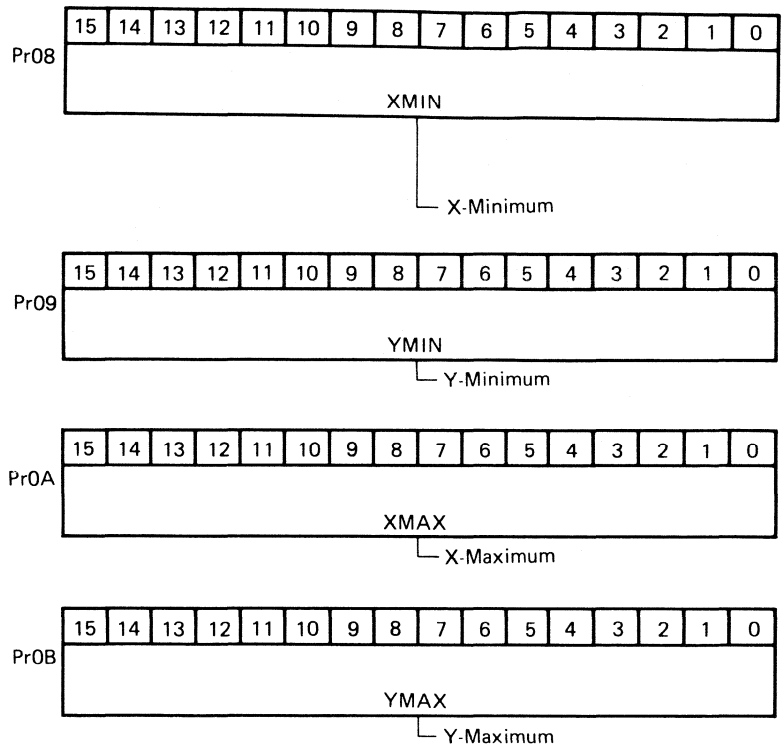


Figure 5.32 Area Definition Register (ADR)

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.

Note) Area is bounded by XMIN, YMIN, XMAX and YMAX, and includes boundaries.

The area judgement may not be properly performed in the graphic drawing just after the area definition, which can be avoided by moving the current pointer (AMOVE or RMOVE).

5.10.2.8 Read Write Pointer (RWP: Pr0C - Pr0D)

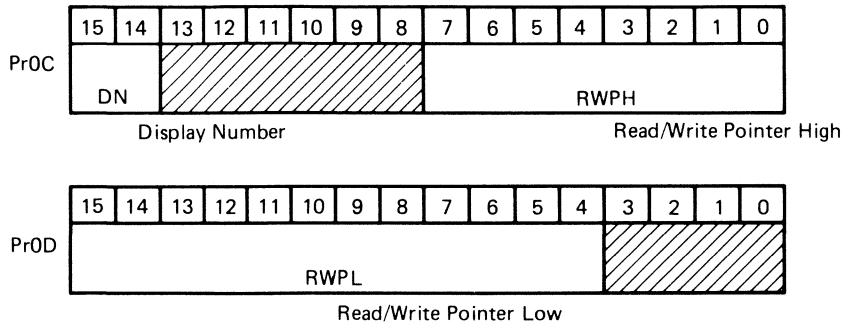


Figure 5.33 Read Write Pointer (RWP)

RWP specifies a 20 bit physical frame buffer 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
15	14	
0	0	Upper Screen
0	1	Base Screen
1	0	Lower Screen
1	1	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 frame buffer address used with the data transfer commands.

Note) Set bit 13 - 8 to 0.

5.10.2.9 Drawing Pointer (DP: Pr10 - Pr11)

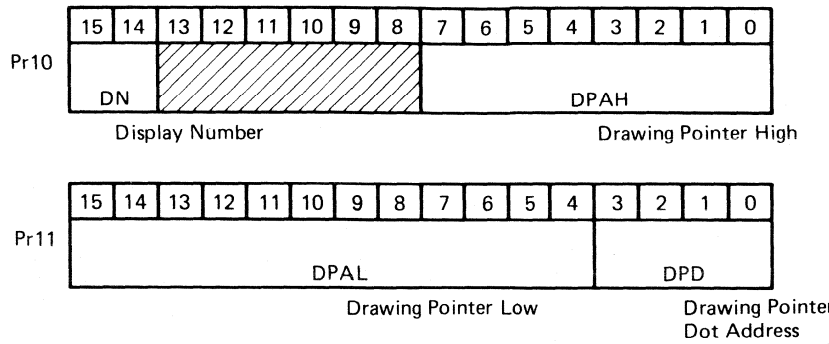


Figure 5.34 Drawing Pointer (DP)

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.
Note) Set bit 13~8 to 0.

○ 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 Graphic Bit Mode (GBM).

GBM	Function of DPD
1 bit/pixel	DPD specifies 1 of 16 logical pixels
2 bits/pixel	DPD specifies 1 of 8 logical pixels using most significant 3 bits of DPD. The least significant bit is not used.
4 bits/pixel	DPD specifies 1 of 4 logical pixels using most significant 2 bits of DPD. The 2 least significant bits are not used.
8 bits/pixel	DPD specifies 1 of 2 logical pixels using the most significant bit of DPD. The 3 least significant bits are not used.
16 bits/pixel	DPD is not used.

5.10.2.10 Current Pointer (CP: Pr12 - Pr13)

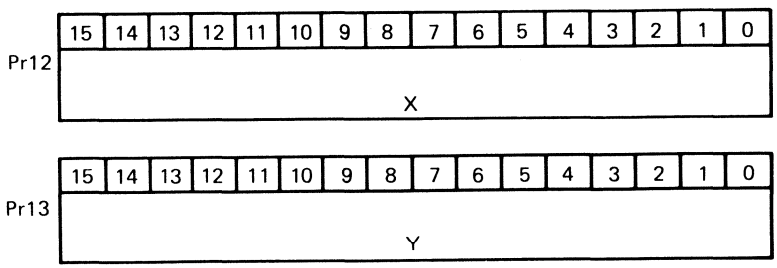


Figure 5.35 Current Pointer (CP)

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-complement format is used to indicate positive and negative values.

6. COMMANDS

6.1 Command Overview

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

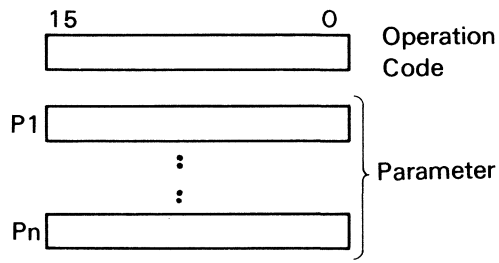
1) Register Access Commands
2) Data Transfer Commands (including DMA Transfer)
3) Graphic Drawing Commands

6.2 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

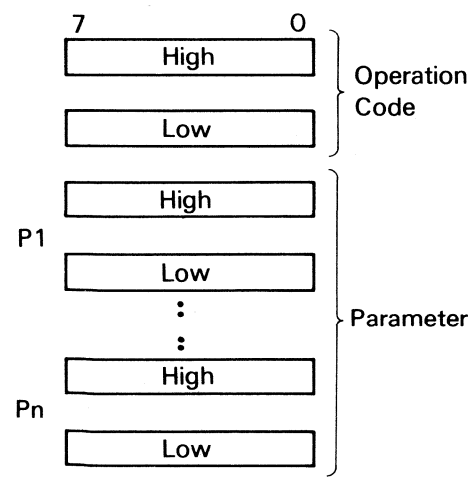
In the case of 16-bit interface, first move the 16-bit operation code and then move necessary 16-bit parameters one by one.



(a) 16 bit Interface

(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.



(b) 8 bit Interface

6.3 Command Transfer Modes

Commands (and associated parameters) can be issued to the ACRTC in one of two ways — program transfer or DMA transfer.

6.3.1 Program Transfer

Program Transfer occurs when the MPU specifies the FIFO entry address and then writes op-code/parameters to the write FIFO under program control (RS = high, R/$\overline{W}$, $\overline{CS}$ = 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 then 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.

6.3.2 Command DMA Transfer

Commands and parameters can be transferred from MPU system memory when using external DMAC. The MPU initiates and terminates Command DMA Transfer mode under software control (bit 12 (CDM) 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.

6.4 Register Access Commands

Registers associated with the Drawing (Pattern RAM and Drawing Parameter Registers) are accessed through the read and write FIFOs using the 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

Note) As for detail, refer to chapter 7.

Figure 6.2 Register Access Commands

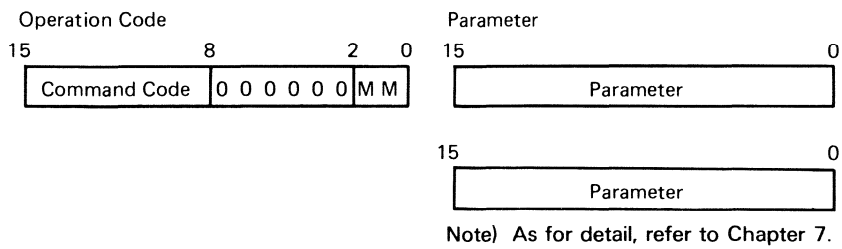
6.5 Data Transfer Commands

Data Transfer Commands are used to move word 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 by the RWP (Read Write Pointer) in Drawing Parameter Register.

The DMA Transfer Commands (DRD, DWT and DMOD) are used to send large amount of data between system memory and frame buffer. The programmer specifies the command and the X and Y logical pixel dimensions of the frame buffer. The ACRTC will automatically control the external DMAC to request data transfers via the read or write FIFOs. In Data DMA Transfer, the ACRTC $\overline{\text{DONE}}$ pin becomes an output by which the ACRTC asserts to the DMAC to terminate the transfer. Also, either cycle steal or burst DMA request mode can be used for data DMA (DRC bit of CCR).

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

Figure 6.3 Data Transfer Commands



Note) As for detail, refer to Chapter 7.

Figure 6.4 Data Transfer Command Format

6.5.1 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.

○ Modify Mode Examples

The following examples show the use of the REPLACE, OR, AND and EOR modify modes. The modifier data (issued as a prameter to the DMOD, MOD, SCLR and SCPY commands) and the non-masked data in the frame buffer are logically operated on, and the result is rewritten to the frame buffer.

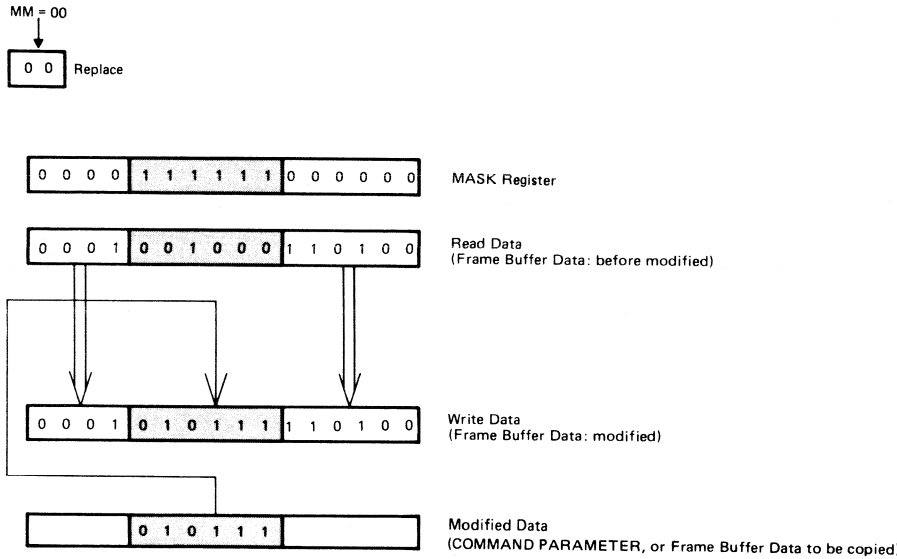


Figure 6.5(a) REPLACE Modify Mode

The read data of modifiable bits (set to "1" in MASK register) are REPLACED with the command parameter modifier data. The result is rewritten into the frame buffer.

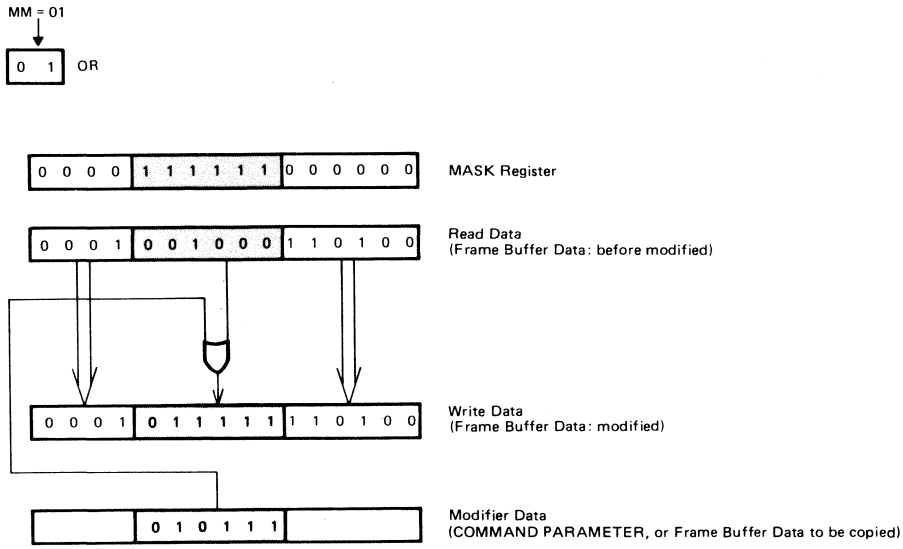


Figure 6.5(b) OR Modify Mode

The read data of modifiable bits (set to "1" in MASK register) are ORed with the command parameter modifier data. The result is rewritten into the frame buffer.

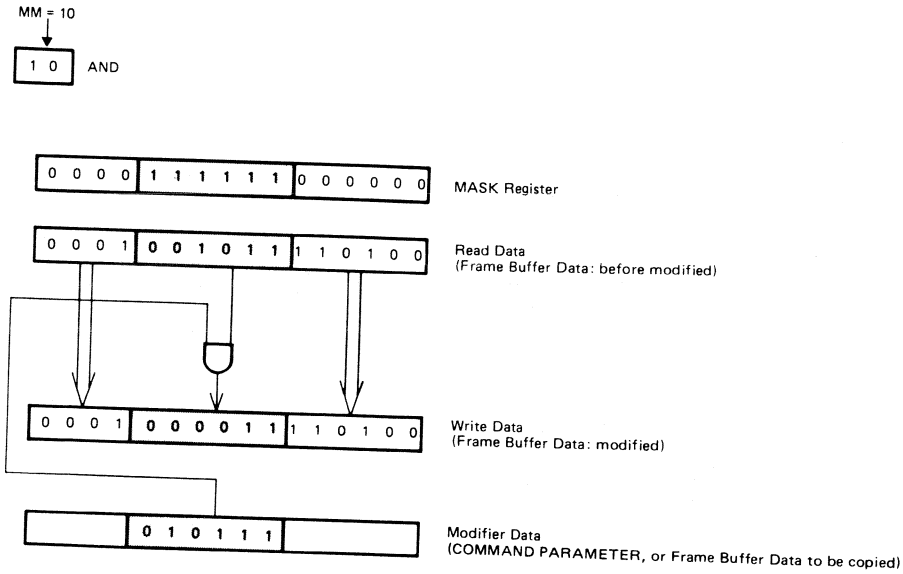


Figure 6.5(c) AND Modify Mode

The read data of modifiable bits (set to "1" in MASK register) are ANDed with the command parameter modifier data. The result is rewritten into the frame buffer.

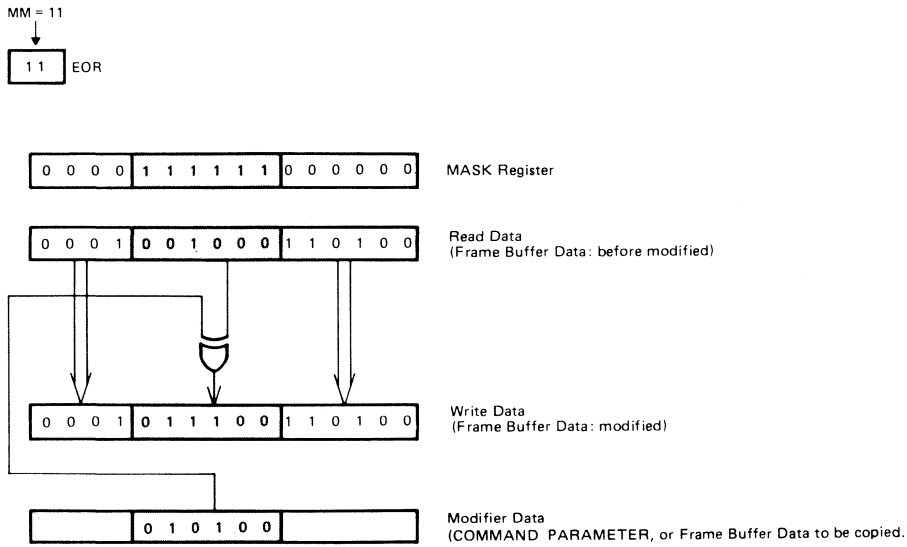


Figure 6.5(d) EOR Modify Mode

The read data of modifiable bit (set to "1" in MASK register) are EORed with the command parameter modifier data. The result is rewritten into the frame buffer.

6.6 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 coordinates. 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, 2's complement numbers are used to represent 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 coordinates (x, y) based on the origin point set by the ORG command.
(Both positive and negative values can be specified.)

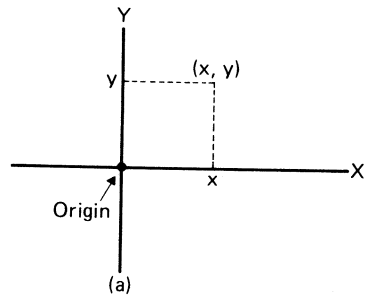


Figure 6.6(a) Absolute Coordinate Specification

(ii) Relative Coordinate Specification
Specifies the relative coordinates (△x, △y) related to the current drawing point.
(Both positive and negative values can be specified.)

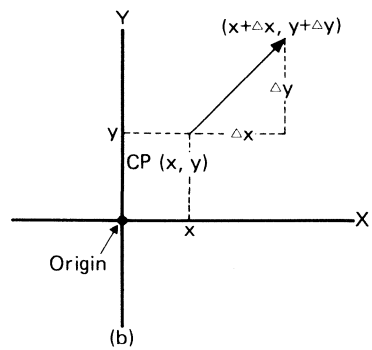


Figure 6.6(b) Relative Coordinate Specification

Command	Function
AMOVE	Movement of current pointer
RMOVE	
ALINE	Drawing of straight line
RLINE	
ARCT	Drawing of rectangle
RRCT	
APLL	Drawing of polyline
RPLL	
APLG	Drawing of polygon
RPLG	
CRCL	Drawing of circle
ELPS	Drawing of ellipse
AARC	Drawing of arc
RARC	
AEARC	Drawing of ellipse arc
REARC	
AFRCT	Painting of rectangle area (Tiling)
RFRCT	
PAINT	Painting of arbitrary area (Tiling)
DOT	Dotting
PTN	Drawing of pattern (rotation angle: 45°)
AGCPY	Rectangle area copy within frame buffer (rotation angle: 90° mirror turnover)
RGCPY	

Figure 6.7 Graphic Drawing Commands

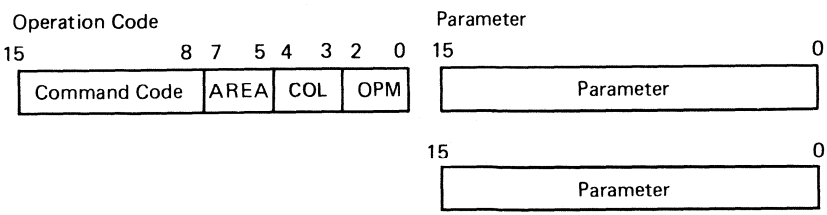


Figure 6.8 Graphic Drawing Command Format

6.6.1 Operation Mode

The Operation Mode (OPM bits) of the Graphic Drawing Command specifies the logical drawing condition.

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 (CL0 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 (CL0 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).

Following are examples of each of the eight operation modes. In these examples, 4 bits pixel mode is assumed.

Figure 6.10 shows examples of a drawing pattern applied with various OPM modes.

Note) In a graphic copy command, the read data from the source area and the destination area are accessed.
The graphic copy command is unique in not using the Pattern RAM data directly for drawing, and so the data operated with the source area data and the destination area data is written to the destination area.
The source area data is used as the color data in this command.

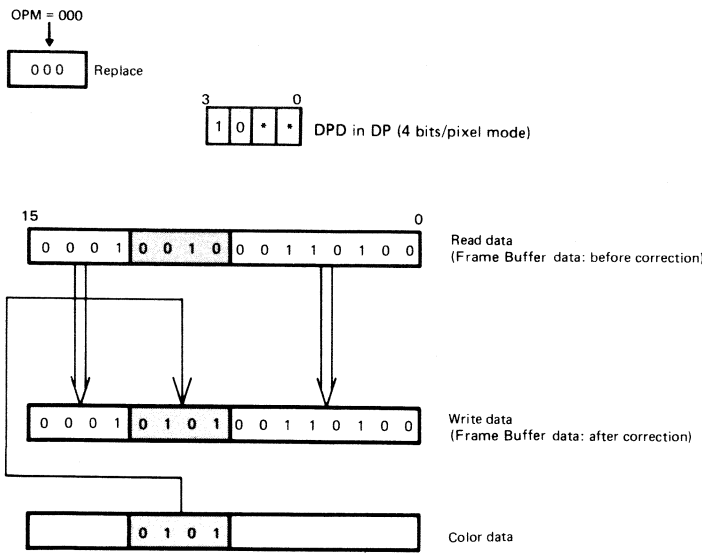


Figure 6.9(a) REPLACE Operation Mode

1-pixel data of the frame buffer is REPLACED with the corresponding color data and the result is rewritten into the frame buffer. The dot pointer serves to extract the pixel from the frame buffer word — in this example, 4 bits/pixel.

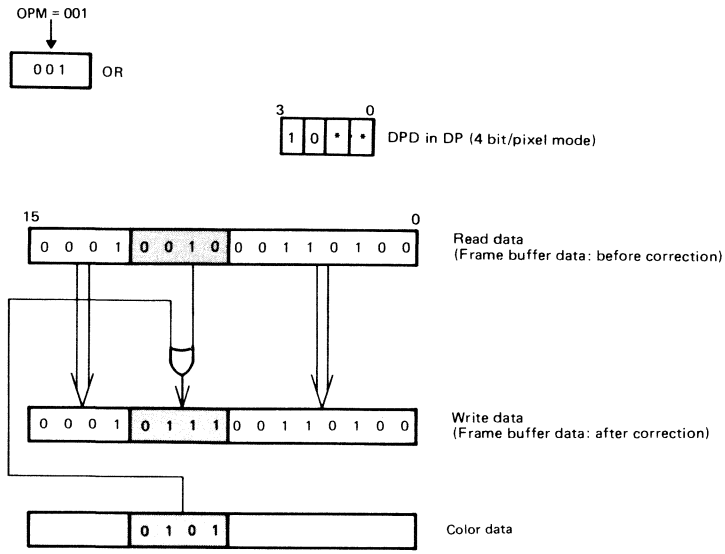


Figure 6.9(b) OR Operation Mode

1-pixel data of the frame buffer is ORed with the corresponding color data and the result is rewritten into the frame buffer. The dot pointer serves to extract the pixel from the frame buffer word — in this example, 4 bits/pixel.

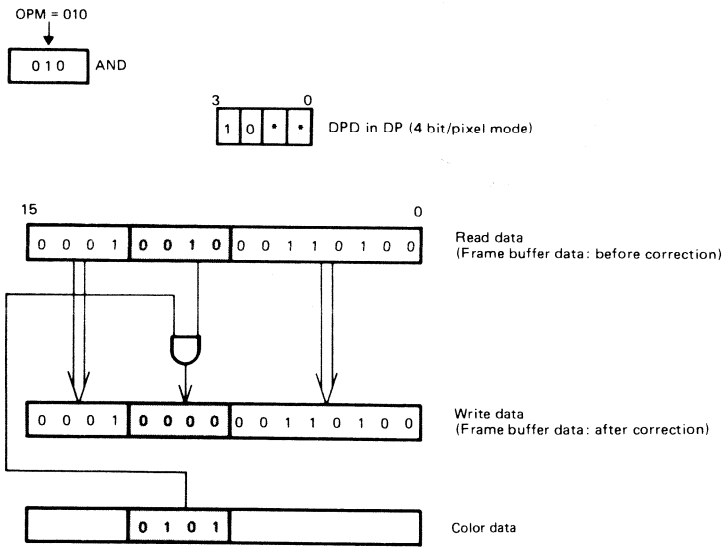


Figure 6.9(c) AND Operation Mode

1-pixel data of the frame buffer is ANDed with the corresponding color data and the result is rewritten into the frame buffer. The dot pointer serves to extract the pixel from the frame buffer word — in this example, 4 bits/pixel.

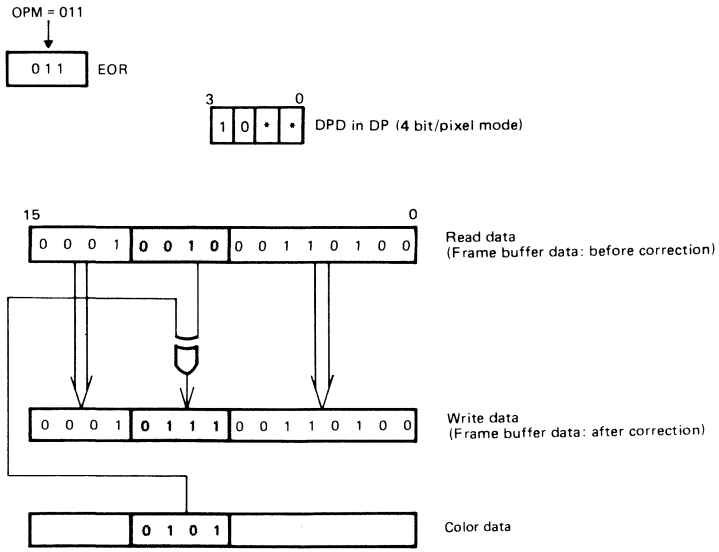


Figure 6.9(d) EOR Operation Mode

1-pixel data of the frame buffer is EORed with the corresponding color data and the result is rewritten into the frame buffer. The dot pointer serves to extract the pixel from the frame buffer word — in this example, 4 bits/pixel.

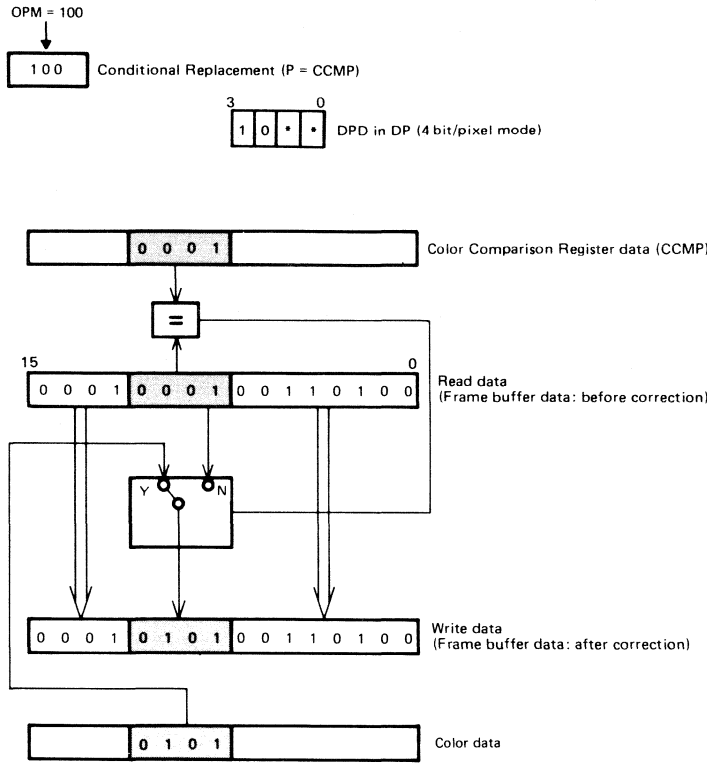


Figure 6.9(e) P=CCMP Operation Mode

Figure 6.9(e) Read Data = CCMP Operation Mode

1-pixel data of the frame buffer is compared with the corresponding pixel data of the Color Comparison Register (CCMP). If equal, the frame buffer data is replaced with the color data and the result is rewritten into the frame buffer. If not equal, the read data (unmodified) is rewritten into the frame buffer. The dot pointer serves to extract the pixel from the frame buffer word — in this example, 4 bits/pixel.

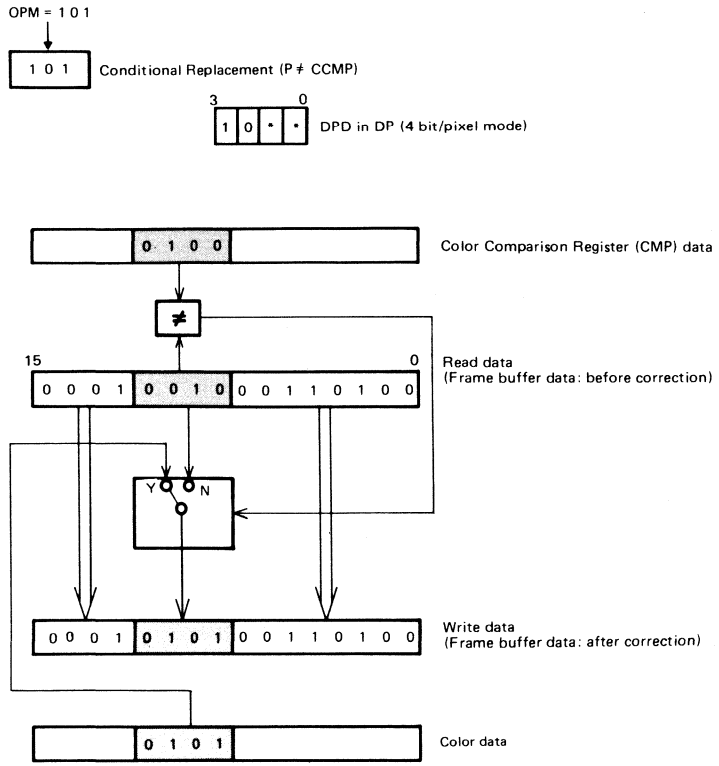


Figure 6.9(f) Read Data ≠ CCMP Operation Mode

1-pixel data of the frame buffer is compared with the corresponding pixel data of the Color Comparison Register (CCMP). If not equal, the frame buffer data is replaced with the color data and the result is rewritten into the frame buffer. If equal, the read data (unmodified) is rewritten into the frame buffer. The dot pointer serves to extract the pixel from the frame buffer word — in this example, 4 bits/pixel.

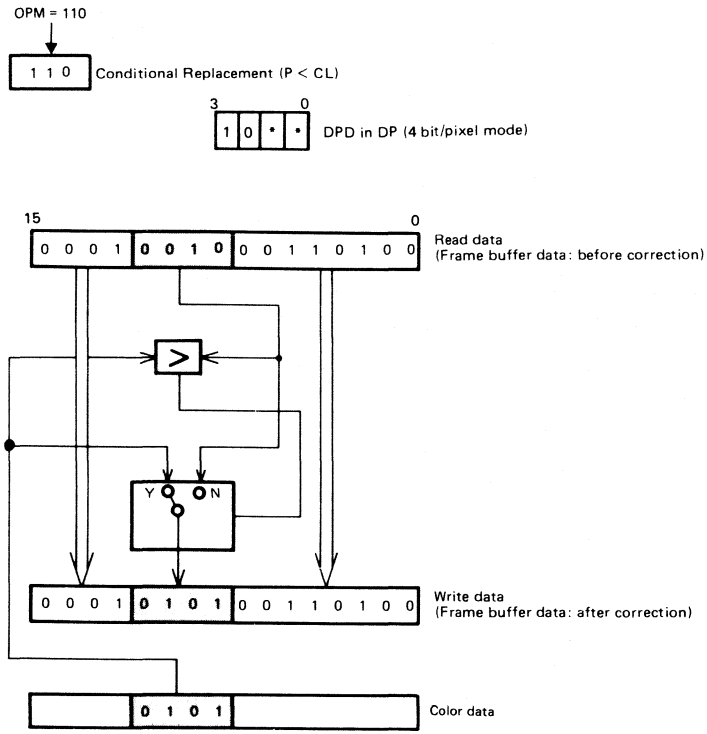


Figure 6.9(g) Read Data < CL Operation Mode

1-pixel data of the frame buffer is compared with the corresponding pixel data of the color register data (CL). If the frame buffer data is LESS than the color register 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 into the frame buffer. The dot pointer serves to extract the pixel from the frame buffer word — in this example, 4 bits/pixel.

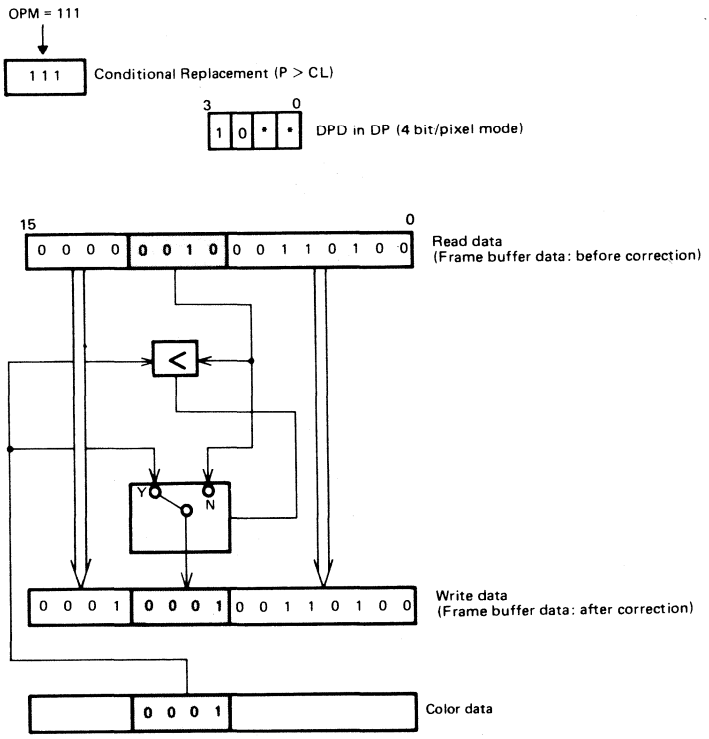
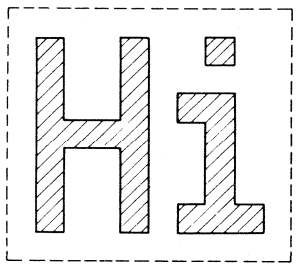
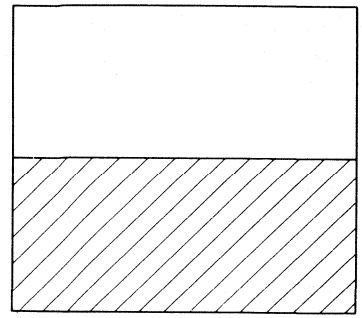


Figure 6.9(h) Read Data > CL Operation Mode

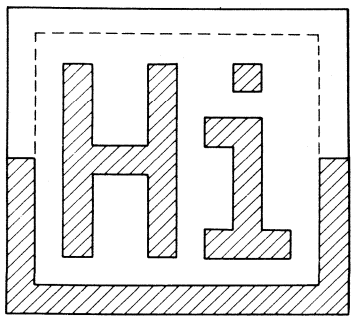
1-pixel data of the frame buffer is compared with the corresponding pixel data of the color register data (CL). If the frame buffer data is GREATER than the color register data, the read data is replaced with the color data and the result is rewritten into the frame buffer. If the read data is LESS than or EQUAL to the color data, the read data (unmodified) is rewritten into the frame buffer. The dot pointer serves to extract the pixel from the frame buffer word — in this example, 4 bits/pixel.



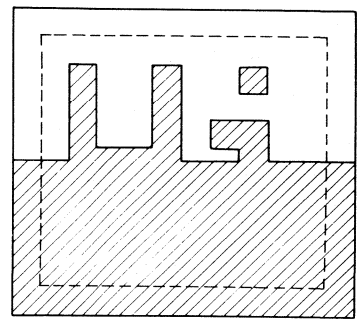
Drawing Pattern



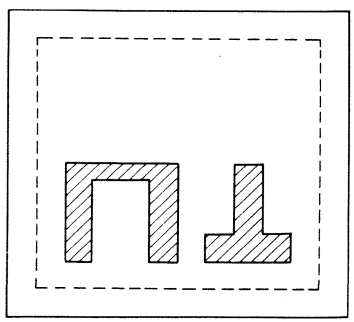
Frame Buffer Image before Drawing



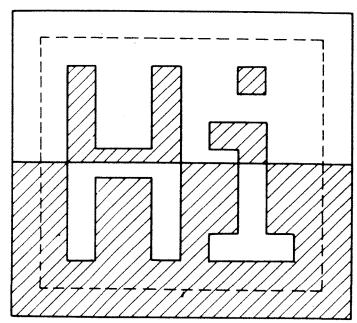
Replacement



OR



AND



EOR

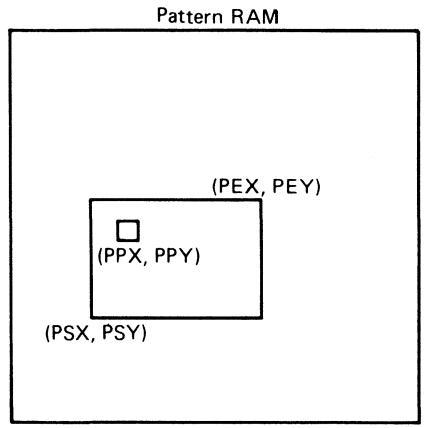
Figure 6.10 Operation Mode Examples

6.6.2 Color Mode

The Color Mode (COL bits) specifies 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.

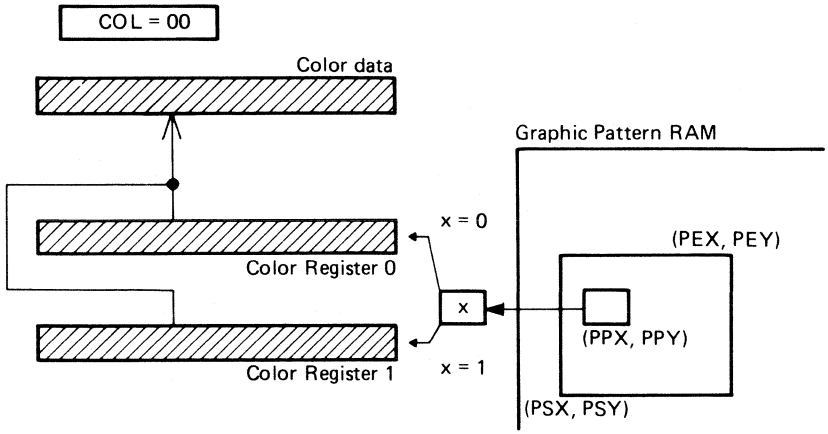
The Color Mode chooses the source for color information based on the contents (0 or 1) of a particular bit in the 16-bit by 16-bit (32 byte) Pattern RAM. A sub-pattern is specified by programming the Pattern RAM Control Register (PRC) with the start (PSX, PSY) and end (PEX, PEY) points which define the diagonal of the sub-pattern. Furthermore, a specific starting point for Pattern RAM scanning is specified by PPX and PPY.



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

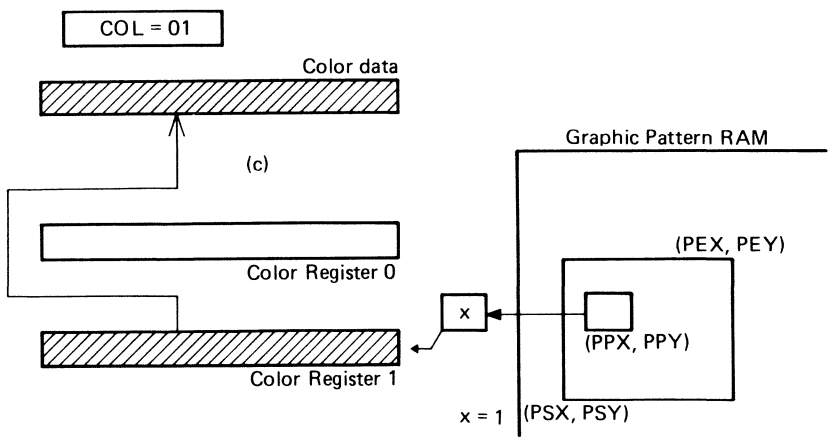
Color Register = | 0 0 0 1 | 0 0 0 1 | 0 0 0 1 | 0 0 0 1 |

In this way, color changes due to changing dot address are avoided.



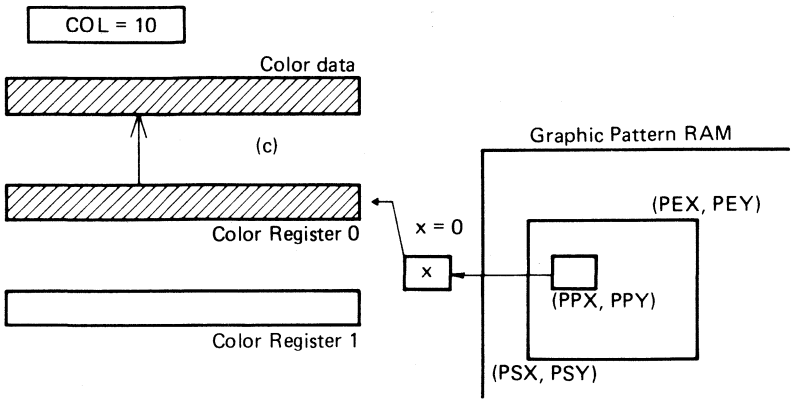
If the scanned Pattern RAM bit is equal to '0', Color Register 0 (CL0) determines the color data. If the scanned Pattern RAM bit is equal to '1', Color Register 1 (CL1) determines the color data.

Figure 6.11(a) Color Mode = 00



If the scanned Pattern RAM bit is equal to '0', the drawing operation is suppressed and the frame buffer is not changed. If the scanned Pattern RAM bit is equal to '1', Color Register 1 (CL1) determines the color data.

Figure 6.11(b) Color Mode = 01



If the scanned Pattern RAM bit is equal to '1', the drawing operation is suppressed and the frame buffer is not changed. If the scanned Pattern RAM bit is equal to '0', Color Register 0 (CL0) determines the color data.

Figure 6.11(c) Color Mode = 10

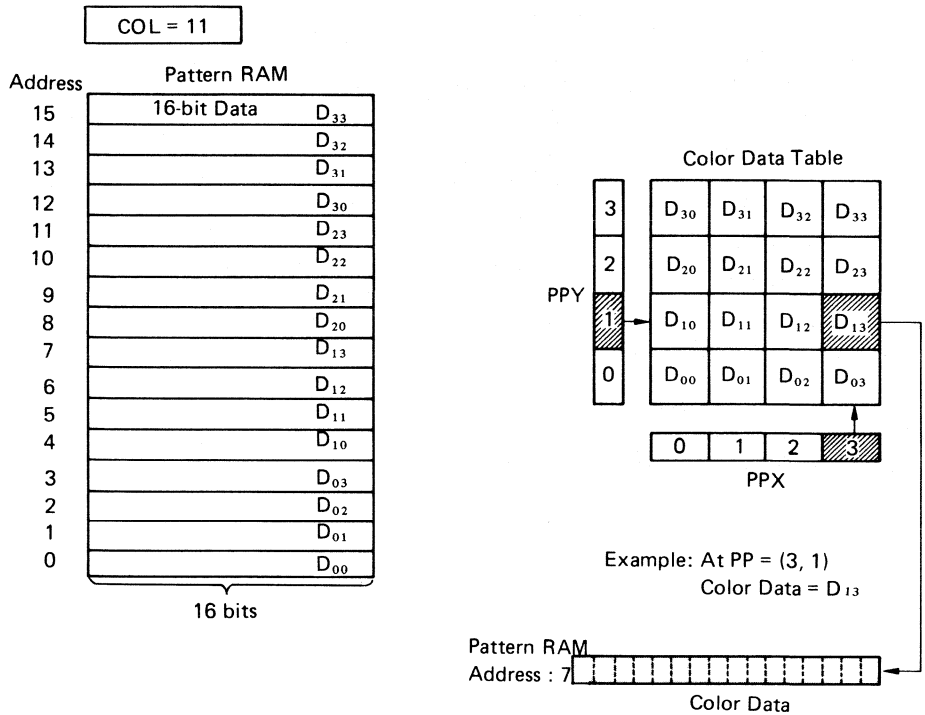


Figure 6.11(d) Color Mode = 11

In the former three color modes (Pattern RAM indirect), the actual color data 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 data. 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-bit color data.

Associated with this logical remapping of the Pattern RAM, the contents of the Pattern RAM Control Register (PRC) are interpreted differently. As shown below the pattern pointer, pattern start and pattern end (PPX, PPY, PSX, PSY, PEX and PEY) are restricted to specify a maximum 4 by 4 logical pixel pattern. Specifically, bits 15-14 and 7-6 must be set to 0.

RN	Register Name	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
05	Pattern RAM Control	0	0	PPY		PZCY				0	0	PPX		PZCX			
06		0	0	PSY	0	0	0	0		0	0	PSX		0	0	0	0
07		0	0	PEY		PZY				0	0	PEX		PZX			

A pattern size less than 4 by 4 logical pixels can be specified (minimum is 1 by 1 logical pixel) as shown below. In this example a 2 by 4 logical pixel pattern is specified by setting PSX = 1, PSY = 0, PEX = 2 and PEY = 3.

	13	14	
	9	10	
	5	6	
	1	2	

As in color register indirect modes, normally one color is repeatedly assigned to the 16-bit color data depending on the number of bits per pixel. For example, when 4-bits/pixel mode is used, and color data for a pixel is 0001, a 16-bit data on the Patttern RAM should be as follows:

0 0 0 1	0 0 0 1	0 0 0 1	0 0 0 1

This prevents color change due to change of dot address.

6.6.3 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.

The following examples show execution of a CRCL (Circle) command using the 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.

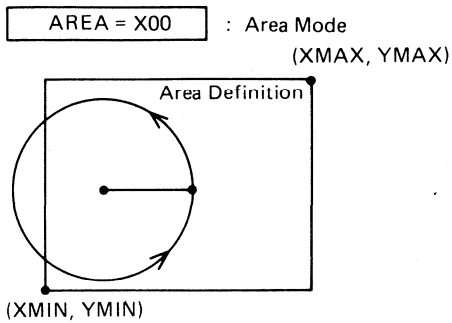


Figure 6.12(a) Area Mode = X00

Drawing is executed without area checking.

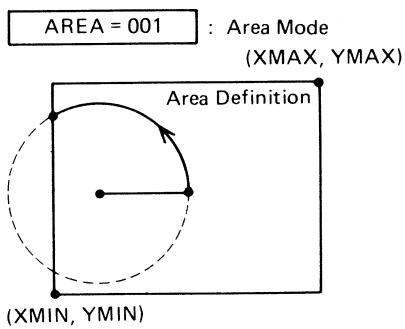


Figure 6.12(b) Area Mode = 001

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 command is terminated after setting the ABT (Abort Bit). Then the FIFO data is cleared and the status flag is reset to initialize the drawing processor.

Note) In case of ABT termination, CP stops by ABT set, but the final point of CP (end point: Pe) may slightly exceed the boundary of the defined area.

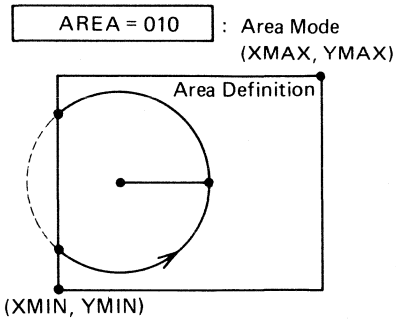


Figure 6.12(c) Area Mode = 010

Wheh 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 flag) in the Status Register (SR) is set to '1'. The ARD (Area Detect flag) in the Status Register is not set to '1' at any time during the drawing command execution regardless of whether CP is inside or outside the defined area.

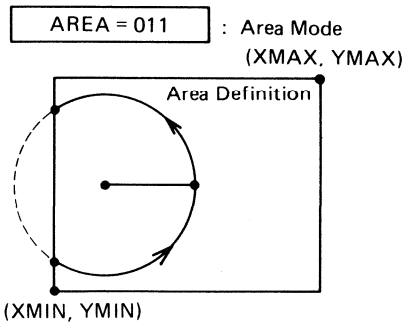


Figure 6.12(d) Area Mode = 011

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 command execution, CP goes outside the defined area, the ARD (Area Detect flag) in the Status Register (SR) will be set to '1'. The ARD can be monitored to determine if the CP is outside the defined area.

Note: In this mode, the ARD is set at every operation for each pixel when the CP is outside the defined area.

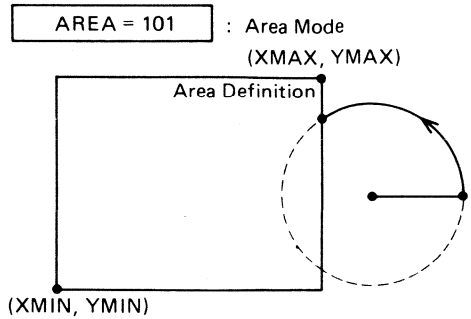


Figure 6.12(e) Area Mode = 101

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 command is terminated after setting the ABT (Abort Bit). Then the FIFO data is cleared and the status flag is reset to initialize the drawing processor.

Note: In case of ABT termination, CP stops by ABT set, but the final point of CP (end point: Pe) may slightly exceed the boundary of the defined area.

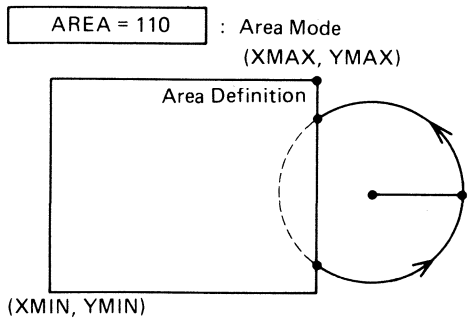


Figure 6.12(f) Area Mode = 110

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) flag in the Status Register (SR) is set to '1'. The ARD (Area Detect flag) in the Status Register is not set to '1' at any time during the drawing command execution regardless of whether CP is inside or outside the defined area.

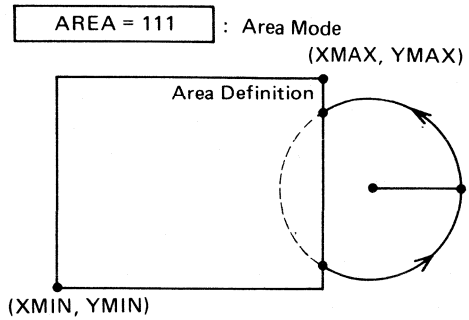


Figure 6.12(g) Area Mode = 111

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 command execution, CP goes inside the defined area, the ARD (Area Detect flag) in the Status Register (SR) will be set to '1'. The ARD can be monitored to determine if the CP is inside the defined area.

Note: In this mode, the ARD is set at every operation for each pixel when the CP is inside the defined area.

6.7 Graphic Drawing Processor

ACRTC Graphic Drawing is performed in units of logical pixels which may be programmed to consist of 1, 2, 4, 8 or 16 physical bits in the frame buffer.

In order to draw, the ACRTC Drawing Processor uses three operation control units.

(a) Drawing Algorithm Control Unit
Interprets graphic commands and parameters and executes the appropriate microprogrammed drawing algorithm. Note that this unit calculates coordinates using logical pixel X-Y addressing.

(b) Drawing Address Generation Unit
Converts logical X-Y addresses from the Drawing Algorithm Control Unit to a bit address in the frame buffer. The frame buffer is organized as sequential 16 bit words. The bit address consists of 20 bits (1M word address space) and bits 0-4 specifying the logical pixel bit address within the physical frame buffer word.

(c) Logic Operation Unit
Using the address calculated in (a) and (b), performs logical operations between the existing (read) data in the frame buffer and the drawing pattern in the Pattern RAM, and rewrites the results into the frame buffer.

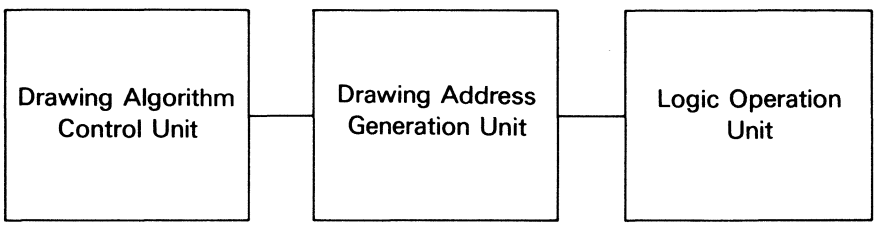


Figure 6.13 Drawing Processor

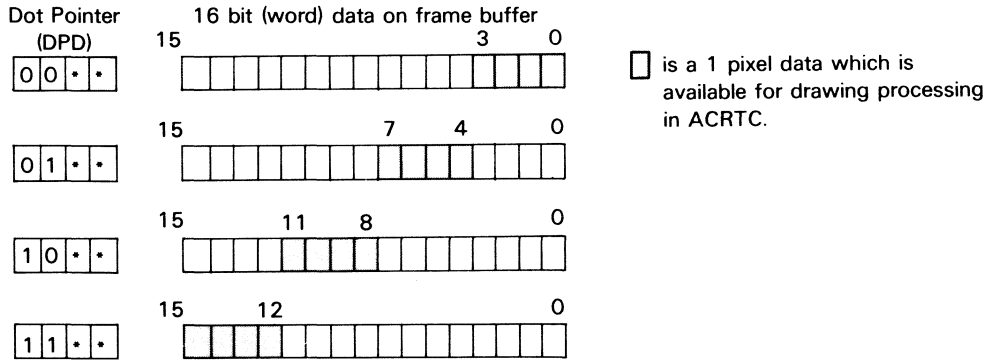
The ACRTC performs various operations as the graphic display processor. The precise graphic drawing is easily provided by accessing the graphic data using a pixel which is the minimum unit of the data contents (bit-mapped method).

The graphic drawing operation with the color images (multi-color tone) can be easily executed in high speed dealing with each pixel data.

The ACRTC sends/receives the data to/from the frame buffer by the 16 bits. 1-pixel data can be presented with maximum 16-bit data, which means that up to 65536 colors displaying is available.

Figure 6.14 (b) shows 1-word data configurations in 5 kinds of GBM (Graphic Bit Mode) and the linear address (dot address) which specifies the data location. Upper 20 bits specify the word address of the screen memory, and the lower 4 bits specify the dot address in each word.

○ Relationship between Dot Pointer (DPD) and directed pixel data
Relationship between Dot Pointer (DPD) and directed pixel data at 4 bits/pixel mode is shown below as an example.



As shown in the figure above, the dot pointer is automatically set by referring to the bits of pixel data depending on the graphic bit mode, and the graphic drawing operation is executed only for the bits corresponding with "1" in the dot pointer. The dot pointer functions like the mask register.

The dot pointer functions like the mask register.

The bits of pixel data is also controlled automatically in the ACRTC, and users can be free from setup of them.

Bit Mode	Data (bit) per pixel	Color or color image number	Number of pixels per word
1 bit/pixel	1	1	16
2 bit/pixel	2	4	8
4 bit/pixel	4	16	4
8 bit/pixel	8	256	2
16 bit/pixel	16	65536	1

Figure 6.14(a) Bits per Pixel

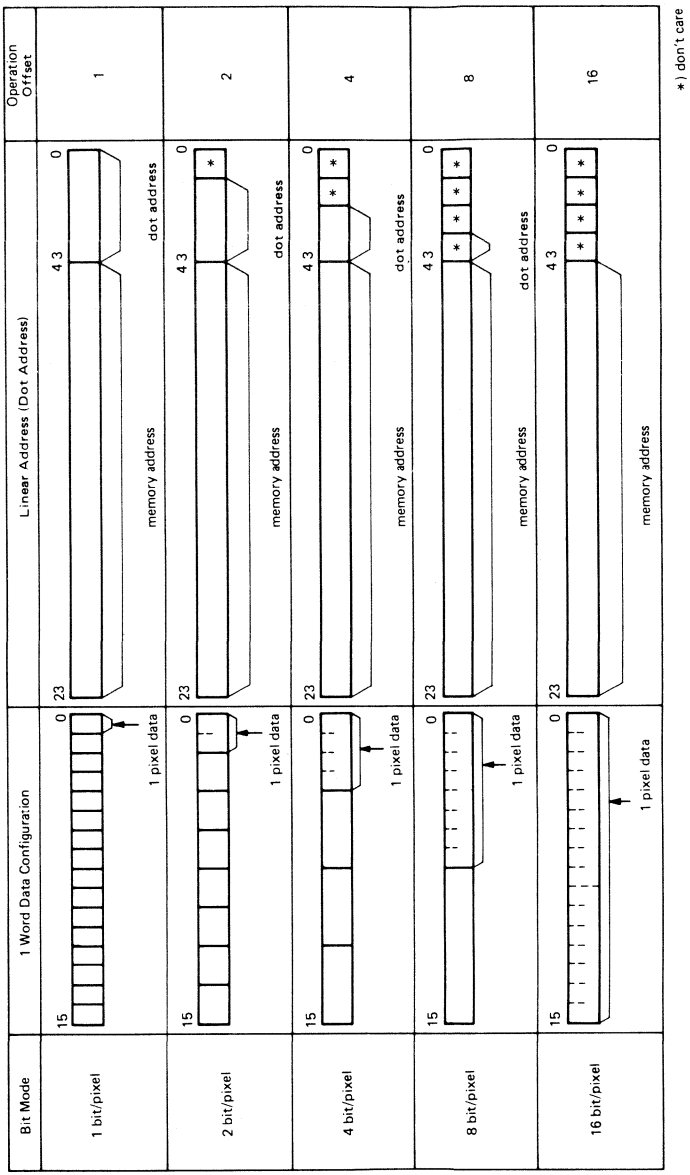


Figure 6.14(b) Pixel Physical Address Specification

(a) 1 bit/pixel mode
(GBM=000)

(b) 2 bit/pixel mode
(GBM-001)

(c) 4 bit/pixel mode
(GBM=010)

(d) 8 bit/pixel mode
(GBM-011)

(e) 16 bit/pixel mode
(GBM=100)

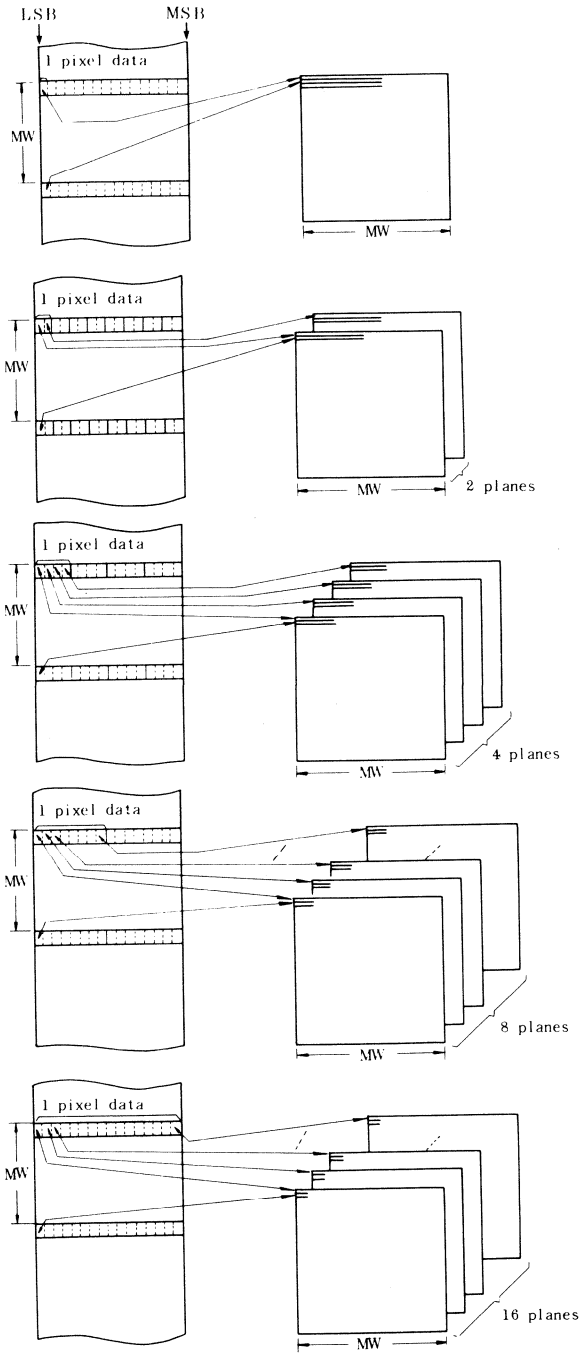


Figure 6.14(c) Logical/Physical Addressing

6.8 Graphic Drawing Operation

The ACRTC performs graphic drawing based on the unit of logical pixel including color information.

I.E., since a logical pixel can consist of multiple bits of frame buffer, a logical pixel is said to contain color information. If a logical pixel is defined as 4 bits or frame buffer, 16 tones of gray scale or 16 colors can be associated with the logical pixel.

6.8.1 Pattern RAM

The 16 by 16 bit Pattern RAM contains the color pattern for graphic drawing. A sub-pattern to be used can be specified by defining the Pattern Start X, Y and Pattern End X, Y addresses. Furthermore, a specific starting point for pattern scanning is defined with the Pattern Pointer X, Y addresses.

6.8.2 Drawing Operation of Each Pixel

For example, Color Regiser Indirect Drawing Mode is used.

Before the drawing color data. This data can be accessed by the MPU using the WPTN (write Pattern RAM) commands. Also, the Drawing Parameter Registers must be initialized using the WPR (write Parameter Register) commands.

After the drawing command is issued, the ACRTC reads the 16 bit word in the frame buffer whose address was calculated by the Drawing Algorithm Control Unit (DACU) and Drawing Address Generation Unit (DAGU). Since the ACRTC reads 16 bit word from the frame buffer; in the case of 4 bits/logical pixel, 4 pixels are read at one time. However, because the drawing can be performed in units of only one pixel, the ACRTC has a Drawing Pointer Dot Address (DPD) which is used to direct appropriates bits and to mask other bits. In Figure 6.15, the logical pixel is bits 4-7 (Cc) of the word.

The DPD mask is also applied to the color registers to select one logical pixel (in this case 4 bits) of color information (C0 and C1).

Depending on the bit value in the Pattern RAM pointed to by Pattern Pointer X and Pattern Pointer Y, the color register is selected. If 0, Color Register 0 is used, if 1, Color Register 1 is used.

The fetched data (Cc) and selected color data (C0 and C1) are logically operated on based on 1 of 8 logical operation modes (OPM) specified with the drawing instruction. the resulting drawing data (Cy) is rewritten to the frame buffer.

Drawing Color Information is normally specified as the 'background' color in CL0 and the 'foreground' or 'drawing' color in CL1. In this case, a dashed line can be drawn with simple manner by loading the dash pattern (0's for OFF, 1s for ON) into the Pattern RAM.

Note in this example (4 bits/pixel) that the color values C0 and C1 are normally repeated in the other three 4 bit subfields of the color register so that, as DPD varies, the same colors will be used. However, there is no restriction in this regard. Each of the four 4 bit logical pixel subfields of Color Register 0 and 1 could be loaded with a different color value. Thus, as IDP varies, different colors will be selected from CL0 and CL1.

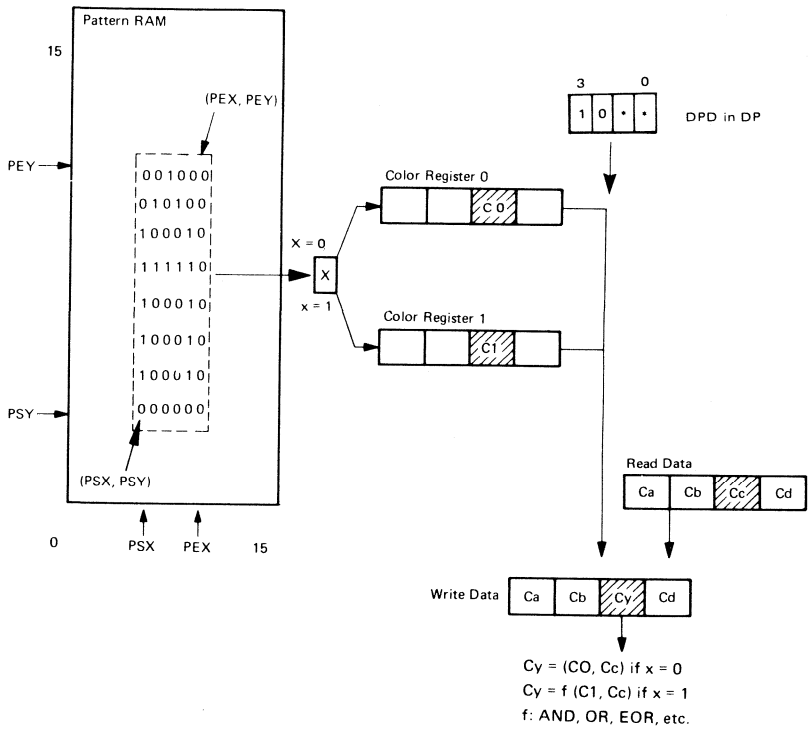


Figure 6.15 One Pixel Drawing Operation

6.8.3 Line Drawing Operation

The following describes an example of the LINE command using Color Register Indirect Drawing Mode and the 'Replace' operation mode.

For line drawing, the drawing pattern is limited to the 16 bit word pointed to by Pattern Pointer Y (PPY). A portion of the word can be extracted based on Pattern Start X and Pattern End X (PSX and PEX).

For the first pixel of the line, the Pattern RAM bit at PPX is used and Color Register 0 and 1 are selected based on this bits value. The selected color is drawn. Then the Pattern RAM pointer is incremented and operation continues until PPX = PEX. Then, PPX is reset to PSX and operation continues. Note that the Pattern RAM horizontal scanning direction is independent of pixel drawing direction.

The drawing pattern can be magnified using the Pattern Zoom Factor (PZX and PZY). For line drawing, only PZX is applicable. The example uses PZX = 0 which is 'by 1' magnification. If PZX = 1 (by 2 magnification) was specified, the selected portion of the Pattern RAM would have each bit scanned twice. Thus, the example '1111101010' pattern would be interpreted as '1111111111001100' during scanning.

6.8.4 Plane Drawing Operation

Figure 6.17(b) shows a Plane drawing example which also uses Color Register Indirect Drawing Mode and Replace Operation Mode.

For plane drawing commands (AFRCT, RFRCT, PAINT and PTN) a two dimensional portion of the Pattern RAM bounded by PSX,PSY and PEX,PEY is used. Pattern scanning starts at PPX and PPY. As each pixel is drawn, the Pattern scanning point is incremented independent of pixel drawing direction. The two dimensional pattern can be independently magnified (i.e. each pattern point repeatedly scanned) in the X and Y directions using the Pattern Zoom Factor (PZX, PZY).

Classification	Applicable Commands
Line drawing command	ALINE, RLINE, ARCT, RRCT, APLL, RPLL, APLG, RPLG, CRCL, ELPS, AARC, RARC, AEARC, REARC, DOT
Plane drawing command	AFRCT, RFRCT, PAINT, PTN

Figure 6.16 Line and Plane Drawing Commands

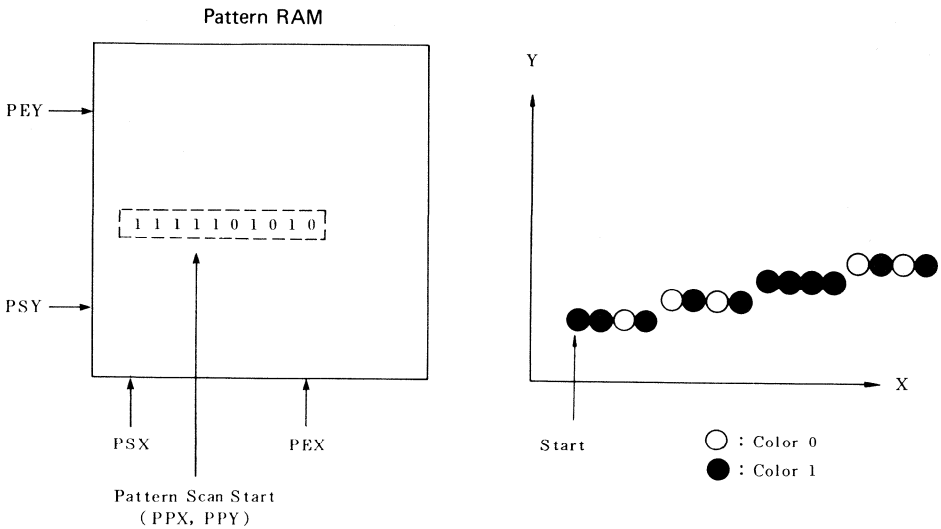


Figure 6.17(a) Line Drawing Example

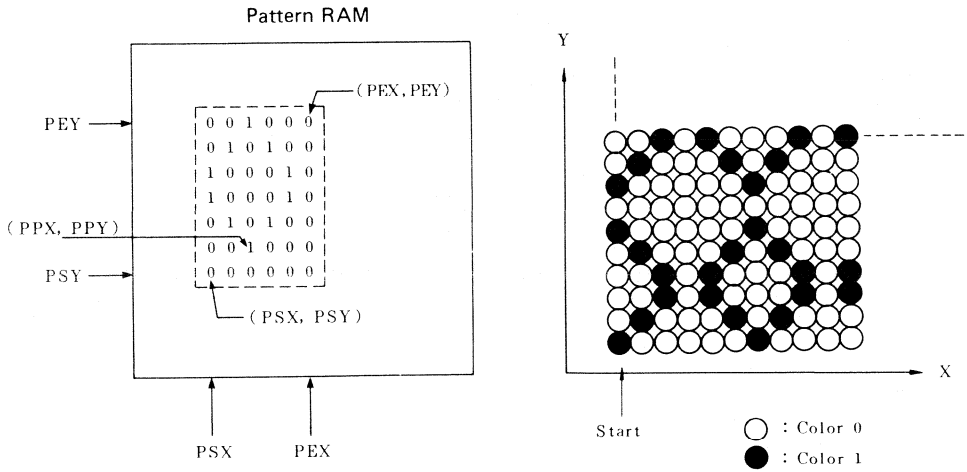


Figure 6.17(b) Plane Drawing Example

6.9 Relation Between the Logical Address and the Physical Address

Captioned relation is shown below with refering to an example that the ACRTC is set to 4 bits/pixel mode to display 1 pixel by means of 4 bits.

Figure 6.18 shows the relation between the logical address and the physical address on the frame buffer defined as 640 × 400 pixels.

The physical address is arranged into two dimensions corresponding to the logical address by setting the physical address 31920 ($07CB0) as the origin point with ORG command and "MW=160" (160 × 16/4 = 640 pixels).

Once the two-dimensional coordinates is defined by ORG command as shown in the figure below, it is preserved inside the ACRTC until the definition is renewed by another command, and complicated control of memory address is not required.

As shown in the figure, the physical address is arranged also outside the area defined by ORG and MW. Inside the defined area, the physical address is arranged in the linear address format, and outside the area, there also exist the physical addresses sequential to that of inside the area. Thus the same addresses are arranged both inside and outside the area.

To secure X-Y coordinate area which is not influenced by the physical address, use Area Mode explained in 6.6.3.

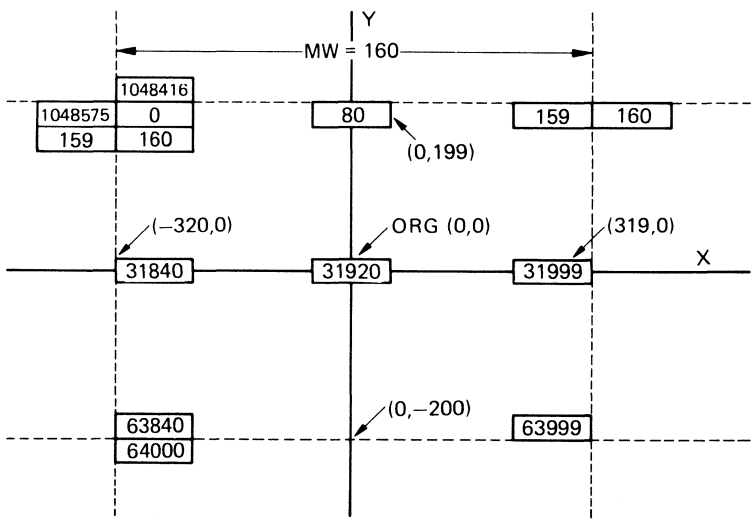


Figure 6.18 Logical Address and Physical Address

6.10 Valid Range of Current Pointer (CP)

Valid range of CP (CPx, CPy):

-32768 ($8000) $\leqq$ CPx $\leqq$ 32767 ($7FFF)

-32768 ($8000) $\leqq$ CPy $\leqq$ 32767 ($7FFF)

Figure 6.19 (a) shows the relation between the valid range of CP and actual memory.

The small square below the figure shows the frame buffer size realizable by 2M bytes in 4 bits/pixel mode. In this case, the valid range of CP is 1024 times as large as the square size (2048 × 2048 pixels).

Note: Avoid that CP moves outside the above-mentioned range during the drawing operation. Otherwise, CP doesn't operate properly. For example, a drawing operation shows in Figure 6.19 (b) cannot be done.

Applicable Commands: CRCL, ELPS, ARC, EARC, RFRCT, PAINT, PTN, GCPY, and every other relative command.

In this connection, even through an area definition is issued, the same fault occurs when CP moves outside the valid range.

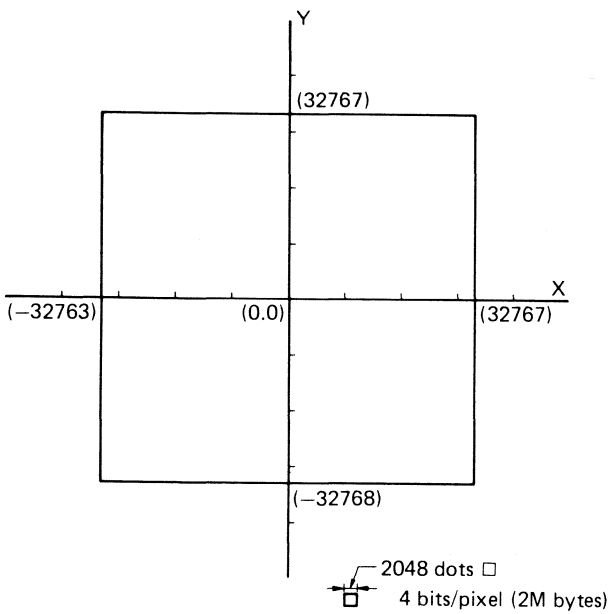


Figure 6.19(a) Valid Range of CP

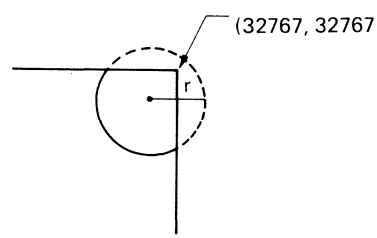


Figure 6.19(b) Setup Prohibited Sample: CRCL Command
(As the part of a circle is outside the valid range, the circle cannot be drawn)

6.11 Drawing Pointer (DP)

6.11.1 Drawing Pointer Operation

○ Specifiable range
$0 \leqq (DPH + DPL + DPD) \leqq \$FFFFFF$

○ Valid range
$0 \leqq DP \leqq \$3FFFFFFF$

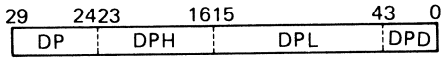


Figure 6.20 Drawing Pointer

DP internalizes 30-bit register as shown above. Upper 26 bits are valid memory address for operation, and lower 4 bits (DPD) are used to specify the physical pixel address. Actually bits 0-23 (DPD, DPL and DPH) are significant when setting parameters at ORG command, and bit 24-29 ought to be set to "0".

DP operates the address by means of 26 bits, and outputs 20 bits on MA. Therefore the drawing operation is performed as shown in Figure 6.21 without overflow even if the drawing point exceeds the area of 2M bytes.

Sample: the area of frame buffer is 2M bytes and the drawing area is not defined.

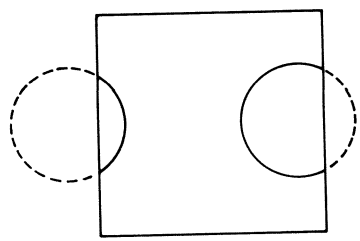


Figure 6.21 Drawing Sample

6.11.2 Area Size of Drawing Pointer

Logical space of DP is 1G bits ($2^{26} \times 16$) for DP operates the address by means of 26 bits, and its physical space is 2M bytes ($2^{20} \times 16$) for MA consists of 20 bits. The figure below show the maximum area sizes of DP which differ depending on the bit mode.

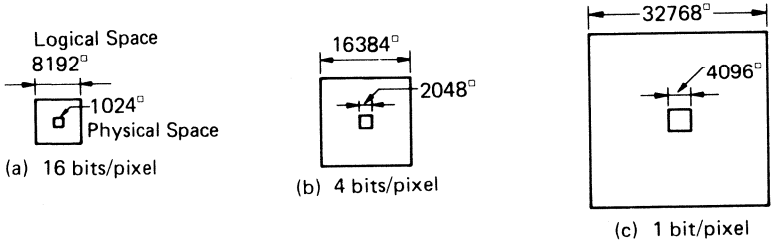


Figure 6.22 Area Size of DP

6.12 Straight Line Drawing Commands

6.12.1 Number Calculation of Drawing Pixels

Method to calculate drawing pixel count (L) is devided into two cases. When a straight line satisfies $|P_{ex} - CP_x| \geq |P_{eY} - CP_Y|$, the line command is regarded as X-direction drawing, and the line is drawn as the line $CP - P_{e1}$ shown below, by locating CP_Y for each CP_x which moves dot by dot. When $|P_{ex} - CP_x| < |P_{eY} - CP_Y|$ as the line $CP - P_{e2}$ in the figure, the drawing is operated in Y-direction.

In each case, drawing pixel count (L) can be calculated as follows.

Case 1. $|P_{ex} - CP_x| \geq |P_{eY} - CP_Y|$

$$L = |P_{ex} - CP_x|$$

Case 2. $|P_{ex} - CP_x| < |P_{eY} - CP_Y|$

$$L = |P_{eY} - CP_Y|$$

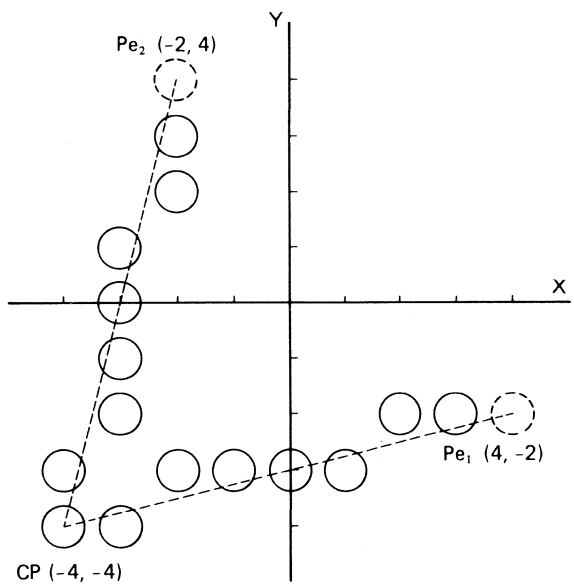
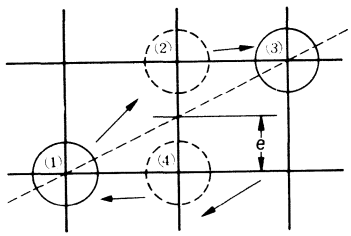


Figure 6.23 Drawing Pixels of Straight Lines

6.12.2 Notes on the Algorithm for Locating Drawing Position

As for straight line drawing commands, DDA* (Digital Differential Analyzer) is used to calculate drawing dot positions. Drawing dot positions can be located only at intersection points of the grid, and the address of the intersection point which indicates the drawing dot position is calculated by the amount of displacement (e) from the proper position to the intersection point.

When the amount of displacement (e) is 1/2, in other words, when the proper position is halfway between two intersection points, selection of the drawing dot position is done according to the drawing direction. The figure below shows an example. In case of drawing a line indicated by the broken line, the drawing order is (1) → (2) → (3) when the line is drawn upward, and (3) → (4) → (1) when the line is drawn downward. Therefore, if the positions of CP and Pe are exchanged, the result of the drawing may not be exactly the same.



* For more information of DDA, see:

W.M. Newman, R. Sproull: "Principles of Interactive Computer Graphics", 2nd Ed., McGraw-Hill, 1979.

J.D. Foley, A. Van Dam: "Fundamentals of Interactive Computer Graphics", Addison Wesley, 1982.

Figure 6.24 Drawing Dot Position

6.13 Circle and Related Drawing Commands

6.13.1 Calculation of the Number of Drawing Pixels

6.13.1.1 Circle and Arc

(a) 1/8 Circle and Whole Circle

The number of drawing pixels of 1/8 circle is calculated by $[\frac{r}{\sqrt{2}}\uparrow]$.

Note: [↑]; With sign unchanged, round up the absolute value to the integer.

When r = 8 pixels (shown in Figure 6.25(a).

The number of drawing pixels is $[\frac{8}{\sqrt{2}}\uparrow] = 6$.

Therefore, the number of the drawing pixels of a whole circle is:

$[\frac{8}{\sqrt{2}}\uparrow] \times 8 - 4 = 44$ (pixels)

Note: In case there exists a dot point on the line which forms an angle of 45° as shown in Figure 5.1 (b), the dot is added to the drawing pixel count of 1/8 circle stated above, and the equation to calculate the number is $[\frac{r}{\sqrt{2}}\uparrow] + 1$

Therefore, that of a whole circle is:

$([\frac{4}{\sqrt{2}}\uparrow + 1) \times 8 - 8 = 24$ (pixels)

In general, the number of drawing pixels of a whole circle is calculated by:

$[\frac{r}{\sqrt{2}}\uparrow] \times 8 + N$

where $\begin{cases} N = 0 \text{ in case there exists a dot on 45°-line} \\ N = -4 \text{ other cases} \end{cases}$

Note: As this counting method is a rough estimate, the number of drawing pixels calculated by the method and the actual number of pixels may be slightly different.

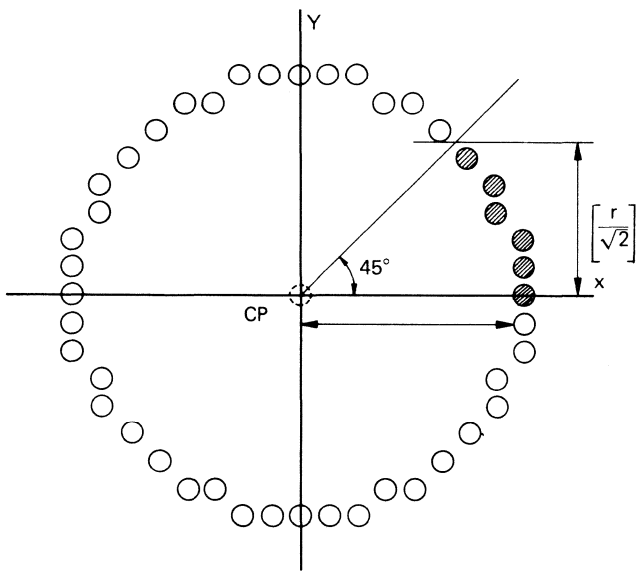


Figure 6.25(a) 1/8 Circle

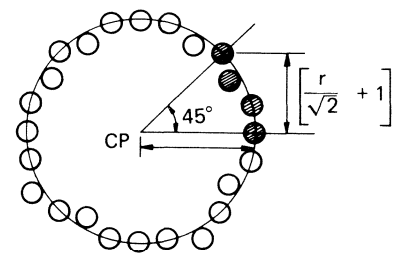


Figure 6.25(b) 1/8 Circle

(b) Arc

The number of drawing pixels of arcs can be calculated in the same method as that of 1/8 circle.

As shown in the figure below, the arc is divided into (1) and (2) by the line which forms the angle of 45° with the X-axis, and pixels are counted separately for each part.

- Number calculation in range (1)
  The number of drawing pixels of 1/8 circle (0-45°) is $[\frac{r}{\sqrt{2}}\uparrow]$, and Y-coordinate deviation of CP to Pc is "Yd".

Hence, the number of pixels of the arc is range (2) is:

$$[\frac{r}{\sqrt{2}}\uparrow] - Yd$$

- Number calculation in range (2)
  The number of drawing pixels of 1/8 circle (45-90°) is $[\frac{r}{\sqrt{2}}\uparrow]$, and X-coordinate deviation of Pe to Pc is "Xd".

Hence, the number of pixels of the arc in range (2) is:

$$[\frac{r}{\sqrt{2}}\uparrow] - Xd$$

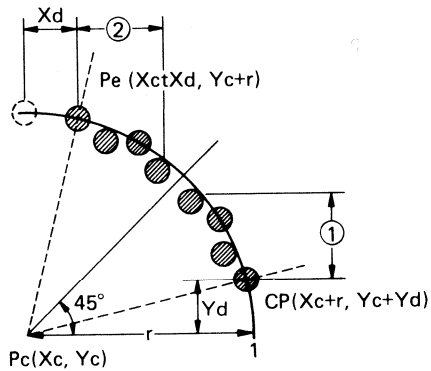


Figure 6.26(a) Arc

The total number of drawing pixels of the arc is calculated as follows:

$$[\frac{r}{\sqrt{2}}\uparrow] \times 2 - (Xd + Yd)$$

In the case of the figure above, it is $[\frac{8}{\sqrt{2}}\uparrow] \times 2 - (2 + 2) = 8$ (pixels)

This calculating method can be applied to the second, third and fourth quadrants alike.

Note) Specifying CP and Pe ranges

Shown in Figure 6.26 (b), the pixel (1) is in the range of 0°~45° and the pixel (2) is in the range of 45°~90°.

Each pixel satisfies the following condition.

(1) $X_1 > Xc + \frac{r}{\sqrt{2}}$ and $Y_1 < Yc + \frac{r}{\sqrt{2}}$

(2) $X_2 < Xc + \frac{r}{\sqrt{2}}$ and $Y_2 > Yc + \frac{r}{\sqrt{2}}$

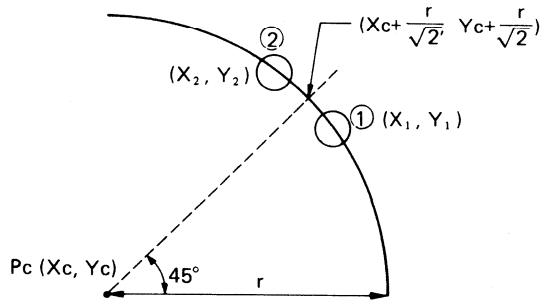


Figure 6.26(b) CP and Pe Ranges

6.13.1.2 Ellipse and Ellipse Arc

The drawing pixels of ellipse and ellipse arc can be calculated in the same method as that of circle and arc explained in 6.13.1.1. However ellipse is divided at points where lines with inclination of "+1" and "−1" touch the ellipse as shown in the following figure. Ellipse arc PS−P corresponds to 1/8 circle explaind in 6.13.1.1(a). The angle θ varies according.

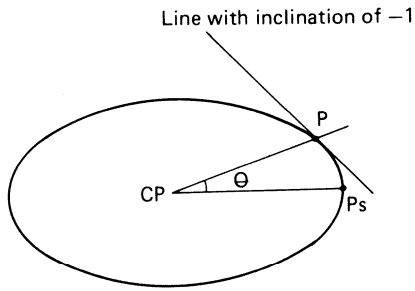


Figure 6.27 Ellipse Devide Point

Note) As this counting method is a rough estimate, the pixel count calculated by the method and the number of pixels actually drawn may be slightly different. This counting method can be applied to the second, third and fourth quadrants alike.

6.13.2 Notes on the Algorithm for Locating Drawing Position

As for circle, ellipse, arc and ellipse arc drawing commands, DDA (Digital Differential Analyzer) is used to calculate the drawing dot positions (see 6.12 Straight Line Drawing Commands). Drawing dot positions can be located only at intersection points of the grid, and the address of the intersection point which indicates the drawing dot position is calculated by the amount of displacement from the proper position to the intersection point. When the amount of displacement is 1/2, in other words, when the proper position is halfway between two intersection points, selection of the drawing dot position is done according to the drawing direction. Therefore, if bit 8 (c) for direction setup in Circle/Ellipse Command is changed, or if the positions of CP and Pe are exchanged in Arc/Ellipse Arc Command, the result of the drawing may not be exactly the same.

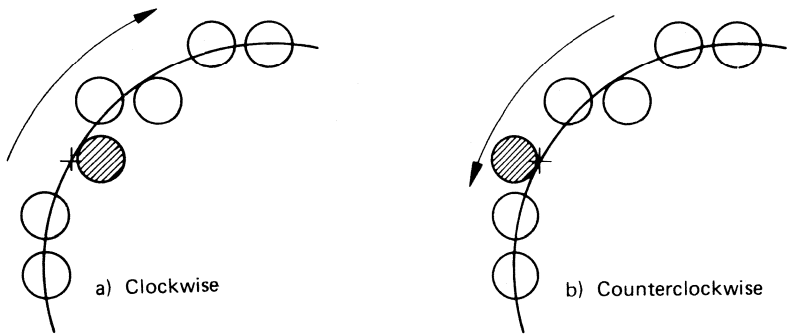


Figure 6.28 Drawing Positions of Circle/Arc

6.13.3 Figure Out Command Parameters

6.13.3.1 Arc

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 θ_1 and drawing end angle θ_2, calculate as follows (counterclockwise rotation):

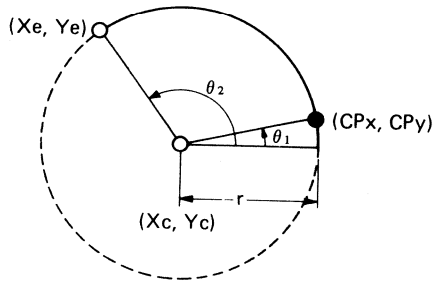


Figure 6.29(a) Arc (1)

(Parameter calculation: ① absolute addressing)

· Calculate the start point (CPx, CPy):

$CPx = Xc + [r \cos \theta_1 \updownarrow]$
$CPy = Yc + [r \sin \theta_1 \updownarrow]$

· Calculate the end point (Xe, Ye):

$Xe = Xc + [R \cos \theta_2 \downarrow]$
$Ye = Yc + [R \sin \theta_2 \downarrow]$ (where, $R = \sqrt{(CPx-Xc)^2+(CPy-Yc)^2} \fallingdotseq r$)

(Parameter calculation: ② relative addressing)

· Calculate the start point (CPx, CPy):

$CPx = Xc + [r \cos \theta_1 \updownarrow]$
$CPy = Yc + [r \sin \theta_1 \updownarrow]$ Same as in absolute addressing

· Calculate the center coordinates (dXc, dYc):

$dXc = -[r \cos \theta_1 \updownarrow]$
$dYc = -[r \sin \theta_1 \updownarrow]$

· Calculate the end point (dXe, dYe):

$dXe = dXc + [R \cos \theta_2 \downarrow]$
$dYe = dYc + [R \sin \theta_2 \downarrow]$ where, ($R = \sqrt{(CPx-Xc)^2+(CPy-Yc)^2} \fallingdotseq r$)

[Example 2] Given center coordinates (Xc, Yc), start point (CPx, CPy) and drawing angle θ, calculate as follows (counterclockwise rotation):

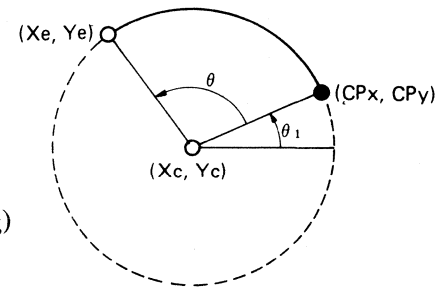


Figure 6.29(b) Arc (2)

(Parameter calculation: ① absolute addressing)

· Calculate the end point (Xe, Ye):

$$Xe = Xc + [R \cos(\theta+\theta_1) \downarrow]$$
$$Ye = Yc + [R \sin(\theta+\theta_1) \downarrow]$$

$$\left(\text{where, } R = \sqrt{(CPx-Xc)^2+(CPy-Yc)^2}\right.$$
$$\left.\theta_1 = \tan^{-1}\left(\frac{CPy-Yc}{CPx-Xe}\right)\right)$$

(Parameter calculation: ② relative addressing)

· Calculate the center coordinates (dXc, dYc):

$$dXc = Xc - CPx$$
$$dYc = Yc - CPy$$

· Calculate the end point (dXe, dYe):

$$dXe = dXc + [R \cos(\theta+\theta_1) \downarrow]$$
$$dYe = dYc + [R \sin(\theta+\theta_1) \downarrow]$$

$$\left(\text{where, } R = \sqrt{(CPx-Xc)^2+(CPy-Yc)^2}\right.$$
$$\left.\theta_1 = \tan^{-1}\left(\frac{CPy-Yc}{CPx-Xe}\right)\right)$$

[Example 3] Calculate parameters for an Arc that passes 3 points, (CPx, CPy), (X, Y) and (Xe, Ye).

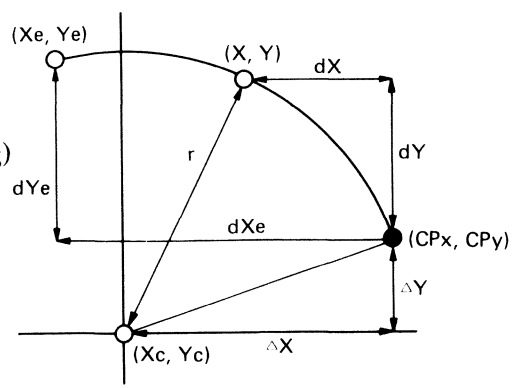


Figure 6.29(c) Arc (3)

(Parameter calculation: Relative addressing)

$$\begin{cases} dX = X - CPx \\ dY = Y - CPy \end{cases}$$

$$\begin{cases} dXe = Xe - CPx \\ dYe = Ye - CPy \end{cases}$$

· Calculate the center coordinates (dXc, dYc):

$$\begin{cases} \triangle X^2 + \triangle Y^2 = r^2 \\ (\triangle X + dX)^2 + (\triangle Y + dY)^2 = r^2 \\ (\triangle X + dXe)^2 + (\triangle Y + dYe)^2 = r^2 \end{cases}$$

where,

$$\begin{cases} dXc = [-\triangle X\uparrow] \\ \quad = [\frac{1}{2} \cdot \frac{(dX^2+dY^2)\cdot dYe-(dXe^2+dYe^2)\cdot dY}{dX\cdot dYe-dXe\cdot dY}\uparrow] \\ \\ dYc = [-\triangle Y\uparrow] \\ \quad = [\frac{1}{2} \cdot \frac{(dXe^2+dYe^2)\cdot dX-(dX^2+dY^2)\cdot dXe}{dX\cdot dYe-dXe\cdot dY}\uparrow] \end{cases}$$

Note)
[↕]: With sign unchanged, rounding the absolute value to the integer.
[↑]: With sign unchanged, round up the absolute value to the integer.
[↓]: With sign unchanged, truncate the absolute value to the integer.

6.13.3.2 Ellipse Arc

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 θ_1 and drawing end angle θ_2, calculate as follows (counterclockwise rotation):

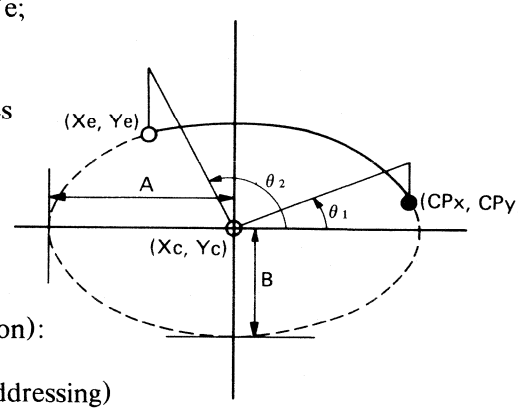


Figure 6.29(d) Ellipse Arc (1)

(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):

$$\begin{cases} CPx = Xc + [A \cos\theta_1 \updownarrow] \\ CPy = Yc + [B \sin\theta_1 \updownarrow] \end{cases}$$

· Calculate the end point (Xe, Ye):

$$\begin{cases} Xe = Xc + [\sqrt{a}\, R' \cos\theta_2 \downarrow] \\ Ye = Yc + [\sqrt{b}\, R' \sin\theta_2 \downarrow] \end{cases}$$

where $R' = \sqrt{\dfrac{(CPx-Xc)^2}{a} + \dfrac{(CPy-Yc)^2}{b}} \fallingdotseq \dfrac{A}{\sqrt{a}}$ or $\dfrac{B}{\sqrt{b}}$

(Parameter calculation: ② relative addressing)

· Calculate the start point (CPx, CPy):

$$\left.\begin{cases} CPx = Xc + [A \cos\theta_1 \updownarrow] \\ CPy = Yc + [B \sin\theta_1 \updownarrow] \end{cases}\right\} \Leftarrow \text{Same as in absolute addressing}$$

· Calculate the center coordinates (dXc, dYc):

$$\begin{cases} dXc = -[A \cos\theta_1 \updownarrow] \\ dYc = -[B \sin\theta_1 \updownarrow] \end{cases}$$

· Calculate the end point (dXe, dYe):

$$\begin{cases} dXe = dXc + [\sqrt{a}\, R' \cos\theta_2 \downarrow] \\ dYe = dYc + [\sqrt{b}\, R' \sin\theta_2 \downarrow] \end{cases}$$

where $R' = \sqrt{\dfrac{(CPx-Xc)^2}{a} + \dfrac{(CPy-Yc)^2}{b}} \fallingdotseq \dfrac{A}{\sqrt{a}}$ or $\dfrac{B}{\sqrt{b}}$

[Example 2] Given center coordinates (Xc, Yc), axial length square ratio a/b, drawing start point (CPx, CPy) and drawing angle θ, calculate as follows (counterclockwise roration):

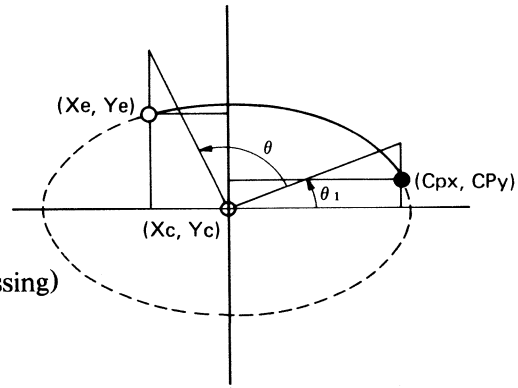


Figure 6.29(e) Ellipse Arc (2)

(parameter calculation: ① absolute addressing)

· Calculate the end point (Xe, Ye):

$$\begin{cases} Xe = Xc + [\sqrt{a}\, R' \cos(\theta+\theta_1)\downarrow] \\ Ye = Yc + [\sqrt{b}\, R' \sin(\theta+\theta_1)\downarrow] \end{cases}$$

where $R' = \sqrt{\dfrac{(CPx-Xc)^2}{a} + \dfrac{(CPy-Yc)^2}{b}}$

$$\theta_1 = \tan^{-1}\left(\sqrt{\frac{a}{b}} \cdot \frac{CPy-Yc}{CPx-Xc}\right)$$

(Parameter calculation: ② relative addressing)

· Calculate the center coordinates (dXc, dYc):

$$\begin{cases} dXc = Xc - CPx \\ dYc = Yc - CPy \end{cases}$$

· Calculate the end point (dXe, dYe):

$$\begin{cases} dXe = dXc + [\sqrt{a}\, R' \cos(\theta+\theta_1)\downarrow] \\ dYe = dYc + [\sqrt{b}\, R' \sin(\theta+\theta_1)\downarrow] \end{cases}$$

where $R' = \sqrt{\dfrac{(CPx-Xc)^2}{a} + \dfrac{(CPy-Yc)^2}{b}}$

$$\theta_1 = \tan^{-1}\left(\sqrt{\frac{a}{b}} \cdot \frac{CPy-Yc}{CPx-Xc}\right)$$

[Example 3] Calculate parameters for an ellipse arc that passes 3 points, (CPx, CPy), (X, Y) and (Xe, Ye) (axial length square ratio: a/b).

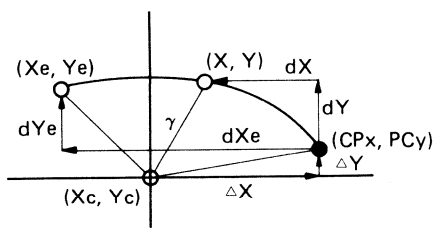


Figure 6.29(f) Ellipse Arc (3)

(Parameter calculation: Relative addressing)

$$\begin{cases} dX = X - CPx \\ dY = Y - CPy \end{cases} \quad \begin{cases} dXe = Xe - CPx \\ dYe = Ye - CPy \end{cases}$$

· Calculate the center coordinates (dXc, dYc):

$$\begin{cases} \dfrac{\triangle X^2}{a} + \dfrac{\triangle Y^2}{b} = r^2 \\ \dfrac{(\triangle X+dX)^2}{a} + \dfrac{(\triangle Y+dY)^2}{b} = r^2 \\ \dfrac{(\triangle X+dXe)^2}{a} + \dfrac{(\triangle Y+dYe)^2}{b} = r^2 \end{cases}$$

we get

$$dXc = [-\triangle X\uparrow]$$

$$= [\frac{1}{2b} \cdot \frac{(b \cdot dX^2 + a \cdot dY^2) \cdot dYe - (b \cdot dXe^2 + a \cdot dYe^2) \cdot dY}{dX \cdot dYe - dXe \cdot dY} \uparrow]$$

$$dYc = [-\triangle Y\uparrow]$$

$$= [\frac{1}{2a} \cdot \frac{(b \cdot dXe^2 + a \cdot dYe^2) \cdot dX - (b \cdot dX^2 + a \cdot dY^2) \cdot dXe}{dX \cdot dYe - dXe \cdot dY} \uparrow]$$

Note)
[↕]: With sign unchanged, rounding the absolute value to the integer.
[↑]: With sign unchanged, round up the absolute value to the integer.
[↓]: With sign unchanged, truncate the absolute value to the integer.

6.13.4 Arc with End Point Not on the Circumference

6.13.4.1 Operation of Arc Command

When the end point of an arc is not on the circumference, temporary end point is defined to draw the arc. Then, as shown in Figure 6.30, a straight line is drawn from the temporary end point to the specified end point to indicate that the specified end point is not on the circumference. In this case, the command execution normally terminates, however, it is recommended to set the end point on the circumference.

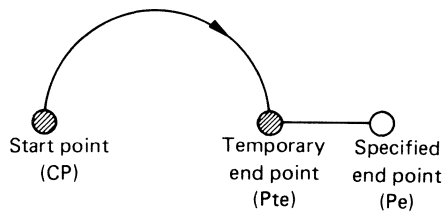


Figure 6.30 Arc Operation

6.13.4.2 Calculation of the Temporary End Point Position

Even if Pe is not specified onto the circumference, the ACRTC executes the command by calculating the position of the temporary end point. Method to calculate the position of the temporary end point varies according to in which area (area A, B, C or D in Figure 6.31) Pe is included. In Figure 6.31, a, b, c and d are where the straight lines with inclinations of "+1" and "−1" touch the circle.

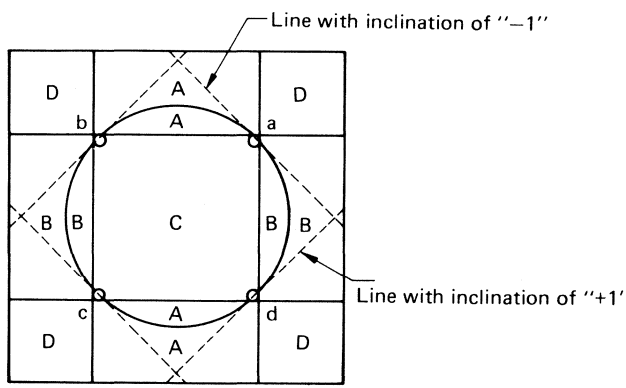


Figure 6.31 End Point Area

(a) Area A

When Pe is specified in this area, the position of Pte satisfies condition:

$$\begin{cases} X = Xe \\ Sgn\ (Y) = Sgn\ (Ye)\ (1) \end{cases}$$

Note) Sgn function (see Figure 6.32 (a))

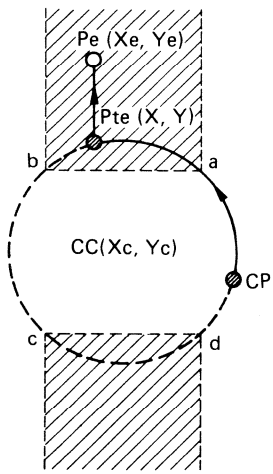
Sgn (Y) function outputs the result of comparison between Y and Yc. Sgn (Y) outputs "1" when Y is greater than or equal to Yc, and "0" when Y is smaller than Yc. As Sgn is used for internal calculation of the ACRTC, it is not provided for users.

During the arc drawing command execution, the coordinates of CP and Pe are compared each time CP moves as the drawing proceeds. In area A, the point whose X component and result of Sgn (Y) function are the same as those of Pe becomes Pte.

(b) Area B

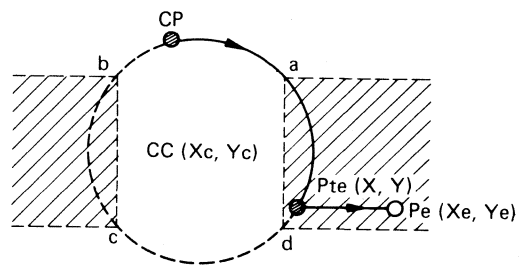
When Pe is specified in this area, the position of Pte satisfies condition:

$$\begin{cases} Sgn\ (X) = Sgn\ (Xe) \\ Y = Ye \end{cases}$$



Note: Border lines (broken lines) are not included in area A.

Figure 6.32(a) Area A



Note: Border lines (broken lines) are not included in this area.

Figure 6.32(b) Area B

(c) Area C

When Pe is specified in this area, the position of Pte varies according to the drawing direction.

For each drawing direction ((A): C.W, (B): C.C.W), arc is drawn as shown in Figure 6.32(c). Even if the positions of CP and Pe of each case (A) and (B) are the same, each Pte is located at different point.

- (A) Clockwise drawing

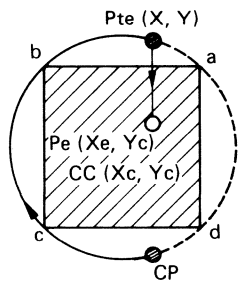
  The position of pte satisfies condition:

  $$\begin{cases} X = Xe \\ Sgn\ (Y) = Sgn\ (Ye) \end{cases}$$

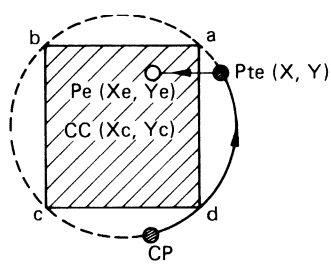
- (B) Counterclockwise drawing

  The position of Pte satisfies condition:

  $$\begin{cases} Sgn\ (X) = Sgn\ (Xe) \\ Y = Ye \end{cases}$$



(A) Clockwise



(B) Counterclockwise

Note: The border line is included in area C.

Figure 6.32(c) Area C

(d) Area D

* Setup prohibited area

Never setup Pe in this area.

If Pe is specified to this area, the command execution does not terminate.

As ACRTC does not have a function to detect erroneous parameters, Pe should be evaluated by conditions explained in 6.13.4.3.

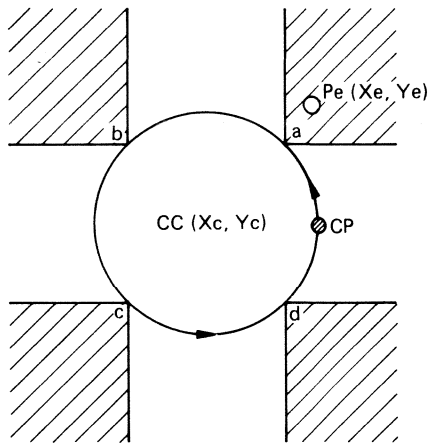


Figure 6.32(d) Area D

6.13.4.3 Setup Prohibited Area of End Point

Figure 6.33 shows setup prohibited area of Pe. Parameters of Pe should be evaluated to use Arc command. If Pe comes under any of the conditions below, change the setup value not to meet the condition.

Condition 1:

$Xe \geqq X_{45}$ and $Ye \geqq Y_{45}$

(except when $Xe = X_{45}$ and $Ye = Y_{45}$: point a)

Condition 2:

$Xe \leqq X_{135}$ and $Ye \geqq Y_{135}$

(except when $Xe = X_{135}$ and $Ye = Y_{135}$: point b)

Condition 3:

$Xe \leqq X_{-135}$ and $Ye \geqq Y_{-135}$

(except when $Xe = X_{-135}$ and $Ye = Y_{-135}$: point c)

Condition 4:

$Xe \geqq X_{-45}$ and $Ye \leqq Y_{-45}$

(except when $Xe = X_{-45}$ and $Ye = Y_{-45}$: point d)

a: $\begin{pmatrix} X_{45} = Xc + [r \cdot \cos 45° \uparrow] \\ Y_{45} = Yc + [r \cdot \sin 45° \uparrow] \end{pmatrix}$

b: $\begin{pmatrix} X_{135} = Xc - [r \cdot \cos 45° \uparrow] \\ Y_{135} = Yc + [r \cdot \sin 45° \uparrow] \end{pmatrix}$

c: $\begin{pmatrix} X_{-135} = Xc - [r \cdot \cos 45° \uparrow] \\ Y_{-135} = Yc - [r \cdot \sin 45° \uparrow] \end{pmatrix}$

d: $\begin{pmatrix} X_{-45} = Xc + [r \cdot \cos 45° \uparrow] \\ Y_{-45} = Yc - [r \cdot \sin 45° \uparrow] \end{pmatrix}$

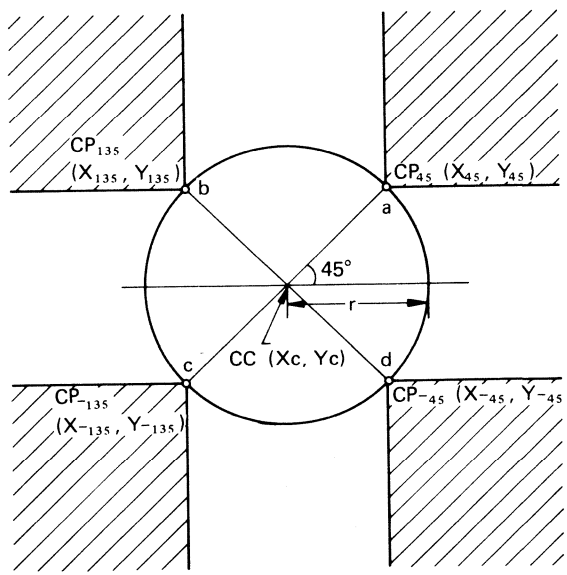


Figure 6.33 Prohibited Area

6.13.4.4 Notes on the End Point Setup Range

- ○ The end point is basically set onto the circumference.
- ○ When the end point deviates from the circumference, it is necessary to set dX and dY within the following range.

$$-16383 \leqq dX \leqq 16383$$
$$-16383 \leqq dY \leqq 16383$$

where dX : difference of X components of Pe and Pte
dY : difference of Y components of Pe and Pte

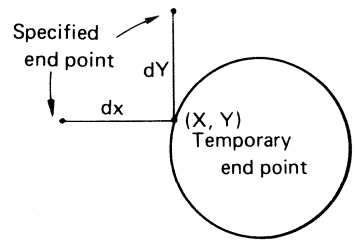


Figure 6.34 End Point Setup Range

6.13.5 Ellipse Arc with End Point Not on the Circumference

6.13.5.1 Operation of Ellipse Arc Command

When the end point of an ellipse arc is not on the circumference, temporary end point is defined to draw the ellipse arc.

Then, as shown in Figure 6.35, a straight line is drawn from the temporary end point to the specified end point. In this case, the command execution normally terminates, however, it is recommended to set the end point on the circumference.

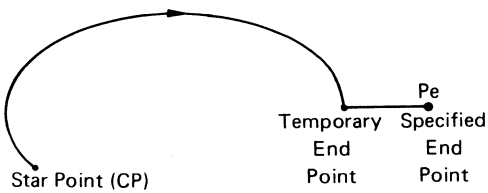


Figure 6.35 Ellipse Arc Operation

6.13.5.2 Calculation of the Temporary End Point Position

Even if Pe is not specified onto the circumference, the ACRTC executes the command by calculating the poisiton of the temporary end point. Method to calculate the temporary end point varies according to in which area of Figure 6.36 Pe is included, and the drawing direction. In the figure, points a, b, c & d indicate positions where lines with the inclinations of "+1" and "−1" touch the ellipse.

Temporary end point position is calculated in the same way as that of arc. See 6.13.4.

The positions of a, b, c and d can be calculated by the following equations.

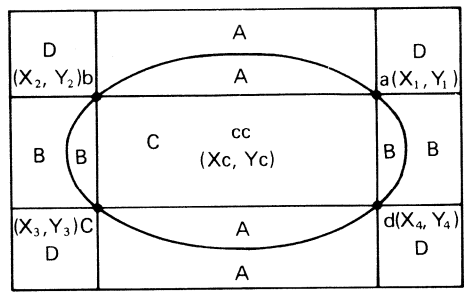


Figure 6.36 End Point Area

a. $X_1 = Xc + [\frac{Dx}{\sqrt{1+b/a}}\uparrow]$, $Y_1 = Yc + [\frac{b/a \cdot Dx}{\sqrt{1+b/a}}\uparrow]$

b. $X_2 = Xc - [\frac{Dx}{\sqrt{1+b/a}}\uparrow]$, $Y_2 = Yc + [\frac{b/a \cdot Dx}{\sqrt{1+b/a}}\uparrow]$

c. $X_3 = Xc - [\frac{Dx}{\sqrt{1+b/a}}\uparrow]$, $Y_3 = Yc - [\frac{b/a \cdot Dx}{\sqrt{1+b/a}}\uparrow]$

d. $X_4 = Xc + [\frac{Dx}{\sqrt{1+b/a}}\uparrow]$, $Y_4 = Yc - [\frac{b/a \cdot Dx}{\sqrt{1+b/a}}\uparrow]$

FUNCTION OF 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	DPH DPL	3	8
	WPR	Write Parameter Register	0 0 0 0 1 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 RN		1	6
	WPTN	Write Pattern RAM	0 0 0 1 1 0 0 0 0 0 0 0 PRA	n $D_1, \ldots, D_n$	n+2	4n+8
	RPTN	Read Pattern RAM	0 0 0 1 1 1 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 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		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 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 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	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 MM	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 AREA COL OPM	X Y	3	P·L+18
	RLINE	Relative Line	1 0 0 0 1 1 0 0 AREA COL OPM	dX dY	3	P·L+18
	ARCT	Absolute Rectangle	1 0 0 1 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 AREA COL OPM	dX dY	3	2P(A+B)+54
	APLL	Absolute Polyline	1 0 0 1 1 0 0 0 AREA COL OPM	n X1, Y1, . . Xn, Yn	2n+2	Σ[P·L+16]+8
	RPLL	Relative Polyline	1 0 0 1 1 1 0 0 AREA COL OPM	n dX1, dY1, . dXn, dYn	2n+2	Σ[P·L+16]+8
	APLG	Absolute Polygon	1 0 1 0 0 0 0 0 AREA COL OPM	n X1, Y1, . . Xn, Yn	2n+2	Σ[P·L+16]+P·Lo+20
	RPLG	Relative Polygon	1 0 1 0 0 1 0 0 AREA COL OPM	n dX1, dY1,...dXn, dYn	2n+2	Σ[P·L+16]+P·Lo+20
	CRCL	Circle	1 0 1 0 1 0 0 C AREA COL OPM	r	2	8d+66
	ELPS	Ellipse	1 0 1 0 1 1 0 C AREA COL OPM	a b DX	4	10d+90
	AARC	Absolute Arc	1 0 1 1 0 0 0 C AREA COL OPM	Xc Yc Xe Ye	5	8d+18
	RARC	Relative Arc	1 0 1 1 0 1 0 C AREA COL OPM	dXc dYc dXe dYe	5	8d+18
	AEARC	Absolute Ellipse Arc	1 0 1 1 1 0 0 C AREA COL OPM	a b Xc Yc Xe Ye	7	10d+96
	REARC	Relative Ellipse Arc	1 0 1 1 1 1 0 C AREA COL OPM	a b dXc dYc dXe dYe	7	10d+96
	AFRCT	Absolute Filled Rectangle	1 1 0 0 0 0 0 0 AREA COL OPM	X Y	3	(P·A+8)B+18
	RFRCT	Relative Filled Rectangle	1 1 0 0 0 1 0 0 AREA COL OPM	dX dY	3	(P·A+8)B+18
	PAINT	Paint	1 1 0 0 1 0 0 E AREA 0 0 000		1	(18A+102)B-58
	DOT	Dot	1 1 0 0 1 1 0 0 AREA COL OPM		1	8
	PTN	Pattern	1 1 0 1 SL SD AREA COL OPM	SZ	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] ORG (Origin)

PAGE ORG-1

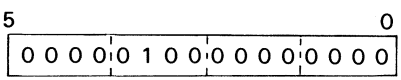
TYPE: Register Access Command

<FUNCTION>
Associates a logical X-Y screen origin with a physical frame buffer address.

<MNEMONIC>
ORG, DN, DPA, DPD

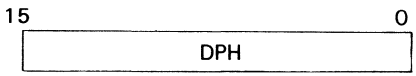
<FORMAT>

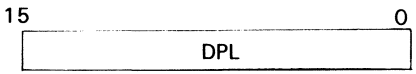
COMMAND CODE



hexadecimal notation ($ 0 4 0 0)

COMMAND PARAMETERS





WORD NUMBER
Wn=3

EXECUTION CYCLES
Cn=8

<DESCRIPTION>

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 with reference to memory width (MW) and character/graphic attributes of each screen, and sets the screen number in which to draw.

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.

The ORG command initializes the Drawing Pointer (DP) to the origin and clears the Current Pointer (CP). Therefore, in case that linear addresses of origin, or memory width (MW), or specification (character/graphic) are different between screens, ORG command must be set for each changing of screen number. On the other hand, in case that these three parameters are same between screens, ORG command is not reqired for each screen changing.

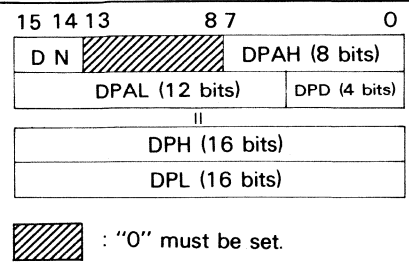
Note) In some case of setting abort bit (ABT) while executing a command, origin slides to move to an other co-ordinates. Therefore after resetting abort bit (ABT), ORG command is required to be set again.
In addition, in case of exceeding the specified parameter area of each commands, origin sometimes slides to mve in same manner. Same consideration is also required.

ORG (Origin)

DN:

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

DP:



: "0" must be set.

- The origin address of the X-Y coordinates is set with the 20-bit linear address using DPAH and DPAL.
- DPD determines the dot position in 16-bit data addressed by DPAH/DPAL. (See the following table. Take notice that there exist "don't care" bits.)
- DN sets screen number for drawing.
  Note: Set the memory width register (MWR) of the specified screen.

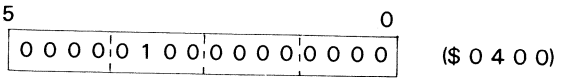
Bit Mode	Linear Address (bit address)
1 bit/pixel	23 … 4 3 0 DPAH + DPAL DPD
2 bits/pixel	23 … 4 3 0 •
4 bits/pixel	23 … 4 3 0 • •
8 bits/pixel	23 … 4 3 0 • • •
16 bits/pixel	23 … 4 3 0 • • • •

•) don't care

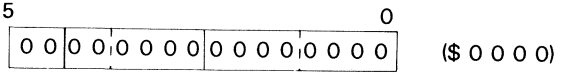
<EXAMPLE>

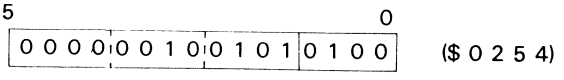
The origin for the Upper screen (screen number 0) is set to dot 2 (bits 4-7) at frame buffer word address $25. 4 bits per pixel and Memory Width (MW) = $10 are assumed.

COMMAND CODE



COMMAND PARAMETERS





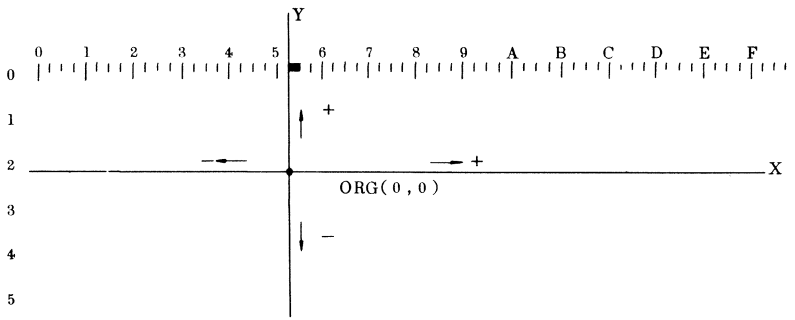


Figure C1-1 ORG Execution Example

[2] WPR (Write Parameter Register)

PAGE WPR-1

TYPE: Register Access Command

<FUNCTION>

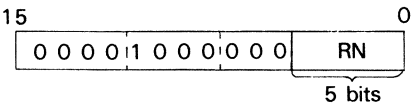
Write the contents of the Drawing Parameter Registers.

<MNEMONIC>

WPR (RN) D

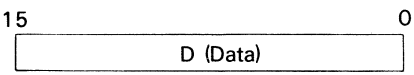
<FORMAT>

COMMAND CODE



hexadecimal notation ($ 0 8 X X)

COMMAND PARAMETERS



WORD NUMBER
Wn=2

EXECUTION CYCLES
Cn=6

<DESCRIPTION>

WPR command writes the data to the drawing parameter register.

The drawing parameter register defines the parameters for drawing, and the register number (RN) is allocated to each register.

To issue WPR command, RN of the parameter register to be written must be specified, and the data must be set to each register.

The data are sent by word (16 bits) in case of 16-bit interface.

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.

To write the data to the consecutive registers continuously, WPR command must be issued and RN must be specified repeatedly.

Note: When WPR is issued to the undefined area and to the unwritable area,

(1) If $0E, 0F, $14~$1F are specified, the command execution is normally terminated without writing to the drawing parameter register. However, the command parameter must be set up.

(2) If $10~$13 are specified, the command execution is normally terminated without writing to DP or CP. However, the command parameter must be set up. ORG command needs to be used to write to $10~$13. The data cannot be written to DP or CP directly.

<EXAMPLE>

The value $1111 is written to the CL1 (Color 1: RN = $01) 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

<DRAWING PARAMETER REGISTERS>

Register No.	Read/ Write	Name of Register	Abbr.	Data (H) 15	14	13	12	11	10	9	8	Data (L) 7	6	5	4	3	2	1	0
Pr00	R/W	Color 0	CL0	CL0															
Pr01	R/W	Color 1	CL1	CL1															
Pr02	R/W	Color Comparison	CCMP	CCMP															
Pr03	R/W	Edge Color	EDG	EDG															
Pr04	R/W	Mask	MASK	MASK															
Pr05	R/W	Pattern RAM Control	PRC	PPY				PZCY				PPX				PZCX			
↓				PSY				▨				PSX							
Pr07				PEY				PZY				PEX				PZX			
Pr08	R/W	Area Definition **	ADR	XMIN															
↓				YMIN															
				XMAX															
Pr0B				YMAX															
Pr0C	R/W	Read Write Pointer	RWP	DN	▨							RWPH							
Pr0D				RWPL								▨							
Pr0E	—	Undefined	—	—															
Pr0F				—															
Pr10	R	Drawing Pointer	DP	DN	▨							DPAH							
Pr11				DPAL														DPD	
Pr12	R	Current Pointer	CP	X															
Pr13				Y															
Pr14 ↓ Pr1F	—	Undefined	—	—															

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

[3] RPR (Read Parameter Register)

PAGE RPR-1

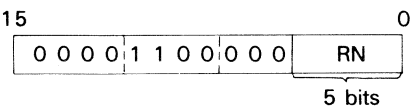
TYPE: Register Access Command

<FUNCTION>
Read the contents of the Drawing Parameter Registers.

<MNEMONIC>
RPR (RN)

<FORMAT>

COMMAND CODE



hexadecimal notation
($ 0 C X X)

COMMAND PARAMETERS
– NONE–

WORD NUMBER
Wn = 1

EXECUTION CYCLES
Cn = 6

<DESCRIPTION>

RPR command reads the data from the drawing parameter register.

The drawing parameter register defines the parameters for drawing, and the register number (RN) is allocated to each register.

To issue RPR command, RN of the parameter register to be read must be specified. The data read from the drawing parameter are successibly set to the Read FIFO.

The data are set by word (16 bits) in case of 16-bit interface.

For 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.

To read the data from the consecutive registers continuously, RPR command must be issued and RN must be specified repeatedly.

Note) When RPR command is issued to the undefined area;
If $0E, $0F, $14~$1F are specified "0" is set to the Read FIFO, therefore, the data (namely "0") must be read out from the Read FIFO.

<EXAMPLE>

The value $1111 in the Drawing Parameter Register (Color Register 1: RN=$01) is loaded into the Read FIFO.

COMMAND CODE

15 0

0 0 0 0	1 1 0 0	0 0 0 0	0 0 0 1

($ 0 C 0 1)

COMMAND PARAMETER

— NONE —

<Color Register 1>

15 0

0 0 0 1	0 0 0 1	0 0 0 1	0 0 0 1

<Read FIFO>

15 0

0 0 0 1	0 0 0 1	0 0 0 1	0 0 0 1

Figure C3-1 RPR Execution Example

<DRAWING PARAMETER REGISTERS>

Register No.	Read/ Write	Name of Register	Abbr.	Data (H) 15 14 13 12 11 10 9 8	Data (L) 7 6 5 4 3 2 1 0
Pr00	R/W	Color 0	CL0	CL0	
Pr01	R/W	Color 1	CL1	CL1	
Pr02	R/W	Color Comparison	CCMP	CCMP	
Pr03	R/W	Edge Color	EDG	EDG	
Pr04	R/W	Mask	MASK	MASK	
Pr05	R/W	Pattern RAM Control	PRC	PPY PZCY	PPX PZCX
↓				PSY ▨	PSX ▨
Pr07				PSE 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	—	Undefined	—	—	
Pr0F				—	
Pr10	R	Drawing Pointer	DP	DN ▨	DPAH
Pr11				DPAL	DPD
Pr12	R	Current Pointer	CP	X	
Pr13				Y	
Pr14 ↓ Pr1F	—	Undefined	—	—	

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

[4] WPTN (Write Pattern RAM)

PAGE WPTN-1

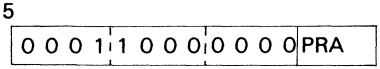
<FUNCTION>
Write data to the Pattern RAM.

<MNEMONIC>
WPTN (PRA) n, D_1, D_2, ... Dn

TYPE: Register Access Command

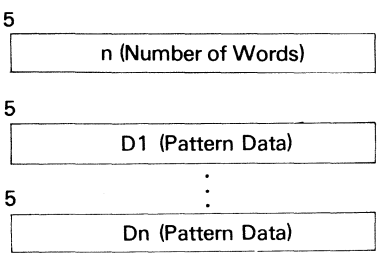
<FORMAT>

COMMAND CODE — hexadecimal notation



($ 1 8 0 X)

COMMAND PARAMETERS



WORD NUMBER
Wn=n+2

EXECUTION CYCLES
Cn=4n+8

<DESCRIPTION>

WPTN command is used to write data into the Pattern RAM.

Pattern RAM Address (PRA) of $0~$F is allocated to the Pattern RAM and each PRA represents 1 word (16 bits) of pattern data.

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).

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.)

<Note for parameters>

(1) When n = 0, the command execution is normally terminated without writing data to the pattern RAM. In this case, do not set the pattern data.
(2) When PRA exceeds $F, (PRA + n ≧ 17), it returns to $0 again and operation is repeated. Therefore, the operation does not necessarily start from $0, and duplicate writing is possible.
(3) PRA cannot be decremented. Arrange the pattern data consecutively.

<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	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	0	0	0	1	0	($ 0 0 0 2)
15															0	
0	0	1	0	0	0	1	1	0	0	0	1	0	1	0	0	($ 2 3 1 4)
15															0	
0	1	0	1	0	1	1	1	0	0	0	1	0	0	1	1	($ 5 7 1 3)

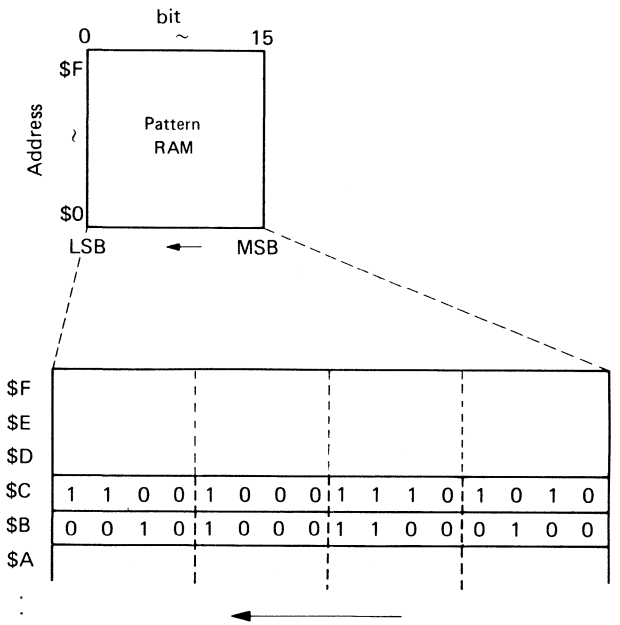


Figure C4-1 WPTN Execution Example

[5] RPTN (Read Pattern)

PAGE RPTN-1

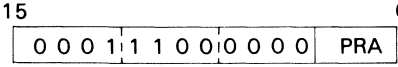
<FUNCTION>
Read Data from the Pattern RAM.

<MNEMONIC>
RPTN (PRA) n

TYPE: Register Access Command

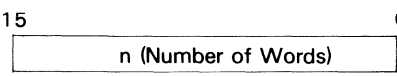
<FORMAT>

COMMAND CODE — hexadecimal notation



($1C0X)

COMMAND PARAMETERS



WORD NUMBER
Wn=2

EXECUTION CYCLES
Cn=4n+10

<DESCRIPTION>

RPTN command is used to read the data in the Pattern RAM.

Pattern RAM address (PRA) of $0~$F is allocated to the Pattern RAM and each PRA represents 1 word (16 bits) of Pattern RAM.

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.

For the 8 bit interface, 1 word of the pattern RAM is divided into high and the low bytes. [Number of words] × 2 of the pattern data is put into the Read FIFO in the order of the high byte, the low byte.

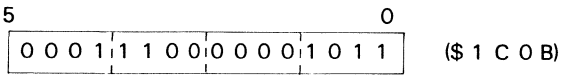
<Note for parameters>

(1) When n = 0, the command execution is normally terminated without reading data from the pattern RAM. In this case, data are not set to the Read FIFO.
(2) When PRA exceeds $F (PRA + n ≧ 17), it returns to $0 again and operation is repeated. Therefore, the operation does not necessarily start from $0, and duplicate reading is possible.
(3) PRA cannot be decremented. The pattern data are consecutively read in the ascending order.

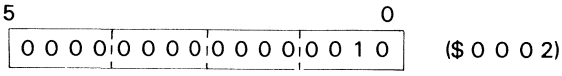
<EXAMPLE>

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



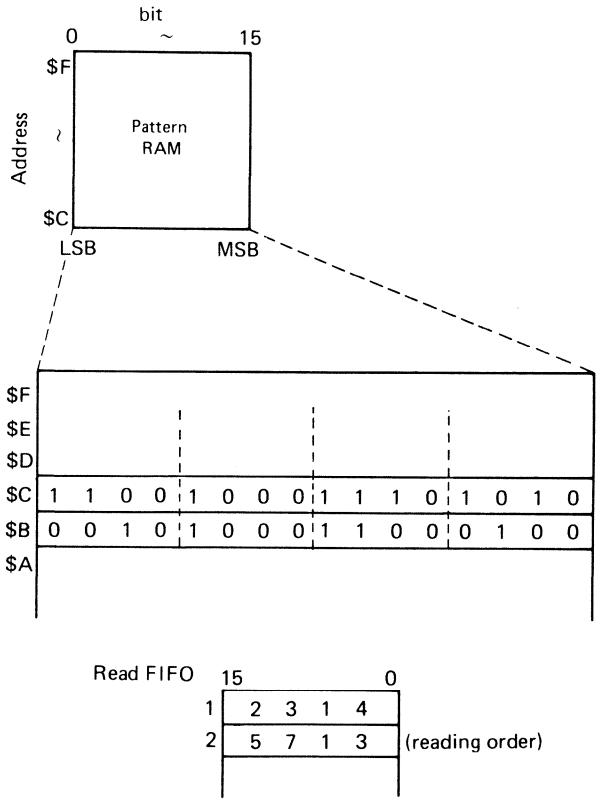


Figure C5-1 RPTN Execution Example

[6] DRD (DMA Read)

PAGE DRD-1

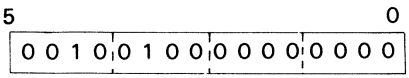
<FUNCTION>
Transfer data, by DMA transfer, from the frame buffer to the MPU system memory.

TYPE: Data Transfer Command

<MNEMONIC>
DRD AX, AY

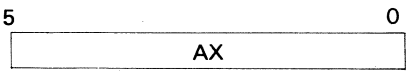
<FORMAT>

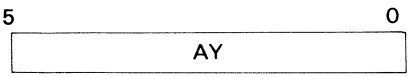
COMMAND CODE — hexadecimal notation



($ 2 4 0 0)

COMMAND PARAMETERS





WORD NUMBER
Wn = 3

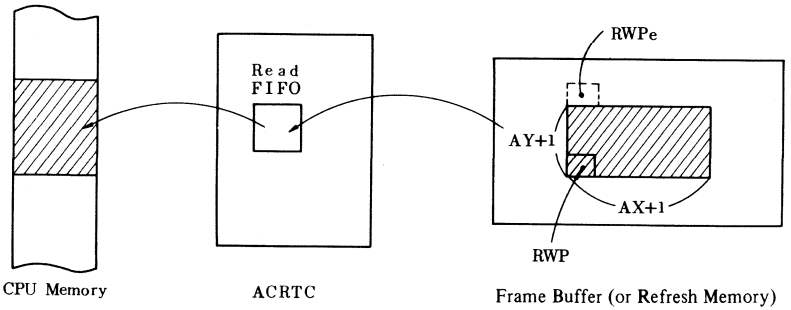
EXECUTION CYCLES

$$Cn = (4x+8)y + 12\left\lceil \frac{x \cdot y}{8} \right\rceil + (62 \sim 68)$$

$x = |AX| + 1$
$y = |AY| + 1$

<DESCRIPTION>

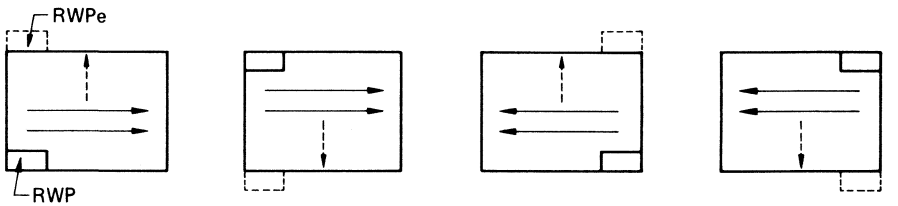
DRD command causes the ACRTC to enter DMA Data Transfer Mode in which the ACRTC will control the external DMAC to transfer data (in unit of words) from the rectangular area in the frame buffer to the MPU memory. The frame buffer data origin must be predefined in the Read Write Pointer (RWP). The parameters, AX and AY 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 (end point).



<NOTE>

The status of the ACRTC Read FIFO should be checked to insure the Read FIFO is empty before the DRD command is issued. If any data is in the Read FIFO before the DRD command issued, that data is read out incorrectly by the DMAC as the first data of the DRD command.

Reading direction



<Note for parameters>

(1) The size of the transfer area can be calculated by the following equation.
$(|AX|+1) \times (|AY|+1)$ = number of words

(2) Reading direction if set by the sign of Ax and AY (see the above figure). 2's complement is used to set the negative value.

(3) Specifiable area range:
-32767 ($8001) $\leqq AX \leqq 32767$ ($7FFF)
-32767 ($8001) $\leqq AY \leqq 32767$ ($7FFF)

[7] DWT (DMA Write)

PAGE DWT-1

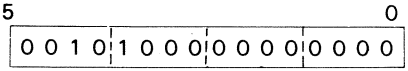
<FUNCTION>
Transfer data, by DMA transfer, from the MPU system memory to the frame buffer.

<MNEMONIC>
DWT AX, AY

TYPE: Data Transfer Command

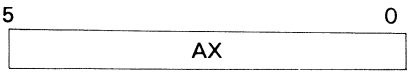
<FORMAT>

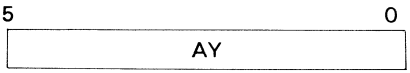
COMMAND CODE



hexadecimal notation ($2800)

COMMAND PARAMETERS





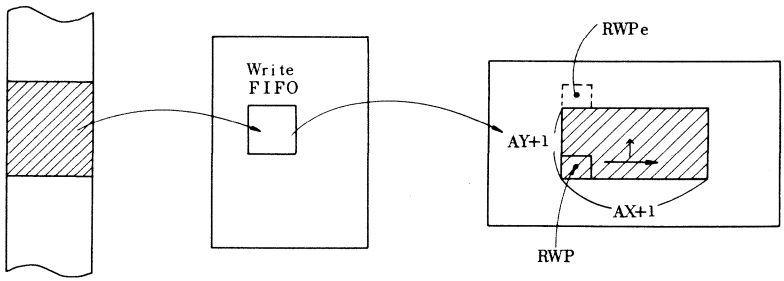
WORD NUMBER
Wn = 3

EXECUTION CYCLES
$Cn = (4x+8)y+16\left\lceil\frac{xy}{8}\right\rceil + 34$

$\begin{cases} x = |AX| + 1 \\ y = |AY| + 1 \end{cases}$

<DESCRIPTION>

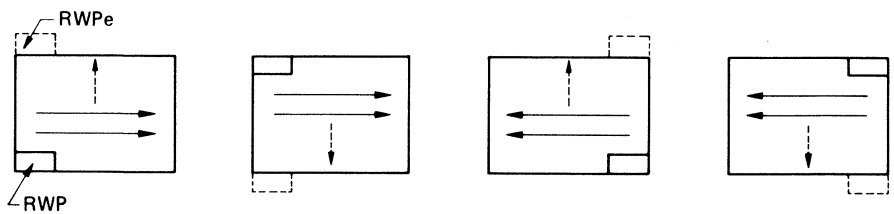
DWT command causes the ACRTC to enter DMA Data Transfer Mode in which the ACRTC will control the external DMAC to transfer data (in unit of words) from the MPU memory to the rectangular area in the frame buffer. The frame buffer data origin must be pre-defined in the Read Write Pointer (RWP). Command parameters (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 (end point).



<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) AX≧0, AY≧0 (2) AX≧0, AY<0 (3) AX<0, AY≧0 (4) AX<0, AY<0

<Note for parameters>

(1) The size of the transfer area can be calculated by the following equation.
$(|AX|+1) \times (|AY|+1) =$ number of words

(2) Reading direction is set by the sign of AX and AY (see the above figure). 2's complement is used to set the negative value.

(3) Specifiable are range:
-32767 ($8001) $\leqq AX \leqq 32767$ ($7FFF)
-32767 ($8001) $\geqq AX \leq 32767$ ($7FFF)
-32767 ($8001) $\leqq AY \leqq 32767$ ($7FFF)

<Note for the mask register>

The mask register (MASK) is not used for the DWT, so in the case the write operation with Masking is required, DMOD command needs to be used.

[8] DMOD (DMA Modify)

PAGE DMOD-1

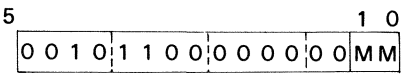
<FUNCTION>
Transfer data, by DMA transfer, from the MPU system memory to the frame buffer subject to logical modification.

TYPE: Data Transfer Command

<MNEMONIC>
DMOD (MM) AX, AY

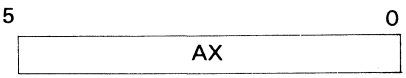
<FORMAT>

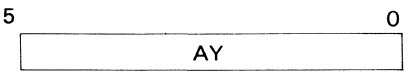
COMMAND CODE



hexadecimal notation ($ 2 C 0 X)

COMMAND PARAMETERS





WORD NUMBER
Wn = 3

EXECUTION CYCLES

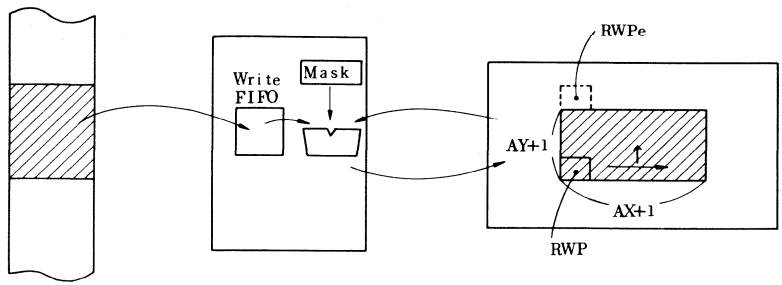
$$Cn = (4x+8)y + 16\left\lceil \frac{xy}{8} \uparrow \right\rceil + 34$$

$$\begin{cases} x = |AX| + 1 \\ y = |AY| + 1 \end{cases}$$

<DESCRIPTION>

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.

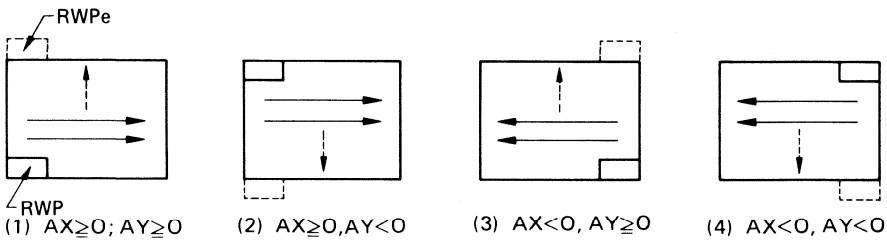
The MM (Modify Mode) field of the command code specifies the DMA data transfer modify mode. Each pixel transferred from MPU system memory is logically operated on the corresponding pixel from the frame buffer, and the result is rewritten to the frame buffer. Logic operation can be enabled and disabled on a bit by bit basis based on the contents of the MASK register.



<NOTE>

As for command issue;

Afrer DMOD is issued, no further commands should be issued until the DMA data is transferred and the DMOD command terminates.



<Note for parameters>

(1) The size of the transfer area can be calculated by the following equation.
$(|AX|+1) \times (|AY|+1) =$ number of words

(2) Modification direction is set by the sign of AX and AY (see the above figure). 2's complement is used to set the negative value.

(3) Specifiable area range
-32767 ($8001) $\leqq AX \leqq 32767$ ($7FFF)
-32767 ($8001) $\leqq Ay \leqq 32767$ ($7FFF)

DRD (DMA Read), DWT (DMA Write), DMOD (DMA Modify)

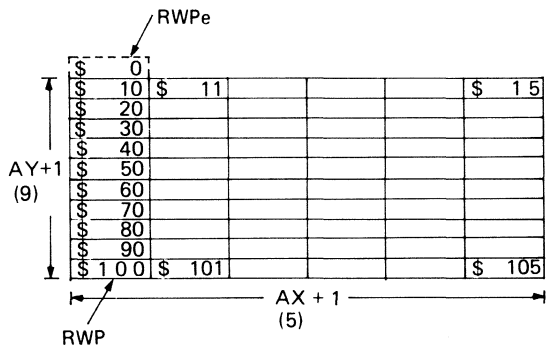
The relation between AX, AY parameters and the frame buffer memory address in DRD, DWT and DMOD.

Data transfer commands sets the rectangular area using AX and AY parameters. The relation between the frame buffer memory address and those parameters are described below (when both AX and AY are of positive value).

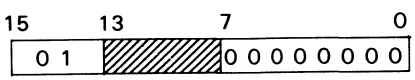
As RWP sets the screen number (DN) as well as the starting point of the transfer area, the memory width of the transfer area and the specification of the screen can be referred to by DN. Therefore, the rectangular area can be set only by specifying AX and AY. To execute the DMA data transfer, the memory width of the frame buffer and the screen number must be set to the RWP before issuing the command.

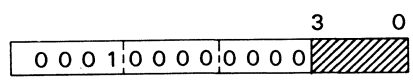
Note) Make sure to set the memory width register (MWR) in the display control register for the screen specified by DN.

Frame Buffer

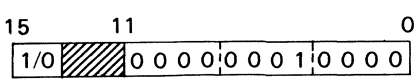


○ Read/Write Pointer (RWP)





○ Base Screen Memory Width



[9] RD (Read)

PAGE RD-1

<FUNCTION>
Read one word of data from the frame buffer specified by the read/write pointer (RWP), and load the word into Read FIFO.

<MNEMONIC>
RD

TYPE: Data Transfer Command

<FORMAT>

COMMAND CODE — hexadecimal notation

15															0	
0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	($ 4 4 0 0)

COMMAND PARAMETER

— NONE —

WORD NUMBER
Wn = 1

EXECUTION CYCLES
Cn = 12

<DESCRIPTION>

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. Then the Read FIFO ready flag is set.

The result may be read from the Read FIFO by the MPU anytime after the RD command is issued and the Read FIFO ready flag = "1" is detected. 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. At the end of the RD command execution, the ACRTC increments RWP by one.

RWP needs to be programmed prior to RD command issue.

DN:

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

RWP:

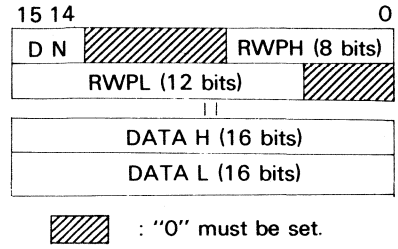


Figure C9-1 RWP Set

- RWPH and RWPL specifies the frame buffer address by setting the linear address of 20 bits.
- DN specifies screen numbers.

<**NOTE**>

As for RD command, the screen number is used to indicate the specification of the screen (graphic/character). Make sure to set the memory width register (MWR) in the display control register for the screen specified by the DN.

<**EXAMPLE**>

RD reads 1-word data from the linear address $56 of the upper screen whose memory width (MW) is set to $1C as shown in Fig. C9-2 and Fig. C9-3.

RWP

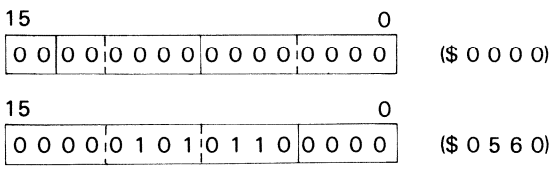
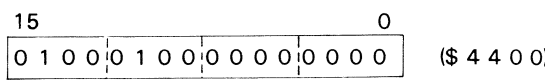


Figure C9-2 Example of RWP Setting

COMMAND CODE



COMMAND PARAMETERS

– NONE –

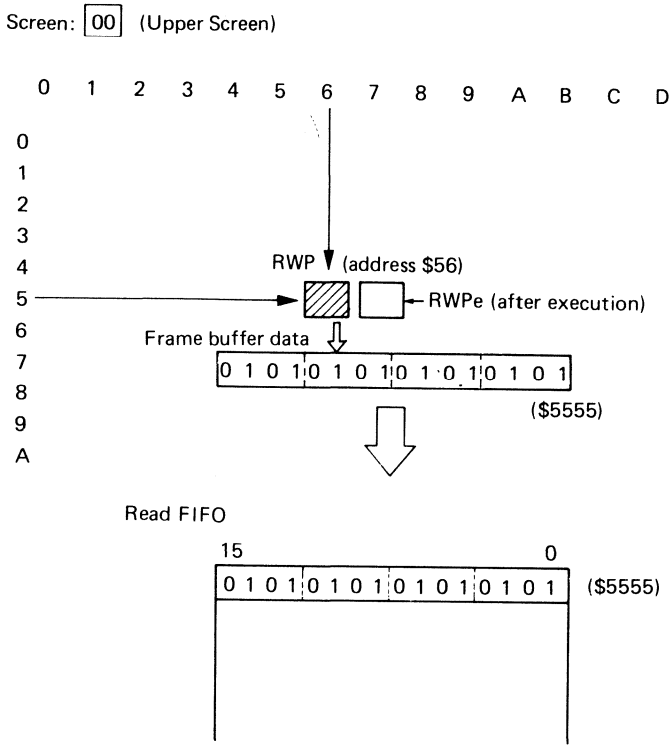


Figure C9-3 RD Execution Example

[10] WT (Write)

PAGE WT-1

<FUNCTION>
Write one word of data to the frame buffer specified by the read/write pointer (RWP).

<MNEMONIC>
WT D

TYPE: Data Transfer Command

<FORMAT>

COMMAND CODE — hexadecimal notation

15															0	
0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	($ 4 8 0 0)

COMMAND PARAMETERS

15	0
D (16 bits)	

WORD NUMBER
Wn = 2

EXECUTION CYCLES
Cn = 8

<DESCRIPTION>

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.

At the end of the WT command execution, the ACRTC increments the RWP by one.

RWP needs to be programmed before issuing the WT.

DN:

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

RWP

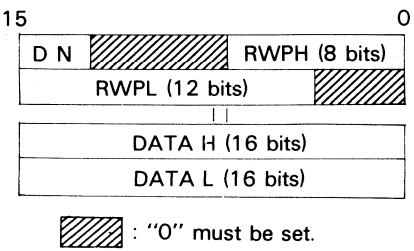


Figure C10-1 RWP Set

- The frame memory sets a 20-bit linear address separated into highorder RWPH (8 bits) and loworder RWPL (12 bits).
- Specify the Screen No. where drawing is executed.

<NOTE>

As for WT command, the screen number is used to indicate the specification of the screen (graphic/character). Make sure to set the memory width register (MWR) in the display control register for the screen specified by the DN.

<EXAMPLE>

WT writes 1-word data ($5555) to the linear address $56 of the upper screen whose memory width (MW) is set to $10 as shown in Fig. C10-2 and Fig. C10-3.

RWP

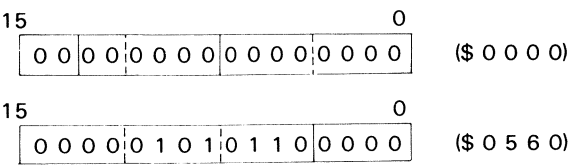
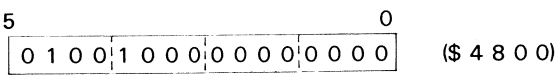
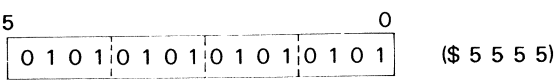


Figure C10-2 Example of RWP Setting

COMMAND CODE



COMMAND PARAMETERS



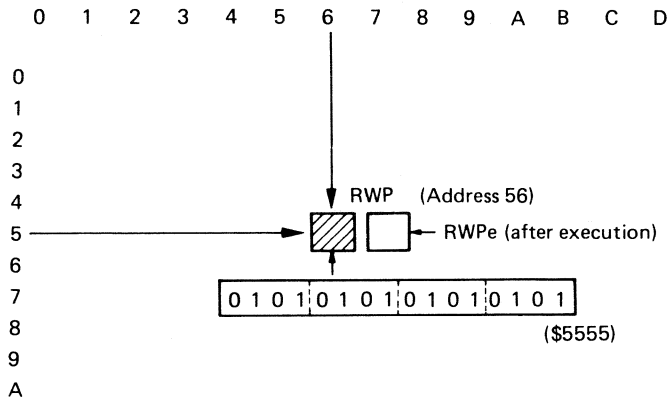


Figure C10-3 WT Execution Example

[11] MOD (Modify)

PAGE MOD-1

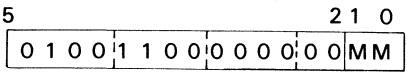
<FUNCTION>
Perform logical operation on one word in the frame buffer specified by the read/write pointer (RWP).

<MNEMONIC>
MOD (MM) D

TYPE: Data Transfer Command

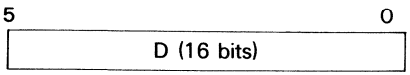
<FORMAT>

COMMAND CODE — hexadecimal notation



($ 4 C 0 X)

COMMAND PARAMETER



WORD NUMBER
Wn=2

EXECUTION CYCLES
Cn=8

<DESCRIPTION>

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).

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.

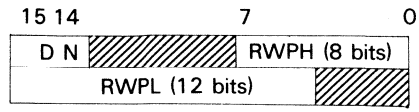
At the end of the MOD command execution, the ACRTC increments the RWP by one.

RWP needs to be programmed before issuing MOD.

DN:

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

RWP:



: "0" must be set.

Figure C11-1 RWP Set

- The frame buffer sets a 20-bit address separated into high order RWPH (8 bits) and low order RWPL (12 bits).
- Specify the Screen No. where data is modified.

<NOTE>

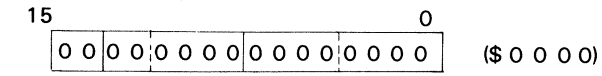
As for MOD command, the screen number is used to indicate the specification of the screen (graphic/character). Make sure to set the memory width register (MWR) in the display address control register for the screen specified by DN.

<EXAMPLE>

OR all bits of the frame buffer word ($5555) at physical address $56 with the 16-bit data word $AAAA. MM = 01 specifies OR modify mode. All bits are selected for logical operation by assuming the MASK register to $0FFF. For this example, Memory Width (MW) is assumed to be $10.

<EXAMPLE>

RWP



MASK

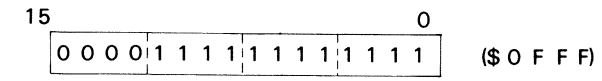
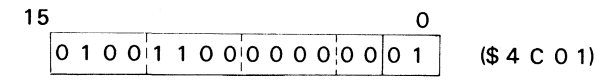
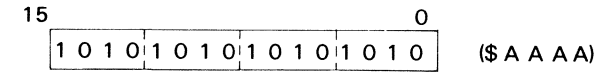


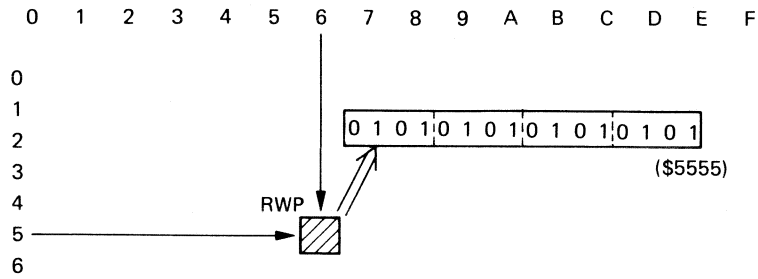
Figure C11-2 Examples of RWP and MASK Setting

COMMAND CODE

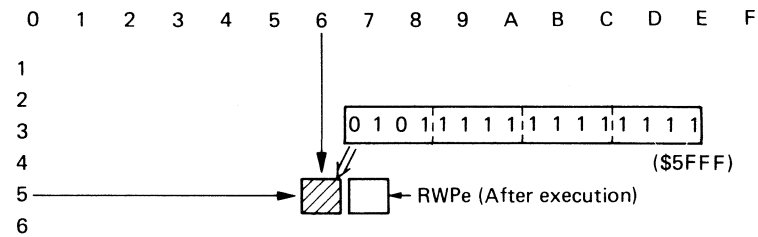


COMMAND PARAMETER





(A) MOD Command Read Cycle



(B) MOD Command Write Cycle

Figure C11-3 MOD Execution Example

[12] CLR (Clear)

PAGE CLR-1

<FUNCTION>

Clear a rectangular area of the frame buffer with a data in the command parameter.

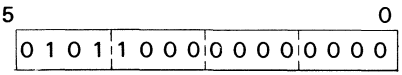
TYPE: Data Transfer Command

<MNEMONIC>

CLR D, AX, AY

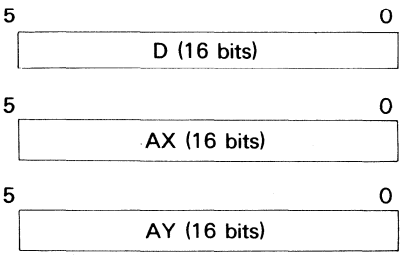
<FORMAT>

COMMAND CODE — hexadecimal notation



($ 5 8 0 0)

COMMAND PARAMETERS



WORD NUMBER

Wn=4

EXECUTION CYCLES

$Cn=(2x+8)y+12$

$\begin{cases} x = |AX| + 1 \\ y = |AY| + 1 \end{cases}$

<DESCRIPTION>

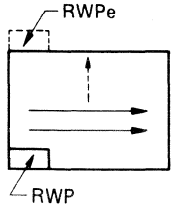
The frame buffer area defined by the physical origin (RWP) and physical frame buffer word address (AX and AY) parameters is filled with the data parameter (D).

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.

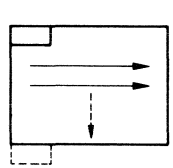
At the end of CLR command execution, RWP will be set to RWPe.

CLR (Clear)

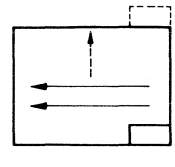
Parameters and the Area to be Cleared



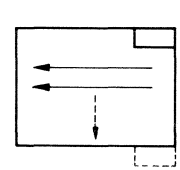
(1) AX≧0



(2) AX≧0, AY<0



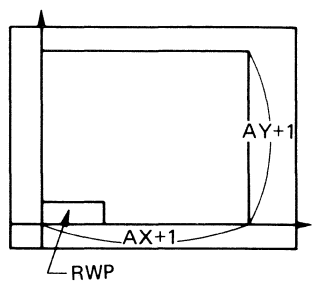
(3) AX<0, AY≧0



(4) AX<0, AY<0

<NOTE>

- 2's complement is used to set the negative value.
- Specifiable area range:
  $-32767 \leqq AX \leqq 32767$
  $-32767 \leqq AY \leqq 32767$



The area is set by parameters AX and AY as shown in the left figure. When both AX and AY are set to "0", 1 word data stored in the address indicated by RWP, is cleared.

DN:

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

RWP:

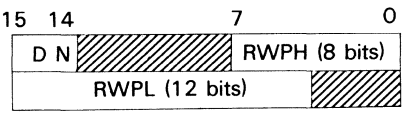


Figure C12-1 RWP Set

- 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 clearing is executed using DN.

<**EXAMPLE**>

After setting the address $56 of the upper screen (MW = $10, GBM = 4 bit/pixel) to RWP as shown in Fig. C12-2, the area specified by AX (−4 [$FFFC]) and AY (−6 [$FFFA]) is cleared with $3333. The result is shown in the Fig. C12-3.

RWP

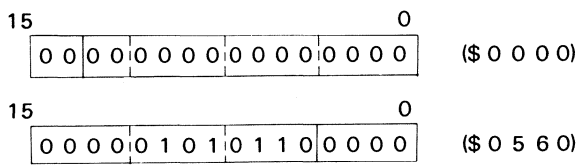
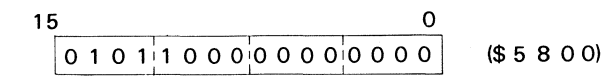
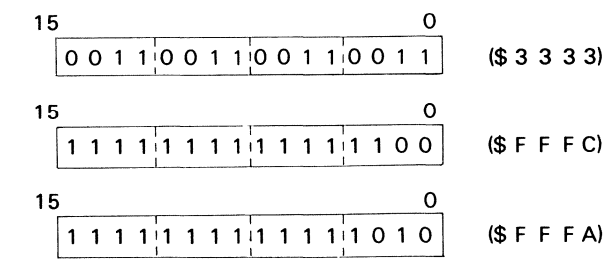


Figure C12-2 Example of RWP Setting

COMMAND CODE



COMMAND PARAMETERS



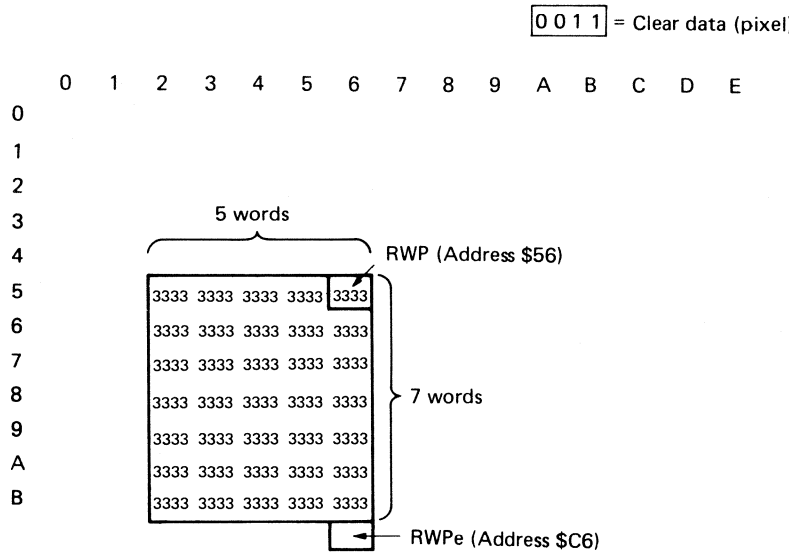


Figure C12-3 CLR Execution Example

[13] SCLR (Selective Clear)

PAGE SCLR-1

<FUNCTION>
Initialize a rectangular area of the frame buffer with 1-word data subject to logical operation.

TYPE: Data Transfer Command

<MNEMONIC>
SCLR (MM) D, AX, AY

<FORMAT>

COMMAND CODE — hexadecimal notation

15															0
0	1	0	1	1	1	0	0	0	0	0	0	0	0	M	M

($ 5 C 0 X)

COMMAND PARAMETERS

15	0
D (16 bits)	

15	0
AX (16 bits)	

15	0
AY (16 bits)	

WORD NUMBER
Wn = 4

EXECUTION CYCLES
$Cn = (4x + 6)y + 12$

$$\begin{cases} x = |AX| + 1 \\ y = |AY| + 1 \end{cases}$$

<DESCRIPTION>

The MM (Modify Mode) field of the command code specifies the data transfer modify mode.

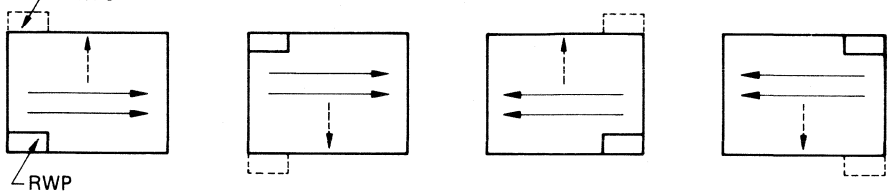
The frame buffer area defined by the RWP origin and the physical frame buffer word address (AX and AY) parameters is selectively cleared. The contents of the frame buffer are read, and that data is logically operated on with the D parameter (excepts bits masked in the MASK register) using the logical operation defined by MM. The result is rewritten to the frame buffer.

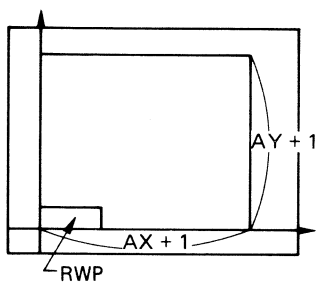
Since the ACRTC performs the selective clear using 16-bit words, multiple logical pixels (if 4 bits/pixel then 4 pixels) are cleared in one access. D is normally specified to contain multiple copies (if 4 bits/pixel then 4 copies) of the color information for a single color selective clear.

At the end of SCLR command execution, RWP will be set to RWPe.

SCLR is especially effective for the modification of the character code and the attribute code on the character screen, and for color modification on the graphic screen.

Parameters and the Area to be Cleared





Note : • 2's competent is used to set the negative value.
: • Specifiable area range

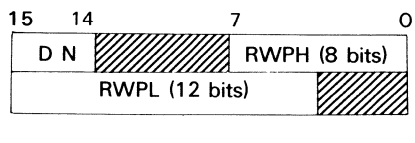
$-32767 \leqq AW \leqq 32767$
$-32767 \leqq AY \leqq 32767$

The area is set by parameters AX and AY as shown in the above figure. When both AX and AY are set to "0", 1 word data, stored in the address indicated by RWP, is modified subject to Modify Mode (MM).

DN:

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

RWP



: "0" must be set.

Figure C13-2 RWP Set

- The frame memory is a 20-bit linear address separated into high order RWPH (8 bits) and low order RWPL (12 bits).
- Specify DN to select the screen to be modified.
  Note) Because the memory width (MW) and the specification (graphic/character) of each screen checked when DN is set, presetting the memory width register (MWR) indicated by DN must be executed in advance.

<EXAMPLE>

For this example 4 bits per pixel is used, the Memory Width (MW) is $10, the MASK register contains $F0F0 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.

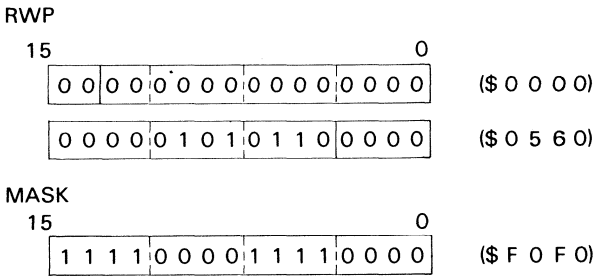


Figure C13-3 Examples of RWP and MASK Setting

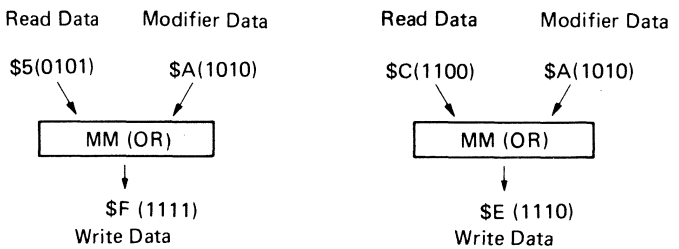
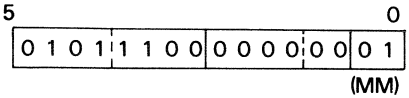


Figure C13-3 Notation of Data (Operation for 1 pixel in the 4 bits/pixel mode)

SCLR (Selective Clear)

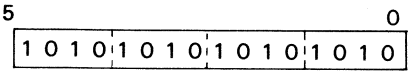
<EXECUTION EXAMPLE>

COMMAND CODE

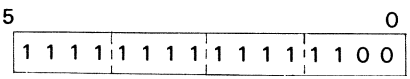


($5C01)

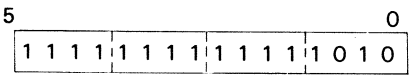
COMMAND PARAMETERS



($AAAA)



($FFFC)



($FFFA)

< EXAMPLE>

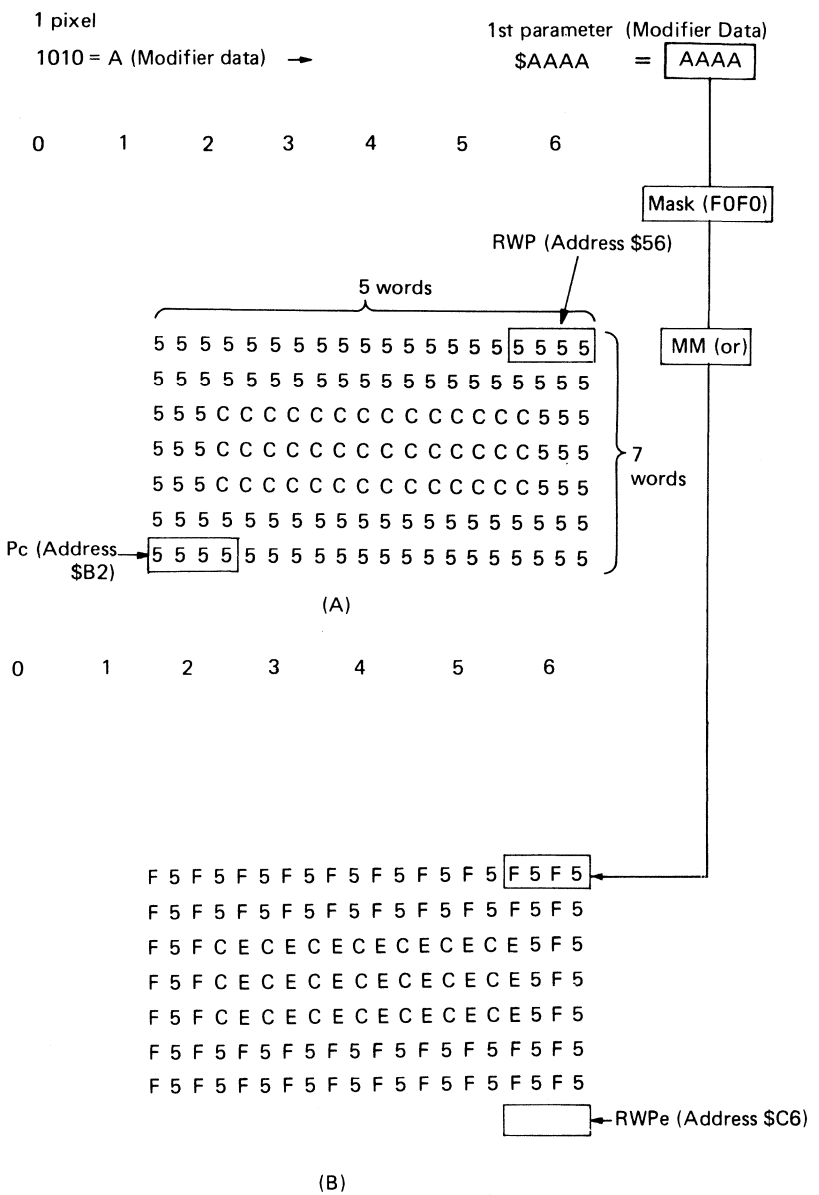


Figure C13-4 SCLR Execution Example

[14] CPY (Copy)

PAGE CPY-1

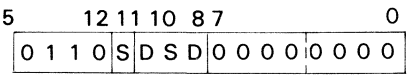
<FUNCTION>
Copy frame buffer data from one area (source area) to another area (destination area).

<MNEMONIC>
CPY (S, DSD) SAH, SAL, AX, AY

TYPE: Data Transfer Command

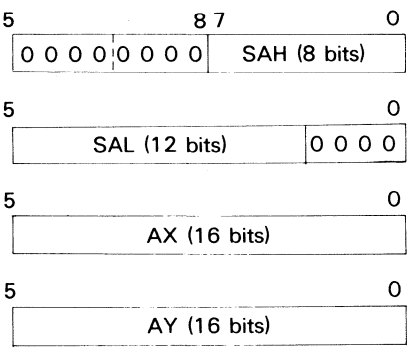
<FORMAT>

COMMAND CODE — hexadecimal notation



($ 6 X 0 0)

COMMAND PARAMETERS



WORD NUMBER
Wn = 5

EXECUTION CYCLES
Cn = (6x + 10)y + 12
$\begin{cases} x = |AX| + 1 \\ y = |AY| + 1 \end{cases}$

<DESCRIPTION>

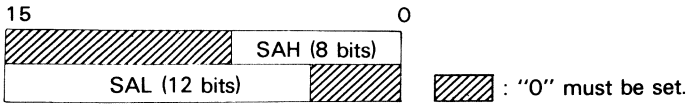
The parameters of the command define the source area. The RWP must be predefined to direct the destination area (including screen number). The source area resides in the same screen as that of the destination area as defined in RWP.

The source area is defined by the origin address (SAH/SAL) and physical frame buffer word (AX and AY) dimensions.

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.

At the end of the CPY command, RWP is set to RWPe.

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

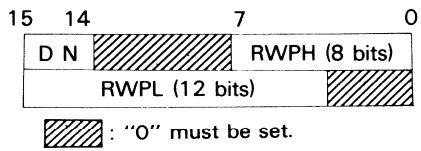


Figure C14-1 Pss and RWP Set

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 copying is executed.

<**NOTE**>

As the memory width and the specification (graphic/character) of each screen are referred to by DN, the memory width register (MWR) must be set by DN in advance. The memory width of the source and destination area need to be same.

2's complement is used to set the negative value.

- Specifiable source area range

  $-32767 \leq AX \leq 32767$

  $-32767 \leq AY \leq 32767$

<CPY Command Scan Direction>

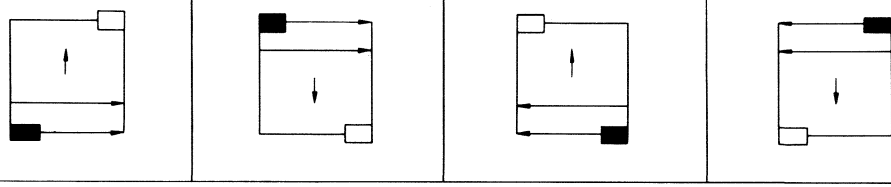
As to CPY, the direction of pointer scanning is specified in command code. (The pointer functions in the unit of word).

(a) Scanning Direction of Source Area (S: Source Scan Direction)
S is set in the command code.

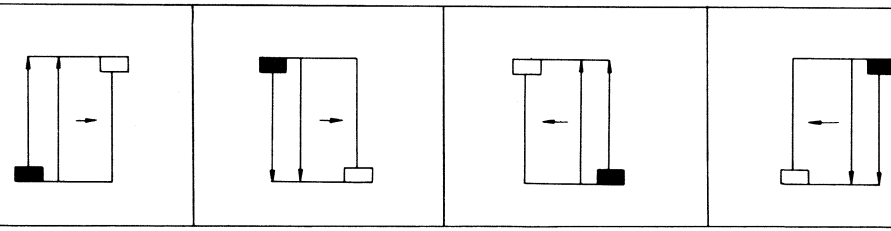
COMMAND CODE

Talbe C14-1 Source Scan Direction

S = 0



S = 1



As shown in Table C14-1, the scanning direction of the source area is decided by the relation between bit 11 in the command codes and the Pss and the Pse.

(a) Scanning Direction of Destination Area (DSD: Destination Scan Direction)
DSD is set in the command code.

COMMAND CODE

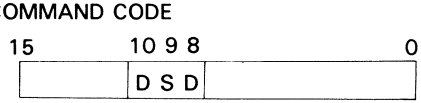
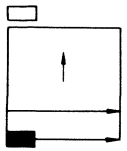
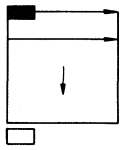
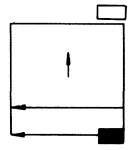
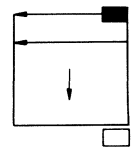
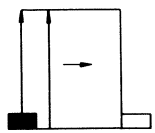
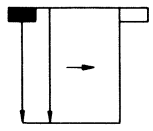
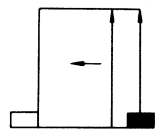
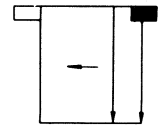


Table C14-2 Destination Scan Direction

DSD = 000	DSD = 001	DSD = 010	DSD = 011
			
DSD = 100	**DSD = 101**	**DSD = 110**	**DSD = 111**
			

■ : RWP □ : RWPe

As shown in Table C14-2, the scanning direction 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.

Note) As CPY command transfers data by word, the patterns in the source area and in the destination area may differ according to the setup of the source and the destination area.

Relation to Linear Address

Fig. C14-2 provides the relation between CPY and specified value when S = 1 and DSD = 000.

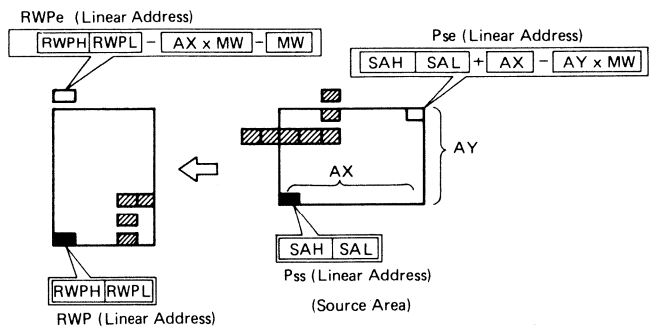


Figure C14-2 Relations with Linear Addresses

<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

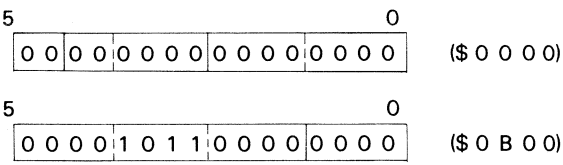
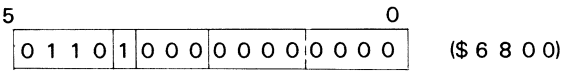


Figure C14-3(A) Example of RWP Setting

COMMAND CODE



COMMAND PARAMETERS

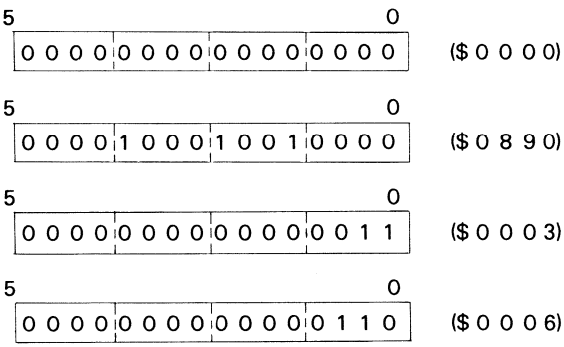


Figure C14-3(B) Example of Command Code/Parameter Setting

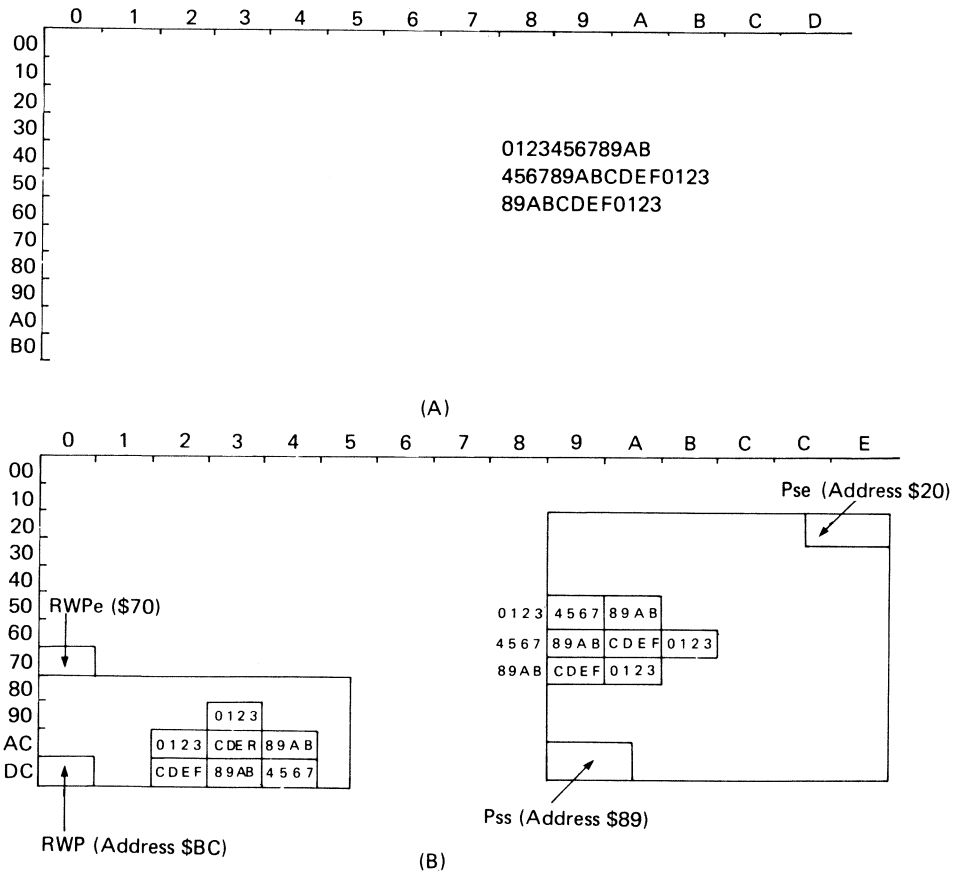


Figure C14-4 Example of CPY Execution

[15] SCPY (Selective Copy)

PAGE SCPY-1

<FUNCTION>

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.

TYPE: Data Transfer Command

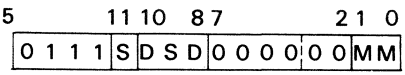
<MNEMONIC>

SCPY (S, DSD, MM) SAH, SAL, AX, AY

<FORMAT>

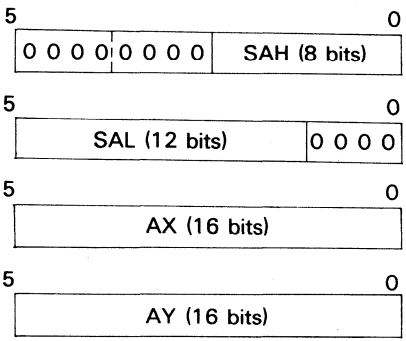
COMMAND CODE

hexadecimal notation



($ 7 X 0 X)

COMMAND PARAMETERS



WORD NUMBER

Wn=5

EXECUTION CYCLES

$Cn=(6x+10)y+12$

$\begin{cases} x = |AX| + 1 \\ y = |AY| + 1 \end{cases}$

<DESCRIPTION>

The parameters of the command define the source area. The RWP must be predefined to direct the destination area (including screen number). The source area resides in the same screen as that of the destination area as defined in RWP.

The source area is defined by the origin address (SAH/SAL) and physical frame buffer word (AX and AY) dimensions.

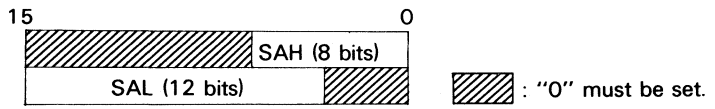
To allow rotation and proper operation for overlapping during copying, the command code contains fields which define the source and destination scanning direction. The S (Source Scan Direction) and DSD (Destination Scan Direction) fields of the command code define the source and destination scanning direction respectively as shown next page.

The MM (Modify Mode) field of the command code specifies the data transfer modify mode. Based on MM, logical operation is performed (except for bits masked in the MASK register) between the source data and the destination data, and the result is written to the destination.

At the end of the CPY command, RWP is set to RWPe.

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:

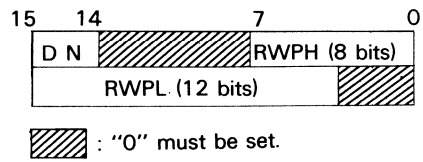


Figure C15-1 P_{SS} and RWP Set

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 copying is executed.

Note) As the memory width and the specification (graphic/character) of each screen are referred to by DN, the memory width register (MWR) must be set by ON in advance. The memory width of the source and destination area must accord.

2's complement is used to set the negative value.

- Specificable source area range

$-32767 \leqq AX \leqq 32767$

$-32767 \leqq AY \leqq 32767$

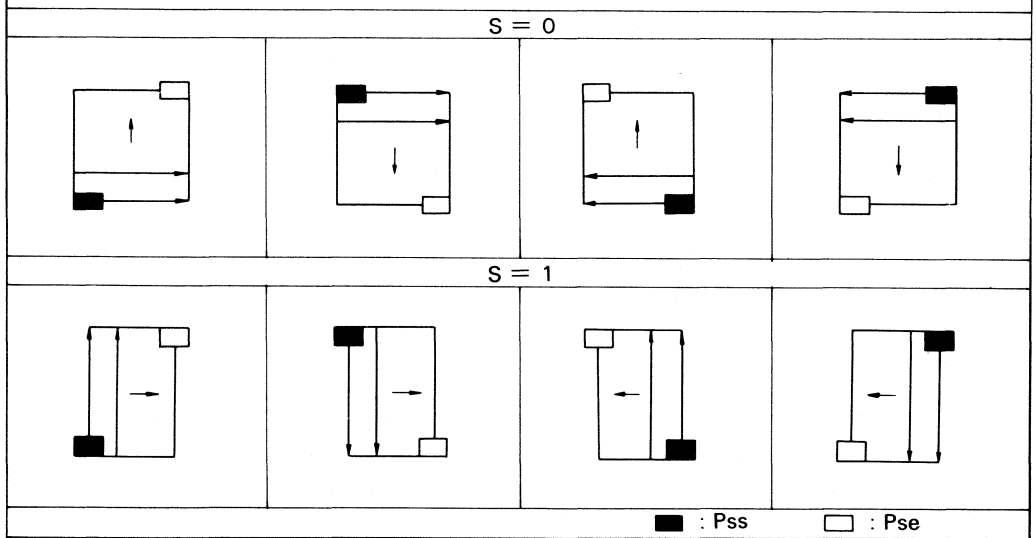
< SCPY Command Scan Direction >

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)
S is set in the command code.

COMMAND CODE

Table C15-1 Source Scan Direction



As shown in Table C15-1, the scanning direction of the source area is decided by the relation between bit 11 (S) in the command code and the Pss and the Pse.

(b) Scanning Direction of Destination Area (DSD: Destination Scan Direction)
DSD is set in the command code.

COMMAND CODE

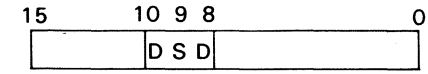
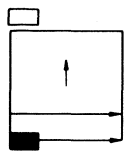
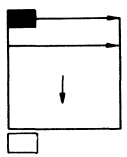
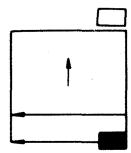
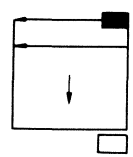
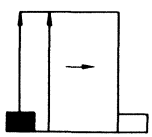
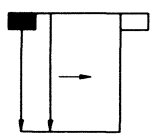
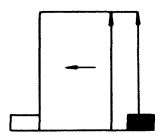
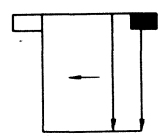


Table C15-2 Destination Scan Direction

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 of the destination area is decided by the relation between bit 10 to 8 (DSD) 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.

Note) As SCPY command transfers data by word, the patterns in the source area and in the destination area may differ according to the setup of the source and the destination area.

Relation to Linear Address

Fig. C15-2 provides the relation between SCPY and specified value when S = 1 and DSD = 000.

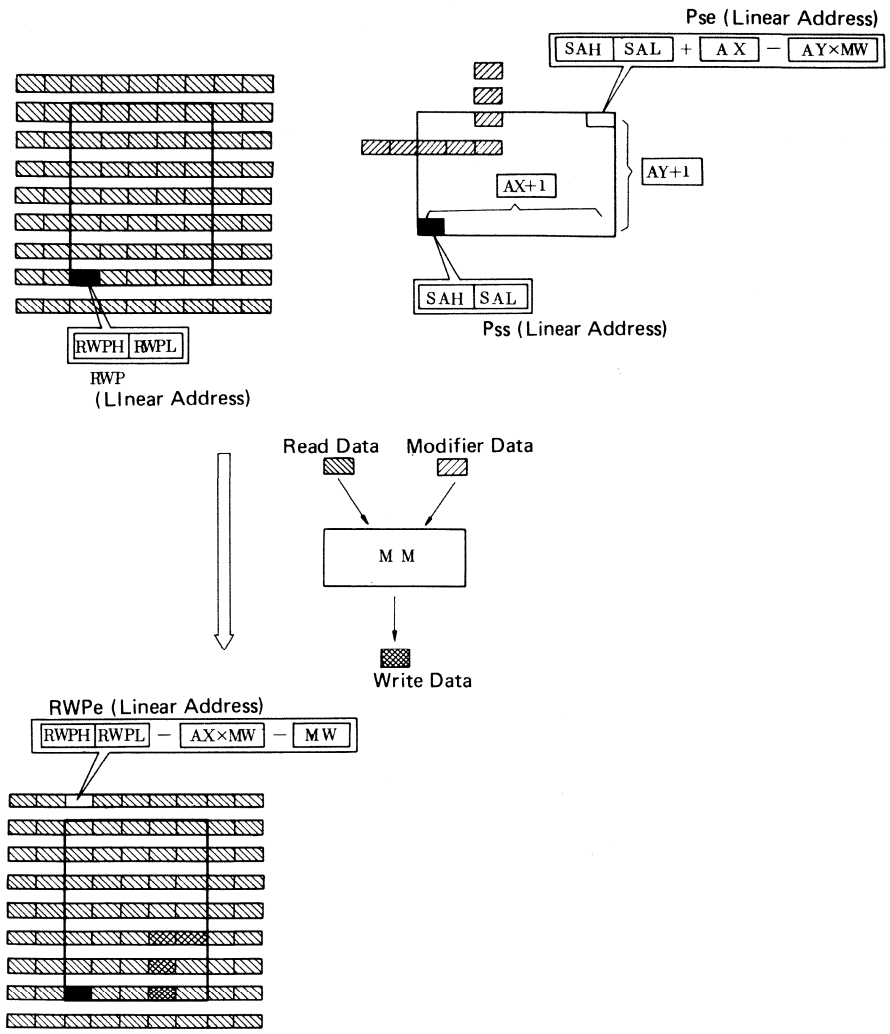


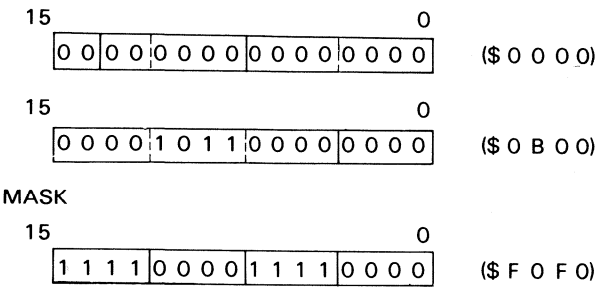
Figure C15-2 Relations with Linear Addresses

<EXAMPLE>

For this example 4 bits per logical pixel is used, the Memory Width (MW) is $10, the MASK register contains $F0F0 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(A) RWP and MASK Setting

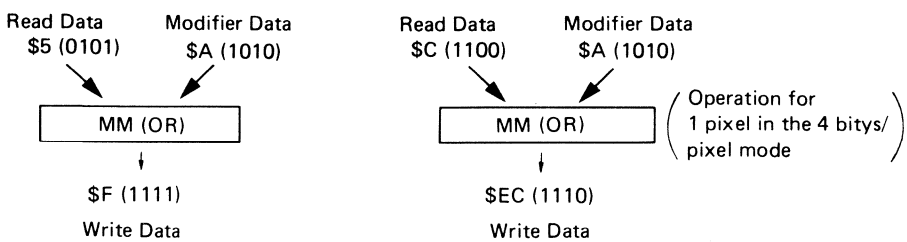
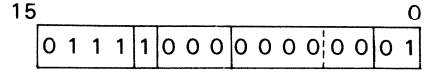
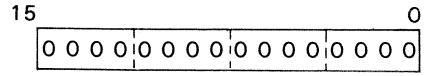


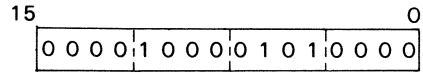
Figure C15-4 Operation of SCPY

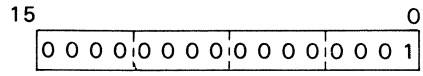
COMMAND CODE



COMMAND PARAMETERS







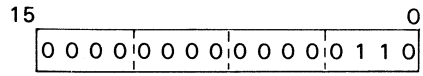
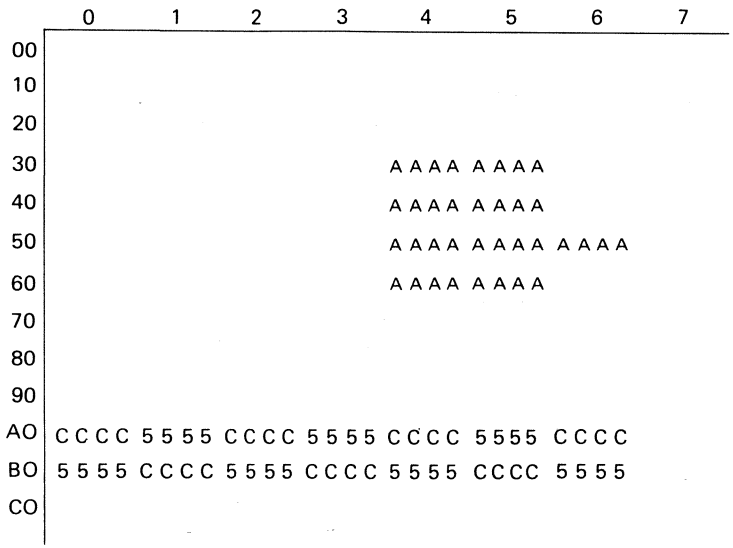
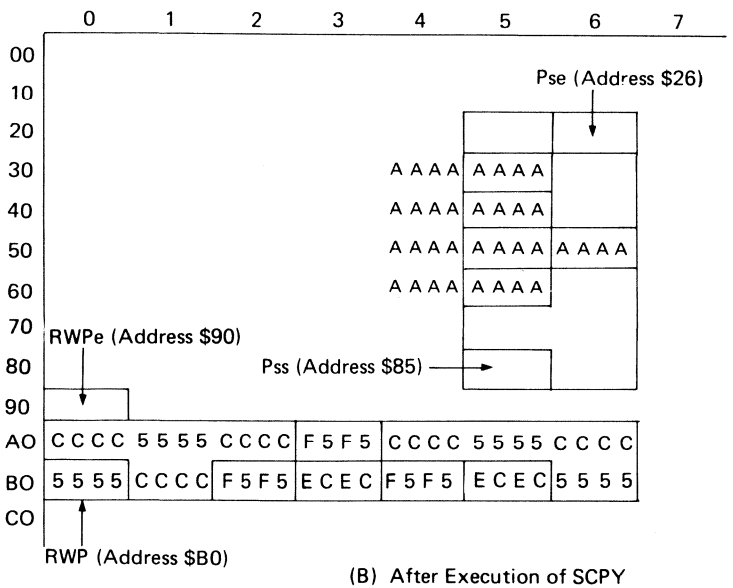


Fig. C15-3(B) Command Code/Parameter Setting



(A) Before Execution of SCPY



(B) After Execution of SCPY

Figure C15-5 Example of SCPY Execution

[16] AMOVE (Absolute Move)

PAGE AMOVE-1

<FUNCTION>
Move the Current Pointer (CP) to an absolute logical pixel X-Y address.

TYPE: Graphic Command

<MNEMONIC>
AMOVE X, Y

<FORMAT>

COMMAND CODE — hexadecimal notation

15															0	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	($ 8 0 0 0)

COMMAND PARAMETERS

15	0
X (16 bits)	

15	0
Y (16 bits)	

WORD NUMBER
Wn=3

EXECUTION CYCLES
Cn=56

<DESCRIPTION>
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.

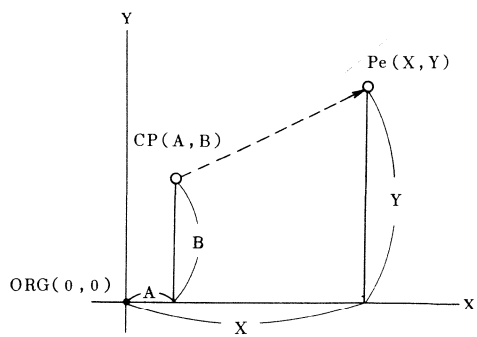


Figure C16-1 Function of AMOVE

<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	($ 8 0 0 0)

COMMAND PARAMETERS

15															0	
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)

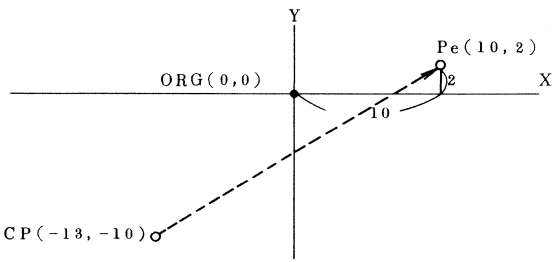


Figure C16-2 Example of AMOVE Execution

<Note for parameter setup>

- 2's complement is used to set the negative value.
- Specifiable range

  $CPX - 16383 \leqq X \leqq 16383 + CPX$

  $CPY - 16383 \leqq Y \leqq 16383 + CPY$
- When Pe is set to the same position as CP, CP does not shifted and the command execution normally terminates.

[17] RMOVE (Relative Move)

PAGE RMOVE-1

<FUNCTION>
Move the Current Pointer (CP) to a relative logical pixel X-Y address.

TYPE: Graphic Command

<MNEMONIC>
RMOVE dX, dY

<FORMAT>

COMMAND CODE — hexadecimal notation

15															0	
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	($ 8 4 0 0)

COMMAND PARAMETERS

15	0
dX (16 bits)	

15	0
dY (16 bits)	

WORD NUMBER
Wn=3

EXECUTION CYCLES
Cn=56

<DESCRIPTION>
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.

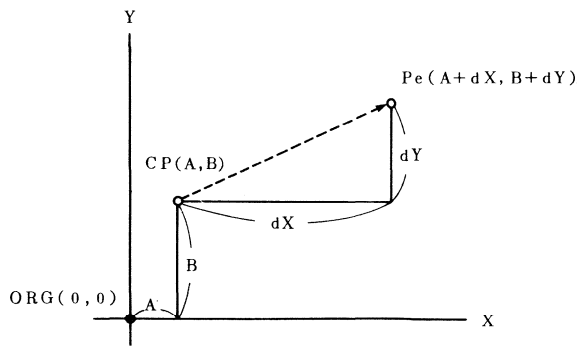


Figure C17-1 Function of RMOVE

<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

15															0	
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	($ 8 4 0 0)

COMMAND PARAMETERS

15															0	
0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	dX ($ 0 0 0 A)

15															0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	dY ($ 0 0 0 2)

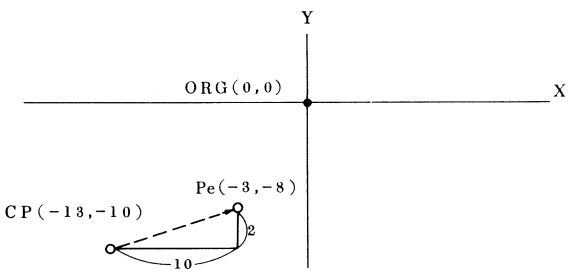


Figure C17-2 Example of RMOVE Execution

<Note for parameter setting>

- 2's complement is used to specify the negative value.
- Specifiable range
  $-16383 \leqq dX \leqq 16383$
  $-16383 \leqq dY \leqq 16383$
- When dX = dY = 0, CP is not shifted and the command execution normally terminates.

[18] ALINE (Absolute Line)

PAGE ALINE-1

TYPE Graphic Command

<FUNCTION>

Draw a straight line from CP to a command specified end point.

<MNEMONIC>

ALINE (AREA, COL, OPM) X, Y

<FORMAT>

COMMAND CODE — hexadecimal notation

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
X (16 bits)	

15	0
Y (16 bits)	

WORD NUMBER
Wn=3

EXECUTION CYCLES
Cn=P·L+18

P=4 (OPM='000'~'011')
P=6 (OPM='100'~'111')

L: Number of pixels being drawn

<DESCRIPTION>

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.

As the line is drawn, CP is moved to Pe. However, the logical pixel at position Pe is not drawn.

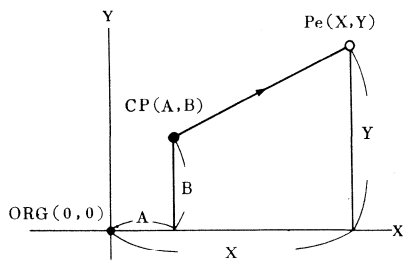


Figure C18-1 Function of ALINE

Note) The draw the logical pixel at position Pe, DOT command should be used. Make sure to set the color register and the pattern RAM before issuing ALINE command.

<EXAMPLE>

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	0	0	0	0	X ($ 0 0 0 A)

15											0	
0	0	0	0	0	0	0	0	0	0	0	0	Y ($ 0 0 0 2)

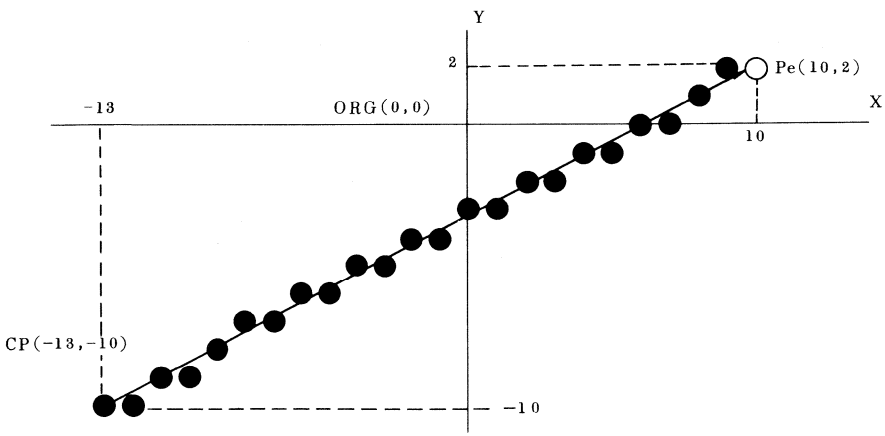


Figure C18-2 Example of ALINE Execution

<Note for parameter setting>

- 2's complement is used to set the negative value.
- Specifiable range
  CPX − 16383 ≦ X ≦ 16383 + CPX
  CPY − 16383 ≦ Y ≦ 16383 + CPY
- When Pe is set to the same position as CP, CP is not shifted and the command execution normally terminates.

[19] RLINE (Relative Line)

PAGE RLINE-1

<FUNCTION>
Draw a straight line from CP to a command specified end point.

<MNEMONIC>
RLINE (AREA, COL, OPM) dX, dY

TYPE: Graphic Command

<FORMAT>

COMMAND CODE — hexadecimal notation

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
dX (16 bits)	

15	0
dY (16 bits)	

WORD NUMBER
Wn=3

EXECUTION CYCLES
Cn=P·L+18

{ P=4 (OPM='000'~'011')
{ P=6 (OPM='100'~'111')

<DESCRIPTION>

The parameters (dX, dY) define the line end point as relative logical pixel X-Y displacements from the CP.

As the line is drawn, CP is moved to Pe. However, the logical pixel at position Pe is not drawn.

Note) To draw the logical pixel at position Pe, DOT command should be used. Make sure to set the color register and the pattern RAM before issuing RLINE command.

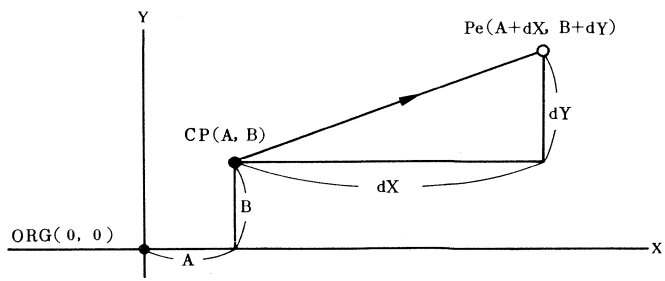
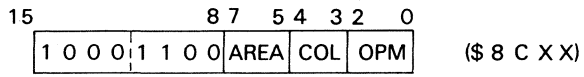


Figure C19-1 Function of RLINE

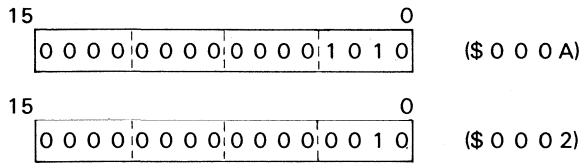
<EXAMPLE>

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



COMMAND PARAMETERS



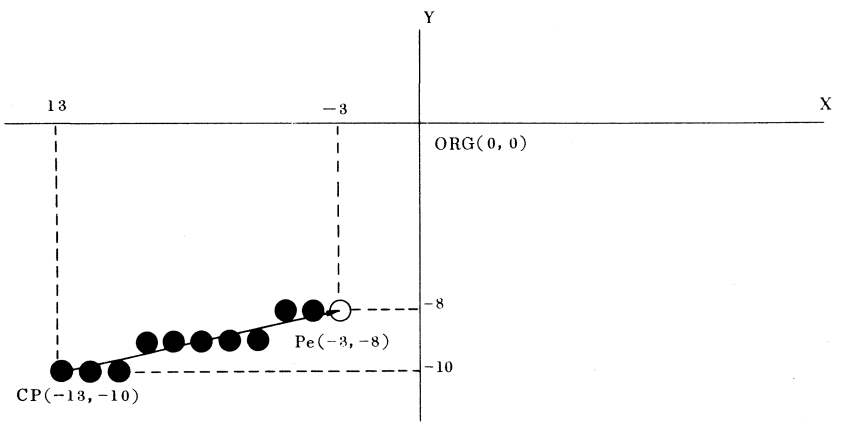


Figure C19-2 Example of RLINE Execution

<**Note for parameter setting**>

- 2's complement is used to set the negative value.
- Specifiable range
  $-16383 \leqq dX \leqq 16383$
  $-16383 \leqq dY \leqq 16383$
- When dX = dY = 0, shifting of CP and the drawing are not performed, and the command execution normally terminates.

[20] ARCT (Absolute Rectangle)

PAGE ARCT-1

<FUNCTION>
Draw a rectangle defined by CP and the command specified diagonal point.

TYPE: Graphic Command

<MNEMONIC>
ARCT (AREA, COL, OPM) X, Y

<FORMAT>

COMMAND CODE — hexadecimal notation

15				8	7	5	4	3	2	0	
1	0	0	1	0	0	0	AREA	COL	OPM		($9 0 X X)

COMMAND PARAMETERS

15	0
X (16 bits)	

15	0
Y (16 bits)	

WORD NUMBER
Wn=3

EXECUTION CYCLES
Cn=2p(A+B)+54

P=4 (OPM='000'~'011')
P=6 (OPM='100'~'111')

$A = |P_{CX} - CP_X| + 1$
$B = |P_{CY} - CP_Y| + 1$

<DESCRIPTION>

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.

Rectangle drawing is executed from CP to Pe via Pc. However, the logical pixel at position Pe is not drawn. CP is located at the position of Pe after the command is executed. Therefore Pe and CP (before command execution) are the same address.

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).

Note) Make sure to set the color register and the pattern RAM before issuing ARCT command.

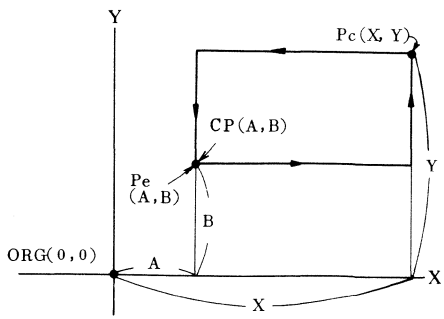


Figure C20-1 Function of ARCT

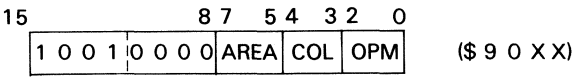
<EXAMPLE>

If CP = (6, −6) and ARCT command is executed with parameters (X, Y) = (−16, 10), then a rectangle is drawn and CP is set to Pe as shown below.

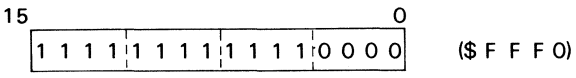
<NOTE>

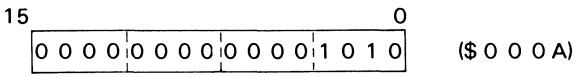
Drawing starts from the X-axis direction.

COMMAND CODE



COMMAND PARAMETERS





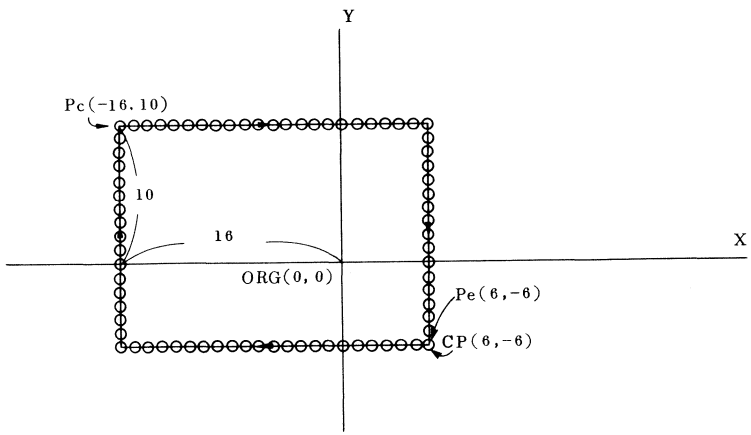


Figure C20-2 Example of ARCT Execution

<Note for parameter setting>

- 2's complement is used to set the negative value.
- Specifiable range
  CPX − 16383 ≦ X ≦ 16383 + CPX
  CPY − 16383 ≦ Y ≦ 16383 + CPY
- If Pc is specified at the same position as CP, shifting of CP and drawing are not performed, and the command execution normally terminates.

[21] RRCT (Relative Rectangle)

PAGE RRCT-1

TYPE Graphic Command

<FUNCTION>
Draw a rectangle defined by CP and the command specified diagonal point.

<MNEMONIC>
RRCT (AREA, COL, OPM) dX, dY

<FORMAT>

COMMAND CODE — hexadecimal notation

15								8	7	5	4	3	2	0
1	0	0	1	0	1	0	0	AREA		COL		OPM		($ 9 4 X X)

COMMAND PARAMETERS

15	0
dX (16 bits)	

15	0
dY (16 bits)	

WORD NUMBER
Wn = 3

EXECUTION CYCLES
Cn = 2P(A+B)+54

P=4 (OPM='000'~'011')
P=6 (OPM='100'~'111')

A = |dX| + 1
B = |dY| + 1

<DESCRIPTION>

The parameters (dX, dY) define the diagonal point of the rectangle as relative logical pixel X-Y displacements from the CP.

Rectangle drawing is executed from CP to Pe via Pc. However, the logical pixel at position Pe is not drawn.

CP is located at the position of Pe after the command is terminated.

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).

Note) Make sure to set the color register and the pattern RAM before issuing RRCT command.

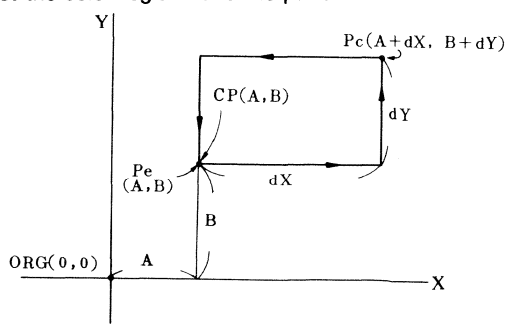
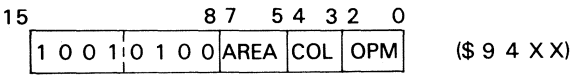


Figure C21-1 Function of RRCT

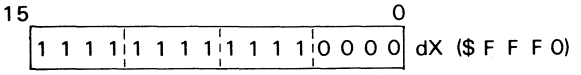
<EXAMPLE>

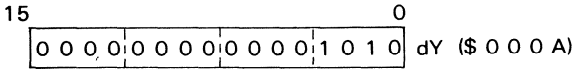
If CP = (6, −6) and RRCT command is executed with parameters (dX, dY) = (−16, 10), then a rectangle is drawn and CP is set to Pe as shown below (Arrows show the moving direction of CP).

COMMAND CODE



COMMAND PARAMETERS





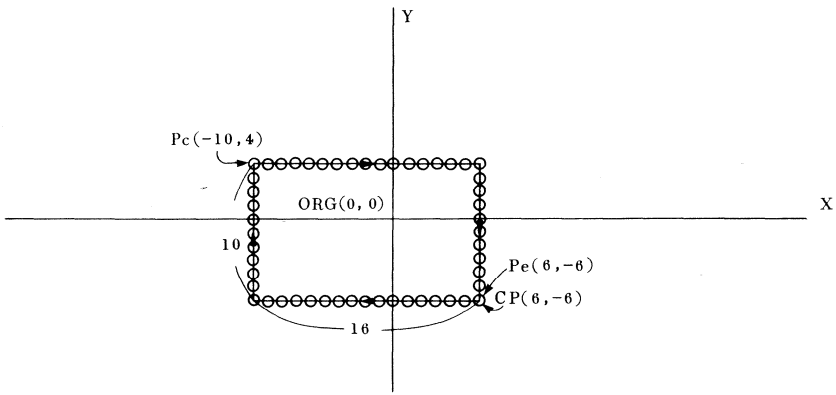


Figure C21-2 Example of RRCT

<Note for parameter setting>

- 2's complement is used to set the negative value.
- Specifiable range
  $-16383 \leqq dX \leqq 16383$
  $-16383 \leqq dY \leqq 16383$
- When dX = dY = 0, shifting of CP and drawing are not performed, and the command execution normally terminates.

[22] APLL (Absolute Polyline)

PAGE APLL-1

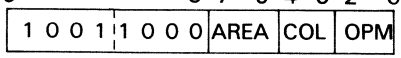
<FUNCTION>
Draw a polyline (multiple contiguous segments) from the CP through command specified points.

TYPE: Graphic Command

<MNEMONIC>
APLL (AREA, COL, OPM) n, X_1, Y_1 ... Xn, Yn

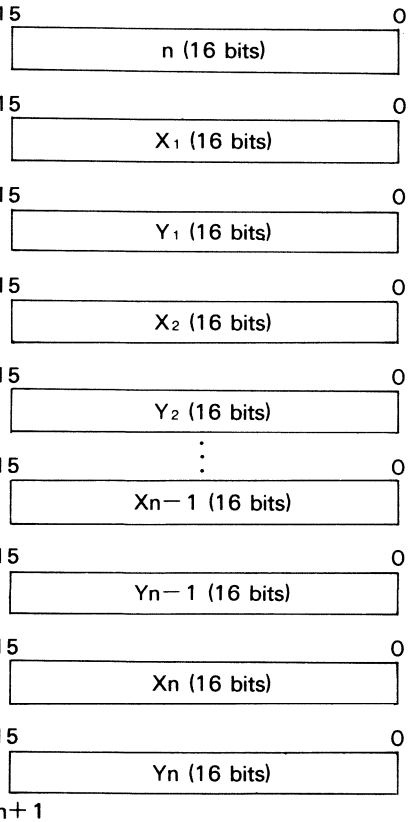
<FORMAT>

COMMAND CODE — hexadecimal notation



($ 9 8 X X)

COMMAND PARAMETERS



P2n+1

n is specified by the absolute value of a 16-bits binary number.
Specifiable range is $0 \leqq n \leqq 65535$. When '0' is specified, command execution normally terminates without drawing.

WORD NUMBER
Wn=2n+2

EXECUTION CYCLES
$Cn=\Sigma\{P \cdot L+16\}+8$

{P=4 (OPM='000'～'011')
{P=6 (OPM='100'～'111')
L=number of pixels being drawn

APLL (Absolute Polyline)

<DESCRIPTION>

The first parameter (n) specifies the number of line segments, that is, n = 1 specifies one line segment. The following parameters (Xn, Yn) 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.

<NOTE>

To draw the logical pixel at position Pe, DOT command needs to be issued. Make sure to set the color register and the pattern RAM before issuing APLL command.

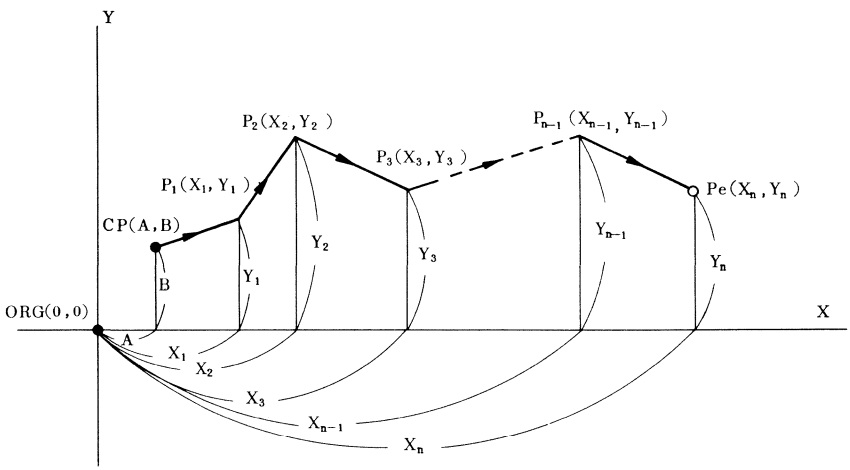


Figure C22-1 Function of APLL

APLL (Absolute Polyline)

<EXAMPLE>

If the CP is at (−8, −6) in the absolute coordinate system. n is set to 3, X1 to −4, Y1 to 4, X2 to 8, Y2 to 6, X3 to 16 and Y3 to −8, then the APLL command draws a polyline as shown below.

COMMAND CODE

15				8	7		5	4		3	2		0
1	0	0	1	1	0	0	0	AREA		COL		OPM	

COMMAND PARAMETERS

15															0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	($0003)
15															0	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	($FFFC)
15															0	
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	($0004)
15															0	
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	($0008)
15															0	
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	($0006)
15															0	
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	($0010)
15															0	
1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	($FFF8)

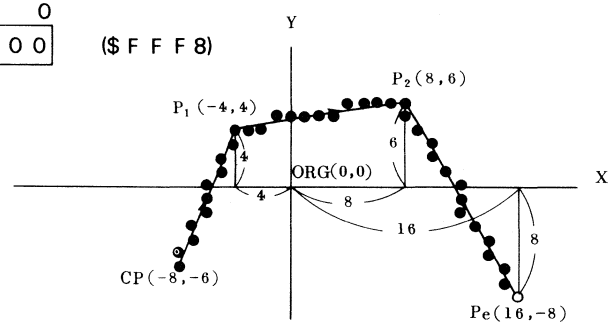


Figure C22-2 Example of APLL Execution

<Note for parameter settihng>

- 2's complement is used to specify the negative value.
- Specifiable range

  $X_{n-1} - 16383 \leqq Xn \leqq 16383 + X_{n-1}$

  $Y_{n-1} - 16383 \leqq Yn \leqq 16383 + Y_{n-1}$
- When Pn is set at the same position as CP, shifting of CP and drawing are not performed, and the command execution normally terminates.

[23] RPLL (Relative Polyline)

PAGE RPLL-1

<FUNCTION>
RPLL command draws a polyline which connects the Start point, current pointer, and each relative coordinate point.

TYPE: Graphic Command

<MNEMONIC>
RPLL (AREA, COL, OPM) n, dX_1, dY_1, ... dXn, dYn

<FORMAT>

COMMAND CODE — hexadecimal notation

15				8 7		5 4		3 2		0
C	1	0	0 1	1	1	0 0	AREA	COL	OPM	($9 8 X X)

COMMAND PARAMETERS

15 0	n (16 bits)	
15 0	dX_1 (16 bits)	
15 0	dY_1 (16 bits)	P_1
15 0	dX_2 (16 bits)	
15 0	dY_2 (16 bits)	P_2
15 0	dX_{n-1} (16 bits)	P_{n-1}
15 0	dY_{n-1} (16 bits)	
15 0	dXn (16 bits)	Pe
15 0	dYn (16 bits)	P_{2n+1}

Set "n" in binary absolute values of 16 bits.
Specifiable range is $0 \leq n \leq 65535$. When '0' is set, command execution is normally terminated without drawing.

WORD NUMBER
$Wn = 2n + 2$

EXECUTION CYCLES
$Cn = \Sigma\{P \cdot L + 16\} + 8$

$\begin{cases} P=4 \text{ (OPM='000'}\sim\text{'011')} \\ P=6 \text{ (OPM='100'}\sim\text{'111')} \end{cases}$

L: number of pixels being drawn

<DESCRIPTION>

As shown in figure below, the relative polyline command (RPLL) draws a polyline which connects the Start point CP, and each relative coordinate (P_1, P_2, P_3, ..., P_{n-1}, Pe).
The total number of points is set in the 1st command parameter (n_1). 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.

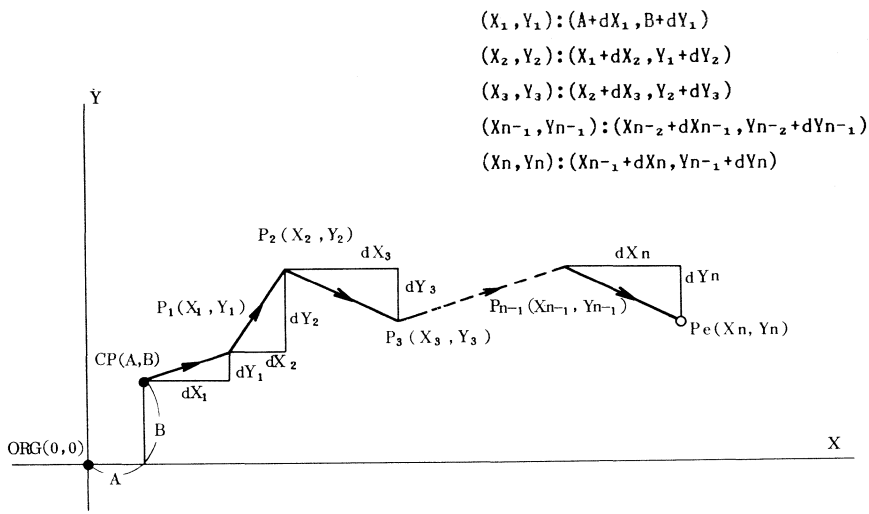


Figure C23-1 Function of RPLL

<NOTE>

To draw the logical pixel at Pe, DOT command needs to be issued. Make sure to set the color register and the pattern RAM before issuing RPLL command.

<EXAMPLE>

If CP is at (−8, −6) in the absolute coordinate system, 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 polyline as shown below.

COMMAND CODE

15				8	7		5	4	3	2	0
1	0	0	1	1	1	0	0	AREA	COL	OPM	

($9CXX)

COMMAND PARAMETERS

15											0	
0	0	0	0	0	0	0	0	0	0	1	1	($0003)
1	1	1	1	1	1	1	1	1	1	0	0	($FFFC)
0	0	0	0	0	0	0	0	0	1	0	0	($0004)
0	0	0	0	0	0	0	0	1	0	0	0	($0008)
0	0	0	0	0	0	0	0	0	1	1	0	($0006)
0	0	0	0	0	0	0	0	0	0	1	0	($0010)
1	1	1	1	1	1	1	1	1	1	1	0	($FFF8)

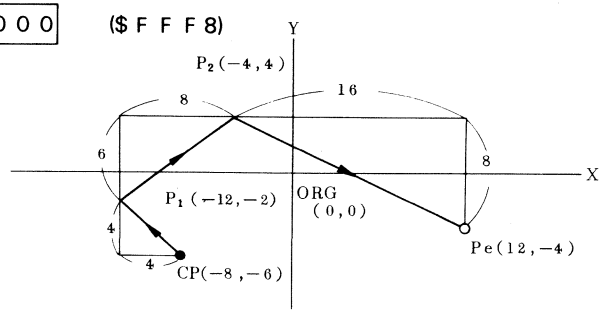


Figure C23-2 Example of RPLL Execution

<Note for parameter setting>

- 2's complement is used to set the negative value.
- Specifiable range
  $-16383 \leqq dXn \leqq 16383$
  $-16383 \leqq dYn \leqq 16383$
- When dXn = dYn = 0, shifting of CP and drawing are not performed, and the command execution normally terminates.

[24] APLG (Absolute Polygon)

PAGE APLG-1

TYPE: Graphic Command

<FUNCTION>

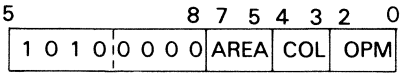
APLG draws a polygon which connects the start point, CP, and each absolute coordinate.

<MNEMONIC>

APLG (AREA, COL, OPM) n, X_1, Y_1 ..... Xn, Yn

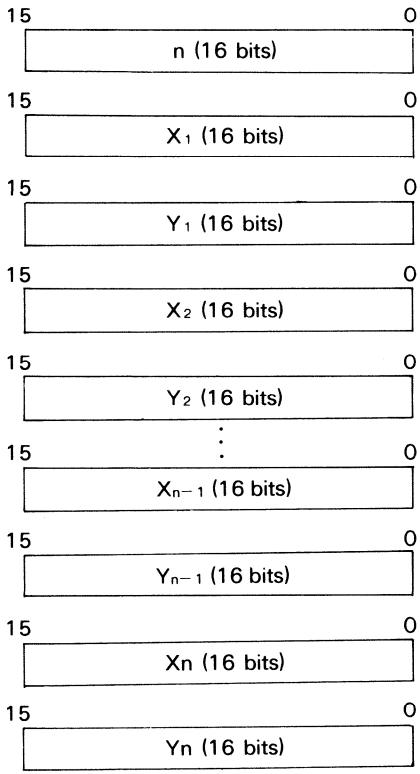
<FORMAT>

COMMAND CODE — hexadecimal notation



($ A 0 X X)

COMMAND PARAMETERS



P2n+1

Set "n" in binary absolute values of 16 bits.
Specifiable range is $0 \leq n \leq 65536$. When '0' is specified, the command execution normally terminates without drawing. When '1' is specified, CP makes a round trip between its present position and (X_1, Y_1) as it draws a line.

WORD NUMBER
Wn = 2n + 2

EXECUTION CYCLES
$Cn = \Sigma \{P \cdot L + 16\} + P \cdot L_0 + 20$

P = 4 (OPM = '000' ~ '011')
P = 6 (OPM = '100' ~ '111')

L = number of pixels being drawn

L_0 = number of pixels of the last line

<DESCRIPTION>

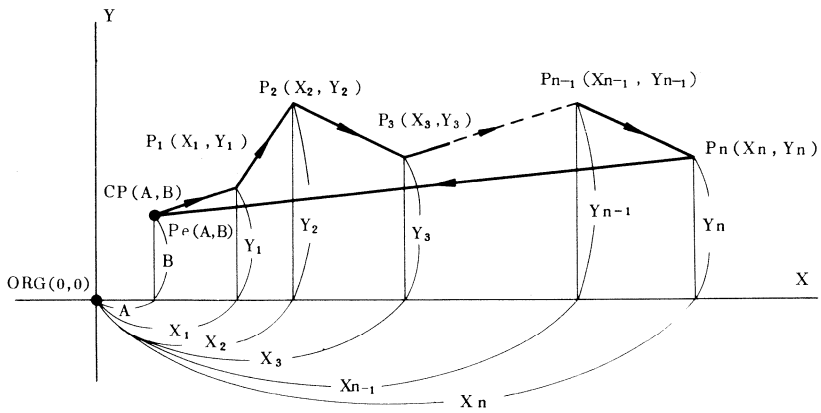


Figure C24-1 Function of APLG

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 ....., P_{n-1}, Pn), 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 polyline. However a dot is not drawn at Pe. CP is the same point as Pe.

Note) Make sure to set the color register and the pattern RAM before issuing APLG command.

<EXAMPLE>

If CP is at (−8, −6) in the absolute coordinate system, 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 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	AREA		COL		OPM				($A0XX)

COMMAND PARAMETERS

15															0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	($0003)

15															0	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	($FFFC)

15															0	
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	($0004)

15															0	
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	($0008)

15															0	
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	($0006)

15															0	
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	($0010)

15															0	
1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	($FFF8)

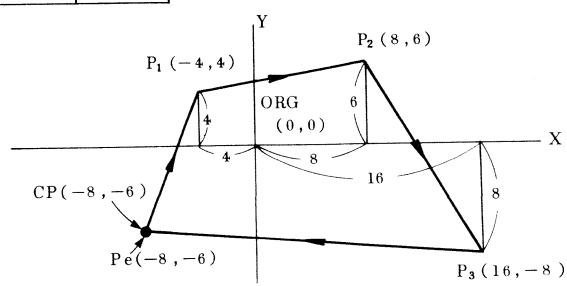


Figure C24-2 Example of APLG Execution

<Note for parameter setting>

- 2's complement is used to set the negative value.
- Specifiable range

  $X_{n-1} - 16383 \leqq Xn \leqq 16383 + X_{n-1}$ (X_0 = CPx, Y_0 = CPr)

  $Y_{n-1} - 16383 \leqq Yn \leqq 16383 + Y_{n-1}$

  $-16383 \leqq Xn - CPX \leqq 16383$

  $-16383 \leqq Yn - CPY \leqq 16383$

[25] RPLG (Relative Polygon)

PAGE RPLG-1

TYPE: Graphic Command

<FUNCTION>

APLG draws a polygon which connects the start point, CP, and each relative coordinate.

<MNEMONIC>

RPLG (AREA, COL, OPM) n, dX_1, dY_1, ... dXn, dYn

<FORMAT>

COMMAND CODE — hexadecimal notation

15 8	7 5	4 3	2 0
1 0 1 0 0 1 0 0	AREA	COL	OPM

($ A 4 X X)

COMMAND PARAMETERS

n (16 bits)	
dX_1 (16 bits)	
dY_1 (16 bits)	P_1
dX_2 (16 bits)	
dY_2 (16 bits)	P_2
⋮	⋮
dX_{n-1} (16 bits)	
dY_{n-1} (16 bits)	P_{n-1}
dXn (16 bits)	
dYn (16 bits)	Pn

P_{2n+1}

Set "n" in binary absolute values of 16 bits.
Specifiable range is $0 \leq n \leq 65536$. When '0' is specified, the command execution normally terminates without drawing. When '1' is specified, CP makes a round trip between its present position and (X_1, Y_1) as it draws a line.

WORD NUMBER

$Wn = 2n + 2$

EXECUTION CYCLES

$Cn = \Sigma \{P \cdot L + 16\} + P \cdot L_0 + 20$

$\begin{cases} P = 4 & (OPM = '000' \sim '011') \\ P = 6 & (OPM = '100' \sim '111') \end{cases}$

L = number of pixels being drawn

L_0 = number of pixels of the last line

< DESCRIPTION >

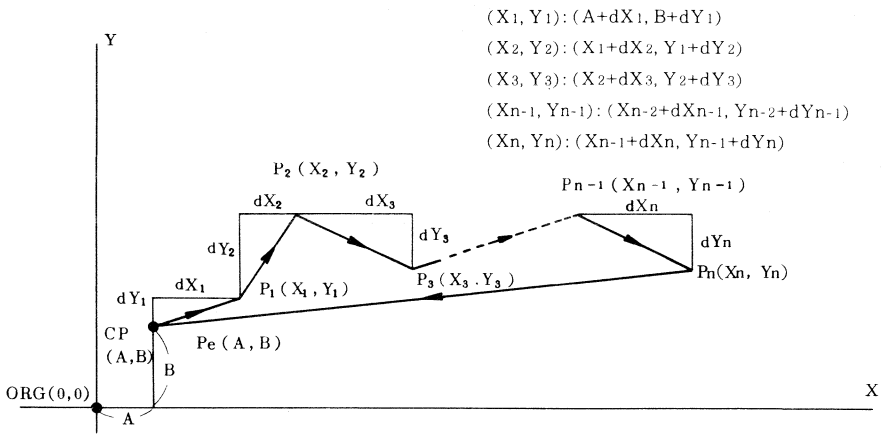


Figure C25-1 Function of RPLG

As shown in above figure, the RPLG command draws a polygon line which connects the start point, CP, and each relative coordinate (P_1, P_2, P_3, ..., P_{n-1}, Pn), then back to CP.
The total number of points are set in the first command parameter. Relative components dX and dY 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.

Note) Make sure to set the color register and the pattern RAM before issuing RPLG.

<EXAMPLE>

If the CP is at (−8, −6) in the absolute coordinate system, n is set to 3, dX_1 to −4, dY_1 to 4, dX_2 to 8, dY_2 to 6, dX_3 to 16 and dY_3 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		($A4XX)

COMMAND PARAMETERS

15															0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	($0003)
1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	($FFFC)
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	($0004)
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	($0008)
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	($0006)
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	($0010)
1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	($FFF8)

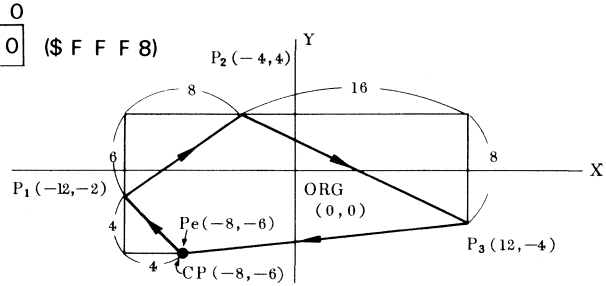


Figure C25-2 Example of RPLG Execution

<Note for parameter setting>

- 2's complement is used to specify the negative value.
- Specifiable range

  $-16383 \leqq dXn \leqq 16383$

  $-16383 \leqq dYn \leqq 16383$

  at the same time,

  $-16383 \leqq Xn - CPx \leqq 16383$

  $-16383 \leqq Yn - CPy \leqq 16383$
- When dXn = dYn = 0, shifting of CP and drawing are not performed, and command execution normally terminates.

[26] CRCL (Circle Command)

PAGE CRCL-1

TYPE: Graphic Command

<FUNCTION>

CRCL Command draws a circle of the radius R placing the CP at the center.

<MNEMONIC>

CRCL (C, AREA, COL, OPM) r

<FORMAT>

COMMAND CODE

hexadecimal notation

15			9	8	7		5	4	3	2	0
1	0	1	0	1	0	0	C	AREA		COL	OPM

C = 1 : ($ A 9 X X)
C = 0 : ($ A 8 X X)

COMMAND PARAMETERS

15	0
r (16 bits)	

WORD NUMBER
Wn=2

EXECUTION CYCLES
Cn=8d+66

<DESCRIPTION>

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.

First the CP moves in the X-direction from the center for the length of the radius r. Now this point is named Ps. The circle drawing starts at Ps and finishes at P_L (=Ps). But, a dot is not drawn at P_L. After the circle has been drawn, the CP moves back to the center and the command is finished. The position of the CP and Pe are the same.

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.

<Note for parameter setting>

- Specifiable range
  $0 < |r| \leq 4095$
- When '0' is set, a dot is drawn in the center of the circle and the command execution normally terminates.
- 2's complement is used to set the negative value.
  In this case Ps is located at opposite position of normal Ps.
  ie: CP moves in the X-negative direction from the center.

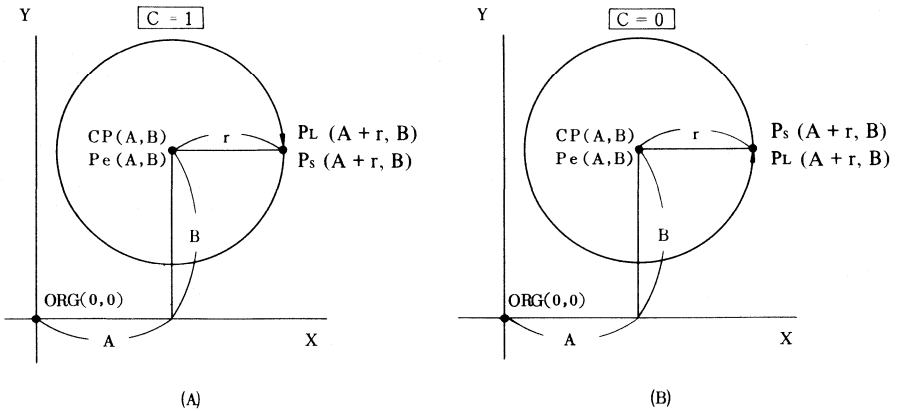
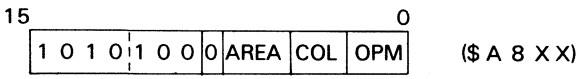


Figure C26-1 Function of CRCL

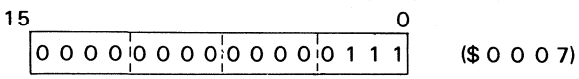
<EXAMPLE>

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



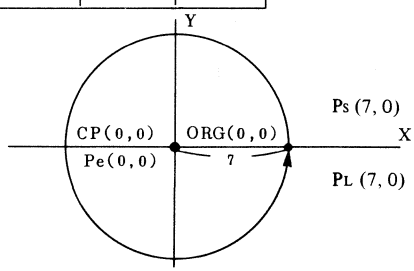


Figure C26-2 Example of CRCL Execution

[27] ELPS (Ellipse Command)

PAGE ELPS-1

TYPE Graphic Command

<FUNCTION>

ELPS Command draws an ellipse placing the CP at the center.

<MNEMONIC>

ELPS (C, AREA, COL, OPM) a, b, dX

COMMAND CODE

hexadecimal notation

15				9	8	7		5	4	3	2	0
1	0	1	0	1	1	0	C	AREA		COL		OPM

C = 1 : ($ A D X X)
C = 0 : ($ A C X X)

WORD NUMBER
Wn=4

EXECUTION CYCLES
Cn=10d+90

d: number of pixels being drawn

COMMAND PARAMETERS

15	0
a (16 bits)	

15	0
b (16 bits)	

15	0
dX (16 bits)	

<DESCRIPTION>

The Ellipse Command (ELPS) draws an ellipse placing the current pointer (CP) at the center.

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.

$$\frac{(X-A)^2}{dX^2} + \frac{(Y-B)^2}{dY^2} = 1 \quad \cdots\cdots\cdots\cdots\cdots (1)$$

In Equation (1), letting the ratio of squared dX and dY be a, b;

$$a : b = dX^2 : dY^2 \quad \cdots\cdots\cdots\cdots\cdots\cdots (2)$$

Then substituting (2) for Equation (1);

$$\frac{(X-A)^2}{a} + \frac{(Y-B)^2}{b} = \frac{dX^2}{a} \quad \cdots\cdots\cdots\cdots (3)$$

The ELPS Command draws an ellipse according to Equation (3). The parameters a, b, and dX are specified in units of pixels.

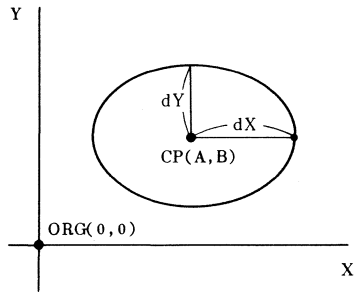


Figure C27-1 Function of ELPS

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 P_L (=Ps). But the dot is not drawn at P_L. 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.

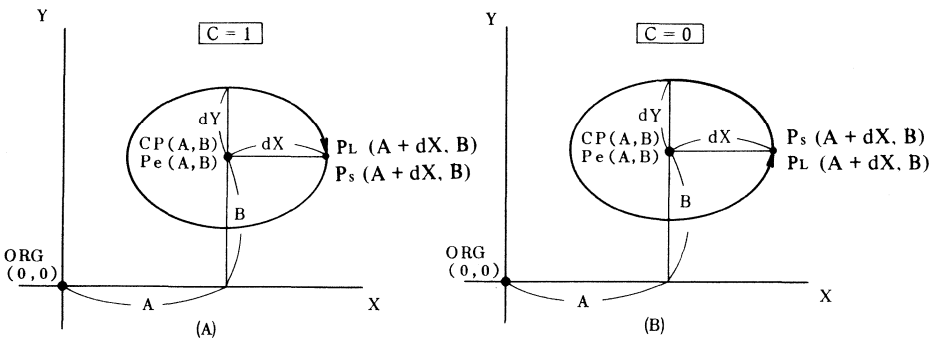


Figure C27-2 Drawing Direction of ELPS

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 above.

<NOTE>

Specifiable range of parameters

$dX > \frac{a}{4b} + \frac{1}{2}$, $dX \leqq \frac{4095}{\max(a, b)}$,

$1 \leqq a \leqq 255$, $1 \leqq b \leqq 255$

where max (a, b) means the greater value among a and b.
(The positive value must be set to dX. "0" cannot be set to a nor b.)

<EXAMPLE>

If the absolute coordinate of CP is (16, 10) 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

15															0	
1	0	1	0	1	1	0	0	AREA				COL			OPM	($ A C X X)

COMMAND PARAMETERS

15															0	
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	a ($ 0 0 0 9)

15															0	
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	b ($ 0 0 0 4)

15															0	
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	dX ($ 0 0 0 9)

$9 : 4 = 9^2 : 6^2$

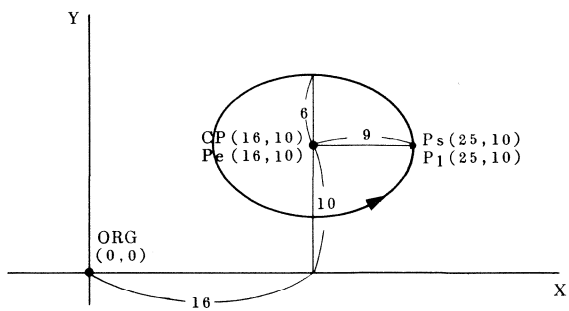


Figure C27-3 Example of ELPS Execution

[28] AARC (Absolute Arc)

PAGE AARC-1

<FUNCTION>
AARC draws an arc by using current pointer (start point), end point, and center point of the absolute coordinates.

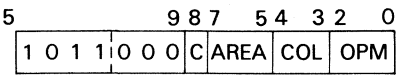
TYPE: Graphic Command

<MNEMONIC>
AARC (C, AREA, COL, OPM) Xc, Yc, Xe, Ye

<FORMAT>

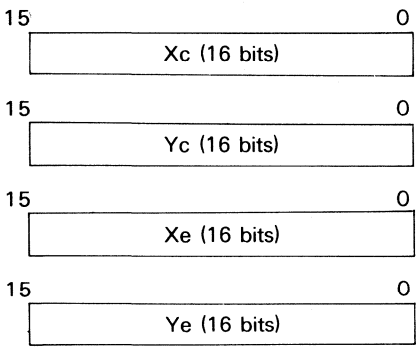
COMMAND CODE

hexadecimal notation



C = 1 : ($ B 1 X X)
C = 0 : ($ B 0 X X)

COMMAND PARAMETERS



WORD NUMBER
Wn=5

EXECUTION CYCLES
Cn=8d+18

d: number of pixels being drawn

<DESCRIPTION>

As shown in Fig. C28-1, the AARC command draws an arc from the current pointer, CP, to Pe of the absolute coordinates (Xe, Ye), 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 each command parameter in unit of pixel. 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.

Note) To draw a pixel at Pe, DOT command needs to be issued after AARC. Make sure to set the color register and the pattern RAM before issuing AARC.

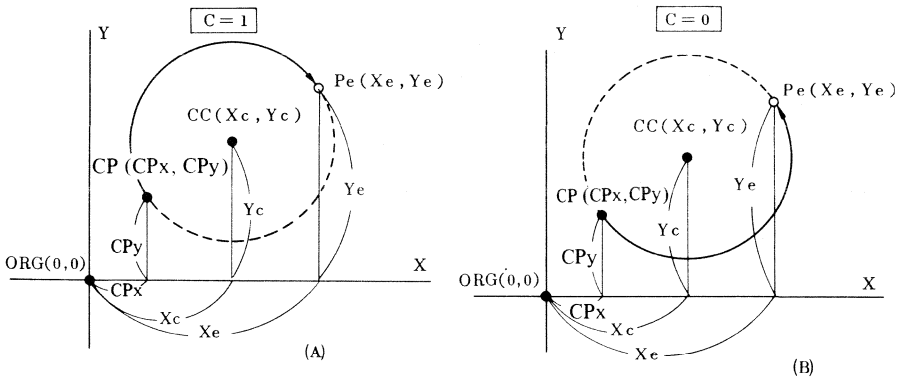
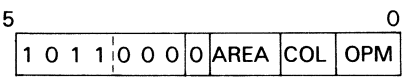


Figure C28-1 Function of AARC Command

<EXAMPLE>

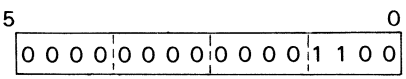
CP is at (12, 4) in the absolute coordinate system, Xc is set to 12, Yc to 10, Xe to 6, and Ye to 10 in each command parameter, then the AARC Command (C = 0) draws an arc as shown in figure next page.

COMMAND CODE

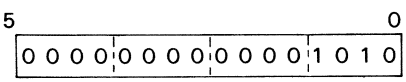


($ B 0 X X)

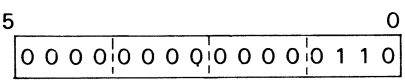
COMMAND PARAMETERS



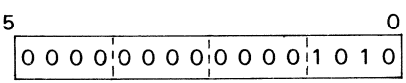
($ 0 0 0 C)



($ 0 0 0 A)



($ 0 0 0 6)



($ 0 0 0 A)

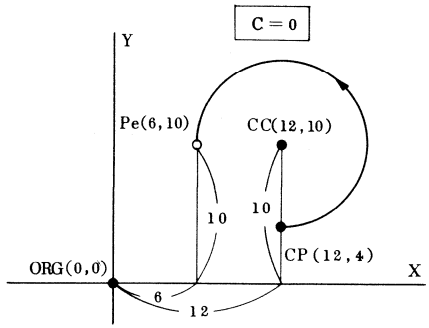


Figure C28-2 Example of AARC Execution

<Note for parameter setup>

- 2's complement is used to specify the negative value.
- Specifiable range: $0 \leqq \sqrt{(X_c - CP_x)^2 + (Y_c - CP_y)^2} \leqq 4095$.
- When $(CP_x, CP_y) = (X_c, Y_c) = (Xe, Ye)$, a pixel is drawn at the start point of the arc, and the command execution normally terminates.
- When $(CP_x, CP_y) = (Xe, Ye)$, a circle is drawn, and the command execution normally terminates.

[29] RARC (Relative Arc)

PAGE RARC-1

<FUNCTION>
RARC draws an arc by using current pointer (start point), end point, and center point of the relative coordinates.

TYPE: Graphic Command

<MNEMONIC>
RARC (C, AREA, COL, OPM) dXc, dYc, dXe, dYe

<FORMAT>

COMMAND CODE

hexadecimal notation

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)

COMMAND PARAMETERS

15	0
dXc (16 bits)	
dYc (16 bits)	
dXe (16 bits)	
dYe (16 bits)	

WORD NUMBER
Wn=5

EXECUTION CYCLES
Cn=8d+18

d: number of pixels being drawn

<DESCRIPTION>

As shown in Fig. C29-1, the RARC command draws an arc from the current pointer, CP, to Pe (CP_X + dXe, CP_Y + dYe) of the relative coordinates, the relative coordinates CC (CP_X + dXc, CP_Y+dYc) being the center point. The X and Y components of the relative coordinates CC and Pe are set in each command parameter unit of pixel. CP moves to the end point Pe after the arc is drawn. However a dot at Pe is not drawn. 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. C29-1.

Note) To draw a pixel at Pe, DOT command needs to be issued after RARC. Make sure to set the color register and the pattern RAM before issuing RARC.

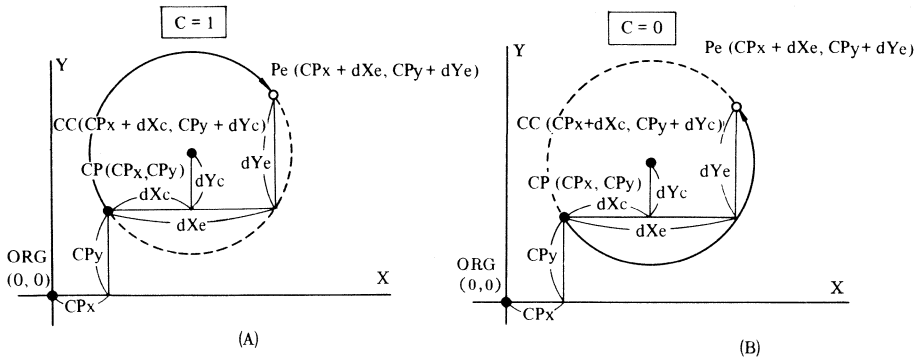


Figure C29-1 Function of RARC

<**EXAMPLE**>

CP is at (6, 10) in the absolute coordinate system, dXc is set to 6, dYc to 0, dXe to 6, and dYe to 6 in each 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	OPM

($ 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)

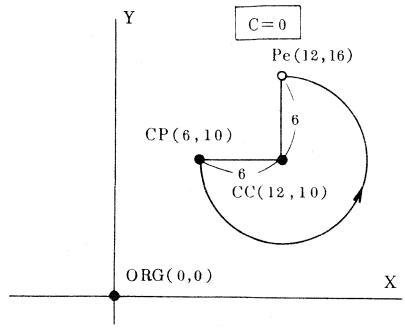


Figure C29-2 Example of RARC Execution

<Note for parameter setup>

- 2's complement is used to set the negative value.
- Specifiable range: $0 \leqq \sqrt{dX^2 + dY^2} \leqq 4095$.
- When dXc = dYc = dXe = dYe = 0, a pixel is drawn at the start point of the arc, and the command execution normally terminates.
- When dXe = dYe = 0, a circle is drawn and the command execution normally terminates.

[30] AEARC (Absolute Ellipse ARC)

PAGE AEARC-1

<FUNCTION>

AEARC draws an ellipse arc by using current pointer (start point), end point, and center point of the absolute coordinates.

TYPE: Graphic Command

<MNEMONIC>

AEARC (C, AREA, COL, OPM) a, b, Xc, Yc, Xe, Ye

<FORMAT>

COMMAND CODE

hexadecimal notation

15			9	8	7		5	4	3	2	0
1	0	1	1	1	0	0	C	AREA		COL	OPM

C = 1 : ($ B 9 X X)
C = 0 : ($ B 8 X X)

WORD NUMBER
Wn=7

EXECUTION CYCLES
Cn=10d+96

d: number of pixels being drawn

COMMAND PARAMETERS

15	0
a (16 bits)	

15	0
b (16 bits)	

15	0
Xc (16 bits)	

15	0
Yc (16 bits)	

15	0
Xe (16 bits)	

15	0
Ye (16 bits)	

<DESCRIPTION>

The AEARC command draws an arc from the current pointer, CP, to Pe of the absolute coordinates (Xe, Ye), 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 each command parameter in unit of pixel.

CP moves to the end point Pe, after the arc is drawn. However a dot is not drawn at Pe.

Note: To draw a pixel at Pe, DOT command needs to be issued after AEARC. Make sure to set the color register and the pattern RAM before issuing AEARC.

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.

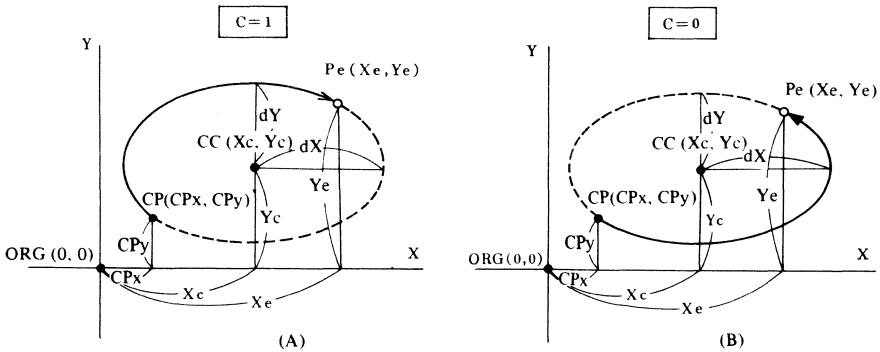


Figure C30-1 Function of AEARC

<DESCRIPTION>

In the X-Y coordinate system, let the center point of the ellipse be CC(Xc, Yc), 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.

$$\frac{(X-Xc)^2}{dX^2} + \frac{(Y-Yc)^2}{dY^2} = 1 \quad \cdots\cdots\cdots\cdots \quad (1)$$

When letting dX^2 and dY^2 be a and b, then

$$a : b = dX^2 : dY^2 \quad \cdots\cdots\cdots\cdots\cdots\cdots \quad (2)$$

by substituting (2) for (1), the result is

$$\frac{(X-Xc)^2}{a} + \frac{(Y-Yc)^2}{b} = \frac{dX^2}{a} \quad \cdots\cdots\cdots \quad (3)$$

The AEARC draws an ellipse ARC according to Equation (3).

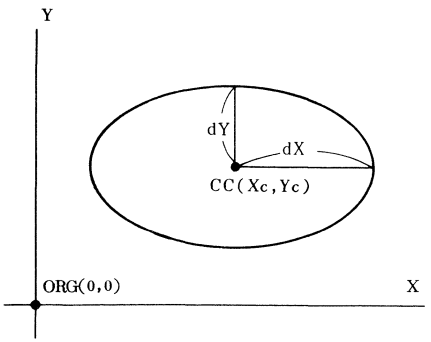


Figure C30-2 Notation of an Ellipse (1)

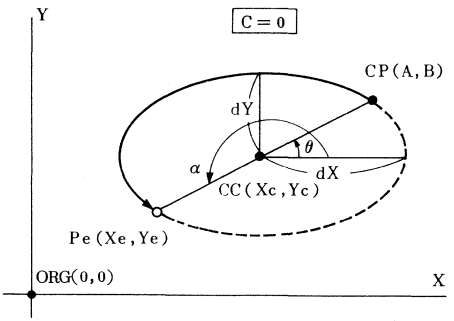


Figure C30-3 Notation of an Ellipse (2)

When setting CP (CP_x, CP_y) and Pe (Xe, Ye) as shown in Fig. C30-3 for an ellipse arc drawing, the following equations are applicable.

$$CP_x = \frac{\sqrt{\frac{b}{a}} \cdot X \cdot \cos\theta}{\sqrt{1 + (\frac{b}{a} - 1)\cos^2\theta}} + Xc \quad \ldots\ldots (4)$$

$$CP_y = \frac{\sqrt{\frac{b}{a}} \cdot X \cdot \sin\theta}{\sqrt{1 + (\frac{b}{a} - 1)\cos^2\theta}} + Yc \quad \ldots\ldots (5)$$

$$Xe = \frac{\sqrt{\frac{b}{a}} \cdot X \cdot \cos\alpha}{\sqrt{1 + (\frac{b}{a} - 1)\cos^2\alpha}} + Xc \quad \ldots\ldots (6)$$

$$Ye = \frac{\sqrt{\frac{b}{a}} \cdot X \cdot \sin\alpha}{\sqrt{1 + (\frac{b}{a} - 1)\cos^2\alpha}} + Yc \quad \ldots\ldots (7)$$

$a : b = dX^2 : dY^2$

<Example>

Suppose that CP is at (10, 3) in the absolute coordinate system. When a is set to 16, b to 9, Xc to 10, Yc to 6, Xe to 6, and Ye to 6, the ellipse arc is drawn as shown in Fig. C30-4.

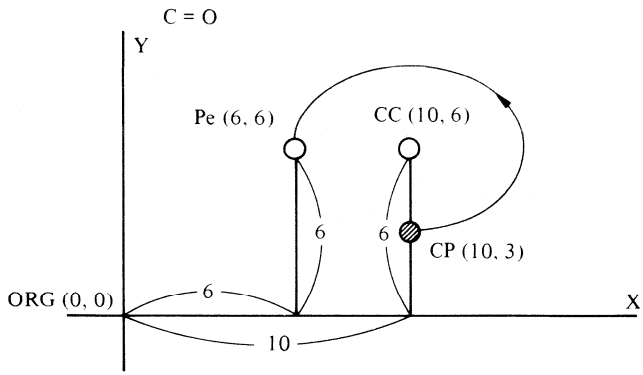


Fig. C30-4 AEARC Execution Example

<Note for parameter setup>

- 2's complement is used to set the netative value.
- Specifiable range

$$0 \leq \sqrt{(Xc - CP_x)^2 + (Yc - CP_y)^2} \leq \frac{4095}{\max(a, b)}$$

(max (a, b) means the greater value among a and b.)

- When $(CP_x, CP_y) = (Xc, Yc) = (Xe, Ye)$, a pixel is drawn at the start point of the ellipse arc, and command execution normally terminates.

- When $(CP_x, CP_y) = (Xe, Ye)$, an ellipse is drawn and command execution normally terminates.

[31] REARC (Relative Ellipse ARC)

PAGE REARC-1

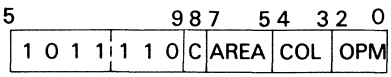
<FUNCTION>
REARC draws an ellipse arc by using current pointer (start point), and, end point and center point of the relative coordinates.

TYPE: Graphic Command

<MNEMONIC>
REARC (C, AREA, COL, OPM) a, b, dXc, dYc, dXe, dYe

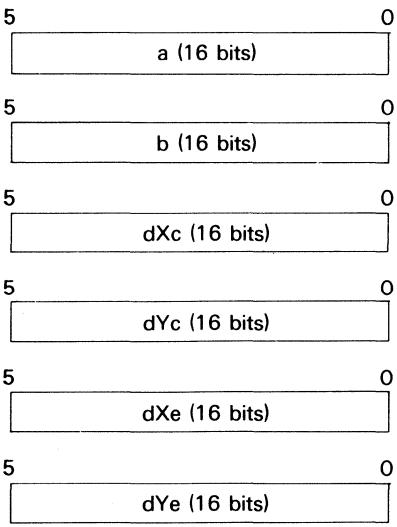
<FORMAT>

COMMAND CODE



hexadecimal notation
C = 1 : ($ B D X X)
C = 0 : ($ B C X X)

COMMAND PARAMETERS



WORD NUMBER
Wn=7

EXECUTION CYCLES
Cn=10d+96

d: number of pixels being drawn

<DESCRIPTION>

As shown in Fig. C31-1, the REARC command draws an ellipse arc from the current pointer, CP, to Pe (dXe, dYe) of the relative coordinates, the relative coordinates CC (dXc, dYc) being the center point.

Relative components dX and dY of CC and Pe are set in each command parameter in unit of pixel.

Note) To draw a pixel at Pe, DOT command needs to be issued after REARC. Make sure to set the color register and the pattern RAM before issuing REARC.

Bit 8 (C) of the command code selects whether an arc is drawn clockwise or counterclockwise. When C is "1", the ellipse arc is drawn clockwise, and when C is "0", the ellipse arc is drawn counterclockwise as shown in Fig. C31-1.

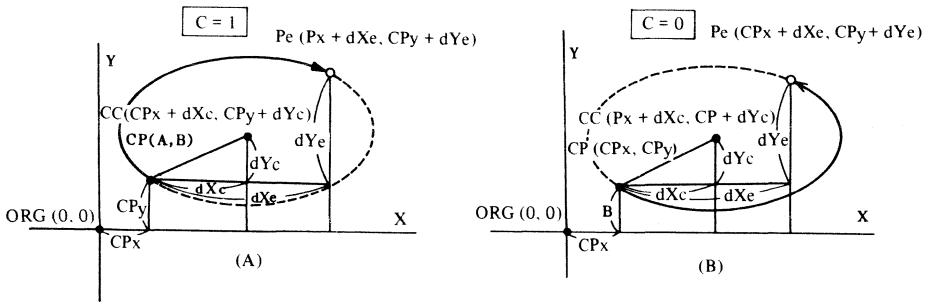


Figure C31-1 Function of REARC

<Example>

Suppose that CP is (10, 3) in the absolute coordinate system. When a is set to 16, b to 9, dXc to 0, dYc to 3, dXe to −4 and dYe to 3, the ellipse arc is drawn as shown in Fig. C31-2.

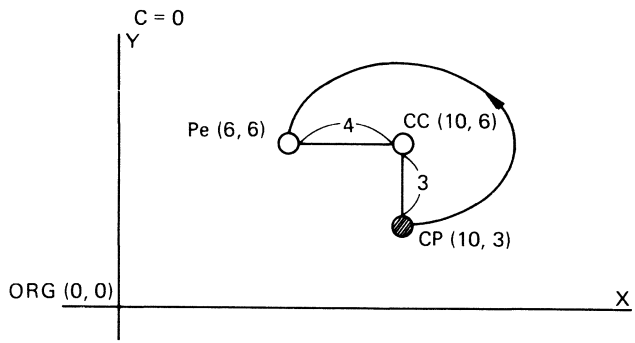


Figure C31-2 REARC Execution Example

<Note for parameter setup>

- 2's complement is used to specify the negative value.
- Specifiable range

$$0 \leq \sqrt{dXc^2 + dYc^2} \leq \frac{4095}{\max(a, b)}$$

(max (a, b) means the greater value among a and b.)

- When dXc = dYc = dXe = dYe = 0, a pixel is drawn at the start point of the ellipse arc, and command execution normally terminates.
- When dXe = dYe = 0, an ellipse is drawn and command execution normally terminates.

[32] AFRCT (Absolute Filled Rectangle)

PAGE AFRCT-1

TYPE: Graphic Command

<FUNCTION>

AFRCT command paints the rectangular area specified with CP (Current Pointer) and the command parameters (the absolute coordinates) according to the figure pattern stored in the Pattern RAM.

<MNEMONIC>

AFRC (AREA, COL, OPM) X, Y

<FORMAT>

COMMAND CODE — hexadecimal notation

15	8	7	5	4	3	2	0
1 1 0 0	0 0 0 0	AREA		COL		OPM	($C0XX)

COMMAND PARAMETERS

15	0
X (16 bits)	

15	0
Y (16 bits)	

WORD NUMBER

Wn = 3

EXECUTION CYCLES

Cn = (P·A + 8)B + 18

$\begin{cases} P=4 \ (OPM='000'\sim'011') \\ P=6 \ (OPM='100'\sim'111') \end{cases}$

$A = |P_{cx} - CP_x| + 1$

$B = |P_{cy} - CP_y| + 1$

<DESCRIPTION>

The Absolute Filled Rectangle Command (AFRCT) paints the rectangle area according to the color information in the pattern RAM. Each side of the rectangle is 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.

Absolute components X and Y of Pc are set in first and second command parameters in unit of pixel.

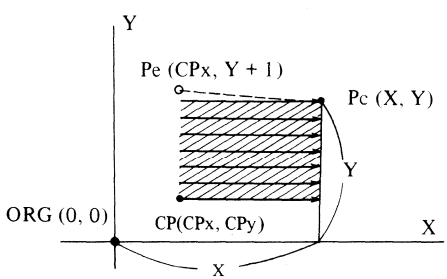
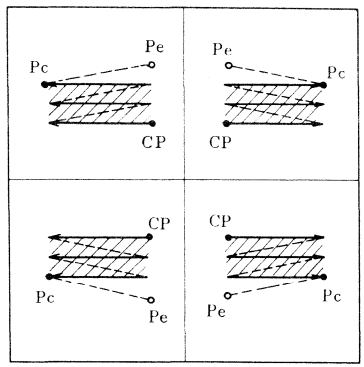


Figure C32-1 Function of AFRCT

Painting direction of AFRCT varies according to the positions of CP and Pc, as shown in Fig. C32-2. CP is moved to Pe at the termination of the command. The drawing at the end point Pe is not performed.



Note) The Pattern RAM is scanned, from low bit to higher bit, and from low address to higher address, independently of the positions of CP and Pc.

Figure C32-2 Painting Direction of AFRCT

<EXAMPLE>

Suppose that CP is (CP_x, CP_y) in the absolute coordinate system, X is set to X_1 and Y to Y_1 in each command parameter, and the Drawing Parameter Registers for the Pattern RAM is set as follows:

Pattern Start Point..... (PSX, PSY)
Pattern End Point..... (PEX, PEY)
Pattern Pointer..... (PPX, PPY)

If $X_1 > CP_x$ and $Y_1 < CP_y$, then, the rectangular area is painted with the AFRCT command as shown in Fig. C32-3.

Note) Make sure that a pattern like "F" in the figure should be drawn in the Pattern RAM in advance.

COMMAND CODE

15															0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

($ C 0 0 0)

COMMAND PARAMETERS

15															0
X_1															

($ X X X X)

15															0
Y_1															

($ Y Y Y Y)

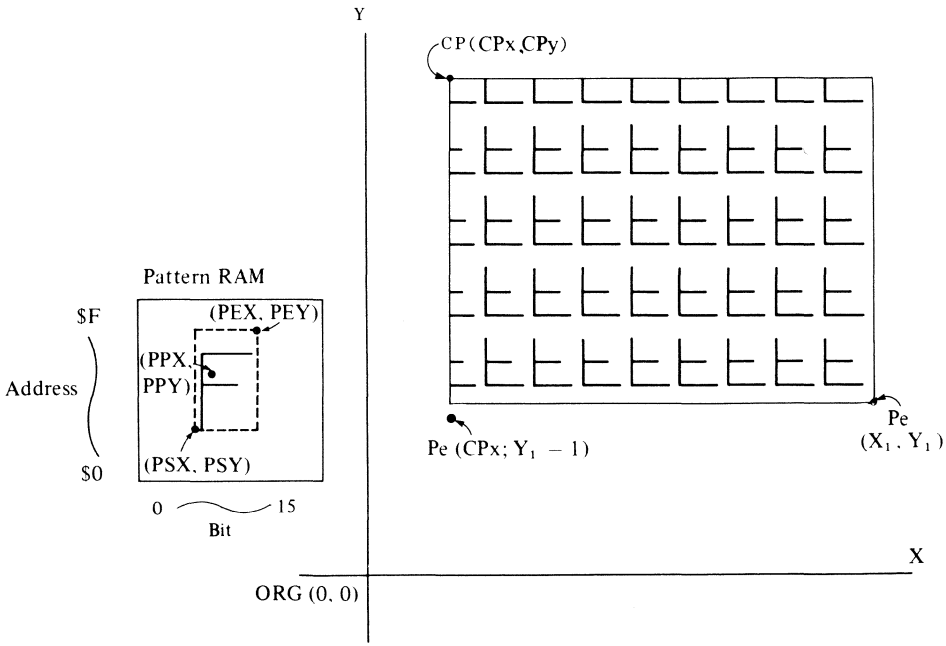


Figure C32-3 Example of AFRCT Execution

<Note for parameter setup>

- Use 2's complement to specify the negative value.
- Specifiable range

$$CP_x - 32767 \leqq X \leqq 32767 + CP_x$$

$$CP_y - 32767 \leqq Y \leqq 32767 + CP_y$$

- When $X = CP_x$ and $Y = CP_y$, the command normally terminates after drawing a pixel at the start point.

[33] RFRCT (Relative Filled Rectangle)

PAGE RFRCT-1

TYPE Graphic Command

<FUNCTION>

RFRCT command paints the rectangular area specified with CP (Current Pointer) and the command parameters (the relative coordinates) according to the figure pattern stored in the Pattern RAM.

<MNEMONIC>

RFRCT (AREA, COL, OPM) dX, dY

<FORMAT>

COMMAND CODE

hexadecimal notation

15								8	7		5	4		3	2		0	
1	1	0	0	0	1	0	0	AREA			COL			OPM				($ C 4 X X)

COMMAND PARAMETERS

15	0
dX (16 bits)	

15	0
dY (16 bits)	

WORD NUMBER

Wn = 3

EXECUTION CYCLES

Cn = (P·A + 8)B + 18

{P = 4 (OPM = '000' ~ '011')
{P = 6 (OPM = '100' ~ '111')

A = |dX| + 1

B = |dY| + 1

<DESCRIPTION>

The Relative Filled Rectangle Command (RFRCT) paints the rectangular area according to the color information in the pattern RAM. Each side of the rectangle is parallel to the coordinate axis. Two corner points on the diagonal are CP and Pe (CP_x + dX, CP_y + dY) at the relative coordinate point from CP.

Relative components dX and dY of Pc are set in first and second command parameters in unit of pixel.

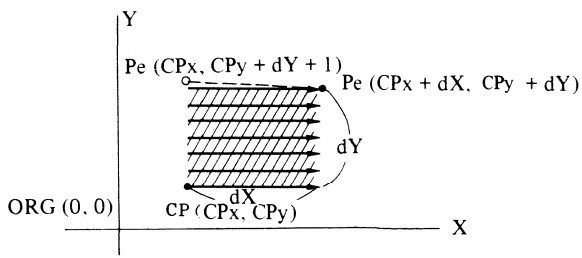
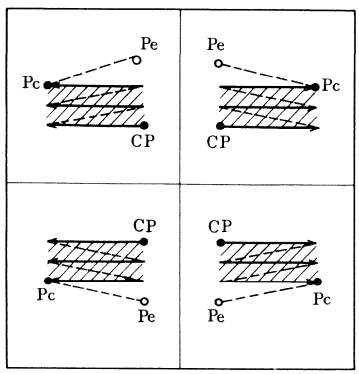


Figure C33-1 Function of RFRCT

Painting direction of RFRCT varies according to the positions of CP and Pe, as shown in Fig. C33-2. CP is moved to Pe at the termination of the command. The drawing at the end point Pe is not performed.



Note: The Pattern RAM is scanned, from lower bit to higher bit, and from lower address to higher address, independently of the positions of CP and Pc.

Figure C33-2 Painting Direction of RFRCT

<EXAMPLE>

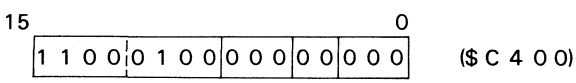
Suppose that CP is (CP_X, CP_Y) in the absolute coordinate system, dX is set to dX_1 and dY to dY_1 in each command parameter, and the Drawing Parameter Registers for the Pattern RAM is set as follows:

Pattern Start Point..... (PSX, PSY)
Pattern End Point..... (PEX, PEY)
Pattern Pointer..... (PPX, PPY)

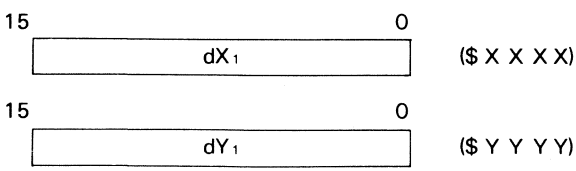
If $dX_1 > 0$ and $dY_1 < 0$, then, the rectangular area is painted with the AFRCT command as shown in Fig. C33-3.

Note: Make sure that a pattern like "F" in the figure should be drawn in the Pattern RAM in advance.

COMMAND CODE



COMMAND PARAMETERS



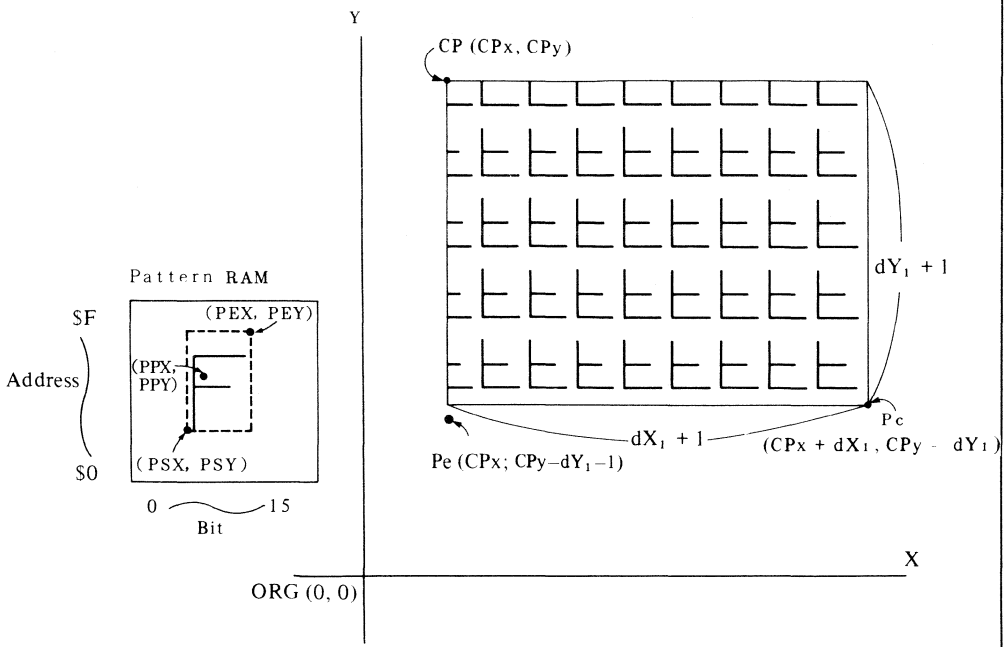


Figure C33-3 Example of RFRC Execution

< Note for parameter setup >

- Use 2's complement to specify the negative value.
- Specifiable range
  $-32767 \leqq dX \leqq 32767$
  $-32767 \leqq dY \leqq 32767$
- When "dX = dY = 0", the command normally terminates after drawing a pixel at the start point.

[34] PAINT (Paint)

PAGE PAINT-1

< FUNCTION >

PAINT command paints the closed area surrounded by edge color using the figure pattern stored in the pattern RAM.

TYPE: Graphic Command

< MNEMONIC >

PAINT (E, AREA)

< FORMAT >

COMMAND CODE — hexadecimal notation

15	9	8	7	5	4	3	0
1 1 0 0	1 0 0	E	AREA	0	0 0 0 0		($ C X X 0)

COMMAND PARAMETERS

— NON —

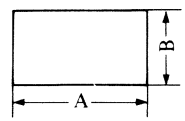
WORD NUMBER

Wn = 1

Command execution cycle number

Cn = (18·A + 102)B − 58

(When painting rectangle)



< DESCRIPTION >

The "Paint" command (PAINT) paints the closed area surrounded by edge color defined in the Drawing Parameter Registers (EDG: edge color, CL0: color 0, CL1: color 1), according to the figure pattern (see Figure C34-3) stored in the Pattern RAM. (Detail definition of edge color is shown in next page.) In this case, the CP must be set inside the closed area, and the paint operation is performed only inside the closed area. If same color dots as EDG, CL0 or CL1 exist in the area, these dots are regarded as already painted.

Above function is the case of E = 0. (Bit 8 of the command) When E = 1 is set to Bit 8, PAINT command only paints the area painted by the color data of the EDG register. (i.e. only dots of the EDG color data are regarded as unpainted.)

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.

If the current pointer (CP) is set on the painted dot when issuing PAINT command, the painting operation is not executed.

Note) In the "Paint" command (PAINT), the Color Mode (COL) and the Operation Mode (OPM) cannot be used.
Therefore, COL (bit 4, 3) and OPM (bit 2 ~ 0) of the command code must be set "00000". (see above FORMAT.)

< Definition of Edge Color >

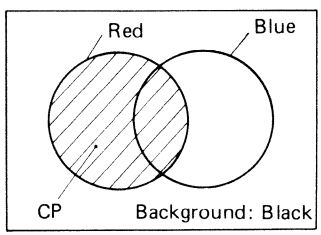
The table below shows the definition of edge color and the conditions to regard a dot as already painted/unpainted.

COL: background color
EDG: the color defined by the EDG register
CL0: the color defined by the CL0 register
CL1: the color defined by the CL1 register

	Edge Color Definition	
	E = 0	E = 1
Dot is regarded as already painted and as edge color.	COL = EDG or COL = CL0 or COL = CL1 (Satisfies one of the conditions above)	COL ≠ EDG or COL = CL0 or COL = CL1 (Satisfies one of the conditions above)
Dot is regarded as unpainted.	COL ≠ EDG and COL ≠ CL0 and COL ≠ CL1 (Satisfies all the conditions above)	COL = EDG and COL ≠ CL0 and COL ≠ CL1 (Satisfies all the conditions above)

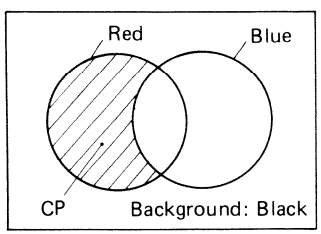
In general case, bit 8 (E) of the command and the EDG register should be used for edge color definition as described below.

E = 0 : The edge color is defined by the data in the EDG register. (See figure C34-1)
E = 1 : The edge color is defined to be all colors except for the color in the EDG register. (See figure C34-2)



E = 0: When "red" is set to the EDG register, and PAINT is executed at E = 0, the area as shown left is drawn.
(In this case, CL0 = CL1 = "red")

Figure C34-1 Paint Function (E=0)



E = 1: When "black" is set to the EDG register, and PAINT is executed at E = 1, the area as shown left is drawn.
(In this case, CL0 = (L1 = "red")

Figure C34-2 Paint Function (E=1)

<NOTE>

- When issuing the PAINT command, the painting operation is not executed at all if the current pointer (CP) is set on the painted dot. (The command is normally terminated.)
- The dots regarded to have been painted cannot be painted any more. For example, when E = 0, the dot painting with the color defined by CL1 is prevented if the same color as the background color (COL) is defined for CL0, because the dot is regarded to have already been painted.
- Set EDG, CL0 and CL1 registers before issuing this command. The color registers (CL0 and CL1) must be set even when all "0" or "1" pattern is drawn on the pattern RAM. CL0 = CL1 is required in this case.
- As described above, the tiling operation is not provided with the PAINT command in 1 bit/pixel mode.
- At E = 1, prohibited conditions are as follows:
  CL0 = EDG, or CL1 = EDG

<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.

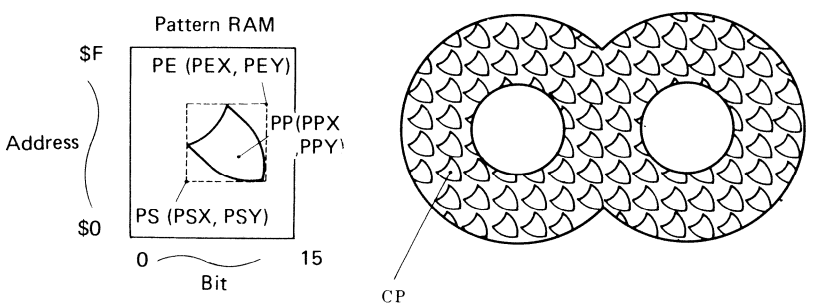


Figure C34-3 Paint Function Using Figure Pattern

＜Paint Procedure＞

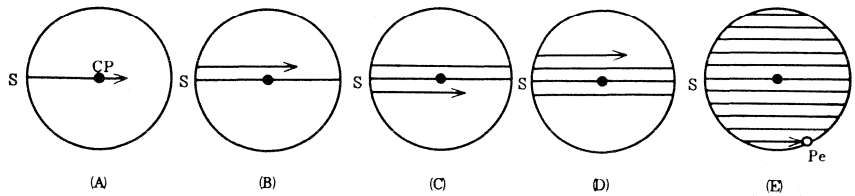


Figure C34-4 Paint Procedure

＜Paint Procedure＞

Painting is continuously performed parallel to the X axis (left to right), and in the Y direction, dot by dot. Fig. C34-4 shows an example of painting the encircled area. First, painting begins from point S on a line which is crossing CP and parallel to the X axis (figure A). Next, painting is executed on the adjacent lines which are above and 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.

＜Complex Figure Painting＞

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 points information being pushed into the stack.

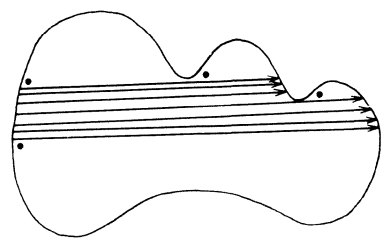
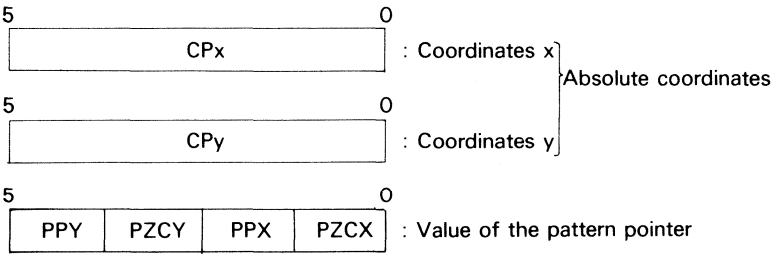


Figure C34-5 Paint Stack Function

The ACRTC can store such four points information. If the points are within four, one PAINT command can completely paint a complex figure.

If the points are five or more, all points information 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. (In the Read FIFO, Two (2) points information can be set at most.)



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, two or less points information can be pushed into the Read FIFO. 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 with using the informations in the Read FIFO.

Figure C34-6 shows a flow chart to complete every type of painting.

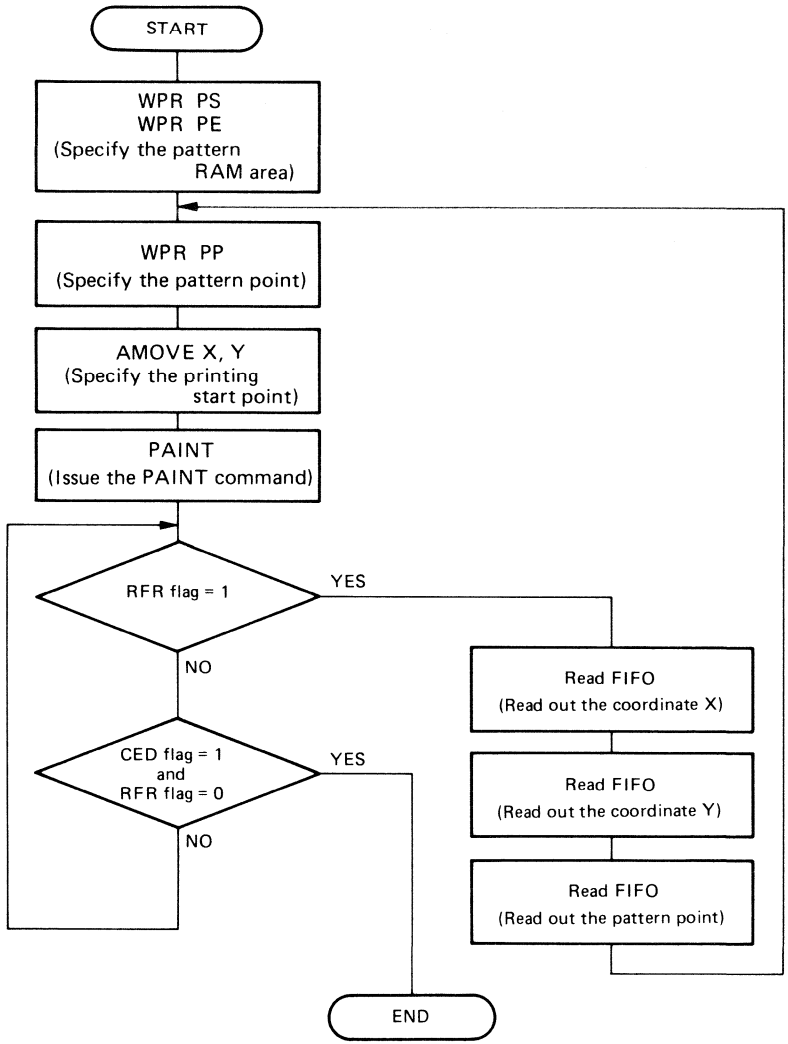


Figure C34-6 Paint Flow of Complex Figures Using PAINT Command

< PAINT Area Detection Mode >

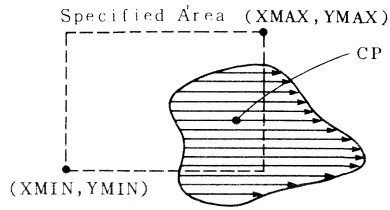
PAINT Area Detection modes have each of the following functions.

AREA	PAINT Command Execution
000	Not check the specified area.
001 **	The ABT bit 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 **	The ABT bit 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.

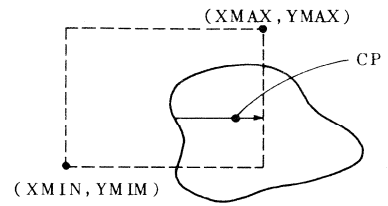
* Set CP with care of the area. See Figure C34-6A.

** While CP is searching point S (see Figure C34-4) at first of the execution, no dot is painted, but the ABT bit is set and the command execution is truncated when CP moves outside/inside the specified area.

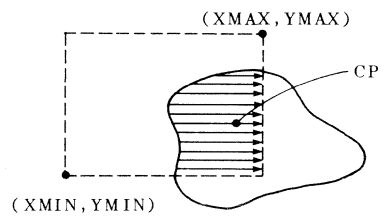
(i) AREA = 0 0 0
AREA = 1 0 0



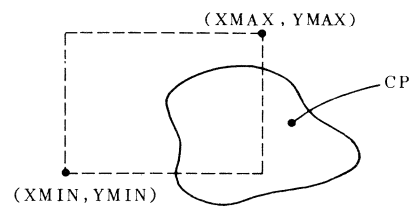
(ii) AREA = 0 0 1
(The ABT bit is set.)



(iii) AREA = 0 1 0
(AREA flag is not changed.)
AREA = 0 1 1
(AREA flag is set.)



(iv) AREA = 1 0 1
(The ABT bit is set.)



(v) AREA = 1 1 0
(AREA flag is not changed.)
AREA = 1 1 1
(AREA flag is set.)

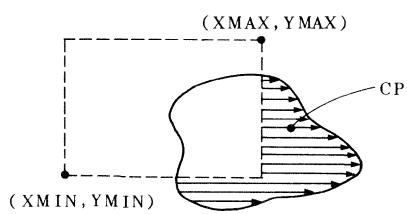


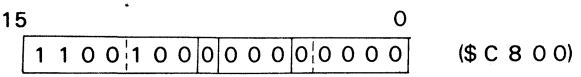
Figure C34-6A Paint Command Example with AREA Modes

<**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 frame buffer, 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.

Note) Assume the background is painted with the color which is not specified by EDG, CL0 nor CL1.

COMMAND CODE



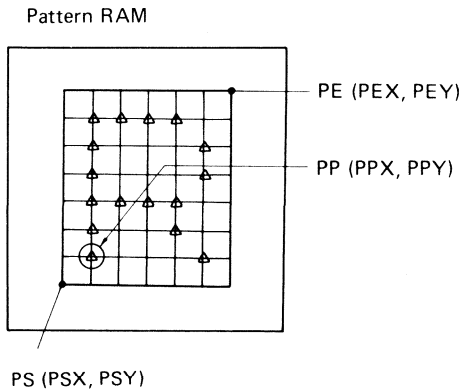


Figure C34-7 Setting of Pattern RAM

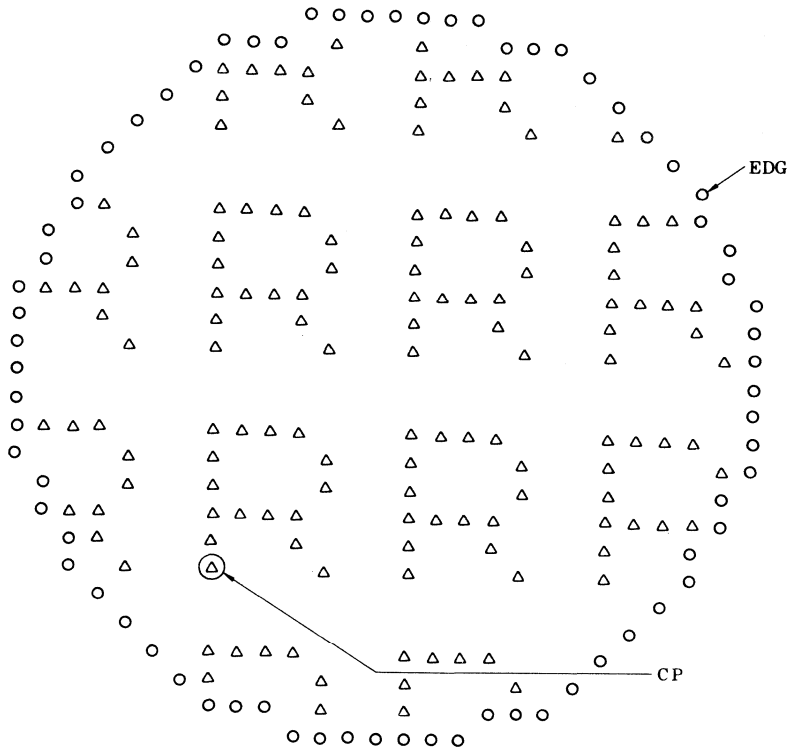
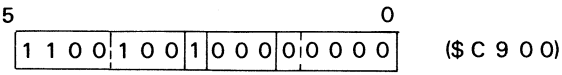


Figure C34-8 Example of PAINT Execution (E = "0")

<**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 frame buffer 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).

COMMAND CODE



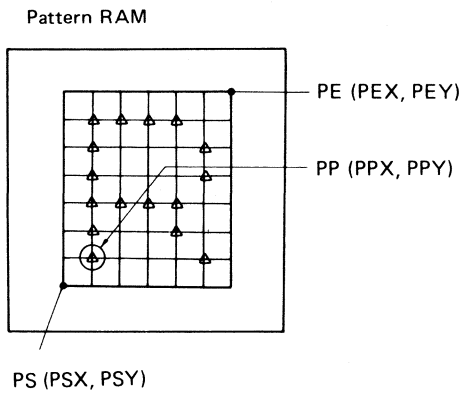


Figure C34-9 Setting of Pattern RAM

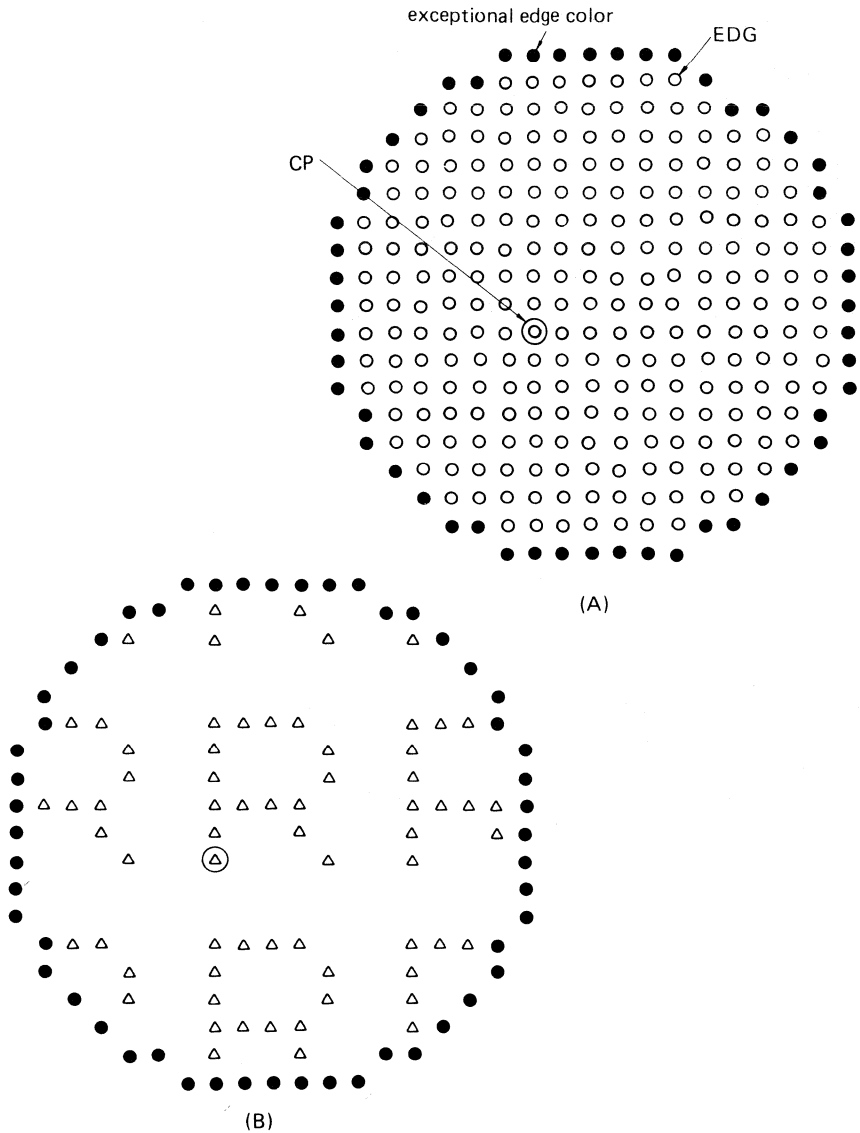


Figure C34-10 Example of PAINT Execution (E = "1")

[35] DOT (Dot Command)

<FUNCTION>
DOT Command marks a dot on the coordinates where the CP indicates.

TYPE: Graphic Command

<MNEMONIC>
DOT (AREA, COL, OPM)

<FORMAT>

COMMAND CODE

hexadecimal notation

15			8	7		5	4		3	2		0
1	1	0	0	1	1	0	0	AREA		COL		OPM

($ C C X X)

COMMAND PARAMETERS

— NON —

WORD NUMBER
Wn=1

EXECUTION CYCLES
Cn=8

<DESCRIPTION>

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.

Note) The Color Registers and the Pattern RAM must be set before issuing this command.
After termination of DOT command, the pattern pointer (PP) doesn't move.

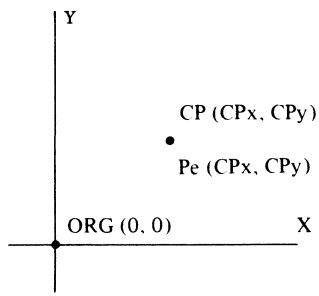


Figure C35-1 Function of DOT

<EXAMPLE>

In the case that the absolute coordinate of the CP is (10, 8) the DOT Command marks a dot as shown in Fig. C35-2.

COMMAND CODE

15					0
1 1 0 0	1 1 0 0	0 0 0	0 0	0 0 0	($ C C 0 0)

COMMAND PARAMETERS

— NONE —

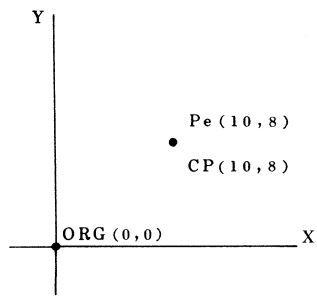


Figure C35-2 Example of DOT Execution

The LINE Commands and ARC Commands do not draw a dot at the finishing point, Pe. The DOT Command can be used to draw a dot at the Pe to draw a complete line or arc.

COMMAND CODE

15		8 7	5 4	3 2	0
1 1 0 0	1 1 0 0	AREA	COL	OPM	

EXAMPLE

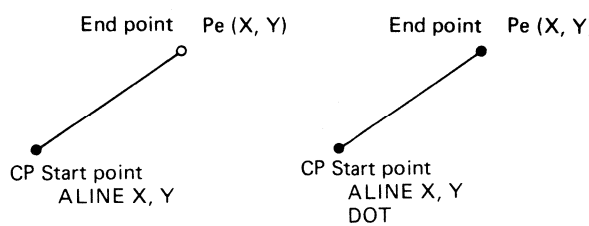


Figure C35-3 DOT Command for the End Point

[36] PTN (Pattern)

PAGE PTN-1

<FUNCTION>
The graphic pattern defined in the Pattern RAM is drawn onto the rectangular area specified by the current pointer and by the pattern size.

TYPE: Graphic Command

<MNEMONIC>
PTN (SL, SD, AREA, COL, OPM) SZ

<FORMAT>

COMMAND CODE — hexadecimal notation

15 12	11	10	8	7 5	4 3	2 0
1 1 0 1	SL	SD		AREA	COL	OPM

($ D X X X)

COMMAND PARAMETERS

15 8	7 0
SZy	SZx

SZx, SZy
Setting: 0~255
Meaning: 1~256
in units of pixels

WORD NUMBER
Wn=2

EXECUTION CYCLES
Cn=(P·A+10)·B+20

{P=4 (OMP='000'~'011')
{P=6 (OMP='100'~'111')
A = SZ_x + 1
B = SZ_y + 1

<DESCRIPTION>

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).

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 CL0 and CL1 for execution of pattern drawing.

Parameter SZ is divided into X component (SZx) and Y component (SZy), each component being set in unit of pixel.

The PTN command has the CP scan direction set up in unit 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).

Note)
- When SL = 0, if SD is set to "001", "011", "101" or "111", the figure pattern in the frame buffer looks different in size from that in the pattern RAM, because the interdots distance is enlarged by $\sqrt{2}$ times in this case.
- Pattern setting
  When "$SZ_y = SZ_x = 0$" is set, the command is normally terminated after drawing a pixel at CP.

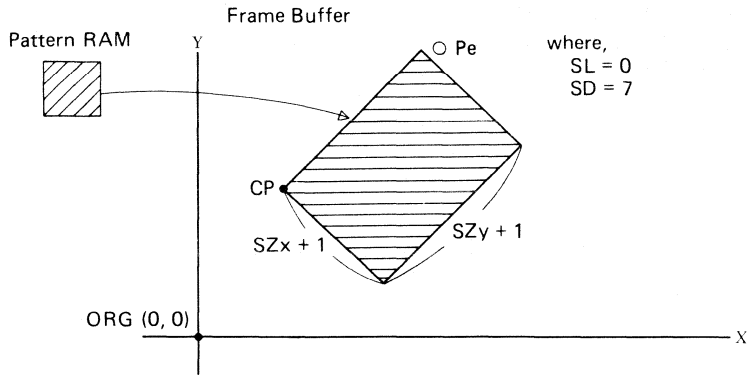
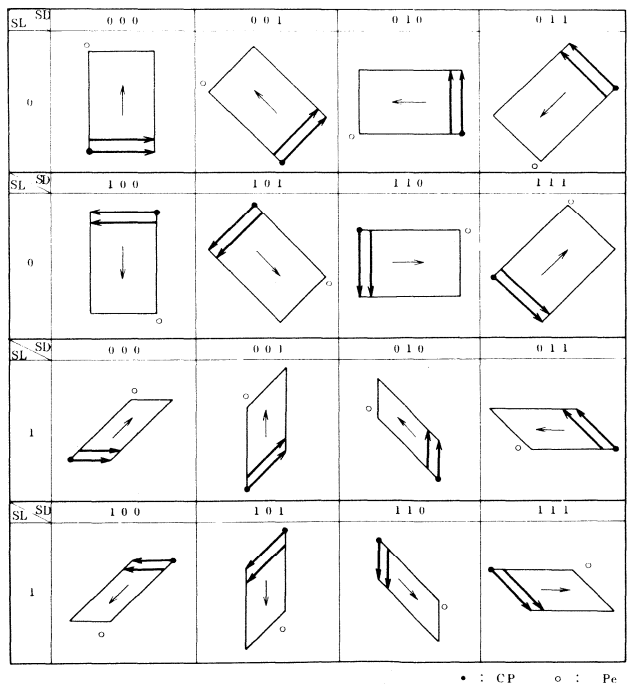


Figure C36-1 Function of PTN

Table C36-1 Directions of CP Scan

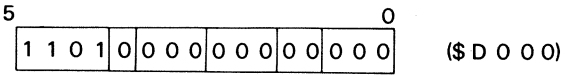


< Example of Command Execution >

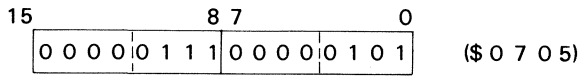
From the pattern RAM, take out a pattern according to the PS (PSX, PSY) and PE (PEX, PEY), and execute the PTN command.

(1) Where PP=PS, PZ=0, SZ=PE−PS, SL=0, SD=0

COMMAND CODE



COMMAND PARAMETERS



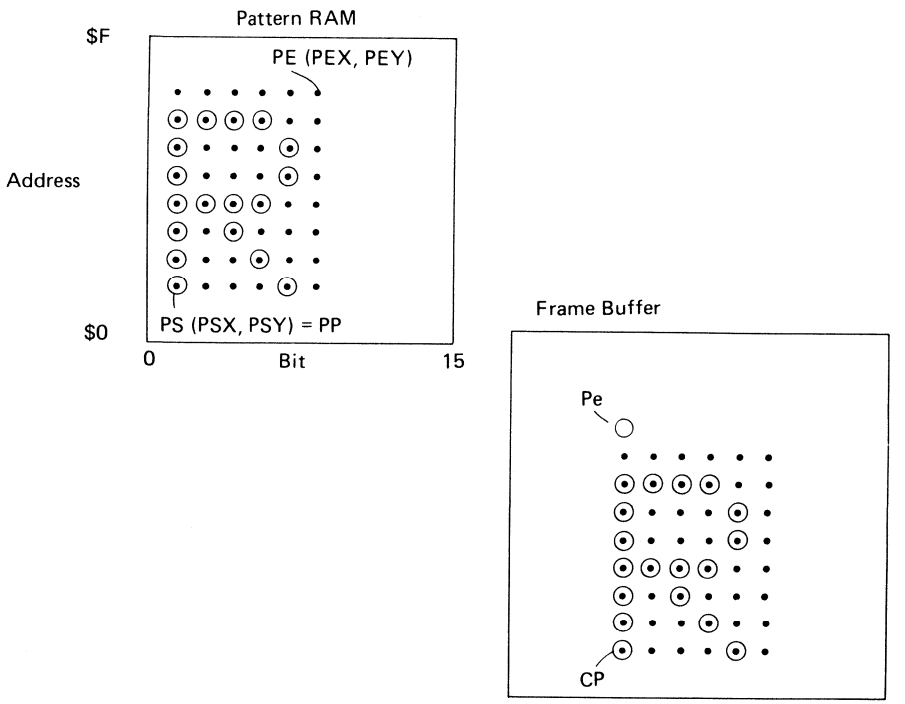
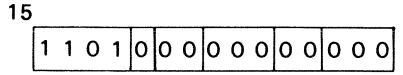


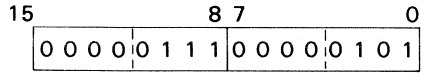
Figure C36-2 Example of PTN Execution (1)

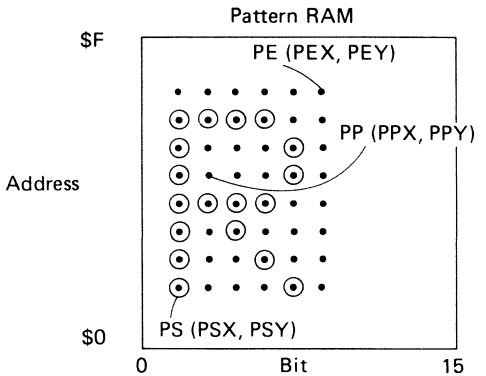
(2) Where PP≠PS, PZ=0, SZ=PE−PS, SL=0, SD=0

COMMAND CODE



COMMAND PARAMETERS





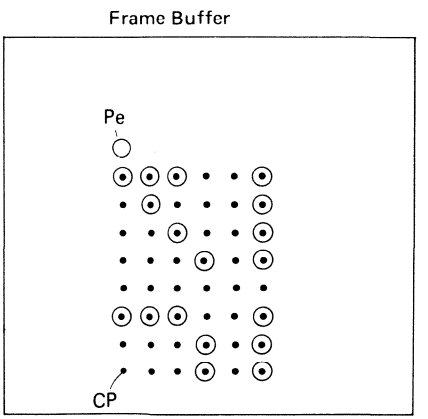
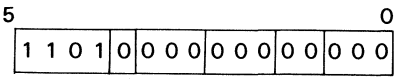


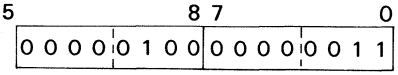
Figure 36-3 Example of PTN Execution (2)

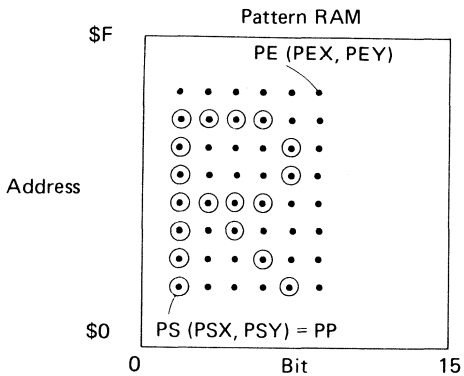
(3) Where PP=PS, PZ=0, SZ<PE−PS, SL=0, SD=0

COMMAND CODE



COMMAND PARAMETERS





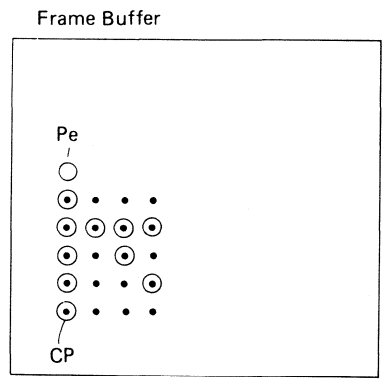
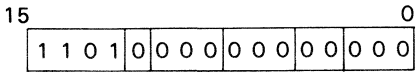


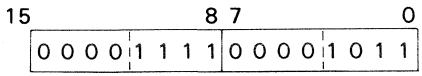
Figure 36-4 Example of PTN Execution (3)

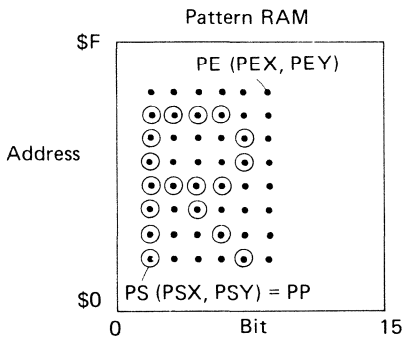
(4) Where PP=PS, PZ=0, SZ>PE−PS, SL=0, SD=0

COMMAND CODE



COMMAND PARAMETERS





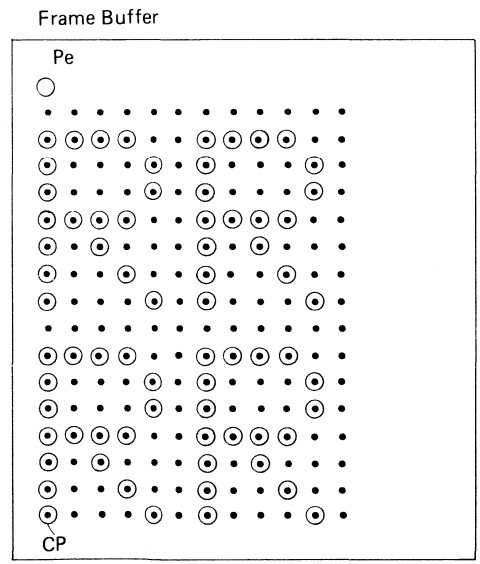
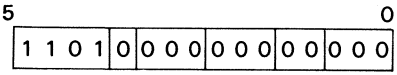


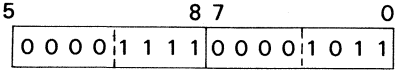
Figure 36-5 Example of PTN Execution (4)

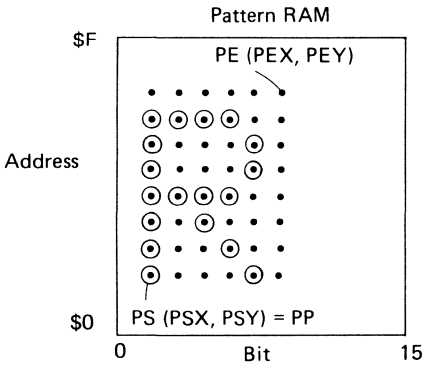
(5) Where PP=PS, PZX=1, PZY=1, SZ>PE−PS, SL=0, SD=0 (Zoom X2)

COMMAND CODE



COMMAND PARAMETERS





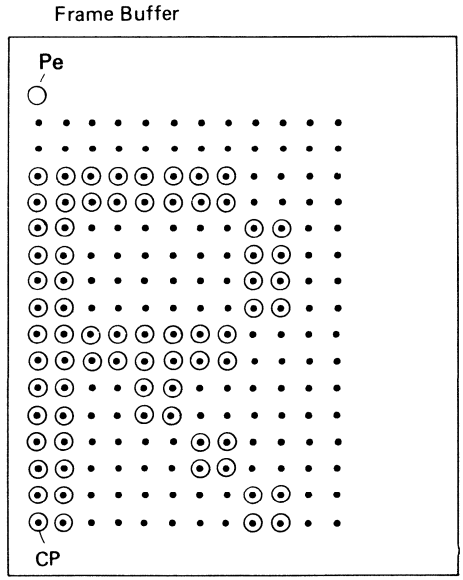
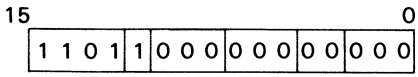


Figure 36-6 Example of PTN Execution (5)

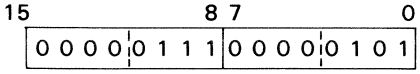
(6) Where PP=PS, PZ=0, SZ=PE−PS, SL=1, SD=0

COMMAND CODE

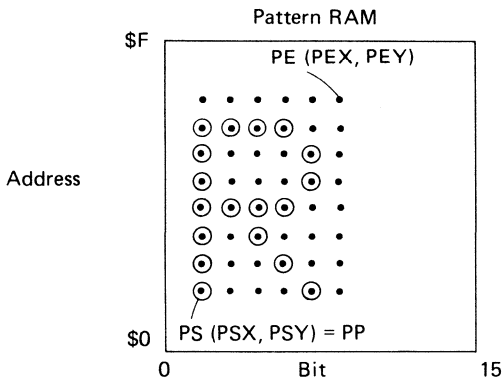


($D800)

COMMAND PARAMETERS



($0705)



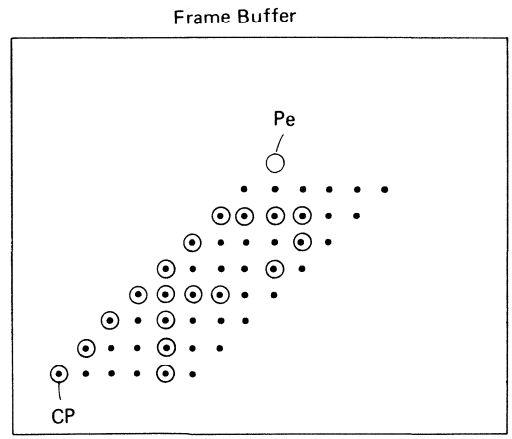
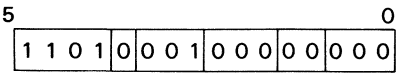


Figure 36-7 Example of PTN Execution (6)

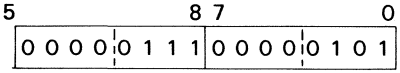
(7) Where PP=PS, PZ=0, SZ=PE−PS, SL=0, SD=1

COMMAND CODE

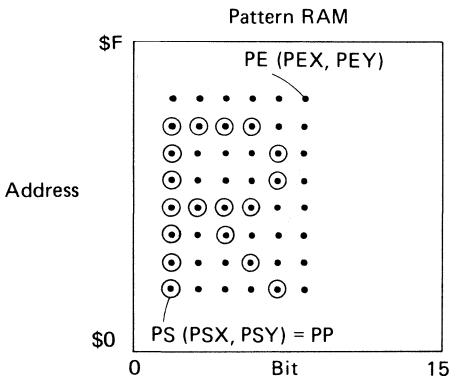


($ D 1 0 0)

COMMAND PARAMETERS



($ 0 7 0 5)



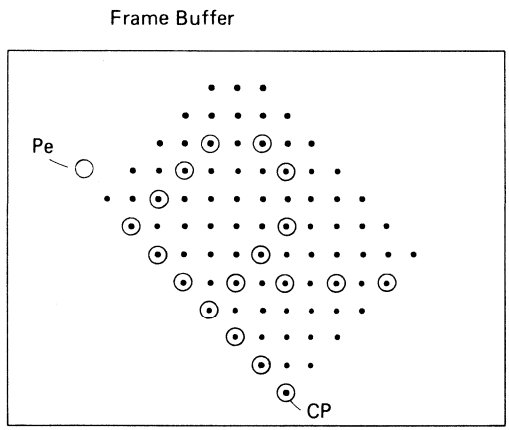


Figure 36-8 Example of PTN Execution (7)

[37] AGCPY (Absolute Graphic Copy)

PAGE AGCPY-1

<FUNCTION>
AGCPY command copies a rectangular area specified by the absolute coordinates to the address specified by CP (Current Pointer)

TYPE: Graphic Command

<MNEMONIC>
AGCPY (S, DSD, AREA, COL, OPM) Xs, Ys, DX, DY

<FORMAT>

COMMAND CODE

hexadecimal notation

15 12	11	10 8	7 5	4 3	2 0
1 1 1 0	S	D S D	AREA	0 0	OPM

($ E X X X)

COMMAND PARAMETERS

15 0
Xs
Ys
DX
DY

WORD NUMBER
Wn = 5

EXECUTION CYCLES
Cn = {(P + 2)A + 10}B + 70

{P = 4 (OPM = '000' ~ '011')
{P = 6 (OPM = '100' ~ '111')

{S = 0 : A = |DX| + 1
{ B = |DY| + 1
{S = 1 : A = |DY| + 1
{ B = |DX| + 1

<DESCRIPTION>

The Absolute Graphic Copy Command (AGCPY) copies data from an rectangular area on the frame buffer (the source area) to another location on the frame buffer (the destination area) specified by the initial starting point CP. Each side of the source rectangular 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.

X and Y components of Pss, Xs and Ys, expressed by absolute X − Y coordinates from the origin, are set in the command parameter in units of pixels.

X and Y components of Pse, DX and DY, expressed by relative X − Y coordinates from Pss, are set in the command parameter in units of pixels.

Note) Pattern RAM is not in use with AGCPY command.

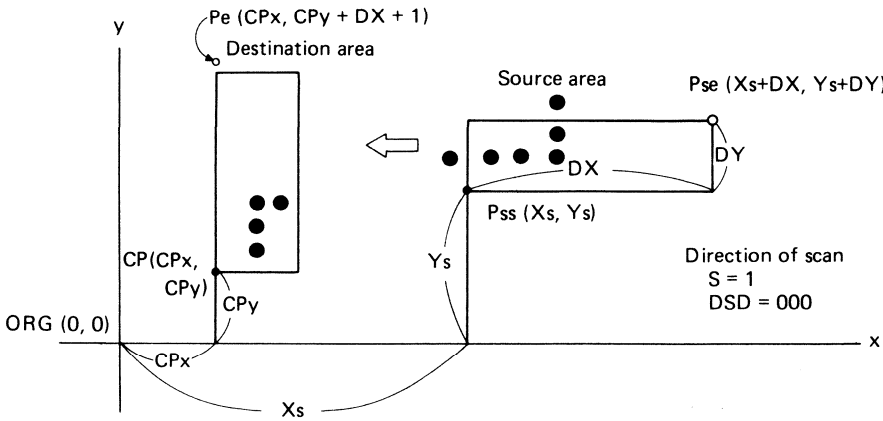


Figure C37-1 Function of AGCPY

<DIRECTION OF POINTER SCAN>

The direction of pointer scan is determined by bit 11 (s) and bit 8-10 (DSD) in the command code through the AGCPY command.

(a) S (Source Scan Direction)

COMMAND CODE

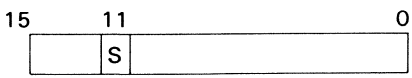
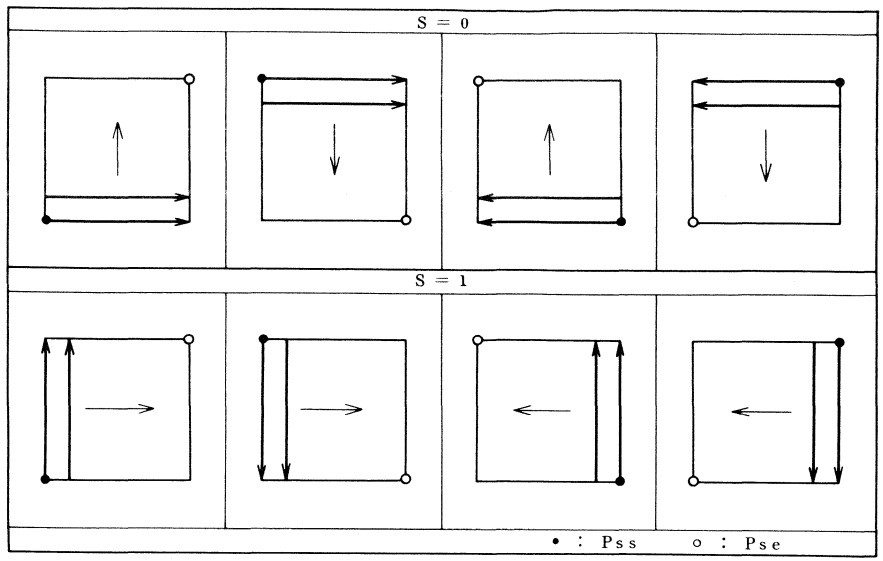


Table C37-1 Direction of Source Data Scan



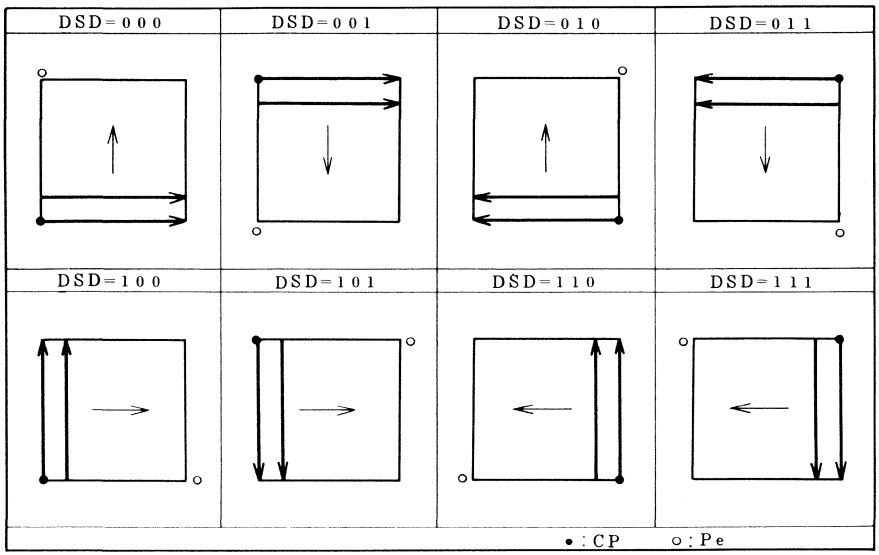
The direction of source data scan on the frame buffer is determined with bit 11 in the command code and the positions of Pss and Pse, as shown in Table C37-1.

(b) DSD (Destination Scan Direction)

COMMAND CODE

15	10 9 8	0
	D S D	

Table C37-2 Direction of Destination Data Scan



As shown Table C37-2, the direction of destination data scan on the frame buffer is determined with bits 10-8 in the command code and positions of CP and Pe.

After termination of the command, CP moves to Pe.

<Note for the parameter setting>

- Use 2's complement to setup negative value.
- Valid range

  $-32767 \leqq Xs \leqq 32767, -32767 \leqq Ys \leqq 32767$

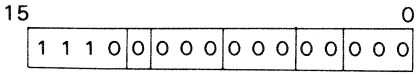
  $-32767 \leqq DX \leqq 32767, -32767 \leqq DY \leqq 32767$

<EXAMPLE>

Assume the absolute coordinates of CP is (4, 2), 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).

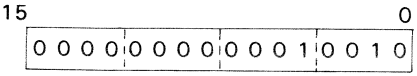
AGCPY (Absolute Graphic Copy)

COMMAND CODE

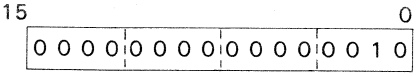


($ E 0 0 0)

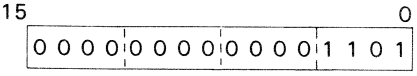
COMMAND PARAMETERS



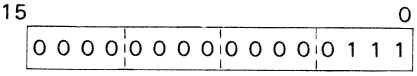
($ 0 0 1 2)



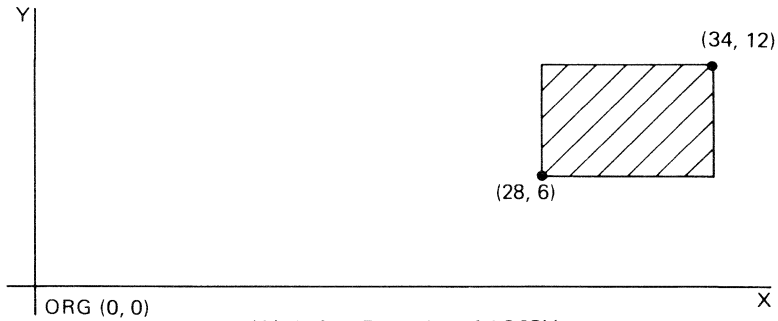
($ 0 0 0 2)



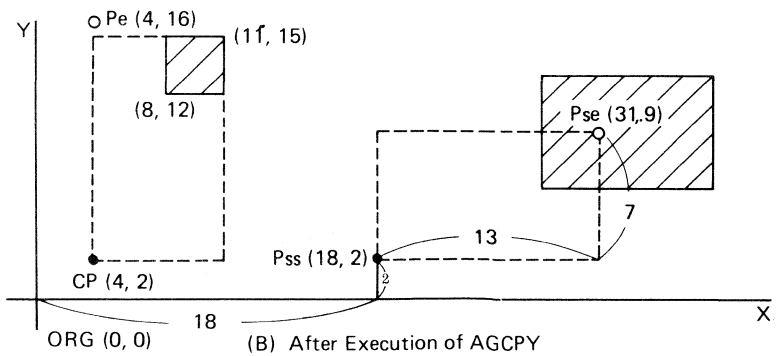
($ 0 0 0 D)



($ 0 0 0 7)



(A) Before Execution of AGCPY



(B) After Execution of AGCPY

Figure C37-2 Example of AGCPY Execution

[38] RGCPY (Relative Graphic Copy)

PAGE RGCPY-1

TYPE: Graphic Command

<FUNCTION>

RGCPY command copies a rectangular area specified by the relative coordinates based on CP (Current Pointer) to the address specified by CP.

<MNEMONIC>

RGCPY (S, DSD, AREA, COL, OPM) dXs, dYs, DX, DY

<FORMAT>

COMMAND CODE — hexadecimal notation

15 12	11	10 8	7 5	4 3	2 0
1 1 1 1	S	D S D	AREA	0 0	OPM

($ F X X X)

COMMAND PARAMETERS

15 0
dXs
dYs
DX
DY

WORD NUMBER
Wn = 5

EXECUTION CYCLES
Cn = {(P+2)A+10}B+70

{P=4 (OPM='000'~'011')
{P=6 (OPM='100'~'111')

{S=0 : A = |DX| + 1
{ B = |DY| + 1
{S=1 : A = |DY| + 1
{ B = |DX| + 1

<DESCRIPTION>

The Relative Graphic Copy Command (RGCPY) copies data from a rectangular area on the frame buffer (the source area) to another location on the frame buffer (the destination area) specified by the initial starting point CP. Each side of the source rectangular is parallel to the coordinate axis. Two diagonal corner points are Pss (A+dXs, B+dYs) at the relative coordinate point from CP and Pse (A+dXs+DX, B+dYs+DY) at the relative coordinate point from Pss.

X and Y components of Pss, dXs and dYs, expressed by the relative X−Y coordinates from CP, are set in the command parameter in unit of pixel.

X and Y components of Pse, DX and DY, expressed by the relative X−Y coordinates from Pss, are set in the command parameter in units of pixels.

Note) The pattern RAM is not in use with RGCPY command.

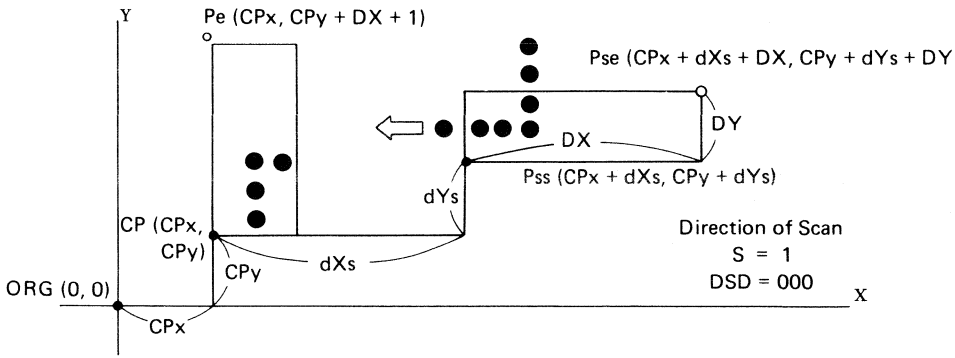


Figure C38-1 Function of RGCPY

<DIRECTION OF POINTER SCAN>

S-bit and DSD bit set the pointer scan directions. Refer to the description about the AGCPY command for details.

<Note for the parameter setting>

- Use 2's complement to setup negative value.
- Valid range

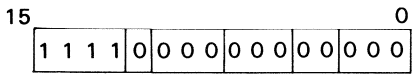
$CP_x - 32767 \leq dXs \leq 32767 + CP_x$, $CP_y - 32767 \leq dYs \leq 32767 + CP_y$
$-32767 \leq DX \leq 32767$, $-32767 \leq DY \leq 32767$

<EXECUTION EXAMPLE>

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).

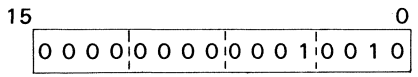
RGCPY (Relative Graphic Copy)

COMMAND CODE

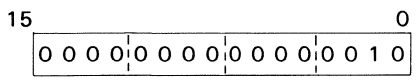


($ F 0 0 0)

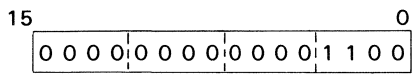
COMMAND PARAMETERS



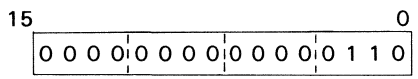
($ 0 0 1 2)



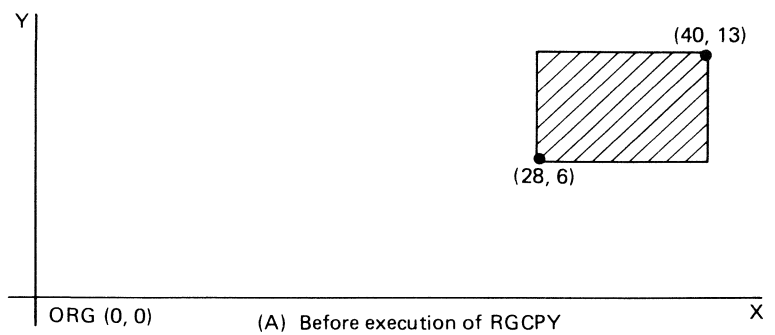
($ 0 0 0 2)



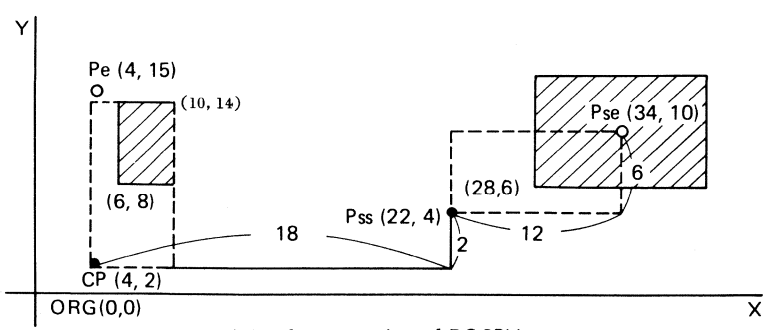
($ 0 0 0 C)



($ 0 0 0 6)



(A) Before execution of RGCPY



(B) After execution of RGCPY

Figure C38-2 Example of RGCPY Execution

ELECTRICAL SPECIFICATION

○ Absolute Maximum Ratings

Item	Symbol	Rating	Unit
Supply voltage	Vcc *	−0.3～+7.0	V
Input voltage	Vin *	−0.3～Vcc+0.3	V
Allowable output current	|Io| * *	5	mA
Total allowable output current	|ΣIo| * * *	120	mA
Operating temperature	Topr	0～+70	°C
Storage temperature	Tstg	−55～+150	°C

* This value is in reference to Vss = 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	Vcc *	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	—	Vcc	
Operating temperature	Topr	0	25	70	°C

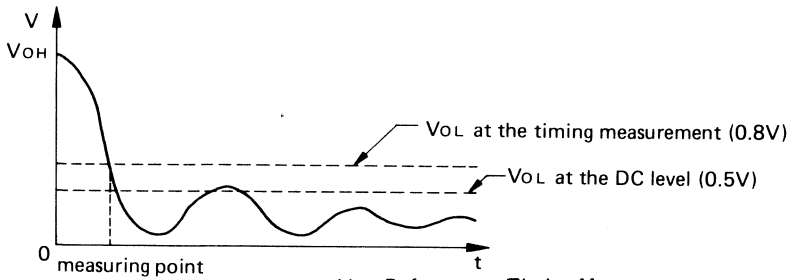
* This value is in reference to Vss = 0V.

○ 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 Vcc−2.0V.



VOL Reference at Timing Measurement

○ Electrical Characteristics

· DC characteristics (Vcc = 5.0V ± 5%, Vss = 0V, Ta = 0 to 70°C unless otherwise noted)

Item		Symbol	Measuring condition	4 MHz Version HD63484-4		6 MHz Version HD63484-6		8 MHz Version HD63484-8		Unit
				min	max	min	max	min	max	
Input "high" level voltage	All Inputs	V_{IH}		2.2	Vcc	2.2	Vcc	2.2	Vcc	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	R/$\overline{W}$,$\overline{CS}$, RS,$\overline{RES}$ $\overline{DACK}$, 2CLK, LPSTB	Iin	Vin=0 ~Vcc	−2.5	2.5	−2.5	2.5	−2.5	2.5	μA
Three state (off state) input current	D0~D15 $\overline{EXSYNC}$ MAD0~ MAD15	I_{TSI}	Vin=0.4 ~Vcc	−10	10	−10	10	−10	10	μA
Output "high" level voltage	D0~D15, MAD0~ MAD15, $\overline{CUD1}$, $\overline{CUD2}$, $\overline{DREQ}$, $\overline{DTACK}$, $\overline{HSYNC}$, $\overline{VSYNC}$, $\overline{EXSYNC}$	V_{OH}	IOH=−400μA	Vcc −1.0	–	Vcc −1.0	–	Vcc −1.0	–	V
Output "low" level voltage	$\overline{DISP1}$, $\overline{DISP2}$, CHR,MRD, $\overline{DRAW}$,$\overline{AS}$, MCYC, RA4, MA16/ RA0~ MA19/ RA3	V_{OL}	IOL=2.2mA	–	0.5	–	0.5	–	0.5	V

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	
Output leak current (off state)	$\overline{IRQ}$, $\overline{DONE}$	I_{LOH}	VOH=Vcc	–	10	–	10	–	10	μA
Input capacity	D0~D15, $\overline{EXSYNC}$, MAD0~ MAD15	Cin	Vin=0V Ta=25°C f=1.0MHz	–	17	–	17	–	17	PF
	R/$\overline{W}$, $\overline{CS}$, $\overline{RS}$, $\overline{RES}$, $\overline{DACK}$, 2CLK, LPSTB		Vin=0V Ta=25°C f=1.0MHz	–	17	–	17	–	17	PF
Output capacity	$\overline{IRQ}$, $\overline{DONE}$	Cout	Vin=0V Ta=25°C f=1.0MHz	–	15	–	15	–	15	PF
Current consumption		Icc	· Chip not selected · Display in progress	–	60	–	80	–	100	mA
			· Data bus in read/write operation · Display in progress · Command execution in progress	–	60	–	80	–	100	mA

· DC characteristics (Vcc = 5.0 ± 5%, Vss = 0V, Ta = 0 to 70°C unless otherwise noted)

Clock

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	tcyc		250	1000	167	1000	125	1000	ns
2	Clock "High" Level Pulse Width	tPWCH	Fig. 1	115	500	75	500	55	500	ns
3	Clock "Low" Level Pulse Width	tPWCL		115	500	75	500	55	500	ns
4	Clock Rise Time	tcr		–	10	–	10	–	10	ns
5	Clock Fall Time	tcf		–	10	–	10	–	10	ns

MPU Read/Write Cycle

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	
6	R/$\overline{W}$ Setup Time	tRWS	Fig. 2.1 Fig. 2.2 Fig. 3.1 Fig. 3.2	70	–	60	–	50	–	ns
7	R/$\overline{W}$ Hold Time	tRWH		0	–	0	–	0	–	ns
8	RS Setup Time	tRSS		70	–	60	–	50	–	ns
9	RS Hold Time	tRSH		0	–	0	–	0	–	ns
10	$\overline{CS}$ Setup Time	tCSS		50	–	40	–	40	–	ns
11	$\overline{CS}$ "High" Level Width	tWCSH		80	–	70	–	60	–	ns
12										
13	Read Wait Time	tRWAI	Fig. 2.1 Fig. 2.2	0	–	0	–	0	–	ns
14	Read Data Access Time	tRDAC		–	120	–	100	–	80	ns
15	Read Data Hold Time	tRDH		10	–	10	–	10	–	ns
16	Read Data Turn Off Time	tRDZ		–	60	–	60	–	60	ns
17	$\overline{DTACK}$ Delay Time (Z to L)	tDTKZL		–	90	–	80	–	70	ns
18	$\overline{DTACK}$ Hold Time (D to L)	tDTKDL	Fig. 2.1 Fig. 2.2	0	–	0	–	0	–	ns
19	$\overline{DTACK}$ Release Time (L to H)	tDTKLH		–	100	–	90	–	80	ns
20	$\overline{DTACK}$ Turn Off Time (H to Z)	tDTKZ		–	100	–	100	–	100	ns
21	Data Bus 3 State Recovery Time 1	tDBRT1		0	–	0	–	0	–	ns
22	Write Wait Time	tWWAI	Fig. 3.1 Fig. 3.2	0	–	0	–	0	–	ns
23	WRITE Data Setup Time	tWDS		80	–	60	–	40	–	ns
24	WRITE Data Hold Time	tWDH		10	–	10	–	10	–	ns

DMA Read/Write Cycle

No.	Item	Symbol	Measuring condition	4 MHz Version HD63484-4		6 MHz Version HD63484-6		8 MHz Version HD63484-8		Unit
				min	max	min	max	min	max	
25	$\overline{DREQ}$ Delay Time1	tDRQD1	Fig. 4 Fig. 5	–	150	–	130	–	110	ns
26	$\overline{DREQ}$ Delay Time2	tDRQD2		–	90	–	80	–	70	ns
27	DMA R/$\overline{W}$ Setup Time	tDRWS		70	–	60	–	50	–	ns
28	DMA R/$\overline{W}$ Hold Time	tDRWH		0	–	0	–	0	–	ns
29	$\overline{DACK}$ Setup Time	tDAKS		50	–	40	–	40	–	ns
30	$\overline{DACK}$ "High" Level Width	tDAKH		80	–	70	–	60	–	ns
31										
32	DMA Read Wait Time	tDRW	Fig.4	0	–	0	–	0	–	ns
33	DMA Read Data Access Time	tDRDAC		–	120	–	100	–	80	ns
34	DMA Read Data Hold Time	tDRDH		10	–	10	–	10	–	ns
35	DMA Read Data Turn Off Time	tDRDZ		–	60	–	60	–	60	ns
36	DMA $\overline{DTACK}$ Delay Time (Z to L)	tDDTZL	Fig. 4 Fig. 5	–	90	–	80	–	70	ns
37	DMA $\overline{DTACK}$ Delay Time (D to L)	tDDTDL	Fig. 4	0	–	0	–	0	–	ns
38	DMA $\overline{DTACK}$ Release Time (L to H)	tDDTLH	Fig. 4 Fig. 5	–	100	–	90	–	80	ns
39	DMA $\overline{DTACK}$ Turn Off Time (L to Z)	tDDTHZ		–	100	–	100	–	100	ns
40	$\overline{DONE}$ Output Delay Time	tDND		–	90	–	80	–	70	ns
41	$\overline{DONE}$ Output Turn Off Time (L to Z)	tDNLZ		–	100	–	90	–	80	ns
42	Data Bus 3 State Recovery Time 2	tDBRT2	Fig. 4	0	–	0	–	0	–	ns
43	$\overline{DONE}$ Input Pulse Width	tDNPW	Fig. 4 Fig. 5	2	–	2	–	2	–	Clk. Cyc
44	DMA Write Wait Time	tDWW	Fig. 5	0	–	0	–	0	–	ns
45	DMA Write Data Setup Time	tDWDS		80	–	60	–	40	–	ns
46	DMA Write Data Hold Time	tDWDH		10	–	10	–	10	–	ns
47										
48	$\overline{AS}$ "Low" Level Pulse Width	tPWASL	Fig. 6~ Fig. 9	80	–	40	–	25	–	ns

Frame Memory Read/Write Cycle

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	
49	Memory Address Hold Time 2	tMAH2	Fig. 7, 8	10	–	10	–	10	–	ns
50	$\overline{AS}$ Delay Time 1	tASD1	Fig. 6~ Fig. 9	–	90	–	75	–	65	ns
51	$\overline{AS}$ Delay Time 2	tASD2		–	90	–	75	–	65	ns
52	Memory Address Delay Time	tMAD		–	95	–	80	–	70	ns
53	Memory Address Hold Time 1	tMAH1		10	–	10	–	10	–	ns
54	Memory Address Turn Off Time (A-Z)	tMAAZ	Fig. 6, 7 Fig. 9	–	50	–	50	–	50	ns
55	Memory Read Data Setup Time	tMRDS	Fig. 7	60	–	50	–	40	–	ns
56	Memory Read Data Hold Time	tMRDH		10	–	10	–	10	–	ns
57	MA/RA Delay Time	tMARDE	Fig. 6~ Fig. 9	–	100	–	90	–	80	ns
58	MA/RA Hold Time	tMARAH	Fig. 6~ Fig. 8	10	–	10	–	10	–	ns
59	MCYC Delay Time	tMCYCD	Fig. 6~ Fig.10	–	60	–	50	–	50	ns
60	MRD Delay Time	tMRDD	Fig. 6~ Fig. 9	–	90	–	80	–	70	ns
61	MRD Hold Time	tMRDH		10	–	10	–	10	–	ns
62	$\overline{DRAW}$ Delay Time	tDRWD		–	90	–	80	–	70	ns
63	$\overline{DRAW}$ Hold Time	tDRWH		10	–	10	–	10	–	ns
64	Memory Write Data Delay Time	tMWDD	Fig. 8	–	90	–	80	–	70	ns
65	Memory Write Data Hold Time	tMWDH		10	–	10	–	10	–	ns
66										

Display Control Signal Output Timing

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	
67	$\overline{HSYNC}$ Delay Time	tHSD	Fig. 9~ Fig.11	–	90	–	80	–	70	ns
68	$\overline{VSYNC}$ Delay Time	tVSD	Fig.10	–	90	–	80	–	70	ns
69	$\overline{DISP1}$, $\overline{DISP2}$ Delay Time	tDSPD		–	90	–	80	–	70	ns
70	$\overline{CUD1}$, $\overline{CUD2}$ Delay Time	tCUDD		–	90	–	80	–	70	ns
71	$\overline{EXSYNC}$ Output Delay Time	tEXD		20	90	20	80	20	70	ns
72	CHR Delay Time	tCHD		–	90	–	80	–	70	ns
73										
74										

$\overline{\text{EXSYNC}}$ Input Timing

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	
75	$\overline{\text{EXSYNC}}$ Input Pulse Width	tEXSW	Fig.11	3	–	3	–	3	–	Clk. Cyc
76	$\overline{\text{EXSYNC}}$ Input Setup Time	tEXS		60	–	60	–	50	–	ns
77	$\overline{\text{EXSYNC}}$ Input Hold Time	tEXH		30	–	30	–	30	–	ns

Input LPSTB Timing

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	
78	LPSTB Uncertain Time 1	tLPD1	Fig.12	70	–	70	–	70	–	ns
79	LPSTB Uncertain Time 2	tLPD2		10	–	10	–	10	–	ns
80	LPSTB Input Hold Time	tLPH		10	–	10	–	10	–	ns
81	LPSTB Input Inhibit time	tLPI		4	–	4	–	4	–	Clk. Cyc

$\overline{\text{RES}}$ Input and $\overline{\text{DACK}}$ Input Timing

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	$\overline{\text{DACK}}$ Setup Time for $\overline{\text{RES}}$	tDAKSR	Fig.13	100	–	100	–	100	–	ns
83	$\overline{\text{DACK}}$ Hold Time for $\overline{\text{RES}}$	tDAKHR		0	–	0	–	0	–	ns
84	$\overline{\text{RES}}$ Input Pulse Width	tRES		10	–	10	–	10	–	Clk. Cyc

$\overline{\text{IRQ}}$ Output, Attribute Control Information Output Cycle Timing

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	
85	$\overline{\text{IRQ}}$ Delay Time 1	tIRQ1	Fig.13	–	250	–	200	–	150	ns
86	$\overline{\text{IRQ}}$ Delay Time 2	tIRQ2		–	500	–	500	–	500	ns
87	ATR Delay Time 1	tATRD1	Fig. 9	–	100	–	90	–	80	ns
88	ATR Hold Time 1	tATRH1		10	–	10	–	10	–	ns
89										
90	ATR Delay Time 2	tATRD2	Fig. 9	–	100	–	90	–	80	ns
91	ATR Hold Time 2	tATRH2		10	–	10	–	10	–	ns

MPU Read/Write Cycle Timing (synchronous bus), DMA Read/Write Cycle Timing (synchronous bus)

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	
100	$\overline{CS}$ Cycle Time	tCSC	Fig. 2 Fig. 3	4	–	4	–	4	–	CLKCYC
101	$\overline{CS}$ "Low" Level Width	tWCSL		2	–	2	–	2	–	CLKCYC
102	$\overline{CS}$ "High" Level Width	tWCSH		2	–	2	–	2	–	CLKCYC
104	$\overline{DACK}$ Cycle Time	tDACKC	Fig. 4A Fig. 5A	4	–	4	–	4	–	CLKCYC
105	$\overline{DACK}$ "Low" Level Width	tWDACKL		2	–	2	–	2	–	CLKCYC
106	$\overline{DACK}$ "Low" Level Width	tWDACKH		2	–	2	–	2	–	CLKCYC

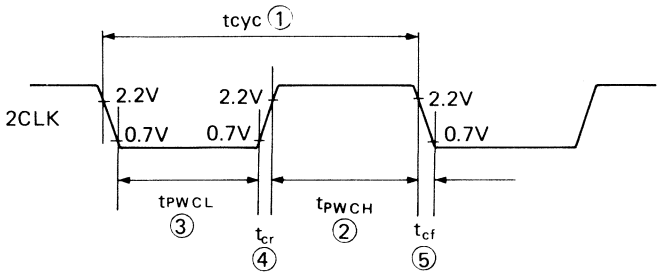
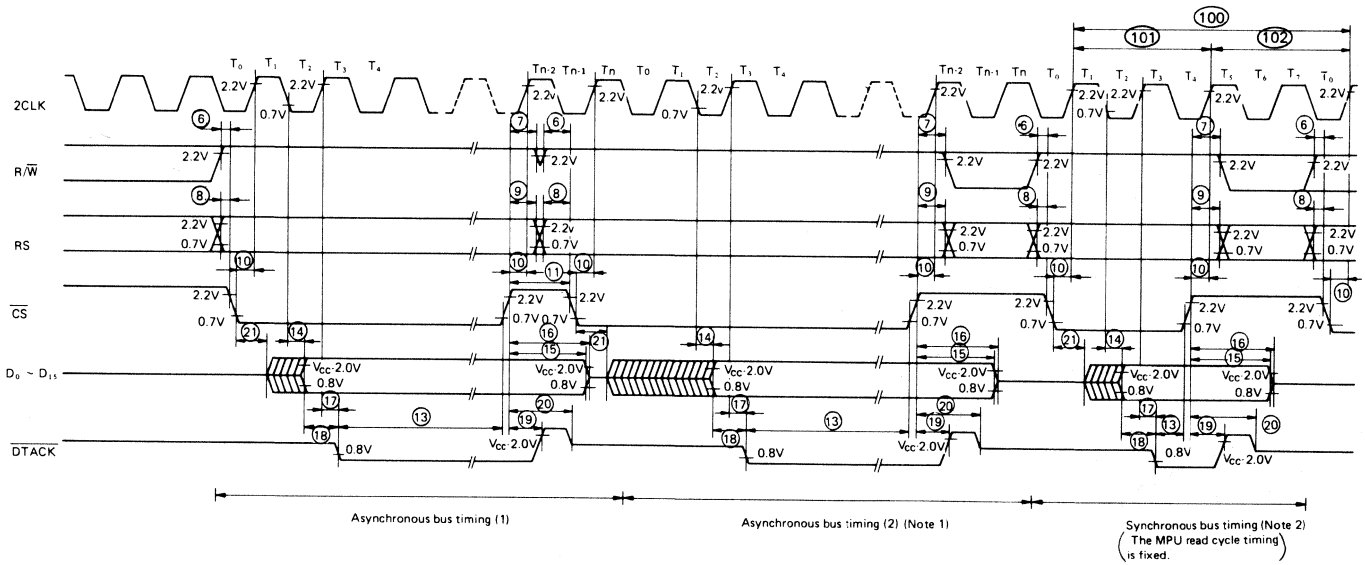


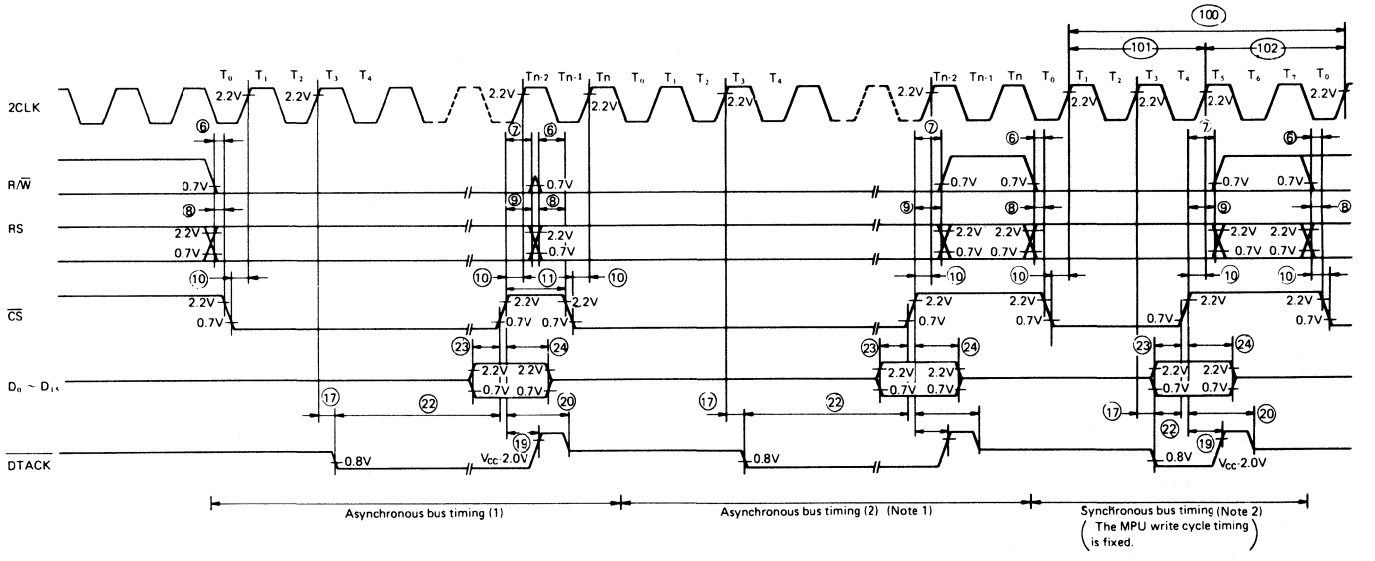
Figure 1 2CLK Waveform



(Note 1) $\overline{CS}$ "high" width must satisfy the specification (11).
Unless satisfying the spec (102), $\overline{DTACK}$ 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.

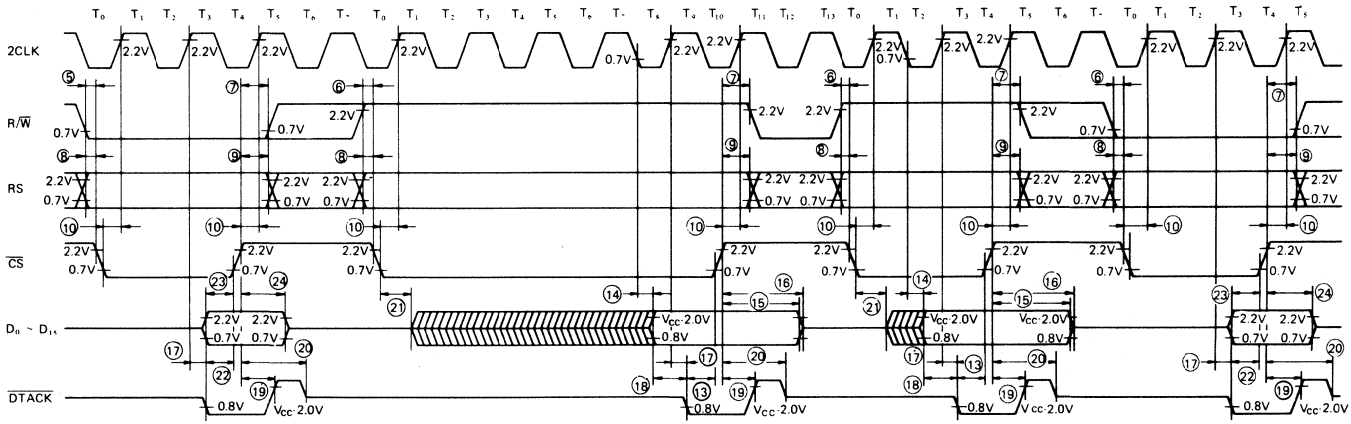
Figure 2 MPU Read Cycle Timing (MPU ← ACRTC)



(Note 1) $\overline{CS}$ "high" width must satisfy the specification (11).
Unless satisfying the spec. (102), $\overline{DTACK}$ 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, 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)

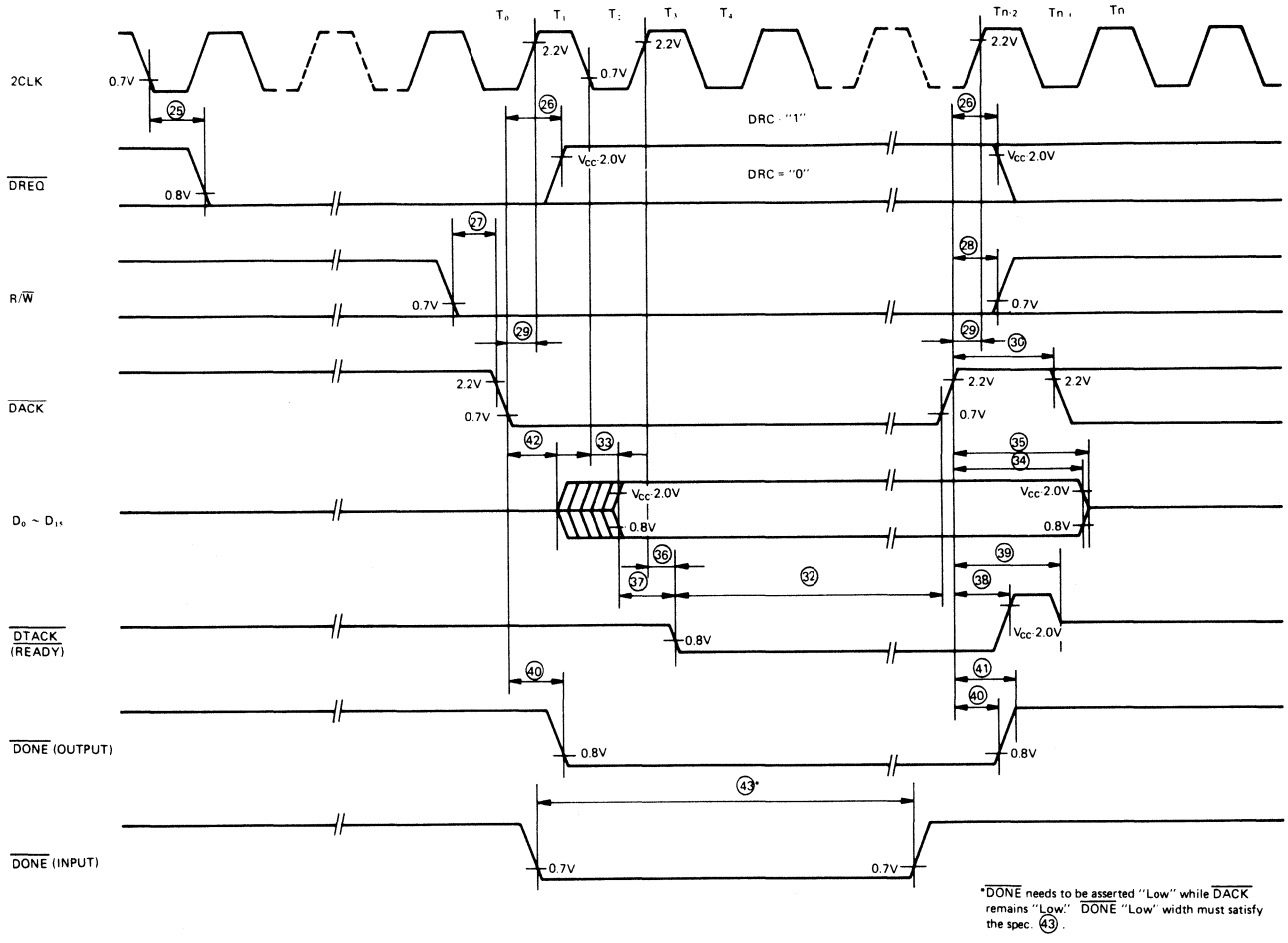
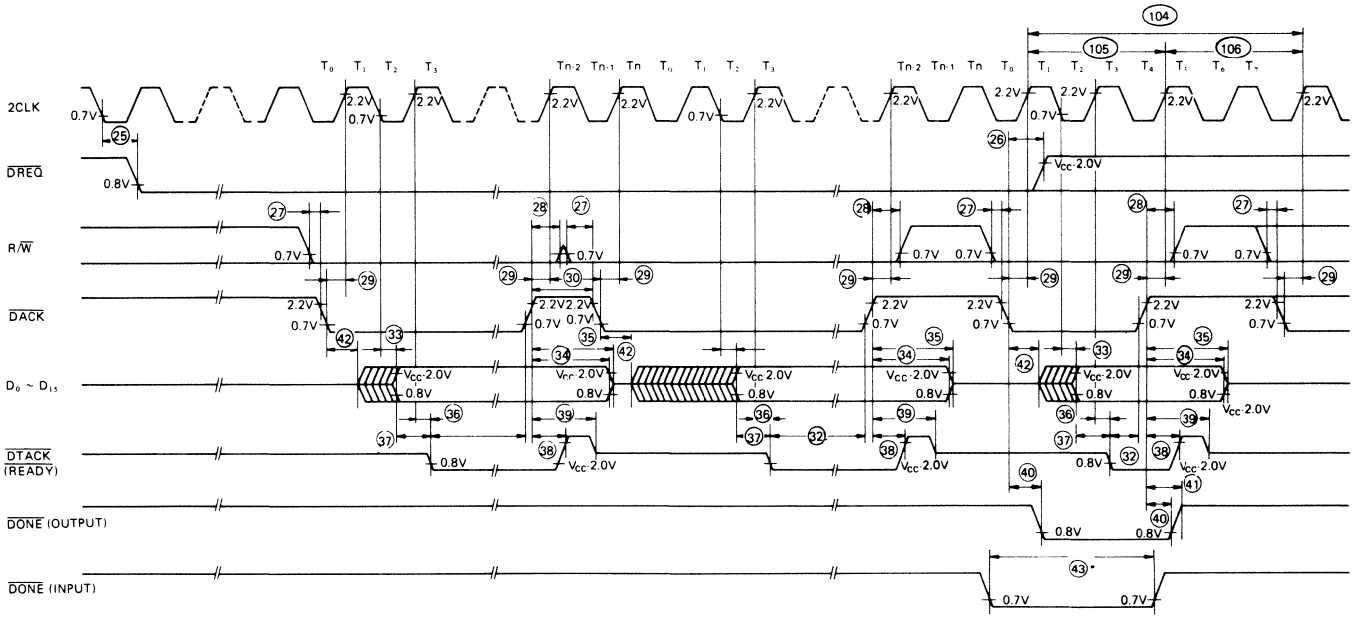


Fig. 4 DMA Read Cycle Timing (Memory ← ACRTC)



*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

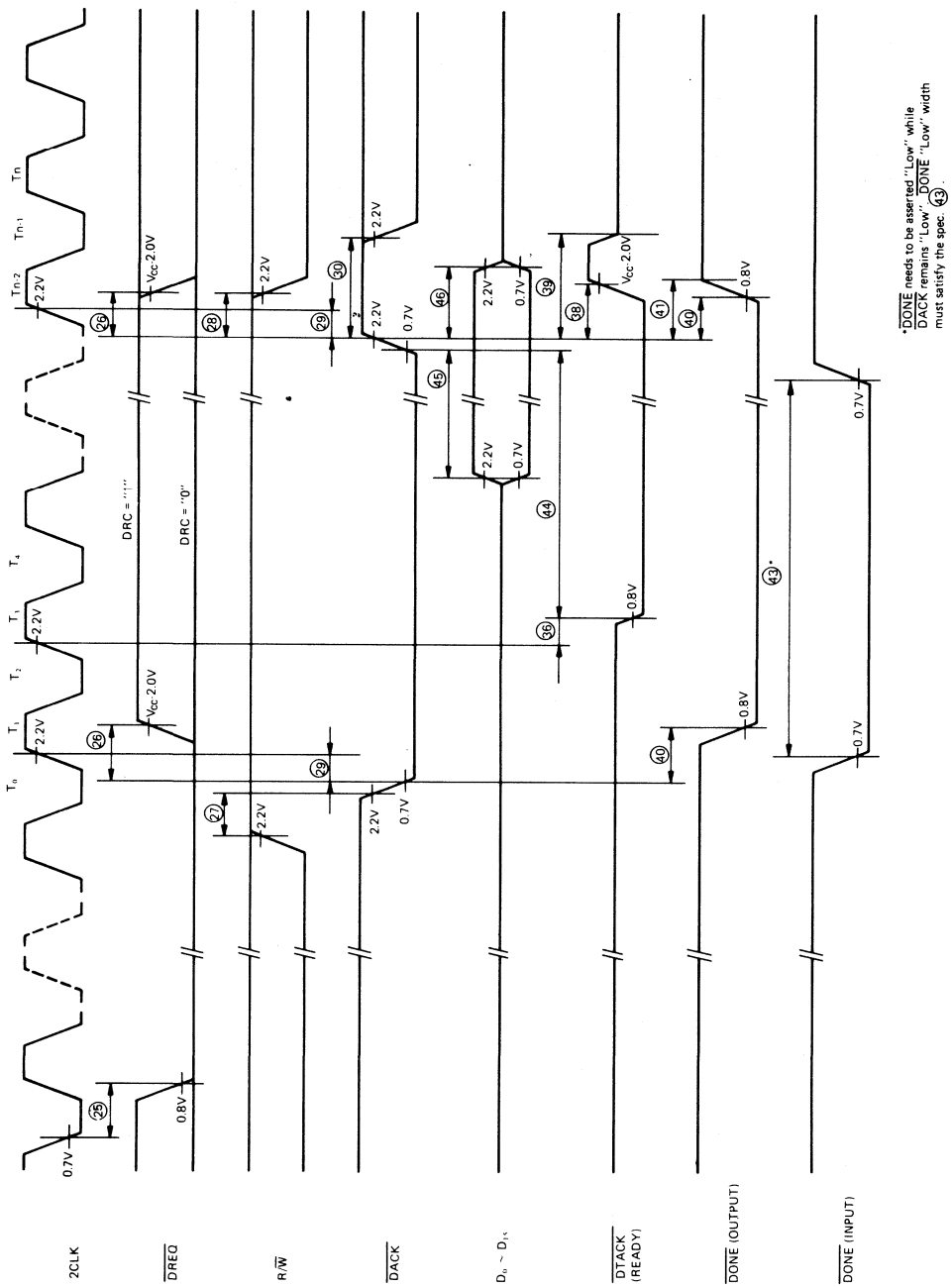
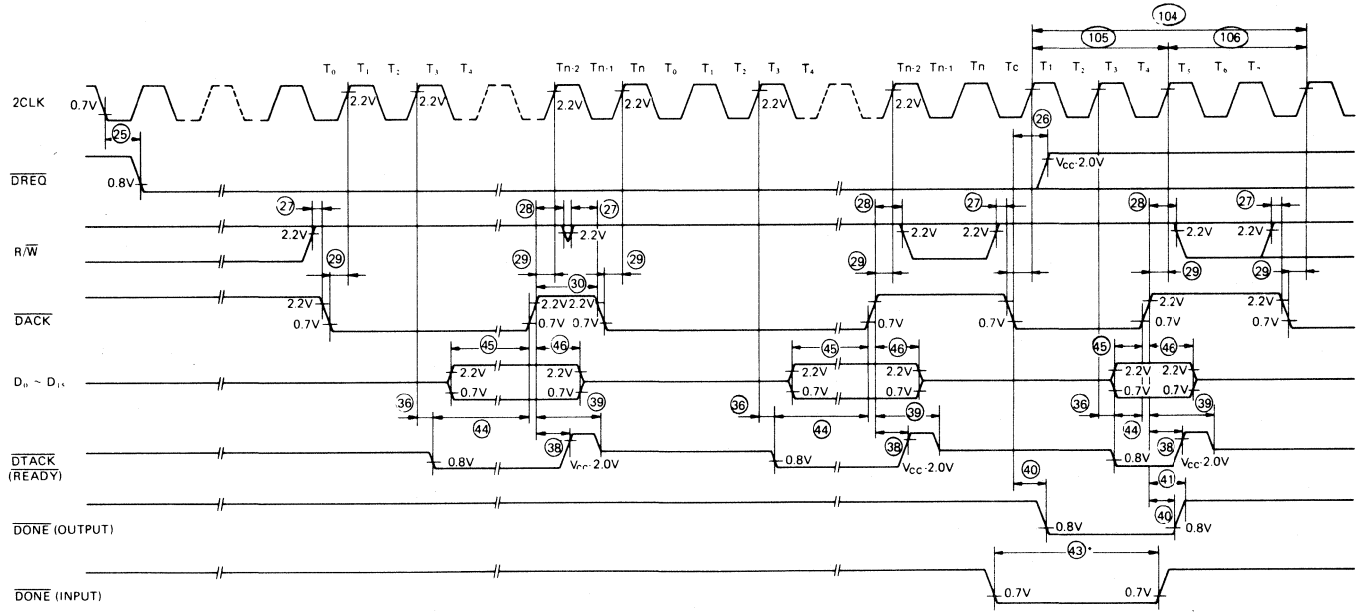


Figure 5 DMA Write Cycle Timing (Memory → ACRTC)



*$\overline{\text{DONE}}$ needs to be asserted "Low" while $\overline{\text{DACK}}$ remains "low".
$\overline{\text{DONE}}$ "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

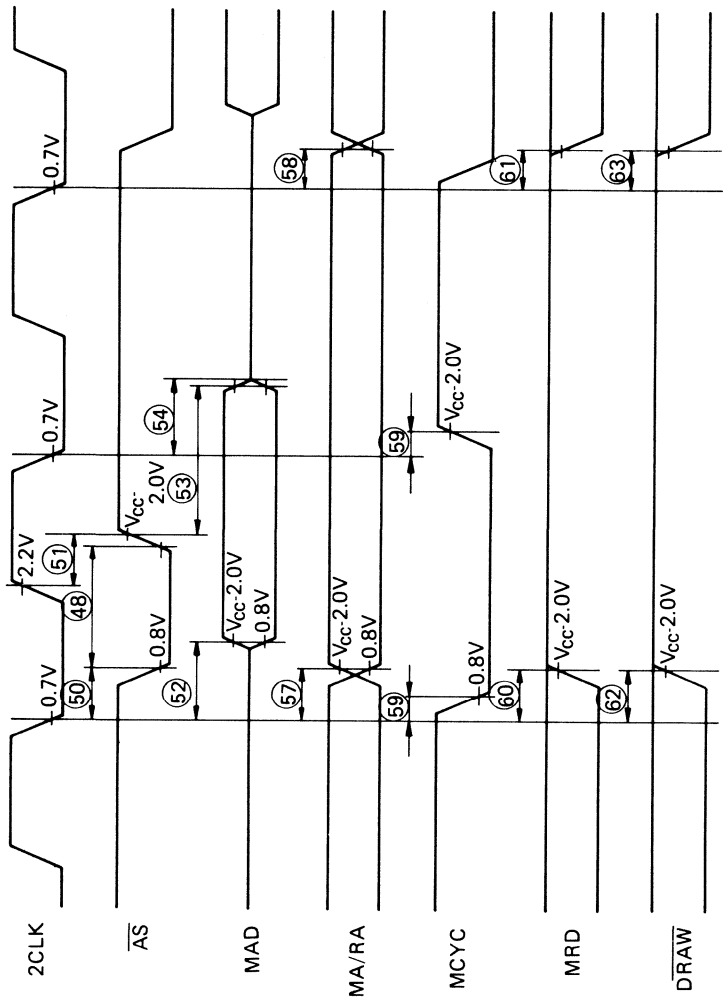


Figure 6 Screen Display Cycle Timing

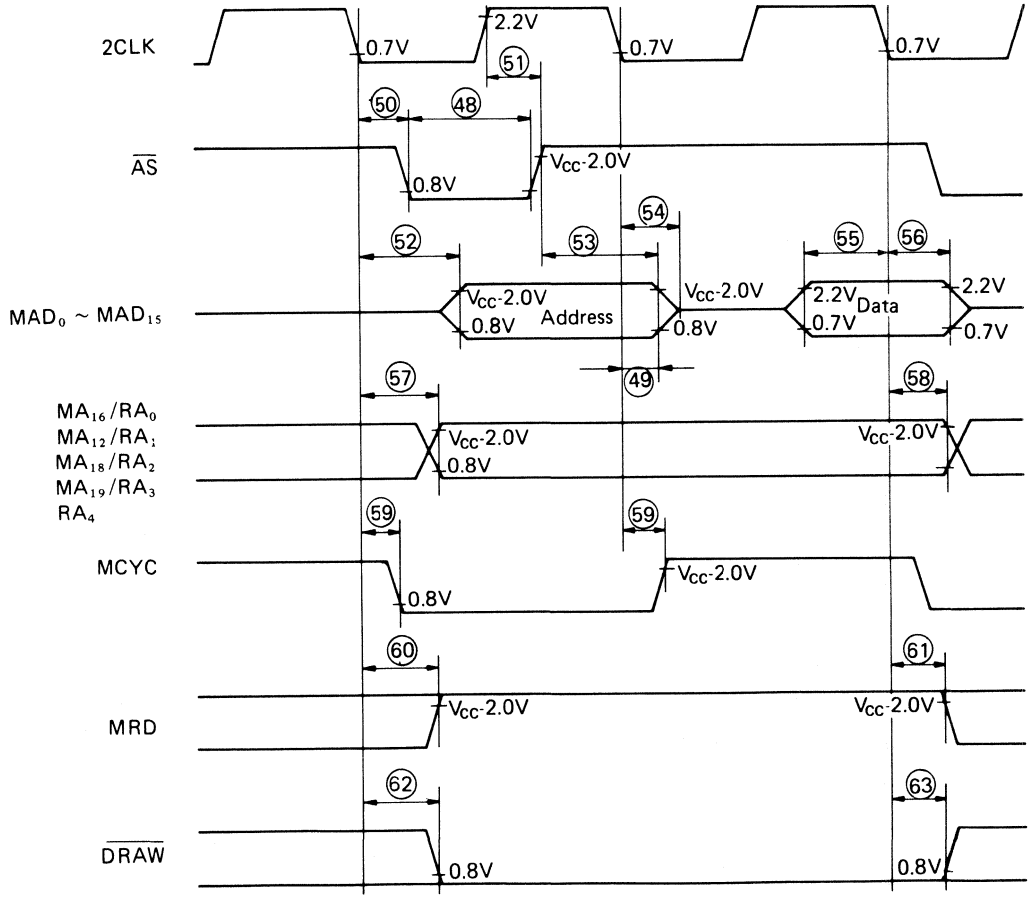


Figure 7 Frame Memory Read Cycle Timing (ACRTC ← Frame Memory)

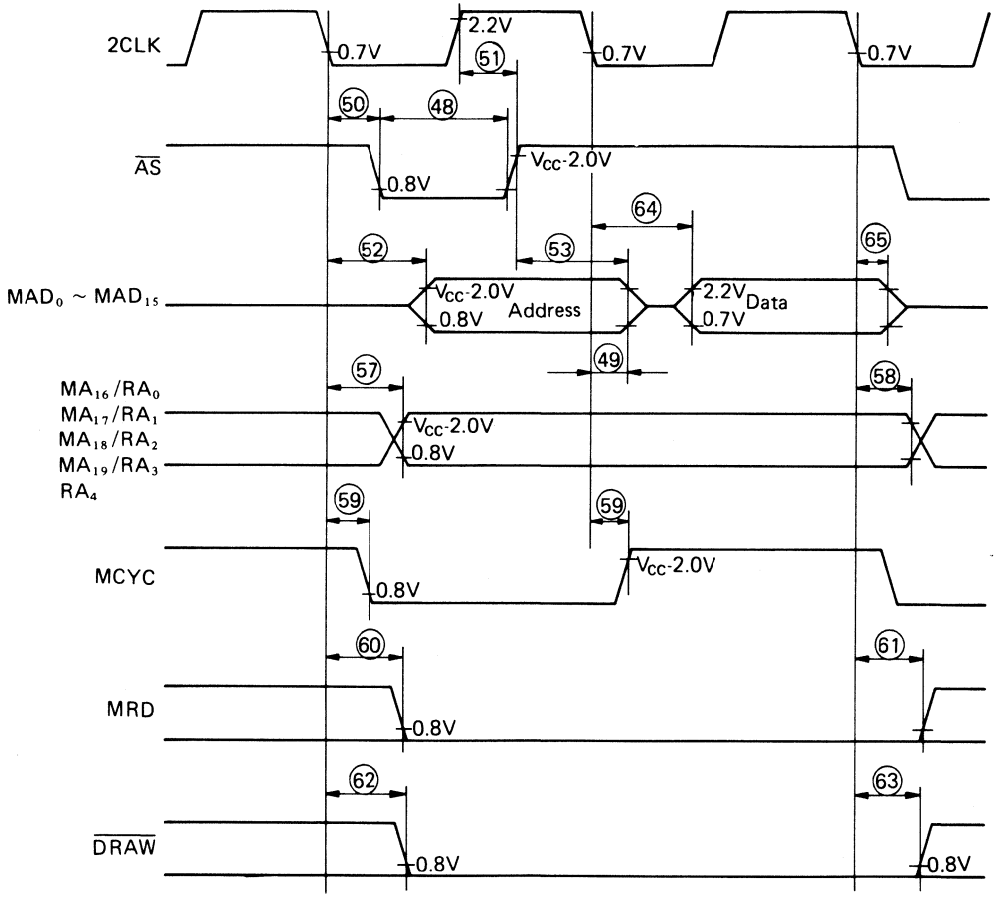


Figure 8 Frame Memory Write Cycle Timing (ACRTC → Frame Memory)

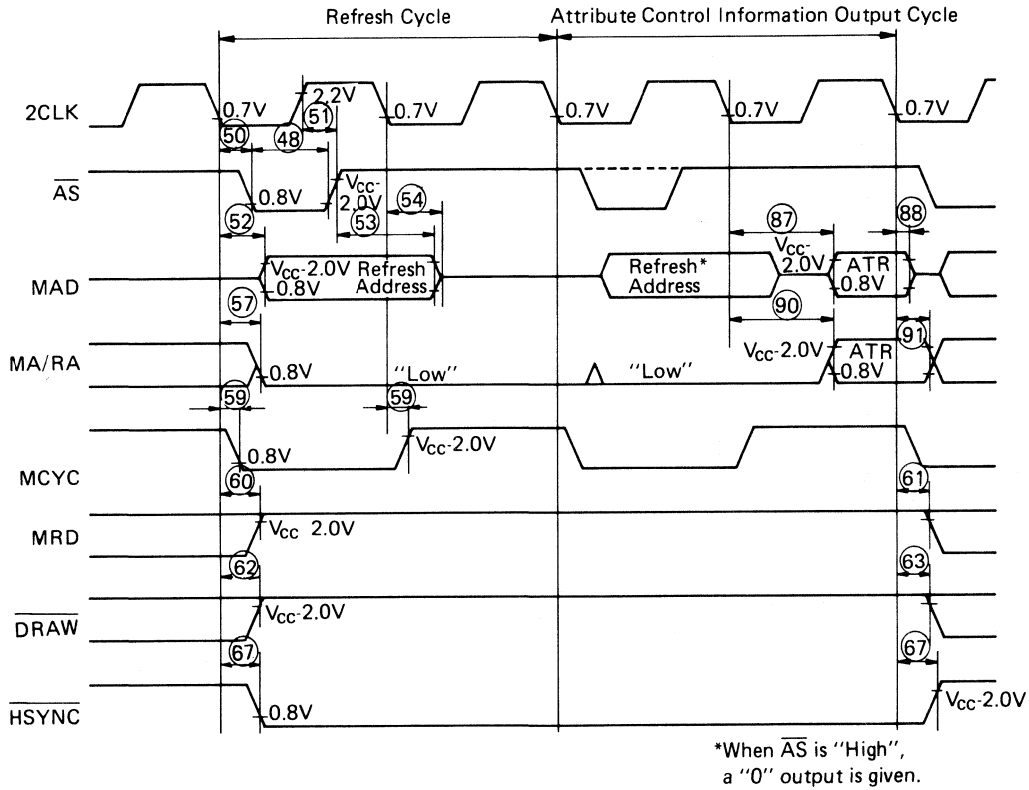


Figure 9 Frame Memory Refresh/Attribute Control Information Output Cycle Timing

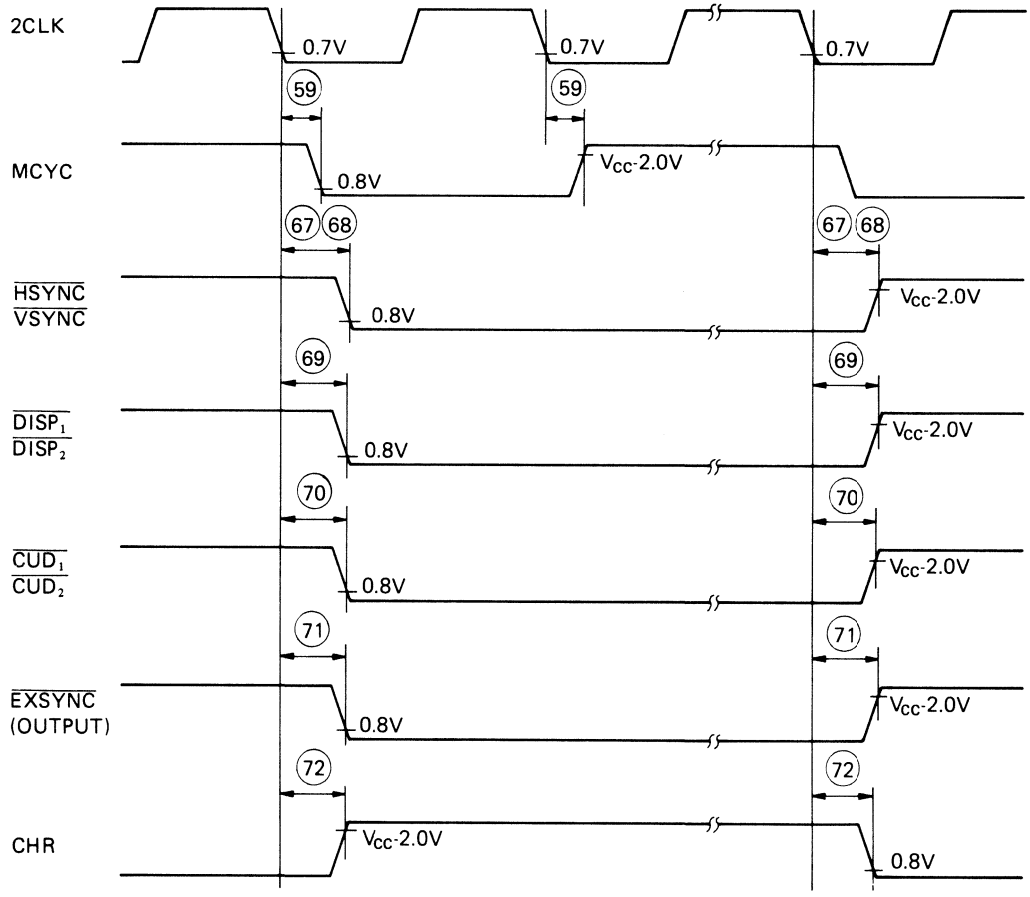
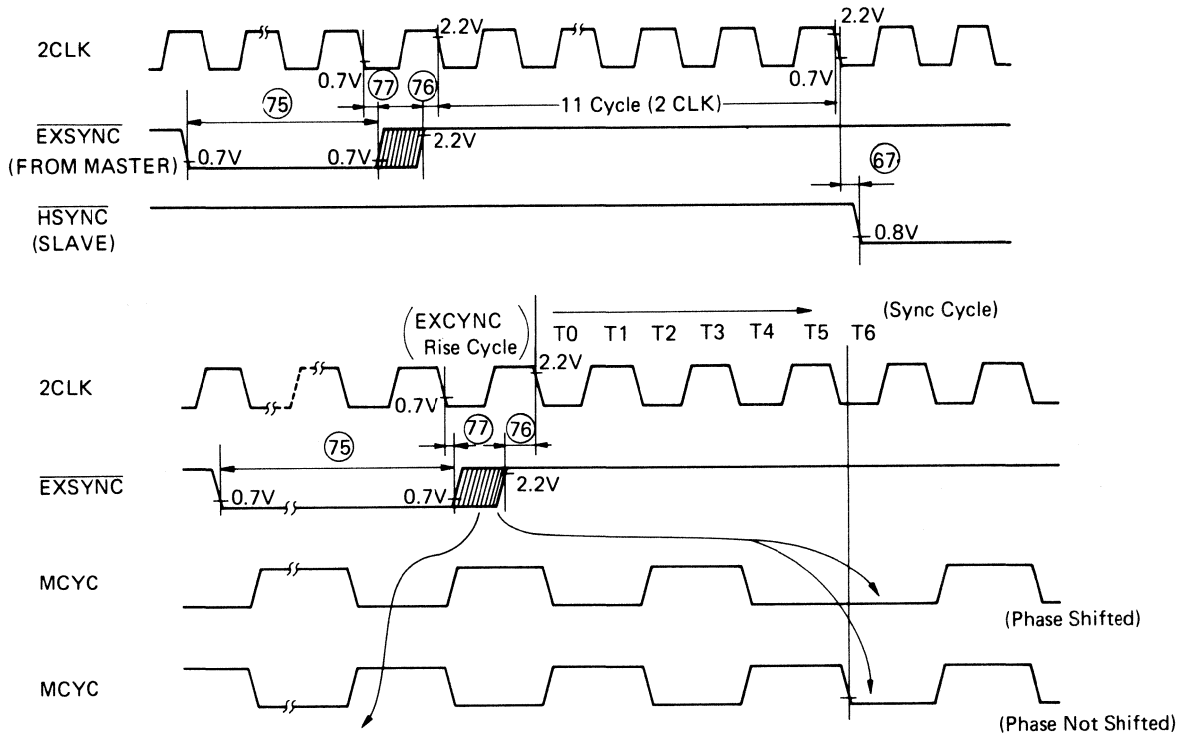
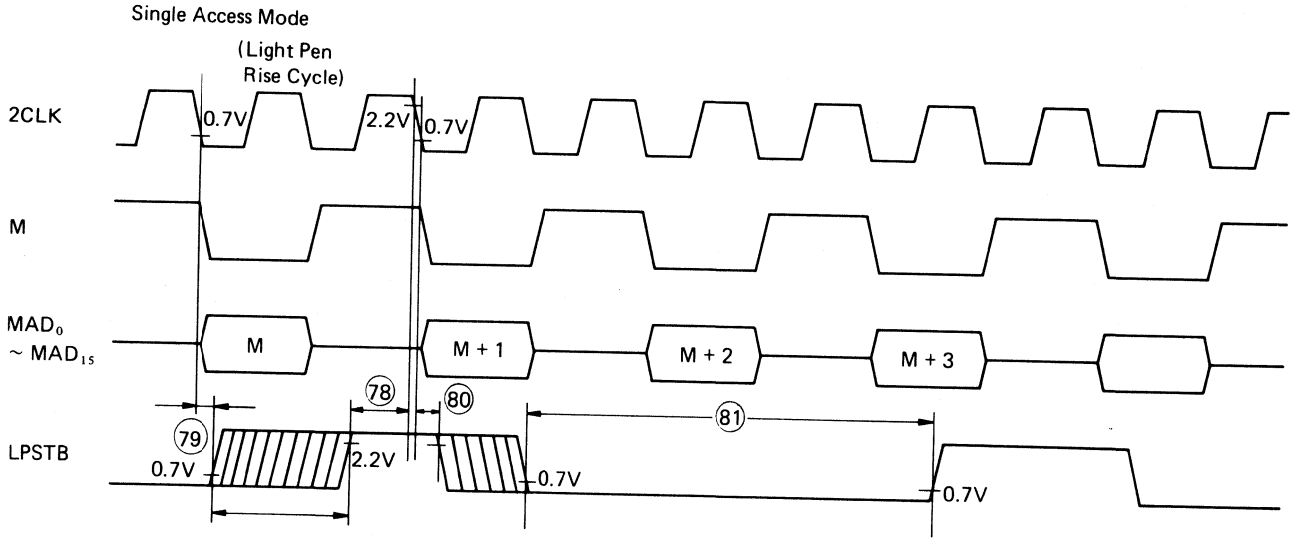


Figure 10 Display Control Signal Output Timing



(When the leading edge of $\overline{\text{EXSYNC}}$ enters this period, $\overline{\text{ACRTC}}$ shifts the internal phase according to the above sequence.)

Figure 11 $\overline{\text{EXSYNC}}$ Input Timing

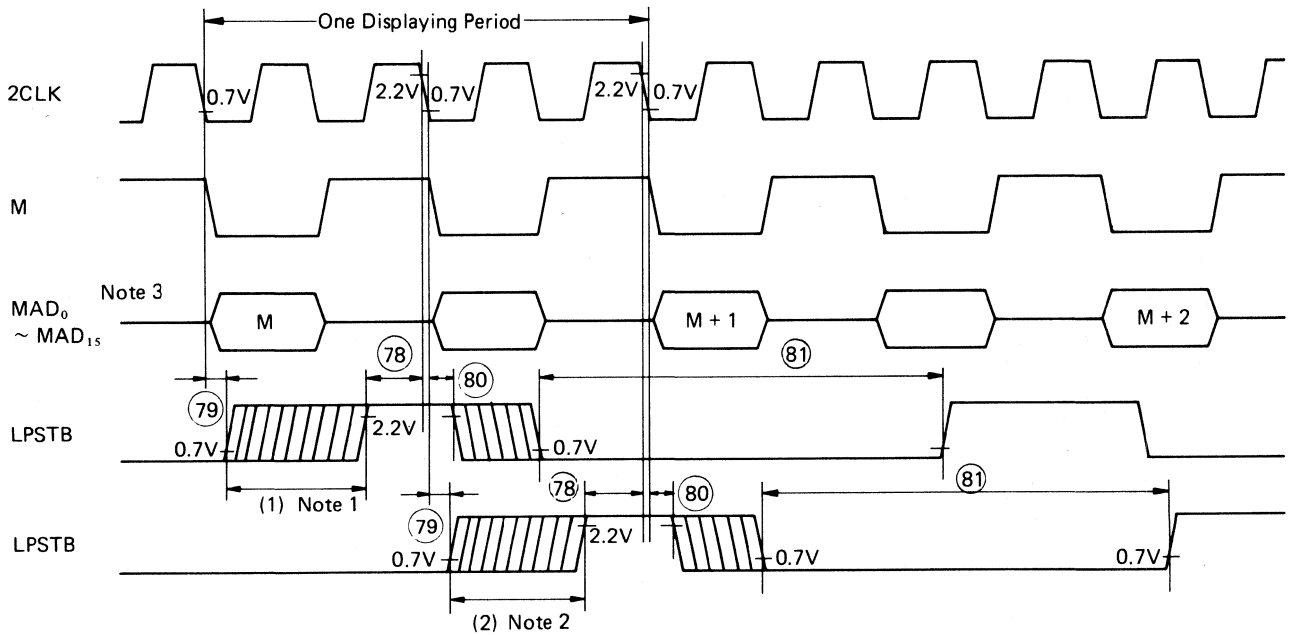


When LPSTB rises in this period, memory address "M + 2" is set in the Light Pen Address register.

Figure 12A LPSTB Input Timing

Interleave Mode/Superimpose Mode

(Dual Access Mode 0/1)



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 12B LPSTB Input Timing

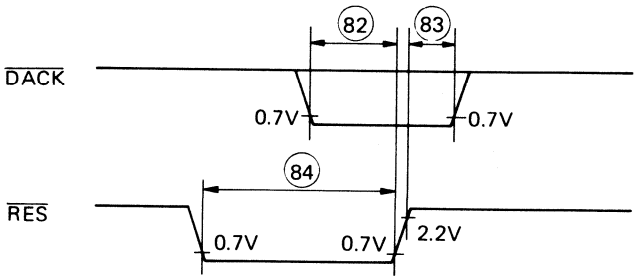


Figure 13 $\overline{\text{RES}}$ Input and $\overline{\text{DACK}}$ Input Timing (System Reset and 16-bit/8-bit Selection)

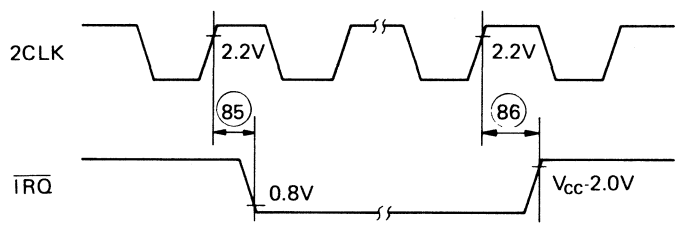
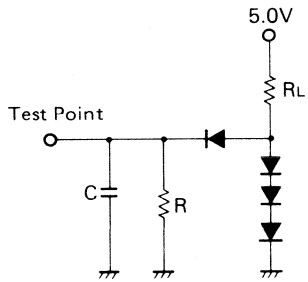


Figure 14 IRQ Output Timing



Signal	Load condition
$D_0 \sim D_{15}$ $\overline{DTACK}$ $\overline{DREQ}$ $MAD_0 \sim MAD_{15}$ $MA_{16}/RA_0 \sim MA_{19}/RA_3$ RA_4 $\overline{VSYNC}$, $\overline{HSYNC}$ $\overline{EXSYNC}$ MCYC , $\overline{AS}$, MRD DRAW, CHR $\overline{DISP_1}$, $\overline{DISP_2}$ $\overline{CUD_1}$, $\overline{CUD_2}$	RL = 1.8K Ω C = 40pF R = 10K Ω All diodes are 1S2074H's or the equivalent

Figure 15 Test Load Circuit A

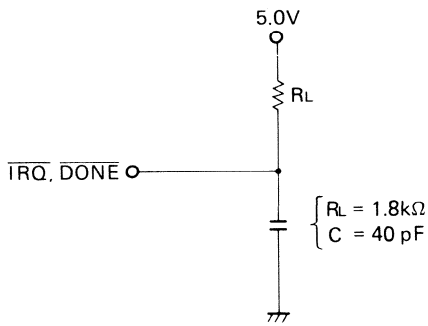


Figure 16 Test Load Circuit B

(Note)
When turning power on, 2CLK and $\overline{RES}$ timings must be as shown in Fig. 17. The delay time from V_{CC} rising to 2CLK rising (t_{2CH}) must be under 100 ms, and that from V_{CC} rising to $\overline{RES}$ rising (t_{REH}) must be above 100 ms.

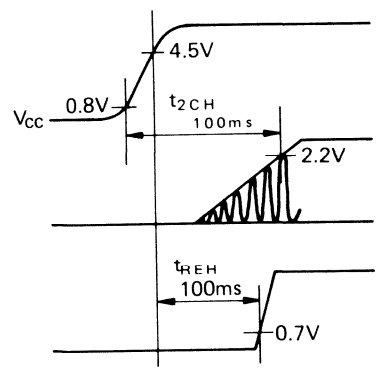


Figure 17 Power On Sequence

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.

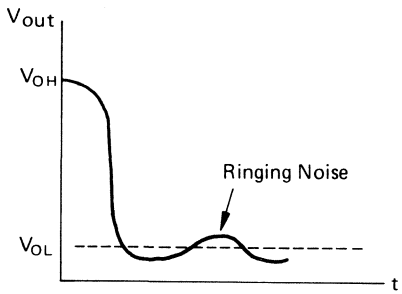


Figure 18 Ringing Noise

Note: The ringing level depends on the load capacity, and it can be V_{OH} + 0.1V.

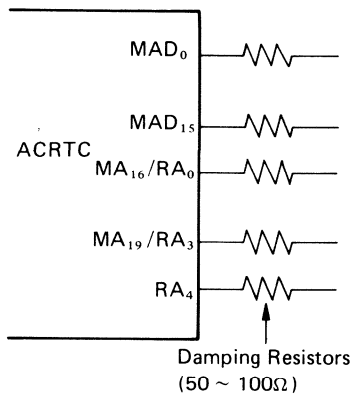


Figure 19 Damping Resistor

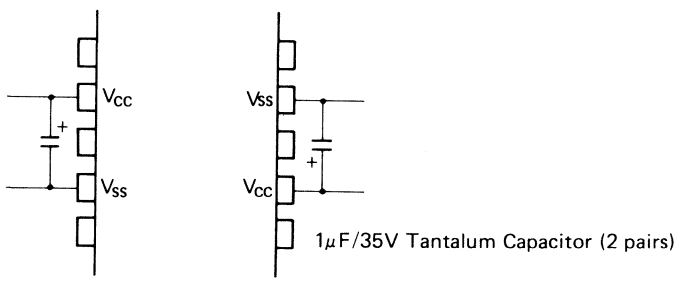


Figure 20 Note for Designing Power Supply Circuit

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.

NOTICE IN SETTING THE ACRTC REGISTERS

1. Notice in Setting the Control Registers for Horizontal Display.
2. Notice in Setting the Control Registers for Vertical Display.
3. Notice in Setting the Graphic Cursor Registers.
4. Notice in Stting the Parameters of the Commands.
5. Notice in Setting the Parameters Registers.
6. ACRTC Internal Reference Timing of the Control Registers.
7. Initial Value After Reset.
   - ○ the Control Registers.
   - ○ the I/O Pins.

1. Notice in setting the Control Registers for horizontal display

Horizontal Front Porch

Raster Scan Mode	Horizontal Front Porch
Non-Interlace, Interlace Sync.	5 or more
Interlace Sync & Video	6 or more

(Unit: Memory Cycle)

Horizontal Back Porch + Horizontal Synchronization Pulse Width (HSW + HDS*)

Raster Scan Mode	Zooming-up Display	Screen	Horizontal Synchronization Pulse Width + Horizontal Back Porch
Non-Interlace, Interlace Sync.	Not Performed	Graphic	6 or more
		Character	7 or more
	Performed	Graphic	7 or more
		Character	8 or more
Interlace Sync. & Video	Not Performed	Graphic	8 or more
		Character	9 or more
	Perfomede	Graphic	9 or more
		Character	10 or more

(Unit: Memory Cycle)

Note) In dual access mode (Interleave, Superimpose mode), the above value needs to be incremented by + 1.

Register Name	Specified Value		Unit
H C*	1~256	Note 2), 8)	Memory Cycle
HSW	2~31	Note 2), 3)	Memory Cycle
HDS*	2~256	Note 2), 4), 5)	Memory Cycle
HDW*	2~256	Note 2), 4), 6)	Memory Cycle
HWS*	HDS*~HDS* + HDW*	Note 5)	Memory Cycle
HWW*	2~HDW*	Note 6), 7)	Memory Cycle

Note 1) In *-marked registers, Setup value = Specified value − 1

Note 2) $HC^* \geq HSW + HDS^* + HDW^* + 5$
(At the non-interlace/interlace mode setup)
$HC^* \geq HSW + HDS^* + HDW^* + 6$
(At the interface sync. & video mode setup)

Note 3) 1 cannot be set to HSW. When the raster counter (RCR) is used, the setup value is to be 3 or more.

Note 4) 1 cannot be set to HDS*, HDW*.

Note 5) When setting the dual access mode, the following rule must be kept.

HDS* is even.	→	HWS* is to be even.
HDS* is odd.	→	HWS* is to be odd.

Note 6) When setting the dual access mode, an even number must be specified.

Note 7) $HWW^* \leqq HDS^* + HDW^* - HWS^*$

Note 8) $HC^* \geq HSW \times 2 + 1$

2. Notice in setting the Control Registers for vertical display

Register Name	Specified Value			Unit
VC	1~4095		Note 2)	Raster
VSW	1~31		Note 3)	Raster
VDS Note 1)	Non-Interlace/ Interlace Sync. Mode	1~255	Note 4)	Raster
	Interlace Sync. & Video Mode	4~255	Note 4)	
SPO Note 1)	Non-Interlace/ Interlace Sync. Mode	2~4095	Note 5), 6)	Raster
	Interlace Synch. & Video Mode	4~4095	Note 5), 6)	
SP1 Note 1)	Non-Interlace/ Interlace Sync. Mode	2~4095	Note 5)	Raster
	Interlace Synch. & Video Mode	4~4095	Note 5)	
SP2 Note 1)	Non-Interlace/ Interlace Sync. Mode	2~4095	Note 5), 6)	Raster
	Interlace Synch. & Video Mode	4~4095	Note 5), 6)	
VWS Note 1)	VDS ~ VDS + SPO + SP1 + SP2		Note 4)	Raster
VWW Note 1)	Non-Interlace/ Interlace Sync. Mode	2~SPO + SP1 + SP2	Note 5), 7)	Raster
	Interlace Sync. & Video Mode	4~SPO + SP1 + SP2	Note 5), 7)	

Note 1) 0 cannot be specified.

Note 2) 0 cannot be specified. The setup value varies as shown below according to the raster scan mode.

- Total raster number scanned in 1 frame [Non-Interlace Mode]
- Total raster number scanned in 1 field [Interlace Sync. Mode]
  (The dummy rasters between the odd/even number fields are not included.)
- Total raster number scanned in 1 frame [Interlace Sync. & Video Mode]
  (The dummy rasters between the odd/even number field are included, and become odd numbers.)

  $VC \leqq VDS + SPO + SP1 + SP2 + VSW$ (NON-interlace/Interlace Sync mode)

  $VC \leqq VDS + SPO + SP1 + SP2 + 2VSW + 1$ (Interlace Sync & Video mode)
  (odd-number)

Note 3) 0 cannot be specified.

Note 4) The specified value must be more than 1 under non-interlace mode or interlace sync. mode, and more than 4 under interlace sync. & video mode.
The specified value must be the raster counts per 1 frame under non-interlace mode, the raster counts per 1 field under interlace sync. mode, and the total raster counts of even field and odd field under interlace sync. & video mode.

Note 5) The specified value must be more than 2 under non-interlace or interlace sync. mode, and more than 4 under interlace sync. & video mode. The specification under each mode depends on the same restriction as described in Note 4).

Note 6) The specified value is regarded as 0, if the split screen is disabled.

Note 7) $VWW \leqq VDS + SPO + SP1 + SP2 - VWS$

3. Notice in setting the Graphic Cursor Registers

Graphic Cursor Signal (r98~r9D)

Register Name	Specified Value		Unit
CXS	HSW + 1~255	Note 2)	Memory Cycle
CXE*	CXS + 1~256	Note 3)	Memory Cycle
CYS	0~VC − VSW − 1	Note 4)	Raster
CYE*	CYS + 1~VC − VSW	Note 5)	Raster

Note 1) Setup value = Specified value − 1, as for the *-marked registers.

Note 2) The specified value must be more than 3. The cursor (X-direction component) is not output, if it is specified as below 3.

Note 3) The specified value must be more than 3. The cursor (X-direction component) is not output, if it is specified as "CXE* < CXS".

Note 4) The cursor (Y-direction component) is not output, if it is specified during the vertical synchronization period.

Note 5) The cursor (Y-direction component) is not output, if it is specified during the vertical synchronization period, and also if it is specified as CYE* < CYS.

4. Notice in setting the Parameters of the Command

○ Data Transfer Command

Command	Parameter
DRD, DWT, DMOD, CLR SCLR, CPY, SCPY	$-32767 \leqq AX \leqq 32767$ $-32767 \leqq AY \leqq 32767$

○ Graphic Drawing Command

Command	Parameter
AMOVE, ALiNE, ARCT	$CP_X - 16383 \leqq X \leqq 16383 + CP_X$ $CP_Y - 16383 \leqq Y \leqq 16383 + CP_Y$
RMOVE, RLiNE, RRCT	$-16383 \leqq dX \leqq 16383$ $-16383 \leqq dY \leqq 16383$
APLL, APLG	$0 \leqq n \leqq 65535$ $X_{n-1} - 16383 \leqq Xn \leqq 16383 + X_{n-1}$ $Y_{n-1} - 16383 \leqq Yn \leqq 16383 + Y_{n-1}$
RPLL, RPLG	$0 \leqq n \leqq 65535$ $-16383 \leqq dXn \leqq 16383$ $-16383 \leqq dYn \leqq 16383$
CRCL	$0 \leqq r \leqq 4095$
ELPS	$0 \leqq a \leqq 255$ $0 \leqq b \leqq 255$ $0 \leqq DX \leqq 4095/\mathrm{Max}\ (a, b)$ *Note
AARC	$0 \leqq \sqrt{(Xc - CPx)^2 + (Yc - CPy)^2} \leqq 4095$
RARC	$0 \leqq \sqrt{dXc^2 + dYc^2} \leqq 4095$
AEARC	$0 \leqq \sqrt{(Xc - CPx)^2 + (Yc - CPy)^2} \leqq 4095/\mathrm{Max}\ (a, b)$ *Note
REARC	$0 \leqq \sqrt{dXc^2 + dYc^2} \leqq 4095/\mathrm{Max}\ (a, b)$ *Note
AFRCT	$CP_X - 32767 \leqq X \leqq 32767 + CP_X$ $CP_Y - 32767 \leqq Y \leqq 32767 + CP_Y$
RFRCT	$-32767 \leqq dX \leqq 32767$ $-32767 \leqq dY \leqq 32767$
AGCPY	$-32767 \leqq Xs \leqq 32767$ $-32767 \leqq Ys \leqq 32767$ $-32767 \leqq DX \leqq 32767$ $-32767 \leqq DY \leqq 32767$
RGCPY	$CP_X - 32767 \leqq dXs \leqq 32767 + CP_X$ $CP_Y - 32767 \leqq dYs \leqq 32767 + CP_Y$ $-32767 \leqq DX \leqq 32767$ $-32767 \leqq DY \leqq 32767$

*Note) Max (a, b) is the greater value of a and b.

5. Notice in setting the Parameter Registers

○ Pattern RAM Control Registers (Pr05 ~ Pr07)

PSX ≦ PPX ≦ PEX Be sure to set these parameters before
PSY ≦ PPY ≦ PEY issuing a drawing command.

0 ≦ PZCX ≦ PZX Note)
0 ≦ PZCY ≦ PZY Note) Usually set PZCX/PZCY to "0"

Note) When it doesn't use the Pattern Zoom, Set the next parmeters to "0"
PZX = PZY = 0
PZCX = PZCY = 0

○ AREA Definition Register (Pr08 ~ Pr0B)

XMIN ≦ XMAX
YMIN ≦ YMAX

Note) It needs that moving CP by using AMOVE, RMOVE, :eg after Area definition.

6. ACRTC Internal Reference Timing of the Control Registers

Reg. No.	Register Bit Name	Reference Cycle	Internal Reference Timing and Notes on Software	Rewriting
4	M/S	Frame	Setup value is not to be changed during operation.	×
	STR		STR is not to be set to "0" during command execution (when set to "0", current command execution stops).	○
	ACP	Raster	ACP is not to be rewritten during command execution.	×
	WSS		WSS is not to be rewritten during display period.	△
	CSK		Setup value is not to be changed during operation.	×
	DSK			×
	RAM		RAM is not to be rewritten during command execution.	×
	GAI		Setup value is not to be changed during operation.	×
	ACM	Frame	Setup value is not to be changed during operation.	×
	RSM		Setup value is not to be changed during operation.	×
6	DSP	Raster	Setup value is not to be changed during operation.	×
	SE0	Frame	To change the screen configuration, BE, SE1, SE2 and WDE are to be rewritten within one frame. When BE, SE1, SE2 and WDE are rewritten in different frames, the screen configuration gradually changes according to the rewriting.	○
	SE1			○
	SE2			○
	WDE			○
	ATR	Raster	Rewritten data is output at the attribute of the next raster, therefore, data is to be input at the appropriate timing.	○

Note) It is not recommended to change the setup value of the graphic bit mode (GBM) of the command control register (CCR) during operation as well as during command execution.

○ : Can be rewritten.
△ : Can be rewritten conditionally.
× : Cannot be rewritten.

Reg. No.	Register Name		Ref. Cycle	Internal Reference Timing and Notes on Software	Ref. Fig.
82	Horizontal Sync.		Raster	This register is referred at each raster. Setup value is not to be changed after the value suited to the CRT is set.	—
84	Horizontal Display		Raster	This register is referred at each raster except the vertical sync. period. Changing of the setup value is to be done during the vertical sync. period.	1
86	Vertical Sync.		Field	This register is referred at the first raster of each field. Setup value is not to be changed after the value suited to the CRT is set.	—
88	Vertical Display	Vertical Sync. Pulse Width/ Vertical Display Start Position	Field	This register is referred at the first raster of each field. Setup value of the vertical sync. pulse width is not to be changed after the value suited to the CRT is set.	—
8A	Split Screen Width	Vertical Display Width	Field	This register is referred at the horizontal front porch preceding the display of the base screen. Changing of the setup value is to be done during the vertical flyback period.	2 ①
8C		Upper Screen Display Width	Field	This register is referred at the horizontal front porch preceding the display of the upper screen. Changing of the setup value is to be done during the vertical flyback period.	2 ②
8E		Lower Screen Display Width	Field	This register is referred at the horizontal front porch preceding the display of the lower screen. Changing of the setup value is to be done during the vertical flyback period.	2 ③
90	Blink Control		Field	This register is referred at the vertical back porch. Setup value is not to be changed during the vertical back porch.	3
92	Horizontal Window Display		Raster	This register is referred at each raster. To change the position and the width of the window, setup value is to be changed during the vertical flyback period.	—
94	Vertical Window Display	Vertical Window Display Start Position	Field	This register is referred at the first raster of each field. Setup value is not to be changed during the vertical back porch.	—
96		Vertical Window Display Width	Raster	This register is referred at each raster except the vertical sync. period. Changing of the setup value is to be done during the vertical flyback period.	1
98	Graphic Cursor	Cursor X Start/ Cursor X End	Field	These registers are referred during the vertical flyback period. they are to be set before the vertical flyback period. When they are rewritten in the different fields, the position and the size of the cursor change according to the gradual rewriting.	—
9A		Cursor Y Start			
9C		Cursor Y End			

Reg. No.	Register Name		Ref. Cycle	Internal Reference Timing and Notes on Software	Ref. Fig.
C0	Upper Screen (Background)	Raster Address 0	Field	These registers are referred during the horizontal sync. period preceding the display of the upper screen, or during the horizontal back porch. They are to be rewritten before the display period of the upper screen. When they are rewritten in the different fields, the display image becomes crooked temporarily according to the gradual re-writing.	2 ③
C2		Memory Width 0			
C4		Display Start Address 0			
C6					
C8	Base Screen (Background)	Raster Address 1	Field	These registers are referred during the horizontal sync. period preceding the display of the base screen, or during the horizontal back porch. They are to be rewritten before the display period of the base screen. When they are rewritten in the different fields, the display image becomes crooked temporarily according to the gradual re-writing.	2 ④
CA		Memory Width 1			
CC		Display Start Address 1			
CE					
D0	Lower Screen (Background)	Raster Address 2	Field	These registers are referred during the horizontal sync. period preceding the display of the lower screen, or during the horizontal back porch. They are to be rewritten before the display period of the lower screen. When they are rewritten in the different fields, the display image becomes crooked temporarily according to the gradual re-writing.	2 ⑥
D2		Memory Width 2			
D4		Display Start Address 2			
D6					
D8	Window	Raster Address 3	Field	These registers are referred at the first raster of each field. They are to be rewritten before the vertical back porch. When they are rewritten in the different fields, the display image becomes crooked temporarily according to the gradual re-writing.	—
DA		Memory Width 3			
DC		Display Start Address 3			
DE					
E0	Block Cursor 1	Block Cursor Start/ Block Cursor End 1	Field (Raster)	These registers are referred at each raster during the vertical flyback period and the display prohibited period. It is necessary to finish rewriting a pair of registers (E0&E2 or E4&E6) during one display period. When E0 and E2, or E4 and E6 are rewritten in the different fields, the position and the size of the cursor change according to the gradual rewriting.	
E2		Block Cursor Address 1			
E4	Block Cursor 2	Block Cursor Start/ Block Cursor End 2			
E6		Block Cursor Address 2			
E8	Cursor Definition		Field (Raster)	This regisrer is referred at each rasrer during the vertical flyback period and the display prohibited period. Changing of the setup value is to be done during the display period.	4
EA	Zoom Factor		Raster	This register is referred at each raster. Setup value is not to be changed during the display period of the base screen.	—

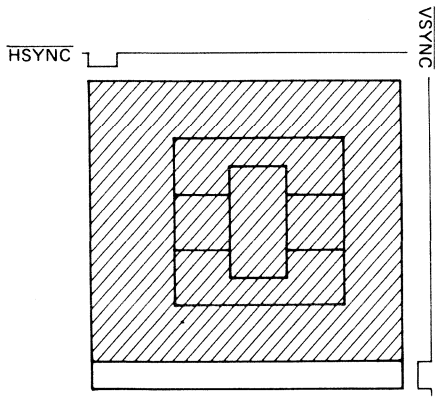


Figure 1

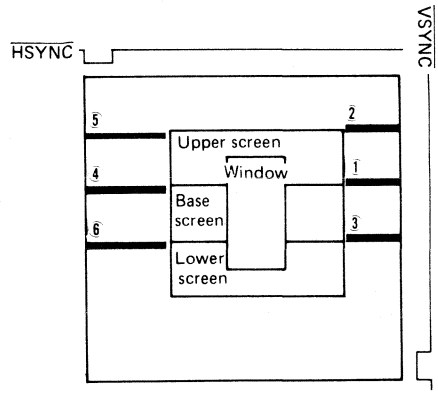


Figure 2

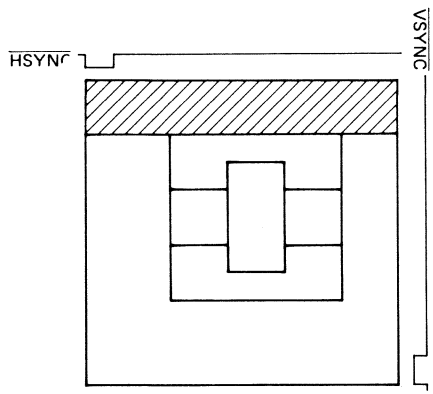


Figure 3

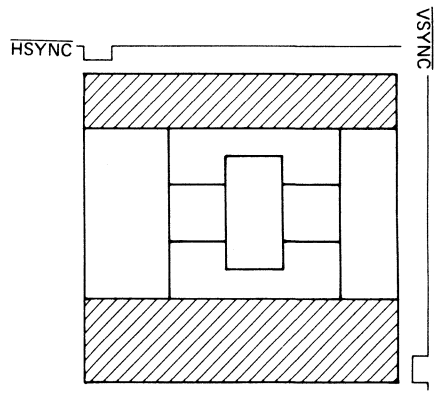


Figure 4

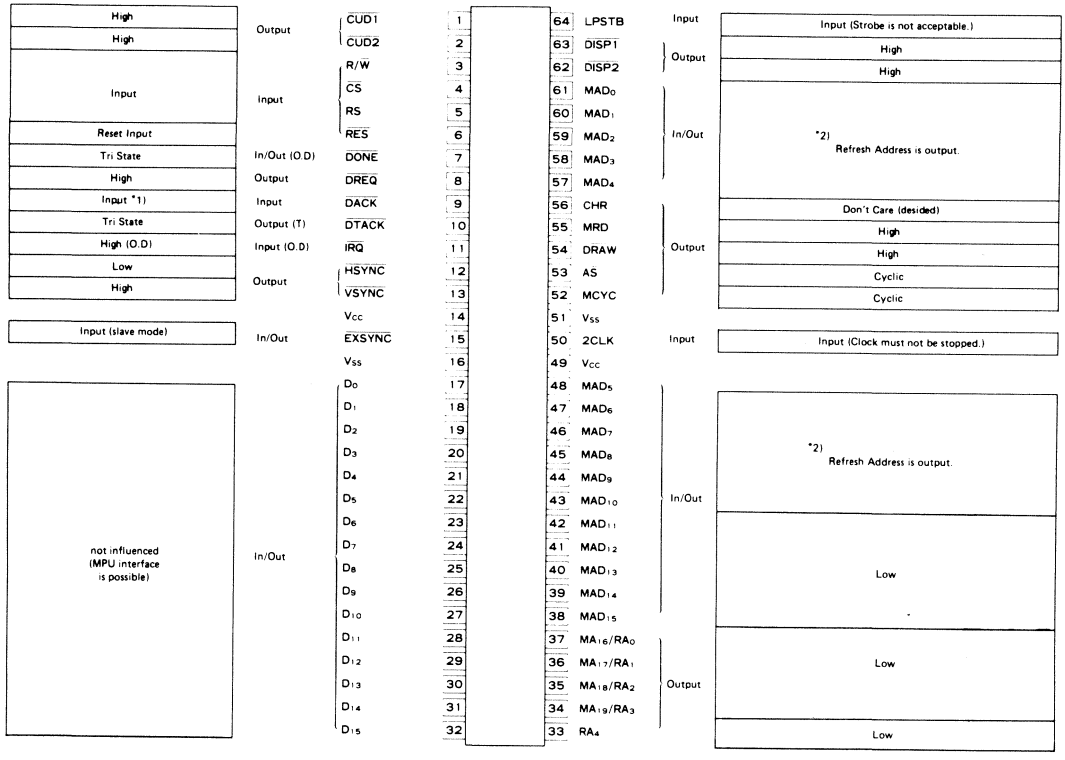
■CONTROL REGISTER

CS	RS	R/W	Reg. No.	Register Name	Abbre.	DATA (H) 15	14	13	12	11	10	9	8	DATA (L) 7	6	5	4	3	2	1	0	Initial value in Reset mode
1	-	-	-	—	—	.....								.....								.....
0	0	0	AR	Address Register	AR	.....								Address								Previous setup values are preserved.
0	0	1	SR	Status Register	SR	.....								CER	ARD	CED	LPD	RFF	RFR	WFR	WFE	CED, WFR, WFE: "High", others: "Low" ($23)
		1/0	r00	FIFO Entry	FE	F E																Initialized by Pointer Clear
		1/0	r02	Command Control	CCR	ABT	PSE	DDM	CDM	DRC	GBM			CRE	ARE	CEE	LPE	RFE	RRE	WRE	WEE	ABT: "High", others: "Low" ($8000)
		1/0	r04	Operation Mode	OMR	M/S	STR	ACP	WSS	CSK	DSK			RAM	GAI			ACM		RSM		M/S, STR: "Low", others: previous preserved
		1/0	r06	Display Control	DCR	DSP	SE1	SE0		SE2	SE3			A T R								Previous setup values are preserved.
		-	r08 ↓ r7E	(undefined)	—	.....								.....								.....
		1	r80	Raster Count	RCR	.....								R C								Indefinite (values at a stop are preserved.)
		1/0	r82	Horizontal Sync.	HSR	H C								.....				H S W				not influenced (Setup values before Reset are preserved.)
			r84	Horizontal Display	HDR	H D S								H D W								
		1/0	r86	Vertical Sync.	VSR	.....								V C								
		1/0	r88	Vertical Display	VDR	V D S								.....				V S W				
		1/0	r8A	Split Screen Width	SSW	.....								S P 1								
		1/0	r8C			.....								S P 0								
		1/0	r8E			.....								S P 2								
		1/0	r90	Blink Control	BCR	BON1				BOFF1				BON2				BOFF2				
		1/0	r92	Horizontal Window Display	HWR	H W S								H W W								
		1/0	r94	Vertical Window Display	VWR	.....								V W S								
		1/0	r96			.....								V W W								
		1/0	r98	Graphic Cursor	GCR	C X E								C X S								
		1/0	r9A			.....								C Y S								
		1/0	r9C			.....								C Y E								
		-	r9E																			
		-	rA0 ↓ rBE	(undefined)	—	.....								.....								.....
0	1	1/0	rC0	Upper Screen Base Address	RAR0	.....				L R A 0				.....				F R A 0				not influenced (Setup values before Reset are preserved.)
		1/0	rC2		MWR0	CHR	.....			M W 0												
		1/0	rC4		SAR0	.....				S D A 0				.....				SA0H/SRA0				
		1/0	rC6			S A 0 L																
		1/0	rC8	Lower Screen Base Address	RAR1	.....				L R A 1				.....				F R A 1				
		1/0	rCA		MWR1	CHR	.....			M W 1												
		1/0	rCC		SAR1	.....				S D A 1				.....				SA1H/SRA1				
		1/0	rCE			S A 1 L																
		1/0	rD0	Upper Screen Window Address	RAR2	.....				L R A 2				.....				F R A 2				
		1/0	rD2		MWR2	CHR	.....			M W 2												
		1/0	rD4		SAR2	.....				S D A 2				.....				SA2H/SRA2				
		1/0	rD6			S A 2 L																
		1/0	rD8	Lower Screen Window Address	RAR3	.....				L R A 3				.....				F R A 3				
		1/0	rDA		MWR3	CHR	.....			M W 3												
		1/0	rDC		SAR3	.....				S D A 3				.....				SA3H/SRA3				
		1/0	rDE			S A 3 L																
		1/0	rE0	Block Cursor 1	BCUR1	B C W1				B C S R 1				.....				B C E R 1				
		1/0	rE2			B C A 1																
		1/0	rE4	Block Cursor 2	BCUR2	B C W 2				B C S R 2				.....				B C E R 2				
		1/0	rE6			B C A 2																
		1/0	rE8	Cursor Definition	CDR	C M		CON1			COFF1			.....		CON2				COFF2		
		1/0	rEA	Zoom Factor	ZFR	H Z F				V Z F				.....								
		1	rEC	Light Pen Address	LPAR	.....								CHR	.....			L P A H				
		1	rEE			L P A L																
		-	rF0 ↓ rFE	(undefined)	—	.....								.....								.....

Note { 1 ·····"High" level
0 ·····"Low" level

Read FIFO Write FIFO	initialized by Pointer Clear
Graphic Pattern RAM	Values before Reset are preserved.
Parameter Register	Values before Reset are preserved.
Command Register	initialized by Pointer Clear (Read is impossible.)

Pin (the period until STR bit becomes "High" after the release of Reset)



*1) 8/16-bit interface switching:
While RES is rising from "Low" to "High"; if DACK is at "High", this is set as 16 bit interface.
; if DACK is at "Low", this is set as 8 bit interface.

*2) It depends on GAI. :

GAI	Refresh Address Output Line	GAI	Refresh Address Output Line
0	MAD 0 ~ 7, Others are "Low"	3	MAD 3 ~ 10, Others are "Low"
1	MAD 1 ~ 8, Others are "Low"	5–6	MAD 0 ~ 7, Others are "Low"
2	MAD 2 ~ 9, Others are "Low"	7	MAD 0 ~ 7, Others are "Low"
		4	MAD 4 ~ 11, Others are "Low"

LIMITATION AND CAUTION ON USING THE HD63484 (ACRTC)

This document describes the limitation on using the R mask and the S mask versions of the HD63484 (ACRTC).

The status of the item numbers described in this document is summarized in the following table.

○ Limitation for the ACRTC function.

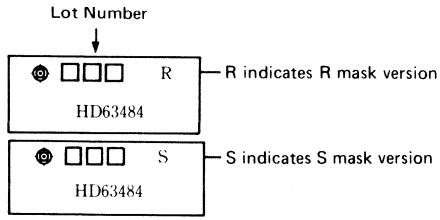
No.	Limitation for the ACRTC function	R Mask	S Mask	Future
1	Light Pen Interface	×	○	–
2	RS Signal during DMA Transfer	×	○	–
3	AREA Mode for AFRCT, RFRCT and PAINT Commands	×	○	–
4	DRD Command	×	○	–
5	DMOD Command	×	○	–
6	PAUSE Bit	×	○	–
7	$\overline{AS}$ Signal output timing during Zooming	×	○	–
8	BLINK Feature	×	○	–
9	CLR, WT and DWT Command	×	○	–
10	Writing to Registers during DRD Command Execution	×	×	△
11	Command DMA Transfer Mode	×	×	△
12	Tiling using the PAINT Command	×	○	–
13	Displaying WINDOW	×	×	△
14	PARAMETER for ELLIPSE Command	×	×	△
15	Light Pen Strobe Detect (LPD) Status Bit	–	×	△

Note) × : It can't be used.
△ : Schedule to fix this limitation is not yet determined.

LIMITATIONS AND CAUTIONS ON USING THE HD63484 (ACRTC)

○ Applicable parts:
HD63484 (R mask) and HD63484 (S mask)

○ Marking of R mask and S mask parts:



As indicated above, parts with "R" after the lot number is the R mask version, and with "S" is the S mask version. Parts with no letter marked after the lot number are the 1st cut sample.

○ Limitations on usage and their countermeasures:
The following pages described earlier, most of these limitations are being fixed, but in the meantime, please observe the appropriate suggested countermeasures when using the ACRTC.

HD63484 ACRTC LIMITATIONS ON USAGE

April 16, 1985

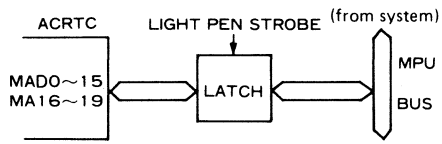
NO. 1 LIGHT PEN INTERFACE

DESCRIPTION:

The light pen function does not work properly.

COUNTERMEASURE:

Please use the following external circuit to latch the address and avoid using the light pen register.



STATUS:

Applicable to the "R" mask (2nd cut) version.

It is fixed in the "S" mark (3rd cut) version, but please refer to item no. 18 when using the light pen function.

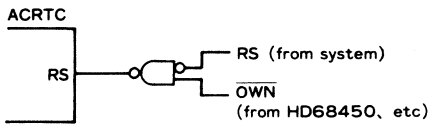
NO. 2 RS SIGNAL DURING DMA TRANSFER

DESCRIPTION:

The RS signal must be held "high" (FIFO selected) during DMA transfer.

COUNTERMEASURE:

Please use the following circuit to keep the RS signal "high" during DMA transfer.



NOTE: $\overline{OWN}$ signal indicates the DMA transfer cycle.

STATUS:

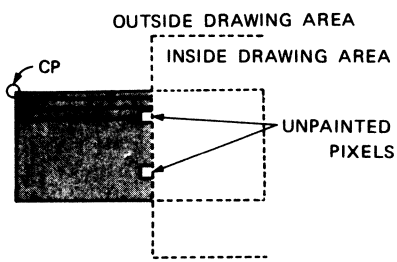
Applicable to the "R" mask (2nd cut) version.
It is fixed in the "S" mask (3rd cut) version.

NO. 3 AREA MODE FOR AFRCT, RFRCT, AND PAINT COMMANDS

DESCRIPTION:

As shown in the following examples, drawing area function (as defined by the Area Definition Register) may not operate correctly for AFRCT, RFRCT, and PAINT commands.

1. AFRCT and RFRCT commands:
   Figure shown below is an example of an execution of the AFRCT or the RFRCT (absolute and relative filled rectangle) command by specifying to draw only outside the "drawing area". In this case, the ACRTC may stop drawing one dot before the specified drawing area. As a result, the side of the rectangle will appear jagged.



The same will result when drawing inside the drawing area.

2. PAINT command:
   Please refer to the appendix, "PAINT Command's AREA Mode Problem", attached at the end.

COUNTERMEASURE:

Please avoid executing the AFRCT, RFRCT, and PAINT commands using the drawing area feature. 000 or 100 should be specified for the Area Mode field (AREA) for these commands.

STATUS:

Applicable to the "R" mask (2nd cut) version.
It is fixed in the "S" mask (3rd cut) version.

NO. 4 DRD COMMAND

DESCRIPTION:

1. When zero is specified for the first parameter of the DRD command, then the ACRTC will read 65536 words in the X direction.
2. When the total words that is read is 7 words or less for the DRD command, the ACRTC will not output $\overline{\text{DREQ}}$ to the DMAC. However, the data read from the display memory is properly set in the Read FIFO. (The Command End flag is not set.)

COUNTERMEASURE:

1. Please avoid using zero as the first parameter of the DRD command.
2. Please set the total number of words to read as 8 or more words.

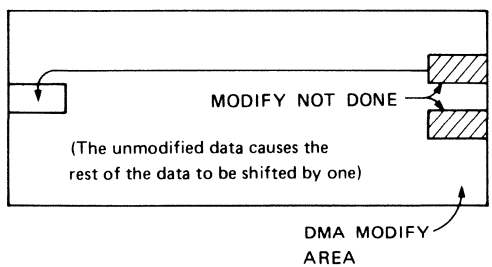
STATUS:

Applicable to the "R" mask (2nd cut) version.
It is fixed in the "S" mask (3rd cut) version.

NO. 5 DMOD COMMAND

DESCRIPTION:

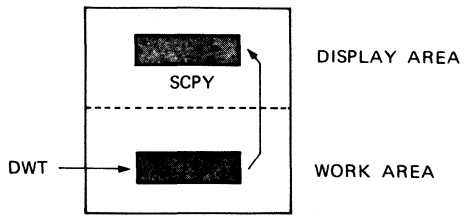
1. When zero is specified for the first parameter of the DMOD command, then the ACRTC will modify 65536 words in the X direction.
2. When using the DMOD command, the last word in the X direction may be skipped and that data used to modify the first word in the X direction in the next line as shown in the figure below.



However, the $\overline{\text{DREQ}}$ signal will be issued the correct number of times, so that, for example, if 2 words were not modified at the correct position, 2 words will be left in the Write FIFO. This will cause the 2 words left in the Write FIFO to be interpreted by the ACRTC as the next command.

COUNTERMEASURE:

Please avoid using the DMOD command. By using the DWT and the SCPY commands, same function as the DMOD command can be realized, as shown below. Otherwise, please use the MOD command.



STATUS:

Applicable to the "R" mask (2nd cut) version.
It is fixed in the "S" mask (3rd cut) version.

NO. 6 PAUSE BIT

DESCRIPTION:

When the Pause Bit is set during command execution, the ACRTC will not operate correctly after the Pause Bit is reset.

COUNTERMEASURE:

Please avoid setting the Pause Bit while the ACRTC is executing a command.

STATUS:

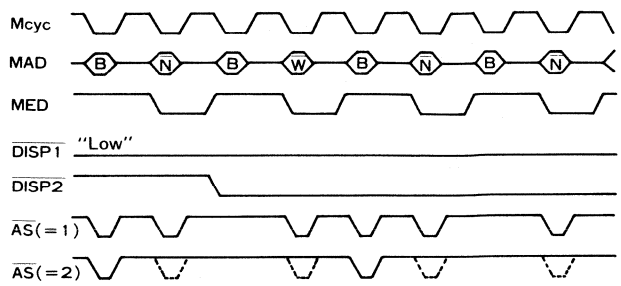
Applicable to the "R" mask (2nd cut) version.
It is fixed in the "S" mask (3rd cut) version.

NO. 7 $\overline{AS}$ SIGNAL OUTPUT TIMING DURING ZOOMING

DESCRIPTION:

When window screen is superimposed on the base screen and the base screen is being zoomed, then the $\overline{AS}$ output for the window screen is not outputted as shown in the following example.

Example of superimposing and zooming the base screen by factor of 2:



B : Display address for base screen.
W: Display address for window screen.
*1: Correct $\overline{AS}$ output for HZF = 1 (zoomed by factor of 2 in the horizontal direction).
*2: Present $\overline{AS}$ output for HZF = 1.

COUNTERMEASURE:

Please generate the $\overline{AS}$ output for the window screen using external circuit with the following logic:

When $\left\{\begin{array}{l} 2CLK - LOW \\ MCYC - LOW \\ MRD - LOW \\ \overline{DRAW} - HIGH \end{array}\right\}$ then $\overline{AS}$ "LOW".

STATUS:

Applicable to the "R" mask (2nd cut) version.
It is fixed in the "S" mask (3rd cut) version.

NO. 8 BLINK FEATURE

DESCRIPTION:

BLK2 (attribute output) signal and $\overline{CUD2}$ signal does not blink.

COUNTERMEASURE:

When using the blink feature, avoid using BLK2 and $\overline{CUD2}$ signals. Use BLK1 and $\overline{CUD1}$ signals for blinking.

The cross hair cursor can be blinked by using the blink rate of the $\overline{CUD1}$.

STATUS:

Applicable to the "R" mask (2nd cut) version.
It is fixed in the "S" mask (3rd cut) version.

NO. 9 CLR, WT, AND DWT COMMANDS

DESCRIPTION:

When CLR, WT, or DWT commands is executed directly after executing AFRCT, RFRCT, or PTN command, then proper drawing is not done.

COUNTERMEASURE:

Before issuing CLR, WT, or DWT command, draw one or more dots in an unused area using the following commands: LINE, RCT, PLL, PLG, CRCL, ARC, ELPS, EARC, DOT. This will insure proper execution of CLR, WT, and DWT commands.

STATUS:

Applicable to the "R" mask (2nd cut) version.
It is fixed in the "S" mask (3rd cut) version.

NO. 10 WRITING TO REGISTERS DURING DRD COMMAND EXECUTION

DESCRIPTION:

When the ACRTC is executing the DRD (DMA read) command and has its $\overline{\text{DREQ}}$ asserted, the MPU cannot write to the ACRTC's registers other than the FIFO register. This condition can occur in cycle steal DMA mode, or at the start of the burst DMA mode.

COUNTERMEASURE:

Avoid writing to the ACRTC's register except for the FIFO register when the ACRTC is executing the DRD command.

STATUS:

Applicable to both the "R" mask (2nd cut) and "S" mask (3rd cut) of the ACRTC. Schedule to fix this limitation is not yet determined.

NO. 11 COMMAND DMA TRANSFER MODE

DESCRIPTION:

The command DMA transfer mode does not operate properly. (The command DMA transfer mode is the mode in which the ACRTC commands are transfered by the DMAC transfers instead of the MPU).

COUNTERMEASURE:

Please avoid using the command DMA transfer mode, and always have the MPU give the commands to the ACRTC. Therefore, please have the Command DMA Mode bit (DCM) in the Command Control Register (CCR) set to 0 (zero) at all times.

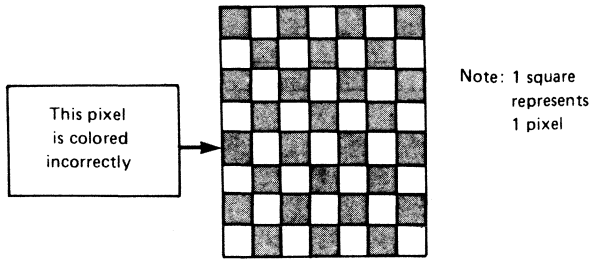
STATUS:

Applicable to both the "R" mask (2nd cut) and "S" mask (3rd cut) of the ACRTC. Schedule to fix this limitation is not yet determined.

NO. 12 TILING USING THE PAINT COMMAND

DESCRIPTION:

When the PAINT command is used, there may occur a pixel with a wrong color at the location next to the left hand border, as shown in the following figure.



Wrong color means that the color of the pixel is not what is specified by the Pattern RAM. More specifically, instead of color register 0 being used, color register 1 may be used, and also the other way around.

This will not always happen at the left hand border, but only some of the time. This occurs when the drawing bus cycles of the ACRTC must wait for more than 2 memory cycles in single access mode, dual access mode 1, or DRAM refresh operation.

COUNTERMEASURE:

The PAINT command can be done properly when the following conditions are met.

1. The same color is set in color register 0 and 1.
2. The current pointer (CP) is initially set at the uppermost portion of the figure to be painted.

STATUS:

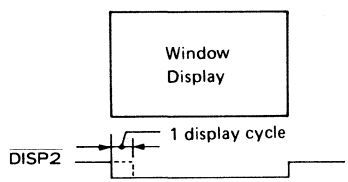
Applicable to the "R" mask (2nd cut) version.
This has been fixed in the "S" mask (3rd cut) version.

NO. 13 DISPLAYING WINDOW

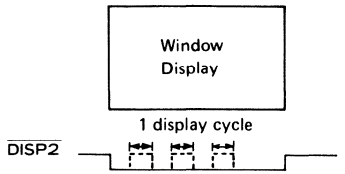
DESCRIPTION:

When only the window is displayed and the display has higher priority over drawing, then the $\overline{\text{DISP2}}$ signal is not asserted properly.

1. Single Access Mode or Dual Access Mode 0
   If drawing operation is in progress during the display period, then the $\overline{\text{DISP2}}$ signal is asserted "low" one memory cycle later than normal as shown in the following figure. This will not occur if WSS = 1 is used.



2. Dual Access Mode 1
   If drawing operation is in progress during the display period, then $\overline{DISP2}$ will oscillate as shown in the following figure.



Note that the drawing operation will proceed normally without any problems in all modes.

COUNTERMEASURE:

Please avoid displaying only the window screen. Always display the window with the other background screens, or, when only 1 screen needs to be displayed, display the background screens.

STATUS:

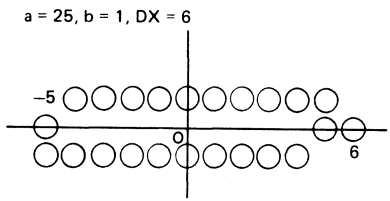
Applicable to both the "R" mask (2nd cut) and "S" mask (3rd cut) of the ACRTC. Schedule to fix this limitation is not yet determined.

NO. 14 PARAMETER FOR ELLIPSE COMMAND

DESCRIPTION:

When the parameters a, b, and DX for the ellipse command satisfies the following condition, the command fails to end and the ACRTC goes into a deadlock: $\mathbf{a \geq 2b(2DX-1)}$.

This condition results in an ellipse that is very flat. The following is one example of bad parameters. (This ellipse is drawn counterclockwise.)



COUNTERMEASURE:

Please use the parameter which satisfies the following condition to avoid the problem: $\mathbf{a < 2b(2DX-1)}$.

STATUS:

Applicable to both the "R" mask (2nd cut) and "S" mask (3rd cut) of the ACRTC. Schedule to fix this limitation is not yet determnined.

NO. 15 LIGHT PEN STROBE DETECT (LPD) STATUS BIT

DESCRIPTION:

When the Status Register is read with values greater than or equal to $C0 in the Address Register, then the Light Pen Strobe Detect (LPD) bit in the Status Register is cleared

This will occur only in the "S" mask version.

COUNTERMEASURE:

When using the LPD bit in the Status Register and it is set, please read the status register with values less than $C0 in the address register.

STATUS:

Applicable to "S" mask (3rd cut) version of the ACRTC only. Schedule to fix this limitation is not yet determined.

APPENDIX 1:

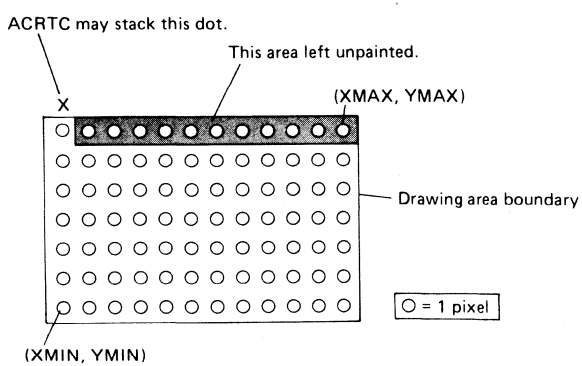
PAINT COMMAND'S AREA MODE LIMITATION

(R mask version only)

1. PROBLEM DESCRIPTION

(1) When drawing inside the drawing area (AREA mode = 001, 010, or 011; when using mode 001, the following description is applicable only when the pointer is set at the uppermost line inside the drawing area)

○ Usually, only 1 dot in the uppermost line inside the drawing area is painted, as shown in the figure below, and the rest is left unpainted.

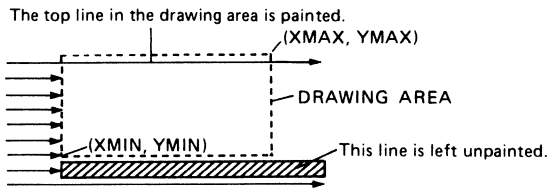


○ Occasionally, more than 1 dot in the uppermost line is painted. (The probability that n dots are painted is r to the nth power, where r is approximately between 0.1 and 0.01.) In this case, more than 1 coordinate may be stacked incorrectly as unpainted area.

○ This problem does not occur at the right, the left, nor the bottom sides of the drawing area. Incorrect coordinates are not stacked as unpainted area.

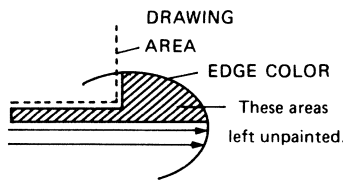
(2) When drawing outside the drawing area (AREA mode = 101, 110, or 111)

○ Usually, 1 line under the drawing area is left unpainted as shown in the following figure. Also, the uppermost line inside the drawing area is painted.



○ Occasionally, a few dots under the drawing area or at the uppermost line inside the drawing area are drawn. (The probability is small.)

○ Depending on the shape of the edge, right side outside the drawing area may be left unpainted as shown in the example figure shown below.

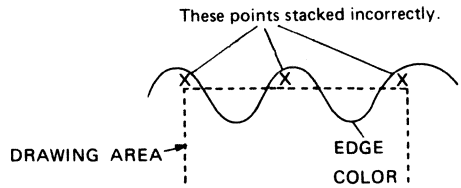


2. SOFTWARE COUNTERMEASURE SUGGESTIONS

(1) When drawing inside the drawing area (AREA mode = 001, 010, or 011)

○ The coordinates outside of the drawing area may be stcked as unpainted area, but the ACRTC will not paint these coordinates even when it is specified with another PAINT command since it is outside the drawing area. Thus, this only causes some reduction in drawing speed, but will not cause incorrect painting operation.

○ Usually, only 1 coordinate is stacked incorrectly as unpainted area at the top left hand point and each of the intersection of the edge color and the drawing area boundary (see figure shown below), so there is no need to have unnecessarily large stack area for unpainted coordinates.



○ Checking the stacked coordinates to see whether it is outside the drawing area is an effective way to screen out incorrect unpainted coordinates.

(2) When drawing outside the drawing area (AREA mode = 101, 110, or 111)

○ The following countermeasure sequence is suggested.

(a) Save the drawing in the drawing area to an used buffer area.

(b)′ Paint without using the AREA mode feature.

(c) Copy the original drawing, saved at (a), back to its original location using GCPY command.