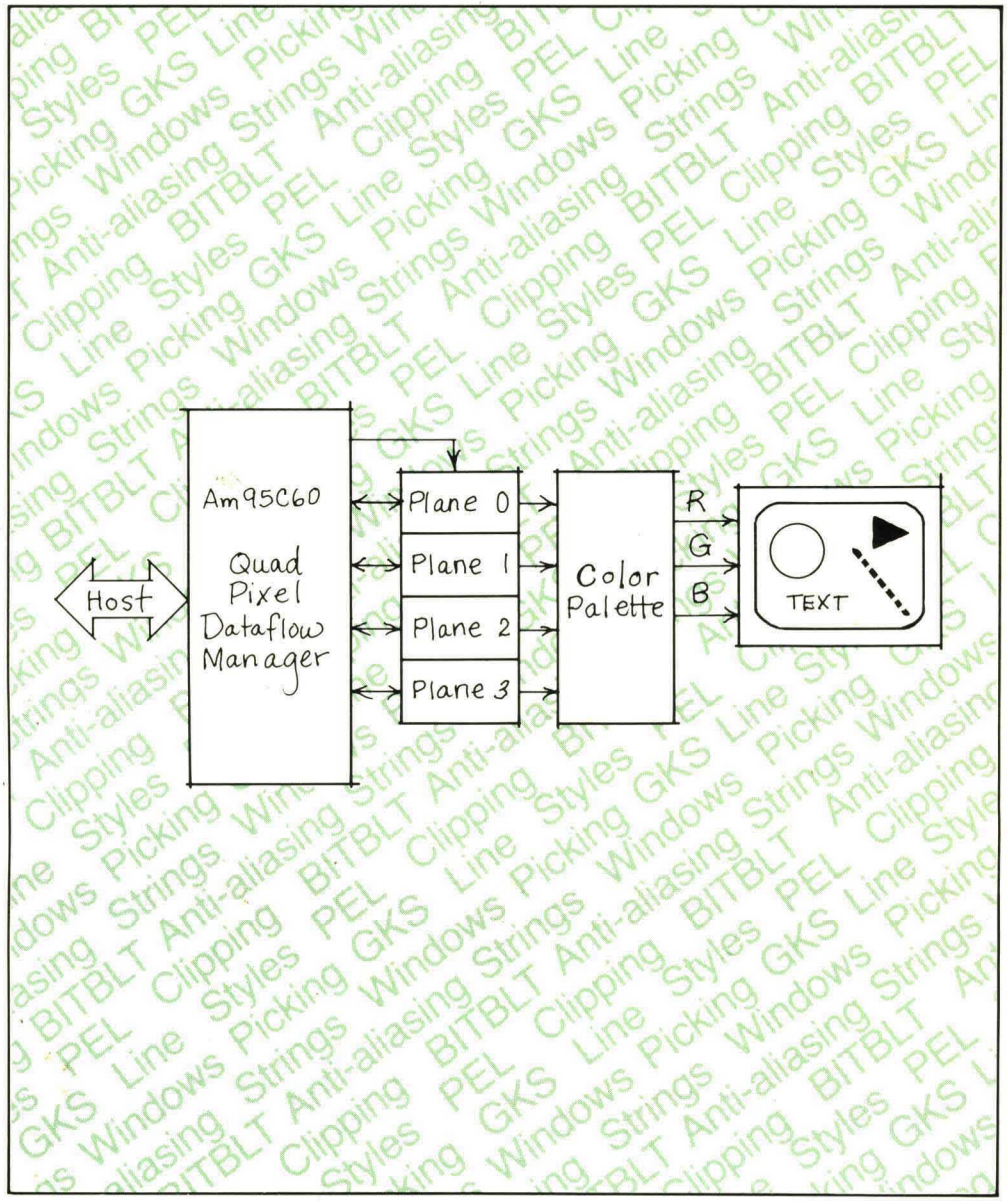
Quad Pixel Dataflow Manager (QPDM) Am95C60

Technical Manual
Revision B



ADVANCED MICRO DEVICES

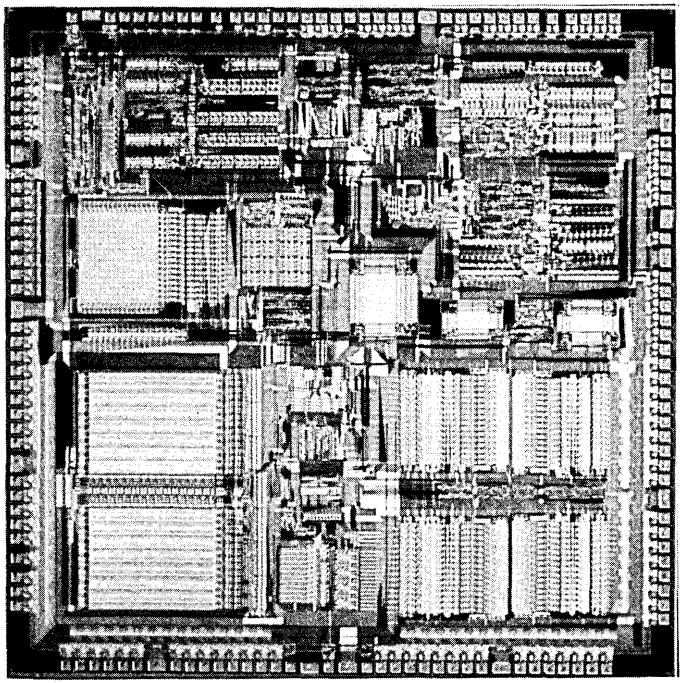
Advanced Micro Devices

Quad Pixel Dataflow Manager (QPDM) Am95C60

Technical Manual

REVISION B

901 Thompson Place, P.O. Box 3453, Sunnyvale, California 94088
(408) 732-2400 TWX: 910-339-9280 TELEX: 34-6306



Written by: Tom Crawford
Senior Applications Engineer
Edited by: Sue Louie Thilking
Technical Editor

Please Register for Product Updates

The Am95C60 QPDM Technical Manual is a "Preliminary Release". It is an accurate description of the device as originally designed and built. However, we anticipate some changes in the device as data is collected from the many systems now being designed with it. To be kept up-to-date on the status of the Am95C60 QPDM, and to receive new issues of the documentation as it becomes available, we ask that you submit your name and address to:

Advanced Micro Devices, Inc.
Mr. Steve Crane (MS-30)
QPDM Updates
P. O. Box 3453
Sunnyvale, CA 94088-3453

Last minute additions for this manual:
Page 3-5, VBLKI Interrupt: Change last sentence to:
The condition is true for one VIDCLK period at the beginning of Vertical Blank.

Page 14-58, Point: Add sentence to Performance:
The time for intermediate points (neither the first nor the last) in a list is 62 SYSCLK cyles.

Page 14-77, Set Scale Factor: Add sentence to Comments:
The ratio of the scale factors must not exceed 16 to 1 (either way).

Page B-1, LS: This is a 1-bit field, not 2-bit as stated.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

The Am95C60 is a graphics processor that manages one to four bit planes, each up to 4 K by 4 K pixels. It is a CMOS microprogrammed machine with a powerful instruction set designed for fast and efficient graphics processing.

This manual consists of 14 chapters and two appendixes.

Chapter 1 introduces the manual and contains a deliberately simplified description of the Am95C60 itself.

Chapter 2 discusses the hardware interface. The pinouts are also covered. For more information on I/O timing, refer to the Am95C60 data sheet.

Chapter 3 discusses the host—Am95C60 communications. It describes how to talk to the Am95C60 at the very lowest level.

Chapter 4 describes the Am95C60 register set.

Chapter 5 discusses addressing modes and scaling.

Chapter 6 describes the end-point and line style options available in drawing lines, arcs and circles. This chapter also covers the logical PEL.

Chapter 7 describes clipping and picking.

Chapter 8 covers graphical operations. Using block transfers or when drawing figures, the new data may be combined with the old.

Chapter 9 describes how pixels are processed to smooth out the steps in diagonal vectors. This is called anti-aliasing.

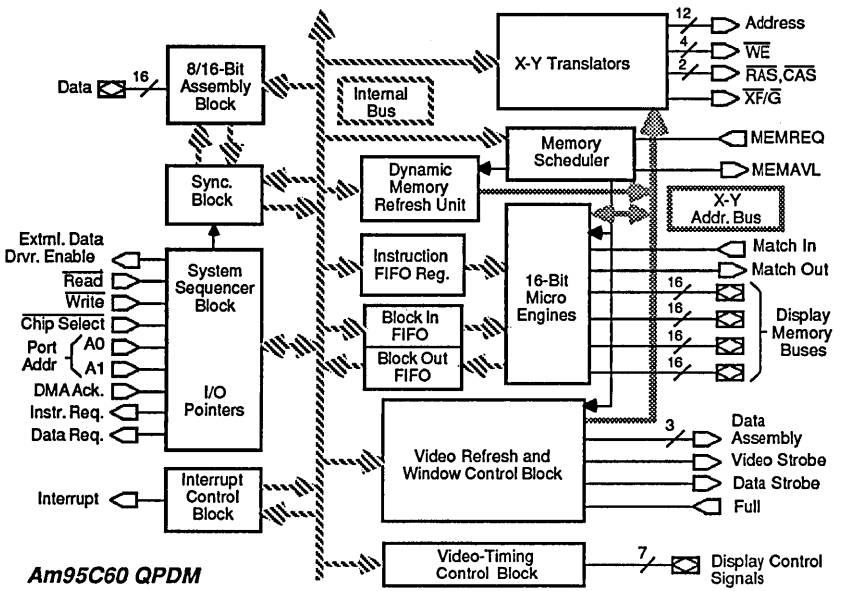
Chapter 10 explains how text can be placed in with the graphics by using the String instruction.

Chapter 11 describes both hardware and software windows. Windows can be used for dragging an object, pop-up menus, and scrolling.

Chapter 12 explains display memory configurations (how much memory is needed and where it is needed).

Chapter 13 contains items that didn't seem to fit anywhere else.

Chapter 14 describes the extensive instruction set of the Am95C60.



Am95C60 QPDM

1.2 DESCRIPTION

Figure 1-1 shows the Am95C60 in a minimum color system. There are a total of four bit planes, called Red, Green, Blue and Intensify. Each bit plane is made up of four VRAM chips, providing a 1 K x 1 K map. Each bit plane has a shift register to provide parallel-to-serial conversion. The Am95C60 provides a 16-bit data path to each of the four bit planes. Address and control terms are provided to the four bit planes in parallel. The host interface consists of a 16-bit data bus and the normal lines necessary to address a peripheral chip.

Figure 1-2 shows the Am95C60 in a large video system. This system includes six Am95C60s allowing 24 bit planes. Each bit plane has a serializer. Eight of the bit planes are used for Red, eight for Green and eight for Blue. The outputs of the serializers go directly to digital-to-analog converters (DACs); there is no requirement for a look-up table in this system. The six Am95C60s are connected to a common host interface; only the chip selects are decoded.

1.3 Am95C60 FUNCTIONS

The Am95C60 performs the three basic functions required in a graphics application: display refresh, display memory update, and dynamic RAM refresh. These functions are discussed in more detail in the following sections.

1.3.1 Display Refresh

The Am95C60 is intended to be used with Video RAMs (VRAMs). During each horizontal blanking time, the Am95C60 generates a transfer cycle. This cycle moves a scan line of display information from the dynamic portion of the VRAM into the shifter portion of the VRAM. The scan line information is then shifted out of the VRAM for presentation on the display.

If a hardware window has been selected, then three transfer cycles take place for each scan line. The first transfer cycle accesses the background information that appears before (to the left of) the window, the second transfer cycle accesses the information in the window and the third transfer cycle accesses the information that appears after (to the right of) the window. External hardware is required to concatenate and buffer the data from the VRAMs to properly accomplish windows. Windows are discussed in detail in Chapter 11 of this manual.

1.3.2 Display Memory Update

The second major task accomplished by the Am95C60 is the updating of display memory. The Am95C60 is a microprogrammed graphics processor that can directly execute high-level graphics operations such as drawing lines and circles or filling an arbitrary bounded area. These instructions and capabilities are discussed throughout this manual. The detailed descriptions of the instructions are in Chapter 14.

1.3.3 Dynamic VRAM Refresh

The third major task accomplished by the Am95C60 is the refresh of the VRAMs that make up the display memory. The Am95C60 generates $\overline{\text{CAS}}$-before-$\overline{\text{RAS}}$ refresh cycles at a programmable rate. A refresh address is supplied with each $\overline{\text{CAS}}$-before-$\overline{\text{RAS}}$ refresh cycle to allow easy conversion to $\overline{\text{RAS}}$-only cycles.

1.4 THE Am95C60 SPEED

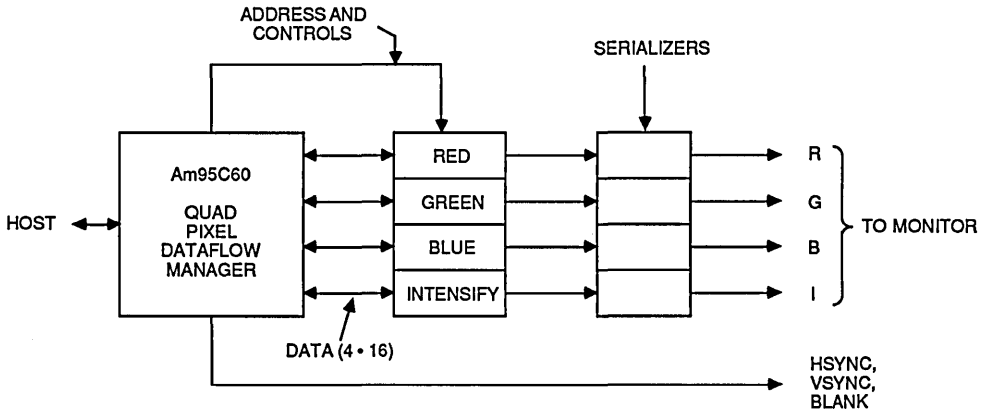
Working with VRAMs, the Am95C60 can easily manage high-resolution screen formats requiring dot clock rates up to 320 MHz (160 MHz if the hardware window is being used). Moreover, it performs the display memory update functions very quickly as indicated in Table 1-1.

The numbers in this table are in microseconds, assume a 20 MHz SYSCLK and are independent of the number of bit planes involved. Three columns are given for each operation. The first column in the instruction overhead; this time is incurred once per instruction. The second column, intermediate overhead, is incurred once per major iteration as indicated. The third column is incurred every pixel or scan line as indicated.

These numbers assume absolute addressing and no scaling.

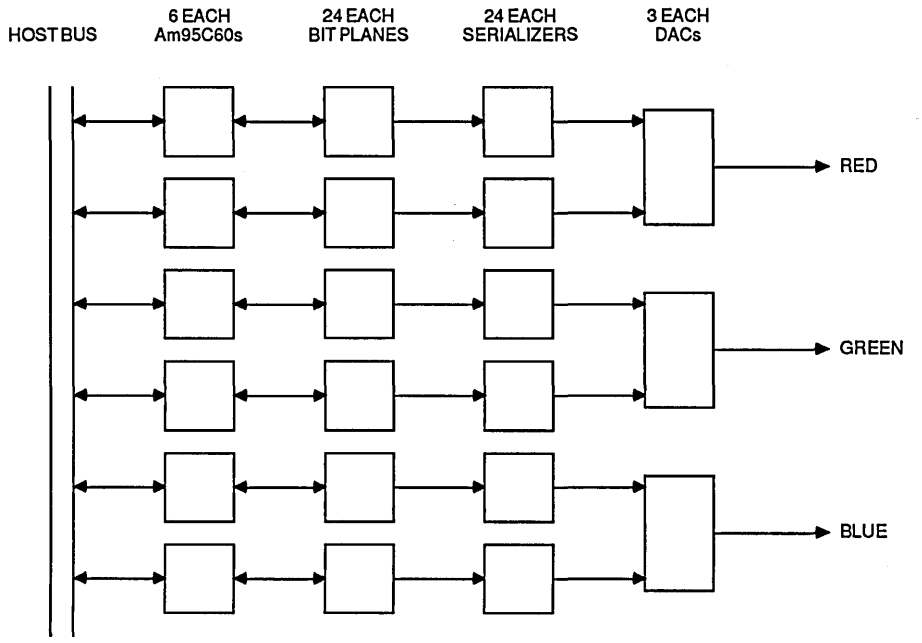
1.5 Am95C60 WINDOWS

The Am95C60 provides the timing and control required to support a single (nondestructive) hardware window. External hardware (Am8171/72 VDAF or equivalent) is required to decode the timing and control to actually implement the window. The size and position of the window is programmed into a set of registers on the Am95C60. The image to appear in the window is



07785A 1-1

Figure 1-1 Am95C60 In a Minimum System



07785A 1-2

Figure 1-2 Large Am95C60 System

located in some area of display memory other than that normally visible on the screen. Since the window parameters (position and size) are dynamically programmable, it is easy to drag a rectangular area containing an object. It is also easy to perform smooth scrolling and panning of either the background or the window. Windows are discussed in detail in Chapter 11.

1.6 Am95C60 DISPLAY MEMORY SUPPORT

The Am95C60 supports display memory sizes ranging from 1024 x 1024 pixels up to 4096 x 4096 pixels in each of four planes. The chart below shows some examples of display memory sizes and RAM sizes.

Display Memory Size	RAM Size	RAMs/ Plane	RAMs/ Am95C60
1024 x 1024	64 K x 4	4	16
1024 x 2048	64 K x 4	8	32
2048 x 2048	64 K x 4	16	64
4096 x 4096	64 K x 4	64	256
4096 x 4096	256 K x 4	16	64

1.7 UP TO 64 Am95C60s CAN BE CASCADED FOR 256 BIT PLANES

Am95C60s can be cascaded to support up to 256 bit planes. No performance degradation occurs in a cascaded system except for some block I/O operations. Line drawing (or BitBLT), for example, takes the same amount of time regardless of the number of bit planes involved.

1.8 Am95C60 SCALING

The Am95C60 can scale addresses using multiplication. Independent scale factors can be set for X and Y. This facility is used with some address modes so that the user can think of the display memory (or some portion of the display memory) as representing a unit square. Scaling is covered in detail in Chapter 5 of this manual.

Table 1-1 Display Memory Update Time

Instruction	Instruction Overhead	Intermediate Overhead	Execution Time	Comments
Line	12.9 µs	(n/a)	300 ns/pixel	
Line	12.9 µs	(n/a)	4750 ns/pixel	Anti-Aliased
Polyline	10.6 µs	4.8 µs/segment	300 ns/pixel	Connected Segments
Arc	28.2 µs	2.7 µs/octant	750 ns/pixel	
Arc	28.2 µs	2.7 µs/octant	4750 ns/pixel	Anti-Aliased
Circle	9.9 µs	2.7 µs/octant	750 ns/pixel	
Circle	9.9 µs	2.7 µs/octant	4750 ns/pixel	Anti-Aliased
Copy Block	10.9 µs	1.8 µs/scan line	60 ns/pixel	BitBLT
Transform Block	11.0 µs	(included)	1280 ns/pixel	3X Zoom
Seed Fill	10.0 µs	12.1 µs/scan line	280 ns/pixel	Intermediate Overhead varies with shape
Filled Rectangle	11.9 µs	2.2 µs/scan line	19 ns/pixel	Graphical SET
Filled Triangle	54.9 µs	8.0 µs/scan line	19 ns/pixel	Intermediate Overhead varies with shape
String	6.3 µs	9.4 µs/character	2000 ns/scan line	

CHAPTER 2

HARDWARE INTERFACE

2.1 INTERFACE BUSES

This chapter explains what the user has to do in order to put an Am95C60 onto a Printed Wiring Board (PWB). The discussion begins with a description of the four interface buses. This is followed by descriptions of the pins for each of the buses. The timing diagrams can be found in the data sheet.

Figure 2-1 is a logic diagram of the Am95C60 showing the four buses. Table 2-1 is the pinout table.

2.1.1 System Bus

The Am95C60 system bus interface is easy to use in most systems. It is always a slave on the bus. There is a 16-bit bidirectional data bus (that can be operated in 8-bit mode). Two address bits select one of four I/O ports while $\overline{CS}$ and $\overline{RD}$ and $\overline{WR}$ control normal transfers.

Three lines are used to implement DMA operations. The instruction FIFO can use only flow-through DMA while the Block Input FIFO and Block Output FIFO can use either flow-through or fly-by DMA.

One signal is used to control data buffers between the data pins and the system data bus. There is an interrupt request and a reset input. A clock input line and a match control pair complete the system bus of the Am95C60.

The following paragraphs are the formal English-language descriptions of the system bus pins on the Am95C60. The pin numbers are in Table 2-1. All signal pins on the Am95C60 have TTL I/O levels.

SYSTEM BUS PINS

D0–D15 System Data Bus (Bidirectional)

These 16 lines are used for transferring information between the host and the Am95C60. The nature of the transaction is controlled by the two address bits and the RD and WR lines. Bit 15 is the most significant bit.

$\overline{RD}$ Read (Input)

This line, when made active with $\overline{CS}$, indicates that the host is executing a read operation from the Am95C60. The I/O port to be read is determined by the two address bits. In fly-by DMA operations (when $\overline{ACKD}$ is active) an active level on this pin will cause a write to the Block Input FIFO (BIF).

$\overline{WR}$ Write (Input)

This line, when made active with $\overline{CS}$, indicates that the host is executing a write operation to the Am95C60. The I/O port that will be written into is determined by the two address bits. In fly-by DMA operations (when $\overline{ACKD}$ is active) an active level on this pin will cause a read from the Block Output FIFO (BOF).

$\overline{CS}$ Chip Select (Input)

This line is used in conjunction with $\overline{RD}$ and $\overline{WR}$ to select the Am95C60 for an I/O operation.

A0–A1 Address 0, Address 1 (Input)

These two lines are used with $\overline{CS}$ to specify the I/O port to be written or read in an I/O operation. See Chapter 3, Section 3.2 for port assignments.

FREQ Instruction FIFO Request (Output, Open Drain)

This is used by the Am95C60 to request instructions in a system with a DMA controller. This is an open drain signal. The Am95C60 can pull this down but not up. An external resistor must be provided in the system to pull this up. This signal is pulled down when the Am95C60 is NOT making a request. In a system with multiple Am95C60s, these pins are tied together. As long as any Am95C60 is not ready to request more instructions the node will be LOW. When the last Am95C60 becomes ready the node will go HIGH (due to the resistor).

If FREQ is active, there is room in the FIFO for at least one word.

DREQ Data FIFO Request (Output, Open Drain)

This is used by the Am95C60 to request that an Input Block or Output Block instruction continue. This is an open drain signal. The Am95C60 can pull this down but not up. An external resistor must be provided in the system to pull this up. This signal is pulled down when the Am95C60 is NOT making a request. In a system with multiple Am95C60s, these pins are tied together. As long as any Am95C60 is not ready to continue with the I/O Block the node will be LOW. When the last Am95C60 becomes ready the node will go HIGH (due to the resistor).

If DREQ is active for an Input Block, there is room in the FIFO for at least one word. If DREQ is active for an Output Block, there is at least one word available in the FIFO.

$\overline{\text{ACKD}}$ Acknowledge DMA (Input)

This line may be driven LOW by a DMA controller that is capable of fly-by operation. When this line is LOW, the Am95C60 will either write to the Block Input FIFO or read from the Block Output FIFO regardless of A0 and A1. $\overline{\text{CS}}$ and $\overline{\text{ACKD}}$ must not be made active simultaneously. A read will take place if $\overline{\text{WR}}$ is active; a write will take place if $\overline{\text{RD}}$ is active. There is no analogous pin for instruction FIFO operations.

$\overline{\text{EDE}}$ External Driver Enable (Output)

This line is used to enable an external data buffer. That will be required in most systems (See the DC characteristics in the data sheet for current drive capability of the Am95C60).

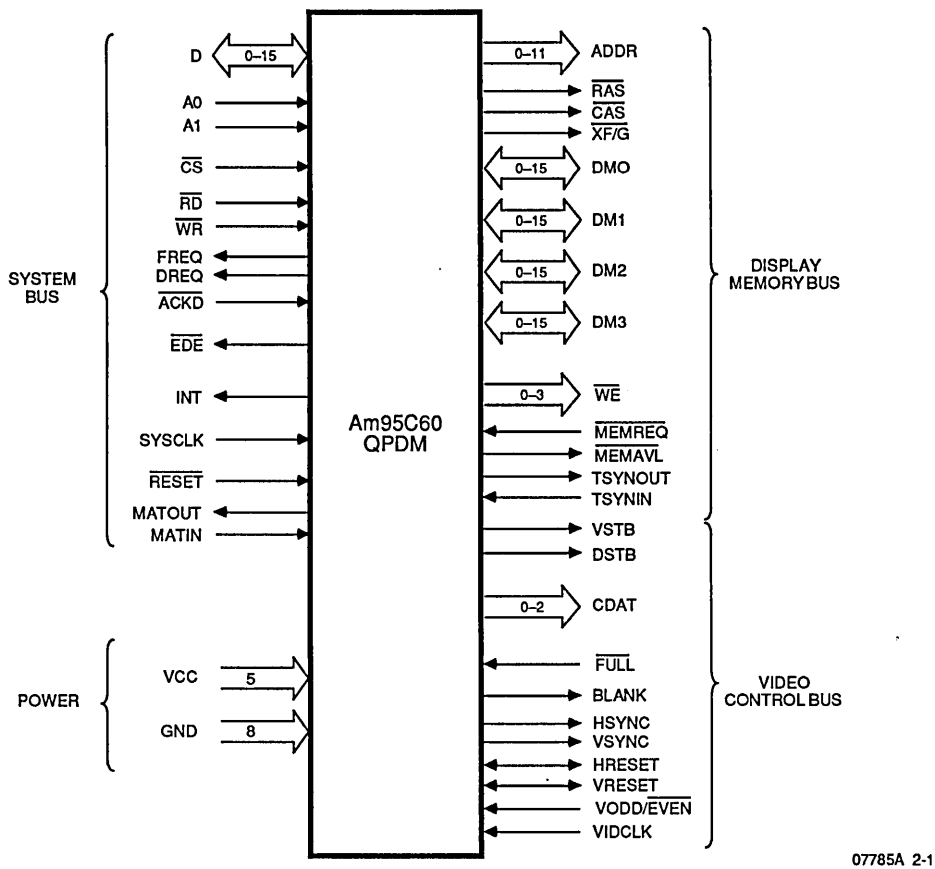


Figure 2-1 Am95C60 Logic Diagram

The logic equation for $\overline{EDE}$ is:

$$\begin{aligned}\overline{EDE} = {} & \overline{CS} \cdot \overline{A0} \cdot \overline{A1} && \text{;Status Register}\\ & + \overline{CS} \cdot A1 \cdot \overline{A0} \cdot ABR && \text{;Block I/O FIFO}\\ & + \overline{CS} \cdot A1 \cdot \overline{A0} && \text{;Register Pointer}\\ & + \overline{CS} \cdot A1 \cdot A0 && \text{;Register}\\ & + \overline{ACKD} \cdot ABR && \text{;Fly-by DMA}\end{aligned}$$

Where ABR is the "OR" of the four activity bits.

In most systems, $\overline{EDE}$ cannot be used by itself to enable the data buffers as this will result in a bus crash. The system designer will have to combine $\overline{EDE}$ with appropriate timing to avoid the crash.

INT Interrupt Request (Output)

This line is used by the Am95C60 to notify the host that an interrupt is pending. The nature of the interrupt can be determined by examining the contents of the status register. Interrupts are discussed in detail in Chapter 3.

SYSCLK System Clock (Input)

This input is the main clock to the Am95C60. All internal timing in the Am95C60 is controlled by this clock. Instruction timing is given in terms of this

---

Table 2-1 Am95C60 Pinouts (Pin-side View)

	A	B	C	D	E	F	G	H	J	K	L	M	N	P	R	
1	DM314	DN313	DM311	DM310	DM307	DM305	DM303	DM301	TEST	DM012	DM010	DM009	DM007	DM005	DM001	1
2	DM200	DM315	DM312	DM308	DM304	DM302	DM300	MATOUT	DM014	DM013	DM011	DM008	DM002	DM003	DM000	2
3	DM202	DM204	DM201	DM309	GND	DM306	VCC	MATIN	VCC	DM015	GND	DM006	DM004	DM104	DM100	3
4	DM206	DM207	DM205	NC									DM101	DM103	DM102	4
5	DM209	DM211	DM203										DM107	DM106	DM105	5
6	DM213	DM208	GND										GND	DM108	DM109	6
7	DM215	DM212	DM210										DM114	DM110	DM111	7
8	DM214	ADDR10	ADDR11										DM112	DM113	DM115	8
9	ADDR8	ADDR9	GND										GND	D14	D15	9
10	ADDR6	ADDR5	VCC										D13	D12	D11	10
11	ADDR4	ADDR7	ADDR1										GND	D10	D9	11
12	ADDR3	ADDR2	GND										D8	D7	D6	12
13	$\overline{FULL}$	ADDR0	$\overline{RAS}$	$\overline{WE0}$	$\overline{CAS}$	$\overline{XF/G}$	VCC	VODD/EVEN	DREQ	INT	VCC	$\overline{WR}$	D5	D3	D4	13
14	VSTB	CDAT2	CDAT0	$\overline{WE1}$	TSYNIN	$\overline{MEMVAL}$	VSYNC	BLANK	HRESET	SYSCKL	$\overline{RESET}$	$\overline{RD}$	$\overline{ACKD}$	D1	D2	14
15	DSTB	CDAT1	$\overline{WE3}$	$\overline{WE2}$	TSYNOUT	$\overline{MEMREQ}$	HSYNC	VRESET	VIDCLK	FREQ	A0	A1	$\overline{CS}$	$\overline{EDE}$	D0	15
	A	B	C	D	E	F	G	H	J	K	L	M	N	P	R	

clock. The maximum frequency for the system clock is 20 MHz and will be less for slower parts.

The microengine as well as the RAM interface is driven by SYSCLK. SYSCLK, VIDCLK and transactions at the system interface are all asynchronous with respect to each other.

SYSCLK should be as close to a square wave as possible. We recommend SYSCLK be generated by dividing a 2x clock with a flip-flop. In any case, parameters 108 and 109 must be met.

$\overline{\text{RESET}}$ Reset (Input)

This input unconditionally suspends any activity in the Am95C60 and forces it to a known state. The specifics are listed in Chapter 3.

MATOUT Match Out (Output)

This pin is used in multiple Am95C60 systems to exchange color matching information. Whenever an Am95C60 is not reporting a match, it will drive this pin LOW. When a match is found it will drive this pin HIGH. These pins must all be ANDed externally with the result driving all the MATIN pins. In addition to color matching, these pins are used to maintain instruction sychronization. The operation of the MATOUT lines is transparent to the user.

MATIN Match In (Input)

This pin must be driven by the ANDed MATOUT lines. In the case of a single Am95C60 system, this pin may be driven directly with MATOUT.

TEST

Test Input must be grounded.

2.1.2 Display Memory Bus

The display memory bus includes signals to completely control four bit planes. In a minimum system, very little other than buffers must be provided externally; in a larger system it will be necessary to decode $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ to implement bank selection.

There is a 12-bit multiplexed address bus. Typically eight or nine of these bits are buffered into the array while one to three bits are decoded to steer $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ to the correct bank within the array.

$\overline{\text{RAS}}$, $\overline{\text{CAS}}$, and $\overline{\text{XF/G}}$ can be buffered into the array in a minimum system and would be steered with high order address bits in a large system where bank selection is required.

One of the most distinguishing features of the Am95C60 is the memory data bus. There are 16 data lines for each of the four bit planes. This large degree of parallelism allows for a very fast BitBLT as well as some other inherently parallel operations. There are four write enables, one for each bit plane.

Two lines, $\overline{\text{MEMREQ}}$ and $\overline{\text{MEMAVL}}$, are used to allow an external device to gain control of the memory array. The external device has to be able to generate all the control and timing into the array.

Two additional lines, TSYNOUT and TSYNIN, in the display memory bus group are used to synchronize Am95C60s to each other in a multiple Am95C60 system.

The following paragraphs are the formal English-language descriptions of the display memory bus pins on the Am95C60. The pin numbers are in Table 2-1. All signal pins on the Am95C60 have TTL I/O levels.

DISPLAY MEMORY BUS PINS

ADDR0–ADDR11 Address 0–Address 11 (Output)

These 12 lines contain the multiplexed address used for accessing the display memory. They contain row and column addresses and bank select bits. See Chapter 12 for information regarding the specific meaning of each bit for various memory configurations. These lines must be buffered.

$\overline{\text{RAS}}$ Row Address Strobe (Output)

This line is used to strobe the row address into the array. This line will have to be buffered. In a large system the high order ADDR bits are decoded to steer $\overline{\text{RAS}}$ to the correct bank of RAMs.

$\overline{\text{CAS}}$ Column Address Strobe (Output)

This line is used to strobe the column address into the array. This line will have to be buffered. In a large system the high order ADDR bits are decoded to steer $\overline{\text{CAS}}$ to the correct bank of RAMs.

$\overline{XF}$/G Transfer/Output Enable (Output)

This line is used to indicate a transfer cycle and to enable the output buffers of the appropriate RAMs during a read cycle. This line must be buffered.

DM000–DM315 Display Memory Data Bus (Bidirectional)

These 64 lines are connected directly to the data I/O pins of the dynamic port of the VRAM chips. There are 16 lines for each of the four bit planes. The naming convention for these lines is DMzxx where z is the plane number (0–3) and xx is the data line number (0–15). The relationship between data line numbers and the actual x addresses of pixels is:

DMz15 is connected to the pixels whose x addresses end in 0000. DMz14 is connected to the pixels whose x addresses end in 0001. And so on, until DMz00 is connected to the pixels whose x addresses end in 111.

Therefore, the serializer should be arranged so that DMz15 is displayed to the left of DMz14. That is, in a raster where the scan is from left-to-right, DMz15 should come out of the serializer first.

$\overline{WE0}$–$\overline{WE3}$ Write Enable 0 – Write Enable 3 (Output)

These four lines indicate whether a random RAM cycle is a read or a write. There is one line for each of the four bit planes. These lines must be buffered.

$\overline{MEMREQ}$ Memory Request (Input)

This line is used by an external controller to request access to the display memory bus. When the Am95C60 can relinquish control of the bus, it will respond by making $\overline{MEMAVL}$ active. The external controller must be capable of generating a multiplexed address and the necessary control signals.

$\overline{MEMAVL}$ Memory Available (Output)

This line is used to indicate to an external controller that the Am95C60 has relinquished control of the memory bus. It does so by making the data lines inputs. The Am95C60 continues to drive its address and control lines; the external buffers must be three-stated. The Am95C60 can make $\overline{MEMAVL}$ not active and regain control of the bus even if $\overline{MEMREQ}$ does not go inactive.

TSYNOUT Timing Synchronization Output (Output)

This line is used to synchronize display memory activities. The TSYNOUT lines of all Am95C60s in a system must be externally ANDed. The result of the AND must be tied to all the TSYNIN pins. In a single Am95C60 system, TSYNOUT may be connected directly to TSYNIN.

TSYNIN Timing Synchronization In (Input)

This pin must be connected to the AND of all TSYNOUT pins. In a single Am95C60 system, TSYNIN may be connected directly to TSYNOUT.

2.1.3 Video Control Bus

The signals in this group are related to video timing and to VRAM/VDAF control.

BLANK, HSYNC and VSYNC are used to control the CRT monitor. HRESET and VRESET are bi-directional signals used to synchronize Am95C60s in a multiple Am95C60 system or to slave an Am95C60 to an external raster. VODD/$\overline{EVEN}$ is used in an interlaced system to force slave Am95C60s to generate the correct field. VIDCLK is the clock signal from which the video timing signals are generated.

The remaining signals in this group are used in a system with hardware windows to control the VDAFs. VSTB is used to clock data out of the VRAMs, and DSTB is used to strobe data into the VDAFs. The CDAT lines are used to control data assembly in the VDAFs, and $\overline{FULL}$ is used to stop the data assembly process when the FIFOs in the VDAFs are full.

The following paragraphs are the formal English-language descriptions of the video control bus pins on the Am95C60. The pin numbers are in TABLE 2-1. All signal pins on the Am95C60 have TTL I/O levels.

VIDEO CONTROL BUS PINS

VSTB Video Strobe (Output)

This signal is used to shift the video data out of the VRAMs in a system that uses VDAFs. The Am95C60 expects that 16 bits of data per bit plane will be shifted with each positive going edge of this clock. VSTB also indicates which half of the 16-bit word is to be clocked into the 8-bit VDAF when DSTB rises. A LOW selects bits 0-7, a HIGH

selects 8-15. This signal must be buffered. In a system that does not use VDAFs, this signal is not used.

DSTB Data Strobe (Output)

This signal is used to clock the video data into the VDAFs. The Am95C60 expects that 8 bits will be clocked into each VDAF with each positive going edge of the clock. This signal is not used in a system that does not use VDAFs. This signal must be buffered. DSTB normally toggles twice as fast as VSTB.

CDAT0–CDAT2 Control Data 0 – Control Data 2 (Output)

These three lines are used to control the assembly process in the VDAFs. During a transfer cycle these lines contain the bit number of the first bit that is to be displayed. At each rising edge of DSTB they will contain the number of valid bits in the byte being loaded. These lines must be buffered.

$\overline{\text{FULL}}$ FIFO Full (Input)

This line signals the Am95C60 that the VDAF FIFO is full and that the data assembly process must temporarily cease. The Am95C60 will respond by transferring up to three more bytes and then stopping. When $\overline{\text{FULL}}$ returns to the nonactive state the Am95C60 will begin transferring data within three SYSCLK cycles.

BLANK Video Blank (Output)

This signal indicates that the video is to be blanked on the screen. BLANK must be synchronized to the Dot Clock in order to insure proper timing. This signal is forced active when the Am95C60 is reset.

HSYNC Horizontal Synchronization (Output)

This signal is intended to initiate horizontal retrace of the CRT electron beam. The timing of this signal is programmable. This signal is forced inactive when the Am95C60 is reset.

VSYNC Vertical Synchronization (Output)

This signal is intended to initiate vertical retrace of the CRT electron beam. The timing of this signal is programmable. This signal is forced inactive when the Am95C60 is reset.

HRESET Horizontal Reset (Bidirectional)

This signal is an output for horizontal video masters and an input for horizontal video slaves. It is used to force horizontal synchronization.

VRESET Vertical Reset (Bidirectional)

This signal is an output for vertical video masters and an input for vertical video slaves. It is used to force vertical synchronization.

VODD/$\overline{\text{EVEN}}$ Vertical Odd/Even (Input)

This signal may be used in an interlaced display system to indicate whether an even field or odd field should be generated.

VIDCLK Video Clock (Input)

This clock is used for generating video timing. This is counted to generate the horizontal timing; HSYNC is counted to generate vertical timing. The maximum frequency of this clock is 15 MHz and will be less for slower parts.

2.1.4 Power Bus

The Am95C60 uses +5 V and ground only. There are five +5 pins and eight ground pins.

POWER BUS PINS

VCC Power (Input)

There are five VCC pins. These must all be connected to a +5 V ±5% supply. Power must be adequately bypassed to ground as close to the pins as possible.

GND Ground (Input)

There are eight GND pins. These must all be connected to the power return. Ground must be distributed throughout the system with a solid layer in the PWB to obtain best results.

CHAPTER 3

HOST–Am95C60 COMMUNICATIONS

This chapter discusses the very lowest level software interface between the Am95C60 and the host. This chapter should be of interest to the person building the software interface at the device driver level.

3.1 RESET FUNCTION

The Am95C60 is reset to a known state when the RESET pin is made active. RESET should be activated as part of the system power-on sequence; this insures that indeterminate sync pulses are not sent to the monitor.

Table 3-1 lists the functions affected by a reset. Any function not mentioned in this list is indeterminate following a reset.

Table 3-1 Functions Affected by a Reset

Function	Condition Following Reset
Video Timing Enable	Disabled
Video Refresh Enable	Disabled
Display Memory Refresh	Disabled
8- or 16-bit System Bus	16-bit System Bus
Program Mode	Not in Program Mode
Instruction FIFO	Empty
Block Input FIFO	Empty
Block Output FIFO	Empty

The reset that results from a write to the reset register (address 27) is the same except that Video Timing is not altered or disabled.

No instructions should be sent to the Am95C60 for at least 100 SYSCLK cycles following any reset.

3.2 I/O PORTS

The Am95C60 is intended to operate as a slave peripheral on the system bus. Typically, but not necessarily, it is placed in the I/O address space since it occupies only four locations.

When the host accesses the Am95C60 as a slave peripheral, the two address lines and $\overline{RD}$ and $\overline{WR}$ are decoded to select one of eight functions. These are indicated in Table 3-2.

Table 3-2 I/O Port Functions

A1	A0	Write Function	Read Function
0	0	Write Instruction FIFO	Read Status Register
0	1	Write Block In FIFO (BIF)	Read Block Out FIFO (BOF)
1	0	Write Address Register	Read Address Register
1	1	Write Register	Read Register

3.2.1 Write Instruction FIFO

Whenever a write is performed to the Am95C60 and A1,A0 are (0,0), the word on the system data bus is written into the instruction FIFO. If the instruction FIFO is already full, the word will be ignored with no notification and some words already in the FIFO may be lost. If the instruction FIFO is not full, the instruction word will be queued in the FIFO following the last one that was written.

Writes to the instruction FIFO may be safely synchronized by using the FREQ (bit 14) status bit. If this bit is a one, there is room in the FIFO for at least one word. In the case where DMA is being used, the FREQ pin provides synchronization to the DMA controller.

It is worth observing that the instruction word will not usually be taken from the FIFO immediately; it has to wait its turn. Short of forcing a reset, there is no way to unconditionally clear the instruction FIFO.

3.2.2 Read Status Register

Whenever a read is issued to the Am95C60 and A1,A0 are (0,0), the status register will be read out onto the system data bus. This operation may be performed at any time without affecting the Am95C60 operating state. Since the internal microengine is not involved, a status read will not cause any performance degradation.

The status register bits are defined in Table 3-3. (See the discussion of interrupts in this chapter for definitions of the interrupt requests.)

3.2.3 Write Block In FIFO (BIF)

When a write is issued to the Am95C60 and A1,A0 are (0,1), the word on the data bus enters the Block Input FIFO. If the FIFO is full, the word is lost with no notification being given to the host. These writes may be safely synchronized using the DREQ (bit 15) status bit. If this bit is a one, there is room for at least two words in the FIFO. In the case where DMA is being used, the DREQ pin provides synchronization to the DMA controller.

3.2.4 Read Block Out FIFO (BOF) – (4 words)

When a read is issued to the Am95C60 and A1,A0 are (0,1), the next word in the Block Output FIFO is placed on the data bus. If the FIFO is empty, indeterminate data will be placed on the bus with no notification being given to the host. These reads may be safely synchronized using the DREQ (bit 15) status bit. If this bit is a one, there are at least two words in the FIFO. In the case where DMA is being used, the DREQ pin provides synchronization to the DMA controller.

3.2.5 Write Address Register

When a write is issued to the Am95C60 and A1,A0 are (1,0), the contents of the data bus are written into the address register. This address is used during subsequent Write Register or Read Register operations to select the actual internal register that is to be involved in the transfer. The register addresses are given in Chapter 4. The address register may be written at any time without affecting the operation of the Am95C60.

3.2.6 Read Address Register

When a read is issued to the Am95C60 and A1,A0 are (1,0), the contents of the address register are placed on the data bus. This is the value that was last loaded with a Write Address Register operation. The address register may be read at any time without affecting the operation of the Am95C60.

Interrupt handling is simplified somewhat by this ability to read the address register. The interrupt service routine can read and save the contents of the address register. It can then set the address register as required to process the interruption and finally restore the address register before it exits. The interruption is thereby made invisible regarding the contents of the address register. When 8-bit bus mode has been selected, this is complicated by the requirement that register accesses must not be split.

3.2.7 Write Register

When a write is issued to the Am95C60 and A1,A0 are (1,1), the contents of the data bus are written into the register selected by the last Write Address

Table 3-3 Status Register Bit Definition

BIT	Mnemonic	Description *(A "1" has the following meaning:)*
0	INT	Any interrupt request is active *and* the corresponding enable bit is set.
1	IDLEI	The IDLE Interrupt request is set.
2	SOFI	The Stack Overflow Interrupt request is set.
3	DMXI	The DMX Interrupt request is set.
4	CLIPI	The Clipping Interrupt request is set.
5	FRAMEI	The Frame Interrupt request is set.
6	FREQI	The FIFO Request Interrupt request is set.
7	DREQI	The Block I/O Request Interrupt request is set.
8	VBLKI	The Vertical Blank Interrupt request is set.
9	SWI	The Software Interrupt request is set.
10	IDLE	The Am95C60 actually is idle. This status bit is cleared by the Am95C60 automatically when it becomes not idle.
11	HRESET	The HRESET pin is HIGH.
12	SPARE	This bit is indeterminate.
13	PCKD	The Am95C60 has been instructed to write within the picking region and picking was enabled. If picking was not enabled at the time the write took place, this bit would not have been set. This status bit is cleared with an Interrupt Acknowledge with bit 9 set.
14	FREQ	The FREQ pin is HIGH.
15	DREQ	The DREQ pin is HIGH.

Register operation. Each register in the Am95C60 is described in Chapter 4, including any restrictions regarding the timing of this operation for that register.

3.2.8 Read Register

When a read is issued to the Am95C60 and A1,A0 are (1,1), the contents of the selected register appear on the data bus if the register has read-back capability. The register will have been selected by the last Write Address Register operation. If the register does not have read-back capability or it is not implemented, indeterminate data will be placed on the bus.

3.3 DMA FACILITIES

The Am95C60 can be used with a DMA controller. This minimizes the load on the host when large amounts of data are being transferred with Block I/O instructions or where long strings of instructions are being sent to the Am95C60.

When DMA operations are being done to the instruction FIFO, flow-through mode must be used. The DMA controller logic must generate the same signals and timing as the CPU. The Am95C60 does not know whether the operation was initiated from the CPU or from the DMA controller.

When DMA operations are being done to the Block Input FIFO or the Block Output FIFO, either flow-through mode or fly-by mode may be implemented. If flow-through mode is chosen, the DMA controller logic must generate the same signals and timing as the CPU.

If fly-by mode is used, then the DMA controller makes $\overline{\text{ACKD}}$ active rather than chip select. The address bits will not be used and the meanings of $\overline{\text{RD}}$ and $\overline{\text{WR}}$ will be interchanged.

3.4 INTERRUPT FACILITIES

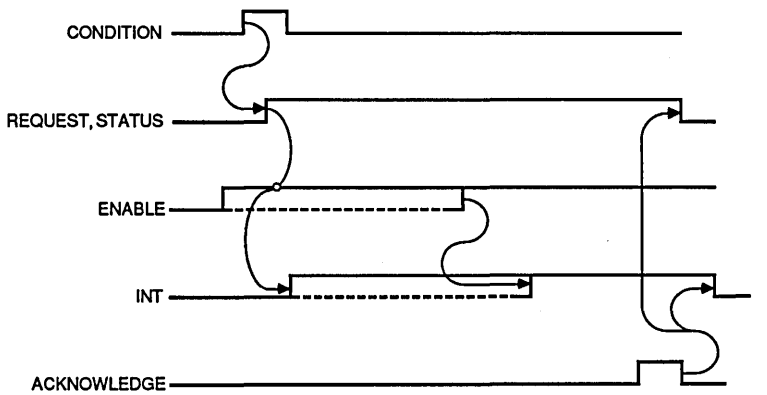
The Am95C60 provides an interrupt capability. This provides a method of reducing the load on the host by avoiding time consuming polling operations.

There are nine conditions in the Am95C60 that can cause interruptions. Each condition has a specific bit in the Interrupt Enable Register and the Interrupt Acknowledge Register, as well as the Status Register.

The Interrupt Enable Register is used to individually enable each of the conditions. The enable bit must be a 1 to allow the condition or to cause an interrupt.

Writes to the Interrupt Acknowledge Register are used to clear interrupt requests. A 1 must be programmed in the word for each interrupt that is to be cleared. A 0 bit has no affect.

Figure 3-1 and Figure 3-2 illustrate the interrupt scheme. When a specific condition occurs (e.g., the Am95C60 becomes idle), a latch is set that remembers the condition. The details of the condition vary from interrupt to interrupt; they are discussed in Table 3-4. The output of this latch is called the interrupt request. This is what goes to the status register. The interrupt enable need not be active for the request to appear in the status register.



07785A 3-1

Figure 3-1 Interrupt Timing

The setting of the latch takes place regardless of whether the corresponding enable bit is set. The latch remains set—even if the condition becomes no longer true—until explicitly cleared with an interrupt acknowledge. If the enable bit had been set prior to the latch being set, the interrupt request to host (INT) becomes active immediately. If the enable bit is not set, the condition cannot generate INT until the enable bit is set.

When a write is directed to the Interrupt Acknowledge Register with the appropriate bit set, the latch is cleared if the condition is no longer active. If the condition is still active, the Interrupt Acknowledge does nothing. If the Interrupt Acknowledge does not clear all the requests that are enabled, INT will remain active.

Table 3-4 describes the interrupt conditions that can occur in the Am95C60. The BIT number is the bit position within the registers.

3.5 CONSIDERATIONS INVOLVING MULTIPLE Am95C60s

Many systems require more than four bit planes. Multiple Am95C60s can be easily cascaded. There are two major considerations.

It is necessary to provide Chip Select ($\overline{CS}$) decoding so that Am95C60s can be selected both individually *and* as a group. When the Am95C60s are addressed as a group, this is called broadcasting. Table 3-5 summarizes the cases.

Table 3-5 Chip Select Decoding

Operation	Individual	Broadcast
Write Register	Video Mode	All other registers
Read Status Register	Always	Never
Send Instructions	Set QPDM Position	All other instructions
Transfer Data	Images	Fonts
Interrupt Acknowledge	Never	Always

When multiple Am95C60s are controlling the bit planes of a display, it is necessary to insure that they remain in synchronization (that is, all Am95C60s are executing the same instruction). This is accomplished using the MATOUT and MATIN pins and the TSYNOUT and TSYNIN pins.

The MATOUT and MATIN pins are used to exchange color match status and are also used to guarantee that the Am95C60s all begin execution of each instruction simultaneously. The TSYNOUT and TSYNIN pins are used to synchronize display memory activities. The actual operation of these pins is invisible to the user.

3.6 EIGHT-BIT INTERFACE

The Am95C60 can be used in a system with an 8-bit data bus. When the Am95C60 is reset it enters 16-bit mode. It will remain in 16-bit mode until (and unless) a write is directed to the System Bus Width Register. When the write has taken place, the Am95C60 enters 8-bit bus mode and remains there until a reset takes place.

When the Am95C60 is in 8-bit mode, system data bus bits 7–0 are used and data bits 15–8 are not used. In the case of a write, the data on bits 15–8 are ignored. In the case of a read, the data placed on bits 15–8 are indeterminate.

When the Am95C60 is in 8-bit mode, each read or write to a register requires two transfers (even if the register actually contains eight or fewer bits), not counting the write to the address register. If the data written to the 8-Bit Bus Byte Order Register was a 0, the high order byte (register bits 15–8) is transferred during the first operation and the low order byte (register bits 7–0) is transferred during the second operation. If the data written to the 8-Bit Bus Byte Order Register was a 1, the low byte is transferred first and the high byte is transferred second. In either case both bytes are transferred on system data bus lines 7–0.

Care must be taken to insure that the two transfers are not split by another access. Suppose, for example, an interrupt takes place after the high byte is written and the interrupt service routine writes to a register. The high byte written in the interrupt service routine will be taken by the Am95C60 to be the low byte of the interrupted sequence, and matters will just get worse from then on. In general, it will be necessary to disable interrupts while transferring the two bytes in a complete register transfer. This consideration applies to register reads as well as writes. It is also necessary to assure that the transfer does not get split by a DMA operation. Suppressing interrupts and DMA request is generally handled in hardware.

When using DMA in an 8-bit system, it is necessary to program the DMA counts to twice the value that would be used in a 16-bit system. The bytes must be ordered in memory consistently with the high byte first or low byte first programming.

Observe the data hold time for write cycles is dissimilar between 8-bit and 16-bit modes.

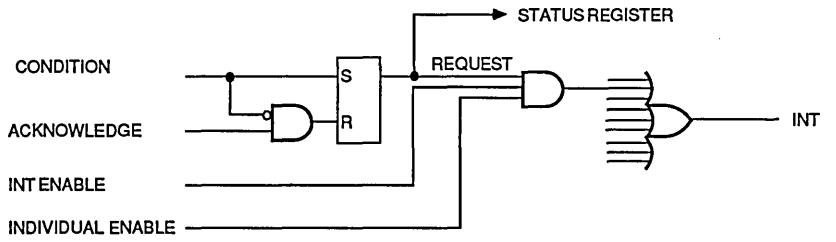


Figure 3-2 Interrupt Logic

Table 3-4 Am95C60 Interrupt Conditions

BIT	Mnemonic	Description of Condition
0	INTI	The logical "OR" of the nine status bits.
1	IDLEI	The Am95C60 has attempted to fetch an instruction and found the instruction FIFO empty. The request remains active until cleared with an interrupt acknowledge. The condition remains true until the Am95C60 is no longer idle.
2	SOFI	The stack has overflowed. The instruction whose execution led to the overflow was not completed. The Am95C60 continues to fetch and execute instructions. The request remains active until cleared with an interrupt acknowledge. The condition remains true until the stack boundaries are loaded again (see Set Stack Boundaries in the Instruction Set of Chapter 14).
3	DMXI	The Am95C60 was instructed to calculate an address outside the maximum (4096 x 4096) memory size when drawing. The instruction was terminated. The Am95C60 continues to fetch and execute instructions. The request remains active until cleared with an interrupt acknowledge. The condition remains active until a write takes place that is inside the maximum (4096 x 4096) display memory.
4	CLIPI	The Am95C60 was instructed to draw a pixel outside the clipping region with clipping enabled. The instruction was not terminated but the write did not take place. The request remains active until cleared with an interrupt acknowledge. The condition remains active until a write takes place that is inside the clipping region.
5	FRAMEI	The registers describing the screen and window parameters can be safely reloaded (without disrupting the screen). The request is raised after the register contents have been transferred into internal counters for the next frame and remains active until cleared with an interrupt acknowledge.
6	FREQI	The FREQ pin has changed from inactive to active (because the instruction FIFO has become half empty). The request remains active until cleared with an interrupt acknowledge. This condition is active when the Am95C60 is reset and remains active until the instruction FIFO is completely filled for the first time. Thereafter, this condition will be active whenever the instruction FIFO is less than half full. Observe the condition is latched.
7	DREQI	The DREQ pin has changed from inactive to active. The request remains active until cleared with an interrupt acknowledge. The condition is true whenever there are at least two words in the output FIFO and an Output Block is in progress or whenever there is room for at least two words in the input FIFO and an Input Block instruction is in progress.
8	VBLKI	Vertical Blank has gone active in the Am95C60. The request remains active until cleared with an interrupt acknowledge. The condition is true whenever Vertical Blank is active.
9	SWI	A SIGNAL instruction has been executed. The request remains active until cleared with an interrupt acknowledge. The condition is a pulse generated by the execution of the instruction.

CHAPTER 4

REGISTER SET

The Am95C60 contains a number of registers that must be programmed by the host. These registers are described in the following sections. In the tables that follow, the address of each register is presented with the number of bits that are implemented. The address is given in decimal. The register name is preceded by a mnemonic. Unused bits in all registers must be programmed to zeroes as shown in Section 4.7.

4.1 VISIBLE SCREEN COORDINATE REGISTERS

These four registers specify the rectangular area of display memory that is to appear on the screen. Two X,Y pairs are required. The Start pair defines the upper-left corner of the screen. The Terminate pair defines the lower-right corner of the screen, and is one pixel past the end of the screen, in both "X" & "Y". These are absolute addresses with respect to the display memory origin (0,0). The pixels defined by the Start address will be visible on the screen. If the values do not make sense (e.g., Terminate is less than Start), the results are indeterminate.

The entire display can be scrolled and panned by changing the contents of these registers. These registers should be changed only at FRAMEI time. These registers are readable.

The relationship between this set of registers and the display memory is shown in Figure 4-1.

Screen X Start and Screen X Terminate must be integer multiples of 16 in a system without VDAFs.

The number of scan lines specified in the Visible Screen Coordinate Registers (SYT-SYS) must equal the number of scan lines specified in the Vertical Active Registers.

Table 4-1 Visible Screen Coordinate Registers

Register Name		Address	Bits
SXS	Screen X Start	1	12
SYS	Screen Y Start	2	12
SXT	Screen X Terminate	3	12
SYT	Screen Y Terminate	4	12

4.2 WINDOW CONTROL REGISTERS

The six Window Control Registers specify the position and size of the hardware window. Four of these registers are called Window Apparent Registers, and the other two registers are called the Window Real Registers. See Chapter 11 for more information on the use of, and restrictions regarding, the window.

The four Window Apparent Registers specify the Start and Termination of the window as it appears on the screen. These are X,Y pairs and are referenced to the upper-left corner of the display memory. The two Window Real Registers specify the upper-left corner of the area of display memory which is to appear in the window. It is unnecessary to specify a Window Real Terminate since the size of the window is set by the Window Apparent Registers.

The relationship between this set of registers and the display memory is shown in Figure 4-1.

The position (and size) of the window on the screen can be modified by changing the contents of the Window Apparent Registers. This would be done, for example, to drag an object on the screen. The origin of the image which appears in the window can be changed by modifying the contents of the Window Real Registers. This would be done, for example, to scroll or pan within the window.

These six registers should be changed only at FRAMEI time. They are readable. If the contents of these registers do not make any sense (e.g., the window terminates before it begins), the results are indeterminate.

If no hardware window is desired, the Window Apparent Start Registers should be programmed to a point outside the visible screen.

Table 4-2 Window Control Registers

Register Name		Address	Bits
WAXS	Window Apparent X Start	14	12
WAYS	Window Apparent Y Start	15	12
WAXT	Window Apparent X Terminate	16	12
WAYT	Window Apparent Y Terminate	17	12
WRX	Window Real X	18	12
WRY	Window Real Y	19	12

4.3 VIDEO TIMING CONTROL REGISTERS

The contents of this group of nine registers define the raster. These values are chosen according to the requirements of the monitor. Normally, these registers are programmed when the Am95C60 is initialized and not changed thereafter. It is safest to disable video timing prior to changing these registers. This precaution prevents sending incorrect sync pulses to the monitor. This group of registers is not readable.

4.3.1 Horizontal Timing Registers

The horizontal timing parameters are programmed in terms of VIDCLK cycles. Depending on how the system designer implements the timing, there are 8, 16, or 32 pixels for each VIDCLK cycle.

Figure 4-2 may be used as an aid to understanding the relationship between the register values and the actual timing. The top waveform conforms to RS-343A and shows composite video. The bottom traces show HBLANK (that exists only inside the Am95C60) and HSYNC. The following terms are those that monitor manufacturers often use:

Monitor Parameters		Description
HFP	Horizontal Front Porch	HBLANK rises to HSYNC rises
HSYNC	Horizontal Sync Width	HSYNC rises to HSYNC falls
HBP	Horizontal Back Porch	HSYNC falls to HBLANK falls
HAV	Horizontal Active Video	HBLANK falls to HBLANK rises

The following are the terms (i.e., registers) used in the Am95C60. They are defined using the preceding monitor terms.

Register Name		Description
HSYNC	Horizontal Sync Width	= HSYNC
HDEL	Horizontal Scan Delay	= HSYNC + HBP
HACT	Horizontal Active	= HAV
HTOT	Horizontal Total	= HFP + HSYNC + HBP + HAV

Table 4-3 Horizontal Timing Registers

Register Name		Address	Bits
HSYNC	Horizontal Sync Width	10	8
HDEL	Horizontal Scan Delay	11	7
HACT	Horizontal Active	12	8
HTOT	Horizontal Total Count	13	9

When the Am95C60 is programmed to be a horizontal video slave HTOT is unused. HSYNC must be programmed to produce a pulse width greater than the (active LOW) incoming signal on HRESET.

If HDEL is programmed to less than HSYNC, the sync signal will persist longer than blanking. This results in a negative horizontal back porch that is required for some monitors.

4.3.2 Vertical Timing Registers

The vertical timing registers, with the exception of VACT, are programmed in terms of half scan lines. VACT is programmed in terms of scan lines. Figure 4-3 may be used as an aid to understanding the relationship between the register values and the actual timing. The traces show VBLANK (that exists only inside the Am95C60) and VSYNC. The following are the terms that monitor manufacturers often use:

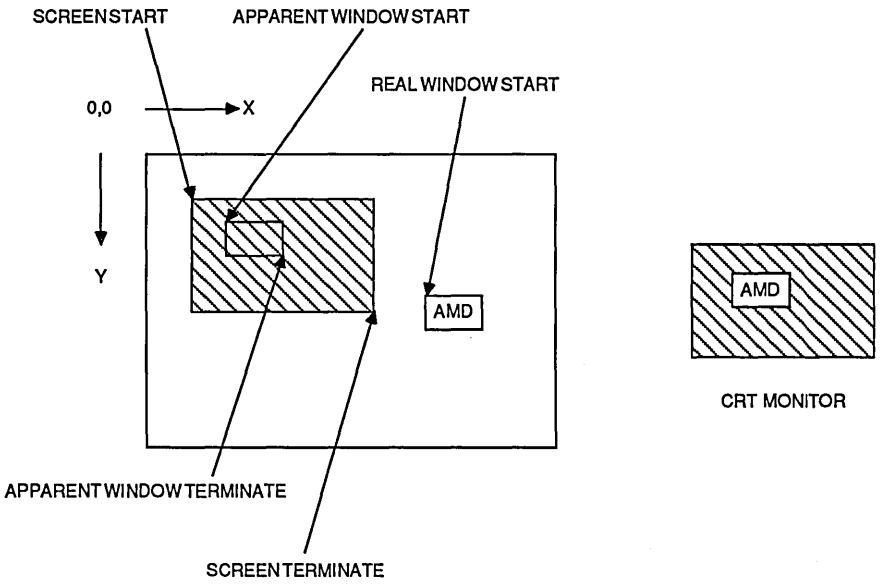
Monitor Parameters		Description
VFP	Vertical Front Porch	VBLANK rises to VSYNC rises
VSYNC	Vertical Sync Width	VSYNC rises to VSYNC falls
VBP	Vertical Back Porch	VSYNC falls to VBLANK falls
VAV	Vertical Active Video	VBLANK falls to VBLANK rises

The following are the terms (i.e., registers) used in the Am95C60. They are defined using the preceding monitor terms.

Register Name		Description
VSYNC	Vertical Sync Width	= VSYNC
VDELODD	Vertical Scan Delay Odd	= VSYNC + VBP
VDELEVEN	Vertical Scan Delay Even	= VSYNC + VBP
VACT	Vertical Active	= VAV
VTOT	Vertical Total	= VFP + VSYNC + VBP + VAV

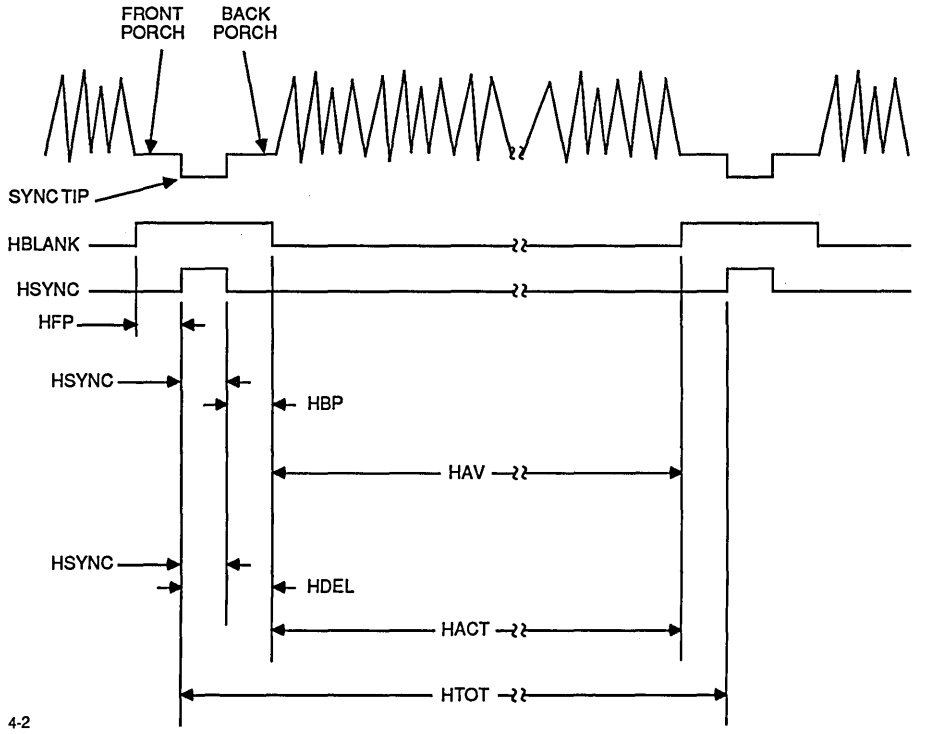
Table 4-4 Vertical Timing Registers

Register Name		Address	Bits
VSYNC	Vertical Sync Width	5	10
VDELODD	Vertical Scan Delay Odd	6	10
VDELEVEN	Vertical Scan Delay Even	7	10
VACT	Vertical Active	8	11
VTOT	Vertical Total	9	13



07785A 4-1

Figure 4-1 Screen and Window Registers



07785A 4-2

Figure 4-2 Horizontal Timing Nomenclature

When the Am95C60 is programmed to be a vertical video slave VTOT is unused. In addition, VSYNC must be programmed to produce a pulse width greater than the (active LOW) incoming signal on VRESET.

If VDEL is programmed to less than VSYNC, the sync signal will persist longer than blanking. This results in a negative vertical back porch that is required by some monitors.

The vertical timing register must be programmed with even or odd values according to the following chart. All vertical timing registers are programmed in terms of half scan lines except VACT which is programmed in terms of full scan lines.

Register	Interlaced	Noninterlaced
VSYNC	Even	Even
VDELODD	Even	Even
VDELEVEN	Odd	Don't Care
VACT	SYT-SYS	SYT-SYS
VTOT	Don't Care	Don't Care

VDELEVEN is not used when the Am95C60 is programmed for noninterlaced operation. If the Am95C60 is programmed for interlaced operation, VDELEVEN must be programmed with an odd number.

4.4 VIDEO CONTROL REGISTERS

The video control group comprises three registers. These registers are normally programmed once during initialization. These registers cannot be read back.

Register Name		Address	Bits
VMR	Video Mode Register	22	16
VTE	Video Timing Enable	28	1
VRE	Video Refresh Enable	29	1

4.4.1 Video Mode Register

The format of the video mode register is shown in Figure 4-4.

In a system containing multiple Am95C60s, one will be the timing master and the others will be timing slaves. If Horizontal Master is programmed to a 1, this Am95C60 is the Horizontal Timing Master; if it is a 0, this Am95C60 is a Horizontal Timing Slave.

The Vertical Master bit is analogous to the Horizontal Master bit. If it is a 1, the Am95C60 is the master; if it is a 0, the Am95C60 is a slave.

Bits 3 and 2 of the Video Mode Register define the display mode, as shown in the chart below:

Bit 3	Bit 2	Display Mode
0	0	Reserved, do not use
0	1	Interlaced (Odd/Even)
1	0	Noninterlaced
1	1	Repeat Field Interlaced

Bit 4 (VODD/$\overline{\text{EVEN}}$ Pin Enable) is used for interlaced mode only; if noninterlaced mode is selected, bit 4 must be programmed to 0. If interlaced operation is programmed VODD/$\overline{\text{EVEN}}$ Pin Enable establishes how the Am95C60 is to determine whether to generate an odd field or an even field. If VODD/$\overline{\text{EVEN}}$ Pin Enable is programmed to a 0, the Am95C60 will use an internal flip-flop; if VODD/$\overline{\text{EVEN}}$ Pin Enable is programmed to a 1, the Am95C60 will sample the VODD/$\overline{\text{EVEN}}$ pin prior to generating each frame. If VODD/$\overline{\text{EVEN}}$ Pin is a 0, the Am95C60 will generate an odd field first.

4.4.2 Video Timing Enable Register

The Video Timing Enable Register is used to enable or suppress the generation of sync and video. If bit 0 of this register is a 0, the Am95C60 will not generate sync or blank. If bit 0 is programmed to a 1, the Am95C60 will generate sync and blank pulses. This bit should not be programmed to a 1 until the timing registers have all been programmed.

When bit 0 of this register is programmed from a 1 to a 0, the Am95C60 will continue to generate sync and blank until the end of the current frame.

4.4.3 Video Refresh Enable

The Video Refresh Enable is used to enable or suppress transfer cycles. If bit 0 is programmed to a 0, transfer cycles will be suppressed allowing nearly exclusive use of the display memory bus for other purposes. If bit 0 is programmed from a 1 to a 0, the Am95C60 will complete the current frame.

VRE must be programmed from 0 to 1 during Vertical Blanking Time. This time can be determined by monitoring the VBLKI bit in the status register or by monitoring the VSYNC pin. If you use the VBLKI bit, you must first ensure it is reset by writing to the interrupt acknowledge register a value with bit 8 set. After you write to the interrupt acknowledge, make sure VBLKI is actually a zero.

4.5 DISPLAY RAM CONTROL REGISTERS

The display RAM control group has two members. They are programmed once during initialization.

These registers cannot be read back.

Register Name		Address	Bits
MMR	Memory Mode	23	16
DMMR	Dynamic Memory Refresh Rate	24	10

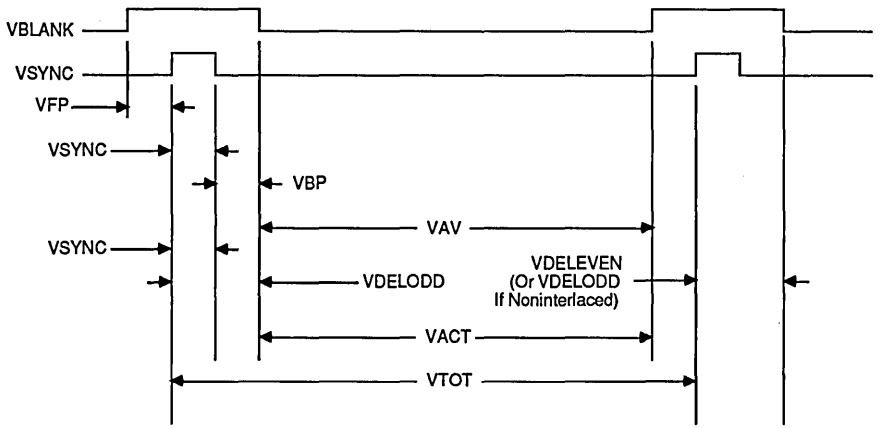
4.5.1 Memory Mode Register

The format of the memory mode register is shown in Figure 4-5.

Bit 3 of the memory mode register is used to indicate the relative priority of dynamic RAM refresh and external access to the display memory bus. External access is what happens when $\overline{\text{MEMAVL}}$ is active. If bit 3 is programmed to a 0, dynamic RAM refresh will have higher priority than external access to the display memory bus. If bit 3 is programmed to a 1, external access will have priority over dynamic RAM refresh. In this case, the external device must be able to guarantee that the VRAMs are refreshed.

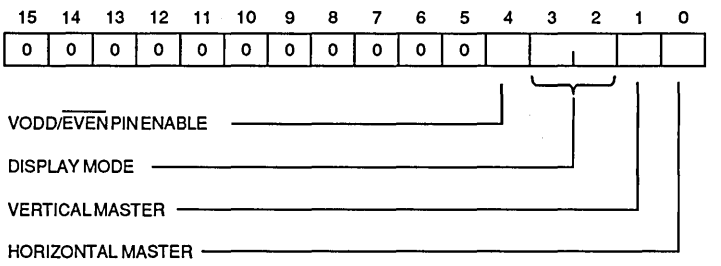
Bits 7 through 4 of the memory mode register indicate both the depth of the memory chips and the width of the display memory. The following chart lists the combinations. See Chapter 12 for Memory Configuration diagrams.

7 6 5 4	RAM Chip Depth	Display Memory Width	Comments
0 0 0 0	1 M	4 K	Configuration A
0 0 0 1	256 K	4 K	Configuration B
0 0 1 0	64 K	4 K	Configuration C
0 0 1 1	256 K	2 K	Configuration D
0 1 0 0	64 K	2K	Configuration E
0 1 0 1	—	—	Reserved
0 1 1 0	256 K	1 K	Configuration F
0 1 1 1	64 K	1 K	Configuration G
1 0 0 0	—	—	Reserved
1 0 0 1	—	—	Reserved
1 0 1 0	64 K	512	Configuration H
1 0 1 1	—	—	Reserved
1 1 0 0	—	—	Reserved
1 1 0 1	—	—	Reserved
1 1 1 0	—	—	Reserved
1 1 1 1	—	—	Reserved



07785A 4-3

Figure 4-3 Vertical Timing Nomenclature



07785A 4-4

Figure 4-4 Video Mode Register

4.5.2 Dynamic Memory Refresh Rate Register

The Dynamic Memory Refresh Rate Register is programmed with the number of SYSCLK cycles between dynamic RAM refresh cycles. The following equation may be used to calculate the value to be written into this register:

X= Refresh Period / SYSCLK period

If the refresh period is 16 µs (that is typical) and the SYSCLK period is 50 ns (20 MHz), the value for X is 320. Alternatively, if the SYSCLK period is 67 ns (15 MHz), the result is 240. Bit 9 must be programmed (a) to a 1 in order to enable refresh. If bit 9 is programmed to a 0, refresh is suppressed.

4.6 HOST–Am95C60 COMMUNICATIONS REGISTERS

The Host-Am95C60 Communications Registers group comprises five registers. These registers have all been described in Chapter 3.

Register Name		Address	Bits
IE	Interrupt Enable	26	16
IA	Interrupt Acknowledge	30	16
SWB	System Bus Width	59	0
RESET	Reset	27	0
BBO	8-Bit Bus Byte Order	31	1

4.6.1 Interrupt Enable Register

The Interrupt Enable Register is used to individually allow or disallow conditions to generate interrupt requests. The bits of this register are described in Chapter 3, Section 3.4 (Interrupt Facilities).

A 1 enables the corresponding condition; a 0 disables it. This register may be written at any time. This register cannot be read.

4.6.2 Interrupt Acknowledge Register

The Interrupt Acknowledge Register is used to individually indicate to the Am95C60 that a condition has been handled by the host and that the latch (Request) associated with the condition may be reset. The bits of this register are enumerated in Section 3.4 (Interrupt Facilities). This register may be written at any time. This register cannot be read.

4.6.3 System Bus Width Register

The System Bus Width Register is used to put the Am95C60 into 8-bit data bus mode. When the Am95C60 is reset, it is put into 16-bit data bus mode. When a write to this register is executed, the Am95C60 enters 8-bit mode and remains in 8-bit mode until it is reset. This register is used in conjunction with the 8-Bit Bus Byte Order Register.

4.6.4 Eight-Bit Bus Byte Order

This register is used to determine the order in which bytes must be written and read in 8-bit data bus mode. This register must be loaded before the write is directed to the System Bus Width Register. If this register is loaded with 00, the high order byte is transferred first. If the register is loaded with 01, the low order byte is transferred first.

4.6.5 Reset Register

Whenever a write is executed to this register the Am95C60 will be reset. The details of this operation are covered in Chapter 3, Section 3.1.

4.7 PICTORIAL REPRESENTATION OF ALL REGISTERS

Figures 4-6 through Figure 4-17 show all the registers. The addresses are given in decimal.

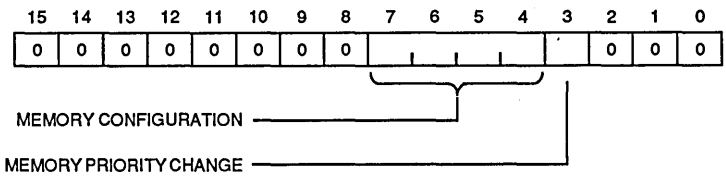


Figure 4-5 Memory Mode Register

MNEMONIC	ADRS	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SXY	1	0	0	0	0	SCREEN X START											
SYS	2	0	0	0	0	SCREEN Y START											
SXT	3	0	0	0	0	SCREEN X TERMINATE											
SYT	4	0	0	0	0	SCREEN Y TERMINATE											

07785A 4-1

Figure 4-6 Visible Screen Coordinate Registers

MNEMONIC	ADRS	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WAXS	14	0	0	0	0	WINDOW APPARENT X START											
WAYS	15	0	0	0	0	WINDOW APPARENT Y START											
WAXT	16	0	0	0	0	WINDOW APPARENT X TERMINATE											
WAYT	17	0	0	0	0	WINDOW APPARENT Y TERMINATE											
WRX	18	0	0	0	0	WINDOW REAL X START											
WRY	19	0	0	0	0	WINDOW REAL Y START											

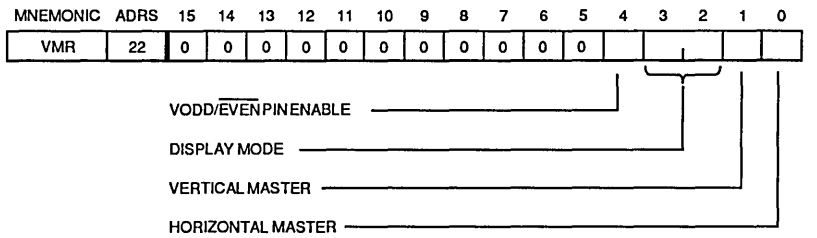
07785A 4-7

Figure 4-7 Window Control Registers

MNEMONIC	ADRS	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSYNC	10	0	0	0	0	0	0	0	0	HOR. SYNC WIDTH						(8 BITS)	
HDEL	11	0	0	0	0	0	0	0	0	0	HOR. SCAN DELAY					(7 BITS)	
HACT	12	0	0	0	0	0	0	0	0	HOR. ACTIVE						(8 BITS)	
HTOT	13	0	0	0	0	0	0	0	HOR. TOTAL COUNT							(9 BITS)	
VSYNC	5	0	0	0	0	0	0	VERT. SYNC WIDTH								(10 BITS)	
VDELOD	6	0	0	0	0	0	0	VERT. SCAN DELAY ODD								(10 BITS)	
VDELEVEN	7	0	0	0	0	0	0	VERT. SCAN DELAY EVEN								(10 BITS)	
VACT	8	0	0	0	0	0	VERT. ACTIVE									(11 BITS)	
VTOT	9	0	0	0	VERT. TOTAL COUNT											(13 BITS)	

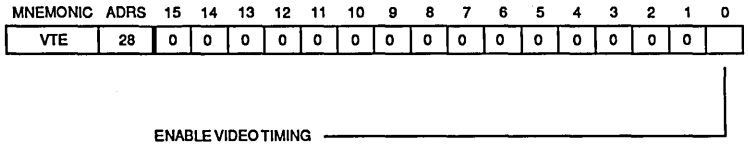
07785A 4-8

Figure 4-8 Video Timing Control Registers



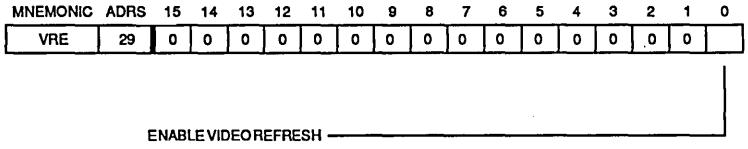
07785A 4-9

Figure 4-9 Video Mode Register



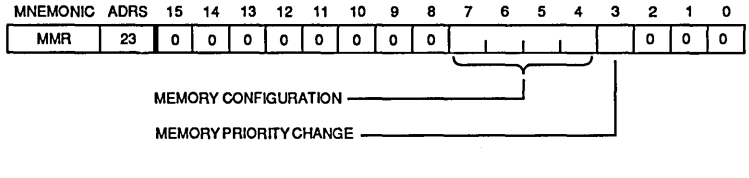
07785A 4-10

Figure 4-10 Video Timing Enable



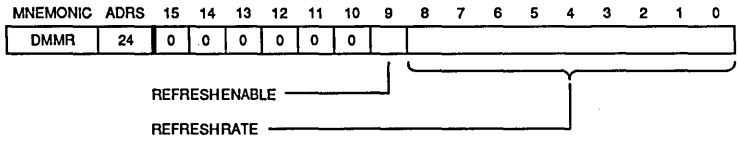
07785A 4-11

Figure 4-11 Video Refresh Enable



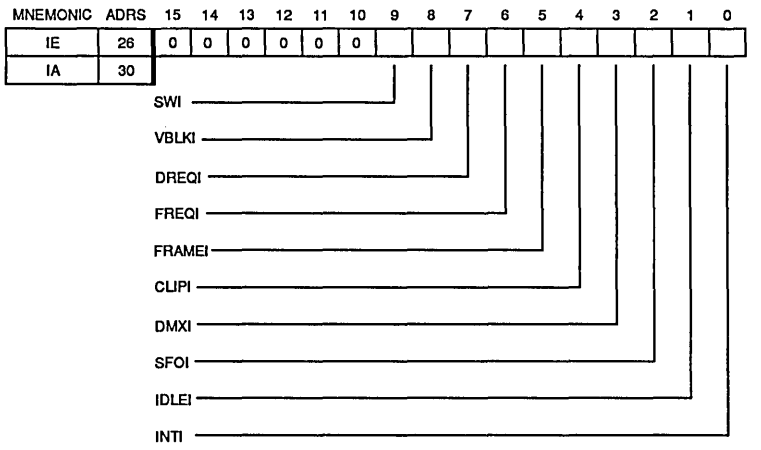
07785A 4-12

Figure 4-12 Memory Mode Register



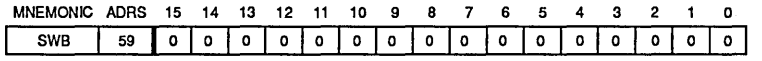
07785A 4-13

Figure 4-13 Dynamic Memory Refresh Rate Register



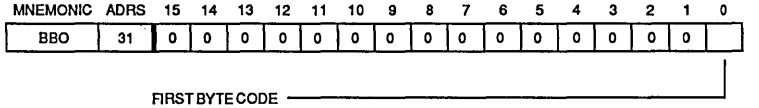
07785A 4-14

Figure 4-14 Interrupt Enable/Acknowledge Registers



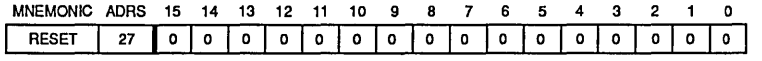
07785A 4-15

Figure 4-15 System Bus Width Register



07785A 4-16

Figure 4-16 8-Bit Bus Byte Order



07785A 4-17

Figure 4-17 Reset Register

CHAPTER 5

ADDRESSING MODES AND SCALING

The purpose of this chapter is to discuss addressing modes and scaling. All Am95C60 instructions that operate on the contents of display memory addresses in a consistent manner.

5.1 STANDARD OPERAND ADDRESS PAIR

Figure 5-1 shows the format of a Standard Operand Address Pair (SOAP) as it appears in any instruction. All Standard Operand Address Pairs have this format.

AM is the 2-bit address mode field. This field chooses one of four addressing modes for both X and Y. The four values of AM are enumerated below.

A M	Addressing Mode	Offset By:
0 0	Absolute	Zero
0 1	Viewport	Viewport Location
1 0	Relative	Current Pen Position
1 1	Indirect	Zero

X/dX is a 14-bit 2's complement number that specifies an absolute address or a relative value to be added to an offset. If the X scale factor was set to any value other than 0, the X/dX field will be multiplied by the X scale factor.

Y/dY is a 14-bit 2's complement number that specifies an absolute address or a relative value to be added to an offset. If the Y scale factor was set to any value other than 0, the Y/dY field will be multiplied by the Y scale factor. The two high order bits of this word must be 0s.

5.2 ADDRESS TRANSLATION

The following sections describe how the SOAP is evaluated for each of the four address modes. This process takes place twice for each operand address pair, once for the X address and once for the Y address.

Some operands must be on 16-bit boundaries in the X dimension. Stack entries and instructions in display memory are examples. The user is responsible for making certain the low order four address bits resolve to zeroes. If not, strange things may happen with no warning or notification.

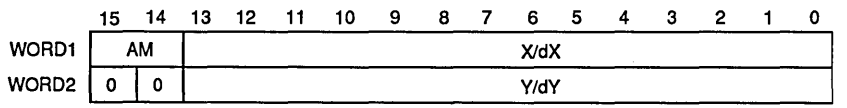
Scale factors will be mentioned frequently in this section. The scale factors are specified using the instruction "Set Scale Factor". In general, scaling is independent for the two dimensions (X and Y). For each dimension, scaling will take place for every addressing mode (except absolute and indirect) unless the scale factor for that dimension was set to 0. If the scale factor for a dimension was set to 0, then scaling will not occur for that dimension. In any case, scaling is never performed for absolute or indirect addressing.

When the multiply takes place, the Am95C60 produces a 29-bit real number as shown in Figure 5-2. There is a 14-bit integer part and a 15-bit fractional part. By integer part we mean that a change of value of 1 will change the address by one pixel. The Am95C60 does not make any assumptions about the position of radix points in either the address field or the scale factor.

One may assume the presence of radix points anywhere one wishes as long as there are a total of 15 bits of fraction. Two especially useful combinations are shown in Figure 5-3. The first combination assumes an integer address field and a fractional scale factor. The second combination assumes a fractional address field and a mostly integer scale factor (there is a single bit of fraction).

In the case where we assume an integer address and fractional scale factor, scale factors will be interpreted as follows:

x7FFF	unity -2^{-14}
x4000	.5
x2000	.25



07785A 5-1

Figure 5-1 Standard Operand Address Pair

5.2.1 Absolute Addressing Mode

When Absolute Addressing is specified, the 14-bit address field is taken as the effective address in the range -8192, +8191 regardless of the scale factors. The Fractional Error is unchanged. See Figure 5-4.

In Absolute Addressing mode, higher Y values will be lower on the screen. The origin is at the upper-left corner of display memory. This is shown in Figure 5-4.

5.2.2 Viewport Addressing Mode

When Viewport Addressing mode is specified and the scale factor was set to 0, the 14-bit address field is taken as an integer to which is added the Viewport Location (also taken as an integer). The sum is the effective address in the range -8192, +8191. The Fractional Error is unchanged. See Figure 5-5.

When Viewport Addressing mode is specified and the scale factor was set to some value other than 0, then the address field is multiplied by the scale factor producing a 29-bit real number. The Viewport Location (taken as an integer) is added to the integer part. The sum is taken as the effective address in the range -8192, +8191. The fractional part is stored as the new Fractional Error. While this address mode can modify the Fractional Error, it does not use it. See Figure 5-6. When the truncation takes place, the pixel is displaced UP on the screen. This is away from the viewport location; that is perhaps not what one might have expected.

In Viewport Addressing, with or without scaling, positive Y values will appear above the viewport location. This is shown in Figures 5-5 and 5-6.

5.2.3 Relative Addressing Mode

When Relative Addressing mode is specified and the scale factor was set to 0, the 14-bit address field is taken as an integer to which is added the Current Pen Position. The sum is the effective address in the range -8192, +8191. The Fractional Error is unchanged. See Figure 5-7.

When Relative Addressing mode is specified and the scale factor was set to some value other than 0, then the address field is multiplied by the scale factor producing a 29-bit real number. The Current Pen Position is added to the integer part and the Fractional Error is added to the fractional part (possibly causing a carry into the integer part). The integer part of the result is the effective address in the range -8192, +8191. The fractional part is stored as the new Fractional Error. This is the only addressing mode which actually uses the Fractional Error. See Figure 5-8. When the truncation takes place, the pixel is displaced UP on the screen. This is away from the Current Pen Position; that is perhaps not what one might have expected.

In Relative Addressing, with or without scaling, positive Y addresses will appear above the Current Pen Position. This is shown in Figures 5-7 and 5-8.

5.2.4 Indirect Addressing Mode

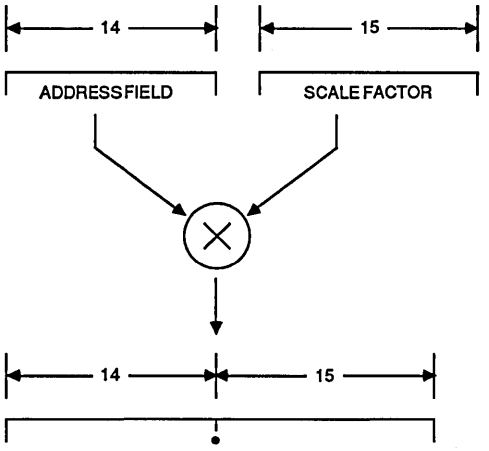
When Indirect Addressing mode is specified, the address fields are taken as an integer address of a standard operand address pair. The pair is at X,Y and X,Y+1 of bit plane 0. The four low order bits of the X address field must be 0.

Two words assumed to be a standard operand address pair (see Figure 5-9) are fetched from the indicated locations in display memory. It is evaluated in the normal manner. It may specify any addressing mode including indirect.

5.2.5 Round-off Errors

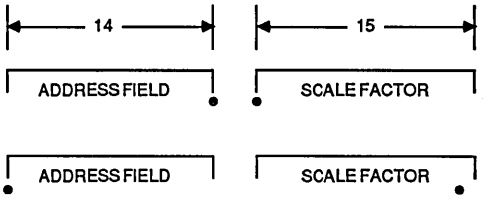
The purpose of maintaining the Fractional Error is to prevent the accumulation of errors when using a series of relative addresses. The Fractional Error resulting from each operand address evaluation is added into the fractional part of the real number resulting from the next scaling operation. The fractional part resulting from that operation is stored as the new Fractional Error for the next address calculation.

In Viewport Addressing mode with scaling the Fractional Error is not added into the real number resulting from the scaling operation. However, the fractional part is still stored as the new Fractional Error.



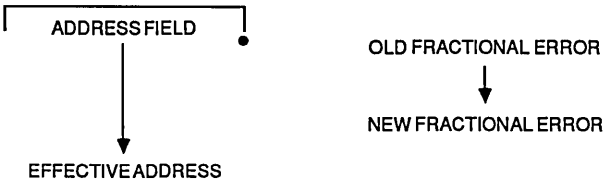
07785A 5-2

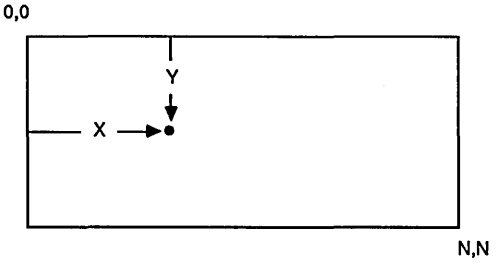
Figure 5-2 Scaling Multiply



07785A 5-3

Figure 5-3 Example of Binary Point Assignments





07785A 5-4

Figure 5-4 Absolute Addressing

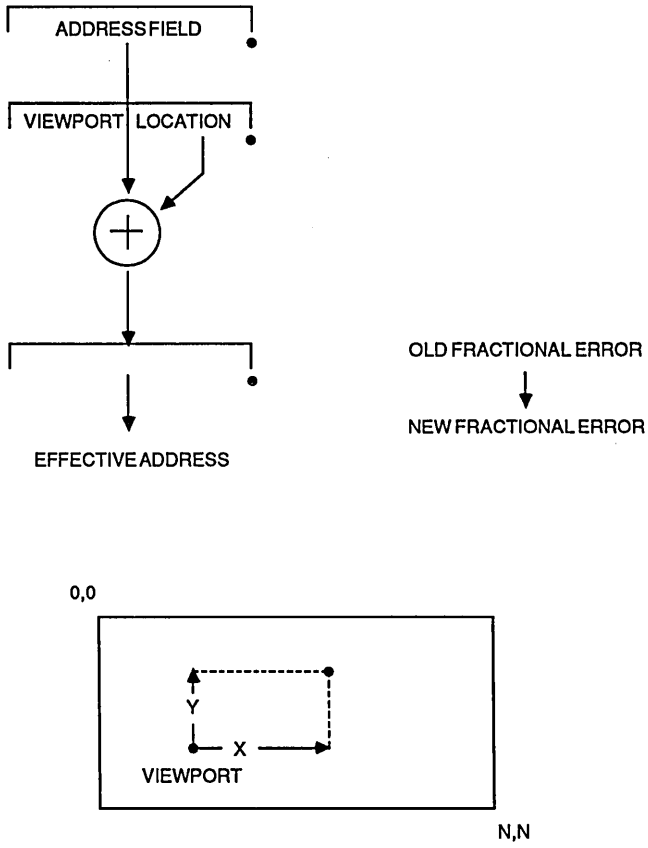
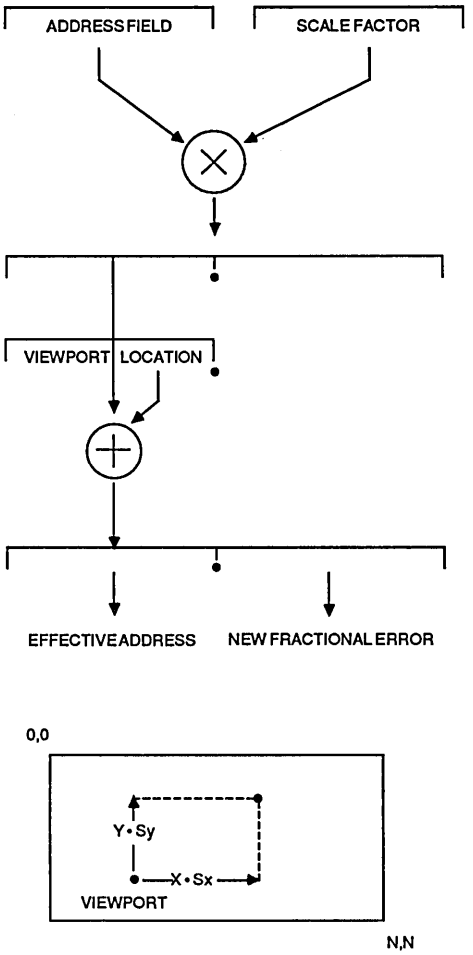


Figure 5-5 Viewport Addressing Without Scaling



07785A 5-6

Figure 5-6 Viewport Addressing With Scaling

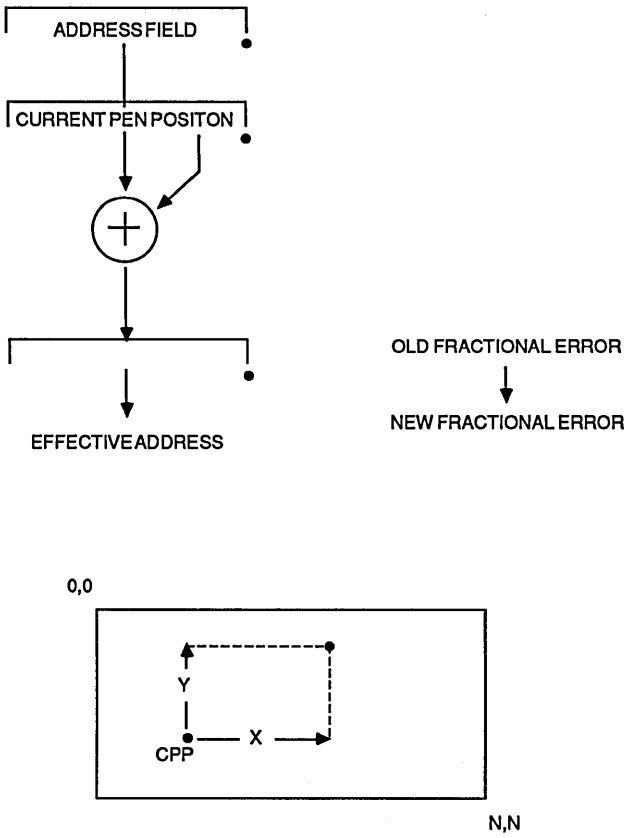
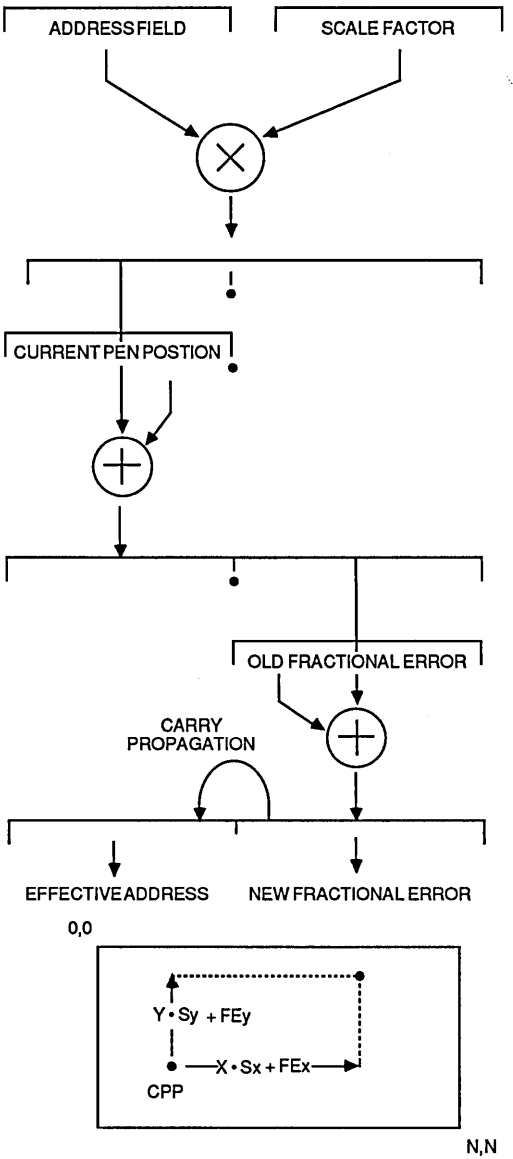
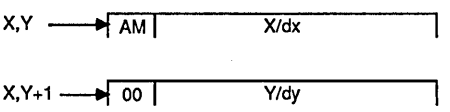


Figure 5-7 Relative Addressing Without Scaling



07785A 5-9

Figure 5-8 Relative Addressing With Scaling



07785A 5-10

Figure 5-9 Indirect Addressing

CHAPTER 6

LINE TEXTURE

Three related topics are covered in this chapter. The first is Line Styles, dealing with plain and fancy lines. The second topic is End-point Options, dealing with how the ends of lines (and arcs) are drawn. The third topic of this chapter is line thickness, dealing with the concept and implementation of the logical Picture Element (PEL). Throughout the remainder of this chapter, only lines will be discussed explicitly, but arcs and circles are treated similarly.

6.1 LINE STYLES

The Am95C60 allows for various line styles for drawing lines, arcs and circles.

6.1.1 Basic Line Styles

The two basic line styles are solid and dashed-dotted. Line styles "dashed" and "dotted" can be generated as special cases using the dashed-dotted line style. The information required by the Am95C60 to draw a specific line comes from three sources. These sources are the Set Line Style instruction, the Set Line Style Phase instruction and the Drawing instruction itself.

Each of the three drawing instructions, Line, Circle, and Arc (and their Current analogs) includes a LS bit (bit position 12) that specifies whether the line style is solid or dashed-dotted. If the LS bit is a zero, the line style is solid and any previous Set Line Style or Set Line Style Phase instructions are don't cares. If the LS bit is a one, the line style is dashed-dotted and the information from previous Set Line Style and Set Line Style Phase instructions is used.

The Set Line Style instruction is used to indicate the length of each of the three elements of a dashed-dotted line. These three elements are the dash length, dot length, and space length. If a dashed-dotted line is required, the dash length would be programmed to a value different from the dot length. If a simple dashed or dotted line is required, the dash length and dot length would be programmed to identical values. It is worth noting that the terms dash and dot do not imply any restrictions on the relative values that may be specified, even for dashed-dotted lines.

Figure 6-1 shows the four basic line styles. Observe that the dashed line style and dotted line style are special cases of the dashed-dotted line style.

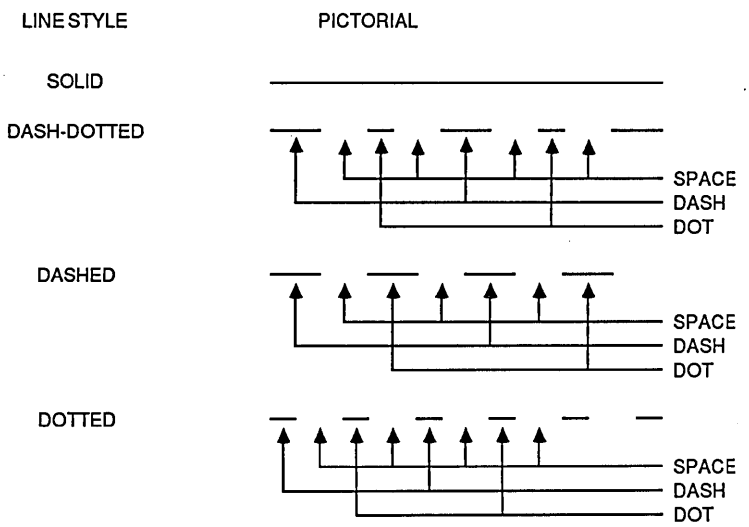


Figure 6-1 Basic Line Styles

The Set Line Style Phase instruction provides the third piece of information needed by the Am95C60, namely where within the line style cycle to begin. The LSS field specifies the element with which the line is to begin and the LSC field specifies the number of pixels that remains in that element. The Set Line Style Phase instruction must proceed every drawing instruction unless the line style is solid or the line style is to be continued from the previous instruction.

Note that the space elements are not drawn with a color or logical operation different from the dash or dot elements; the Am95C60 simply does not execute any writes to the bit map.

6.1.2 Complex Line Styles

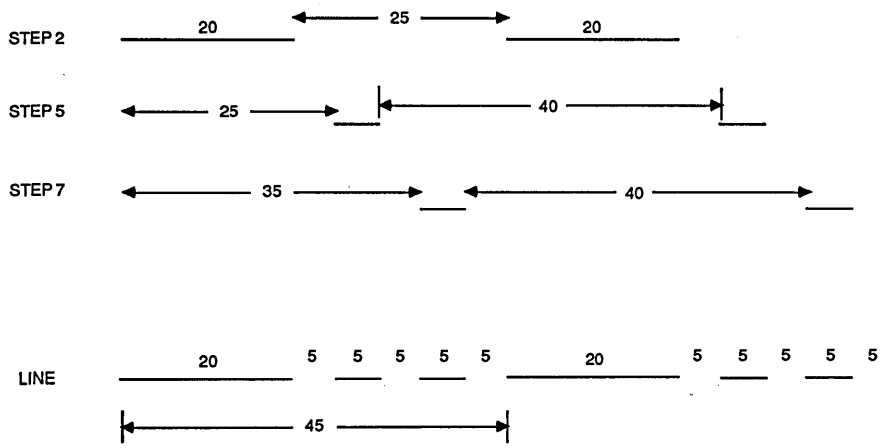
It is possible to draw with line styles more complex than dashed-dotted. This involves drawing the line (or arc or circle) more than once. Figure 6-2 illustrates a line with two dots between dashes. The lengths of each of the elements are indicated in the figure.

This particular complex line style is formed by drawing the same line three times (as indicated in the figure). The first iteration produces the dashes, the second produces the leftmost dots and the third produces the rightmost dots. The step-by-step procedure is:

1. *Set Line Styles:*
   Dash=Dot=20, Space=25,
2. *Set Line Style Phase:*
   LSS=01 (Dash), LSC=20 (full length dash)
3. *Draw the Line:*
   LS=Dashed-dotted, EP=conditional
4. *Set Line Style:*
   Dash=Dot=5, Space=40
5. *Set Line Style Phase:*
   LSS=00 (Space before dash),
   LSC=25 (25 space pixels)
6. *Draw the Line:*
   LS=Dashed-dotted, EP=conditional
7. *Set Line Style Phase:*
   LSS=00 (Space before dash),
   LSC=35 (35 space pixels)
8. *Draw the Line:*
   LS=Dashed-dotted, EP=conditional

The first three steps draw the first iteration of the line. This is a dashed-dotted line with 20 pixel dashes and dots and 25 pixel spaces. The line style phase is set to begin with a full-length dash.

Steps four through six draw the second iteration of the line. The line style is set to five pixel dashes and dots and 40 space elements. The line style phase is set to begin with a space element with 25 pixels remaining. Thus, we skip over the first 25 pixels before beginning a dash.



07785A 6-2

Figure 6-2 Complex Line Style

Steps seven and eight draw the third iteration of the line. The line style is still correct from the second iteration (we still want five pixel dash-dots with 40 pixel spaces). The line style phase is set to begin with a space element with 35 pixels remaining. Thus, we skip over the first 35 pixels before beginning a dash.

Steps five and seven (the second and third Set Line Style Phase instructions) could just as well have specified LSS of 10 (space before dot) rather than 00. Since the dash length and dot length are set the same, the result would have been identical. Furthermore, step two could have begun with a dot region with identical results.

It is possible to draw arcs and circles with complex line styles using the same approach.

6.1.3 Line Styles with Diagonal Lines

The line drawing algorithms used in the Am95C60 normally count pixels along either the X or Y axis, never along the diagonal. This will result in the elements of nonsolid lines drawn near the diagonal appearing to be longer than the elements of lines drawn near the X or Y axis. This can be observed by drawing a series of dashed-dotted lines radiating out from a common point.

This can be compensated for, in part, by using the Pixel Length Diagonal (PLD) option in the Set Scale Factor instruction. This would be done regardless of whether your application would otherwise call for scaling.

When the Am95C60 is choosing the next pixel of a line, it has two basic choices: it can choose the next pixel that is adjacent along the X or Y axis or it can choose the next pixel that is diagonally away from the current pixel. When it chooses a pixel adjacent along an axis, it always subtracts one from the remaining length of the current line style element. When it chooses a pixel that is diagonal, it subtracts the value programmed with the PLD option. If this value is the Square Root of 2 (~1.414), elements at the diagonal will have the same physical length (not the number of pixels) as elements along the axis. In addition, lines at intermediate angles will be adjusted by an intermediate amount (they will be generated with a mixture of orthogonal and diagonal moves).

The PLD option is programmed in terms of sixteenths of a pixel. The default value that is programmed when the Am95C60 is reset is 16. Thus, the default if ONE. The values that are closest to Square Root of 2 are 23/16 (1.4375) and 22/16 (1.375).

6.1.4 Line Styles with Scaling

If scaling is used and the X and y scale factors are set differently (to compensate for a pixel aspect ration of non-unity, for example), another correction will take place. This is because the number of pixels per inch is different from X to Y. The length of lines is adjusted to compensate for this; the length of line style elements must be adjusted as well.

The correction that is performed for Line when the dashed-dotted line style is chosen (LS=1) involves the use of three different pixel lengths; one for each possible incremental direction of movement: X, Y, and D(iagonal). The increment in X is always unity. The increment in Y is calculated in the Line setup (if LS=1) as the ratio of the scale factors SX/SY. The increment for diagonal movement is chosen in the Set Scale Factor instruction as described above.

The Set Line Style and Set Line Style Phase instructions should always be executed whenever the scale factors are changed.

6.2 END-POINT OPTIONS

The user can control whether the end-points (that is, the first and last points) of lines and arcs are drawn.

The first point is always drawn for a Line or Arc instruction, even if the line style and line style phase are such that it falls within a space element. The first point is never drawn for a Line Current or Arc Current even if the line style and line style phase are such that it falls within a dash or dot element.

The last point of a Line or Arc is controlled by the EP bit (bit position 11). If EP is zero, the last point is drawn only if it falls within a dash or dot element. If EP is a one, the last point is drawn even if it falls within a space element. If the line style is solid, the last point is always drawn.

The line style phase is never reset at the beginning of a drawing instruction, but rather continues from where the previous instruction left off. This is useful for continuing lines around an inflection point. If you wish to force the phase, you must execute a Set Line Style Phase instruction.

6.3 LOGICAL PEL

The logical PEL provides a means of drawing points, lines, arcs, and circles using a rectangular

brush containing an arbitrary pattern. The pattern may vary from plane to plane or it may be stored only in a single plane and replicated into the other planes.

For lines, arcs and circles, the PEL is useful for drawing with various line width parameters; for points, it is useful for drawing with various marker types.

The logical PEL is controlled with the Define Logical PEL instruction, the Set Block Size instruction, and three bits of the drawing instruction itself.

The Define Logical PEL instruction enables or disables the logical PEL and defines where in display memory it is located. The addresses derived from the Define Logical PEL instruction point to the upper-left corner of the PEL. There is only a single current PEL, although many may be stored in memory for subsequent use. Set Block specifies the size of the PEL; there is no limit to the size of the PEL.

The AA, SI, and M bits of the drawing instructions control how the PEL is used.

If the logical PEL is enabled and the AA bit is a zero, the PEL is taken from all planes. If the AA bit is a one, the PEL is taken from the bit plane that was specified in the FP field of the last Set Character Font Position instruction. In this case, the PEL is replicated into all those planes whose activity bits are set. The replication does not span QPDMs. If the logical PEL is not enabled, the AA bit is used to control anti-aliasing.

If the SI bit is a one, the PEL is inverted (1's complement) before being used. If it is a zero, the PEL is used as it appears in the display memory. If the logical PEL is not enabled, the SI bit is ignored.

If the M bit is a one, the PEL is stored only where the destination matches, on a pixel-by-pixel basis, the search color. This allows clipping with irregular regions. Observe that the PEL could be a single bit. If the M bit is a zero, the PEL is stored without matching. The SOAXZ field still applies in all these cases. If the logical PEL is not enabled, the M bit is ignored.

When the Set Block Size instruction is executed, offsets for each of the two dimensions (X and Y) are calculated by dividing the block size by two (using truncation). These offsets are stored whenever a Set Block Size is executed and are used with the logical PEL as described below.

6.3.1 Using the Logical PEL with the Point Instruction

The logical PEL is especially useful for drawing markers with the Point Instruction.

When the logical PEL has been enabled, the Point instruction (and its Current analog) proceed as follows (see Figure 6-3). The standard operand address pair is evaulated in the normal manner, producing an X, Y address. The offsets described above are SUBTRACTED from the X, Y address and the resultant address is used as the upper-left corner of an area into which the PEL is written.

When the logical PEL is being used with the Point instruction, the center of the logical PEL is deposited on the point and the remainder is deposited around the point. If the size of the PEL is even in either of both dimensions, then it does not contain a pixel that occupies the center. In this case, the PEL cannot be deposited symmetrically about the point but rather will be displaced up or to the left 1/2 pixel.

6.3.2 Using the Logical PEL with the Line Instruction

When the logical PEL has been enabled, the Line instruction (and its Current analog) proceed as follows (see Figure 6-4). The start and end points are each modified by SUBTRACTING the offsets described above. Then the line is interpolated and the PEL is deposited at its upper-left corner at each pixel. The effect is that "the PEL is dragged by its center".

Depending on the pattern contained in the logical PEL, this will draw a wide line regardless of the vector direction. A filled circle whose diameter is equal to the desired line width works especially well. If the SOAXZ field specifies graphical SET, and the PEL is drawn into all bit planes, the line will be the color specified with the current drawing color.

Using the logical PEL with Line instructions may lead to unexpected results if the SOAXZ field specifies XOR since the PEL will cancel itself in some cases.

The logical PEL cannot be used with anti-aliased lines but can be used with dashed-dotted line style.

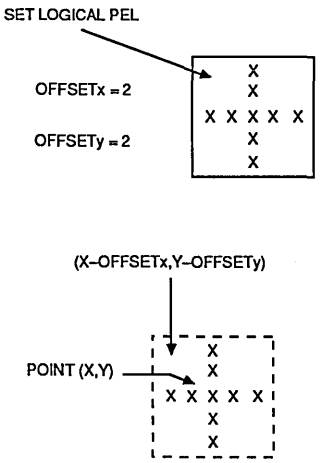
6.3.3 Using the logical PEL with Arc and Circle instructions

When the logical PEL has been enabled, the Arc and Circle instructions proceed as follows (see Figure 6-4). The center is modified by SUBTRACTING the offsets described above. Then the arc is interpolated and the PEL is deposited at its upper-left corner at each pixel. The effect is that "the PEL is dragged by its center".

Depending on the pattern contained in the logical PEL, this will draw a wide figure regardless of the vector direction. A filled circle whose diameter is equal to the desired line width works especially well. If the SOAXZ field specifies graphical SET, and the PEL is drawn into all bit planes, the figure will be the color specified with the current drawing color.

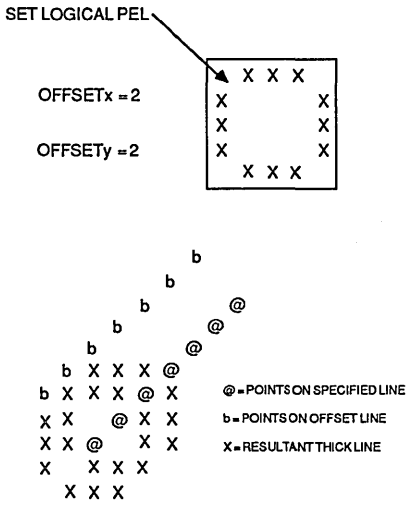
Using the logical PEL with Arc and Circle instructions any lead to unexpected results if the SOAXZ field specifies XOR since the PEL will cancel itself in some cases.

The logical PEL cannot be used with anti-aliased arcs and circles but can be used with dashed-dotted line style.



07785A 6-3

Figure 6-3 Logical PEL and Point



07785A 6-4

Figure 6-4 Logical PEL and Line

CHAPTER 7

CLIPPING AND PICKING

Two related topics are covered in this chapter. These are clipping that involves altering only a certain area of display memory, and picking that allows the host to determine what operations affect a certain area of display memory.

7.1 CLIPPING

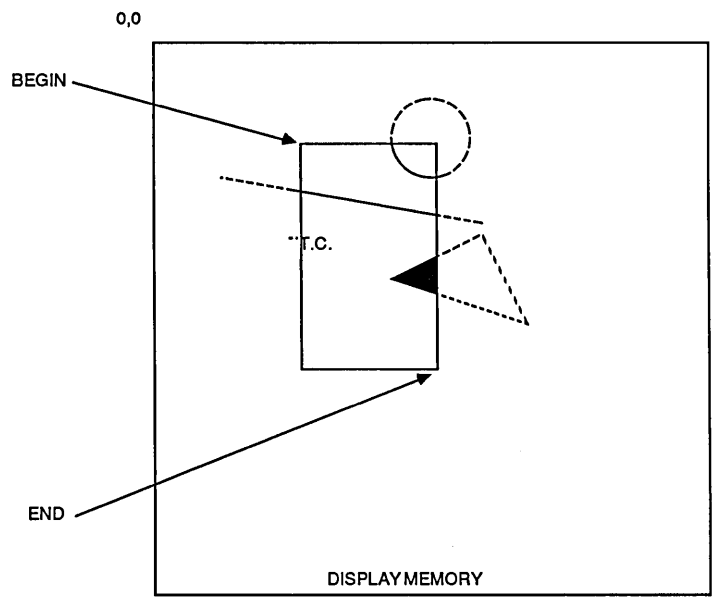
The Am95C60 provides a clipping facility. If clipping is enabled, operations that draw into display memory will actually write only within the clipping window. This is illustrated in Figure 7-1 where the solid figures indicate writes that take place and the dotted figures indicate writes that do not take place. Note that there is no particular relationship between the screen boundaries and the clipping boundaries.

The instructions Input Block and Store Immediate always ignore clipping. The writes to display memory take place as though clipping were disabled.

The clipping window is established using the Set Clipping Boundary instruction. This instruction takes as its operands two sets of X,Y points that define the clipping window (always a rectangle). The pixels that lie on the boundaries are taken to be within the window.

If clipping is enabled (see Chapter 14, Control Clipping instruction), instructions that alter pixels in display memory operate precisely as they normally would except that no writes will take place outside the clipping window. In addition, whenever an instruction generates an address that would result in a write outside the clipping window, the CLIPI condition is raised (see Chapter 3 for a description of the interrupt system). Once raised, this condition will remain active until a write takes place within the clipping window. The CLIPI request remains active until cleared with an acknowledge.

The Am95C60 calculates all the points (of a circle for example) it would otherwise have calculated; only the actual writes to display memory are



07785A 7-1

Figure 7-1 Clipping

suppressed. This implies that drawing time cannot be reduced by clipping. If a point is outside maximum display memory, DMXI will be set even if the point is clipped.

For the purposes of write enable control, the clipping boundaries may be on any pixel address. For correct operation of the CLIPI condition for operations other than Arc, Circle, Point and Line, the X coordinate of the clipping boundary must be word aligned.

7.2 PICKING

The Am95C60 provides a picking facility. If picking is enabled, operations that would otherwise write into display memory do not actually perform writes at all. Rather, the addresses are calculated and a decision is made to raise the Pick Detect (PCKD) status bit based on whether the write would have taken place within the picking region.

The picking region is established using the Set Picking Region instruction. This instruction takes as its operands two sets of X,Y points that define the picking region (always a rectangle). The pixels that lie on the boundaries are taken to be within the picking region.

If picking is enabled (see Chapter 14, Control Picking instruction), instructions that alter pixels in display memory operate precisely as they normally would except that no writes will take place. In addition, whenever an instruction would write into the picking region, the PCKD status bit is set. After the PCKD status bit has been set and cleared, you must draw something outside the picking region in order to clear the pick condition. This will "arm" the PCKD status bit.

For the Arc, Circle, Line, and Point instructions (and their Current analogs), if the logical PEL is disabled, the picking boundary may be set to any pixel boundary. The writes will be suppressed and the pick detect condition will be set correctly.

For most other operations, the picking boundary must be word aligned (in the X dimension). Examples of such operations are Copy Block, Fill Bounded Region, Fill Connected Region, Filled Rectangle, Filled Triangle (and their Current analogs) as well as Arc, Circle, Line, and Point if the logical PEL is enabled. The writes will be suppressed but pick detect may or may not be set correctly if this restriction is not met.

If a Copy Block with match is executed, and a write that would otherwise set picking detect is suppressed because of no match, pick detect may or may not be set.

7.2.1 Example of Picking

The host needs to determine whether a particular drawing instruction would cause any writes within the picking region. The step-by-step method is given below:

1. Set the picking region and enable picking, using the Set Picking Region instruction.
2. Enable interrupts.
3. Issue the drawing instruction in question.
4. When an interruption occurs, the state of PCKD and IDLE may be evaluated as follows:

PCKD	IDLE	Condition
0	0	Some other interrupt occurred
0	1	No pick took place
1	X	A pick took place

CHAPTER 8

GRAPHICAL OPERATIONS

This chapter discusses four related fields that appear in most drawing instructions. The fields are the SOAXZ field, the M bit field, the SI bit field and the SP field.

This discussion will refer to color bits and search color. There is one color bit per plane that is specified with Set Color Bits. There is one bit per plane in the search color; it is specified with Set Search Color.

8.1 SOAXZ FIELD

When blocks are being copied or when images (e.g., lines) are being written into display memory, it is possible to perform graphical operations combining the data being written with the current contents of display memory. These operations are done on a pixel-by-pixel basis and are done independently for each bit plane in the system. The SOAXZ field contains three bits and specifies which graphical operation is used. The acronym is derived from Set, Or, And, Xor, Zero.

8.1.1 SOAXZ Field in Instructions Involving a Source Operand

The instructions that involve a source operand are Copy Block, String, and Transform Block (and their Current analogs). If the logical PEL is enabled, then Arc, Circle, Line and Point behave as if the logical PEL were the source operand. The source operand is combined with the destination operand under control of the SOAXZ, the color bit for the particular bit plane and the SI field. See Figure 8-1.

8.1.1.1 Logical SET

If a SOAXZ field of 000 is specified, the destination operand is simply replaced with the source operand. This mode ignores the color bit. This SOAXZ field should be used to accomplish a simple bit-by-bit copy. The activity bits can be used to selectively suppress the transfer of planes, allowing a transparent effect.

8.1.1.2 Logical OR

If a SOAXZ field of 001 is specified, the destination operand is replaced with the logical OR of the destination operand and the source operand. This logical operation is done independently for each bit plane. This mode ignores the color bit.

8.1.1.3 Logical AND

If a SOAXZ field of 010 is specified, the destination operand is replaced with the logical AND of the destination operand and the source operand. This logical operation is done independently for each bit plane. This mode ignores the color bit.

8.1.1.4 Logical XOR

If a SOAXZ field of 011 is specified, the destination operand is replaced with the logical XOR of the destination operand and the source operand. This logical operation is done independently for each bit plane. This mode ignores the color bit.

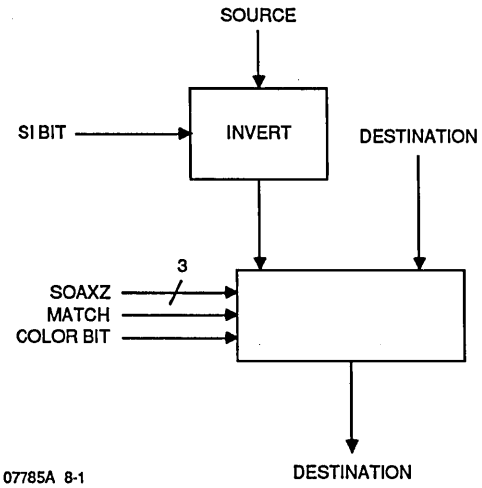
8.1.1.5 Logical ZERO

If a SOAXZ field of 100 is specified, the destination operand is replaced with the logical AND of the destination operand and the 1's complement of the source operand. This is done regardless of the color bit. This is done independently for each bit plane.

A 1 in the source field forces a 0 in the destination, and a 0 in the source field results in no change in the destination. This mode ignores the color bits.

8.1.1.6 Graphical SET

If a SOAXZ field of 101 is specified and the color bit for a given bit plane is a 1, the destination operand is replaced with the logical OR of the destination operand and the source operand. If a SOAXZ field of 101 is specified, and the color bit for a given bit



SOAXZ	Label	Color Bit = 1	Color Bit = 0
0 0 0	Logical SET	D ← S	D ← S
0 0 1	Logical OR	D ← S OR D	D ← S OR D
0 1 0	Logical AND	D ← S AND D	D ← S AND D
0 1 1	Logical XOR	D ← S XOR D	D ← S XOR D
1 0 0	Logical ZERO	D ← !S AND D	D ← !S AND D
1 0 1	Graphical SET	D ← S OR D	D ← !S AND D
1 1 0	Graphical OR	D ← S OR D	D ← D (n/c)
1 1 1	Graphical XOR	D ← S XOR D	D ← D (n/c)

D	destination operand
←	is replaced by
S	source operand
!S	ones complement of source operand
OR	logical OR operation
AND	logical AND operation
XOR	logical exclusive OR operation
n/c	No Change

SOAXZ	SOURCE	DESTINATION	RESULT	SOAXZ	SOURCE	DESTINATION	RESULT
000	0	0	0	100	0	0	0
	0	1	0		0	1	1
	1	0	1		1	0	0
	1	1	1		1	1	0
001	0	0	0	101	0	0	0
	0	1	1		0	1	1
	1	0	1		1	0	Color Bit
	1	1	1		1	1	Color Bit
010	0	0	0	110	0	0	0
	0	1	0		0	1	1
	1	0	0		1	0	Color Bit
	1	1	1		1	1	1
011	0	0	0	111	0	0	0
	0	1	1		0	1	1
	1	0	1		1	0	Color Bit
	1	1	0		1	1	!Color Bit

Figure 8-1 Instructions With a Source Operand

plane is a 0, the destination operand is replaced with the logical AND of the destination operand and the 1's complement of the source operand. This is done independently for each bit plane.

If the color bit is a 1, a 1 in the source field forces a 1 in the destination, and a 0 in the source field causes no change. If the color bit is a 0, a 1 in the source field forces a 0 in the destination, and a 0 in the source field causes no change. The other way to view this is: a 0 in the source field places the color bit in the destination, and a 0 in the source field causes no change in the destination.

8.1.1.7 Graphical OR

If a SOAXZ field of 110 is specified and the color bit for a given bit plane is a 1, the destination operand is replaced with the logical OR of the destination operand and the source operand. If a SOAXZ field of 110 is specified, and the color bit for a given bit plane is a 0, the destination operand is unchanged. This is done independently for each bit plane.

If the color bit is a 1, a 1 in the source field forces a 1 in the destination, and a 0 in the source field causes no change. If the color bit is a 0, the destination is unchanged regardless of the source field. The other way to view this is: a 1 in the source field ORs the color bit into the destination, and a 0 in the source field causes no change in the destination.

8.1.1.8 Graphical XOR

If a SOAXZ field of 111 is specified, and the color bit for a given bit plane is a 1, the destination operand is replaced with the logical XOR of the destination operand and the source operand. If a SOAXZ field of 111 is specified, and the color bit for a given bit plane is a 0, the destination operand is unchanged. This is done independently for each bit plane.

If the color bit is a 1, a 1 in the source field inverts the destination and a 0 in the source field causes no change. If the color bit is a 0, the destination is unchanged regardless of the source field. The other way to view this is: a 1 in the source field XORs the color bit into the destination, and a 0 in the source field causes no change in the destination.

8.1.2 SOAXZ Field in Instructions Not Involving a Source Operand

The instructions that do not involve a source operand are Arc, Circle, the Fills, Line, and Point (and their Current analogs). In these instructions, the Am95C60 calculates where a point or series of points should be placed. In this case there is no source operand (or rather, the interpolation mask is the source operand). If the logical PEL is enabled, then Arc, Circle, Line and Point behave as if there were a source operand (the image of the logical PEL). See Figure 8-2.

8.1.2.1 Logical SET

A SOAXZ field of 000 should not be used with instructions not involving a source operand. Since the interpolation mask is used, fifteen bits of each word will be forced to zero. This is almost certainly not the intended result. But you should try it once just to satisfy your curiousity.

8.1.2.2 Logical OR

If a SOAXZ field of 001 is specified, 1s are forced regardless of the color bit.

8.1.2.3 Logical AND

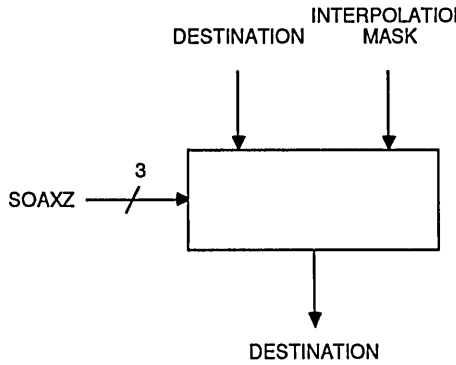
A SOAXZ field of 010 should not be used with instructions not involving a source operand. Since the interpolation mask is used, fifteen bits of each word will be forced to zero. This is almost certainly not the intended result.

8.1.2.4 Logical XOR

If a SOAXZ field of 011 is specified, the destination data is inverted regardless of the color bit.

8.1.2.5 Logical ZERO

If the SOAXZ field is 100, a 0 is written regardless of the color bit. This is done independently for each bit plane.



07765A 8-2

SOAXZ	Label	Color Bit = 1	Color Bit = 0
000	Logical SET	D ← 1	D ← 1
001	Logical OR	D ← 1	D ← 1
010	Logical AND	D ← D (n/c)	D ← D (n/c)
011	Logical XOR	D ← !D	D ← ! D
100	Logical ZERO	D ← 0	D ← 0
101	Graphical SET	D ← 1	D ← 0
110	Graphical OR	D ←1	D ← D (n/c)
111	Graphical XOR	D ← !D	D ← D (n/c)

D	destination operand
←	is replaced by
1	binary one
0	binary zero
!D	ones complement of destination operand
n/c	No change

SOAXZ	DESTINATION	COLOR	RESULT	SOAXZ	DESTINATION	COLOR	RESULT
000	0	0	1	100	0	0	0
	0	1	1		0	1	0
	1	0	1		1	0	0
	1	1	1		1	1	0
001	0	0	1	101	0	0	0
	0	1	1		0	1	1
	1	0	1		1	0	0
	1	1	1		1	1	1
010	0	0	0	110	0	0	0
	0	1	0		0	1	1
	1	0	1		1	0	1
	1	1	1		1	1	1
011	0	0	1	111	0	0	0
	0	1	1		0	1	1
	1	0	0		1	0	1
	1	1	0		1	1	0

Figure 8-2 Instructions Without a Source Operand

8.1.2.6 Graphical SET

If the SOAXZ field is 101, the destination operand is simply replaced with the color bit of the particular bit plane. This is the correct SOAXZ field for drawing lines, etc., with the current drawing color.

8.1.2.7 Graphical OR

If the SOAXZ field is 110, the destination field is read and ORed with the color bit for the particular bit plane. If the color bit for a plane is a 1, the destination is replaced with a 1. If the color bit for a plane is a 0, the destination is not changed. This is done independently for each bit plane.

8.1.2.8 Graphical XOR

If the SOAXZ field is 111, the destination field is read and XORed with the color bit for the particular bit plane. If the color bit is a 1, the destination is inverted (1's complement). If the color bit is a 0, the destination is unchanged. This is done independently for each bit plane.

8.1.3 Color Strategy

Essentially nothing can be said about a general color strategy without making some assumptions about the relationship of bit planes and colors on the screen. Since a typical system will include a color look-up table, there is no definitive or typical relationship. Thus, we will say nothing further about color strategy except to reiterate that each pixel of each plane is dealt with independently. We have enumerated the rules; the rules go by bit plane.

8.2 THE M BIT

Some instructions include a single-bit field called the M bit. These instructions are Copy Block and String (and their Current analogs) as well as the drawing instructions Point, Line, Arc, and Circle (and their Current analogs). In the case of Point, Line, Arc, and Circle, this bit is effective only if the logical PEL is enabled.

If the M bit is set, the instruction will write only into those pixels whose color matches the search color (see instruction Set Search Color). This match is done on a pixel-by-pixel basis and include those planes whose listen bits are zeroes.

The M bit can be used to implement a fill with a pattern. If it is necessary to fill an arbitrary closed polygonal area with a pattern, one may proceed as follows. There must be a copy of the pattern in memory.

First, fill the polygon with some otherwise unused color using Fill Bounded Region or Fill Connected Region. Then set the search color to the unused color with which the polygon was filled. Finally copy the pattern into an area that encloses the polygon using Copy Block with the M bit set. Everywhere the color matches the pattern will be copied, thus filling the polygonal area. Figure 8-3 illustrates this where a small pattern is copied repetitively.

The M bit can be used with the logical PEL to implement clipping using an irregular region (as opposed to the normal clipping rectangle). This might involve assigning an otherwise unused color (pixel combination) to the bits that are to be matched.

8.3 THE SI BIT

Some instructions include a single-bit field called the SI bit. These instructions are Copy Block and String (and their Current analogs) as well as the drawing instructions Point, Line, Arc, and Circle (and their Current analogs).

If this bit is set in a Copy Block, the contents of the source field are inverted (1's complement) prior to the SOAXZ operation.

If this bit is set in a String, the contents of the font pattern words (but not the attribute word) are inverted (1's complement) prior to the SOAXZ operation. This can be used to fill in the character cells with a background color.

If this bit is set in a Point, Line, Arc, or Circle instruction AND the logical PEL is enabled, the contents of the logical PEL are inverted (one's complement) prior to the SOAXZ operation.

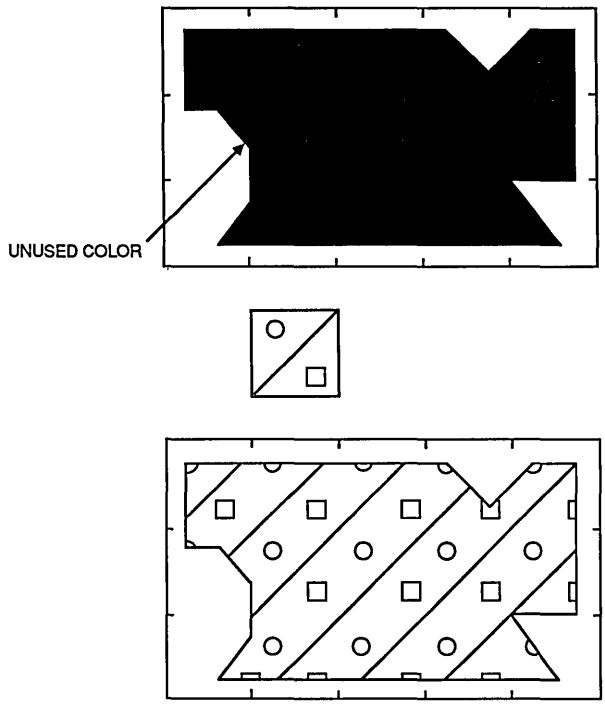
8.4 THE SP BIT

Some instructions include a single-bit field called the SP bit. These instructions are Copy Block and String (and their Current analogs) as well as the drawing instructions Point, Line, Arc, and Circle (and their Current analogs).

If this bit is set in a Copy Block, the source is taken from a single plane (specified in the FP of Set Character Font Base) and replicated for each plane whose activity bit is set.

If this bit is set in a String, the attribute word and the font pattern is taken from a single plane (specified in the FP field of Set Character Font Base). The pattern words are replicated for each plane whose activity bits are set.

If this bit is set in a Point, Line, Arc, or Circle instruction AND the logical PEL is enabled, the logical PEL is taken from a single plane (as specified in the FP field of Set Character Font Base) and replicated for each plane whose activity bits are set.



07785A 8-3

Figure 8-3 The "M" Bit

CHAPTER 9

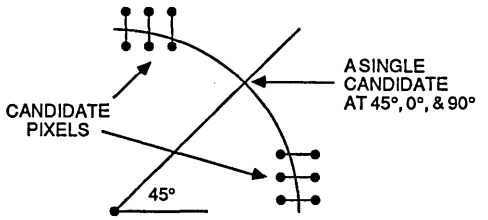
ANTI-ALIASING

The Line, Arc and Circle instructions (and their Current analogs) may invoke anti-aliasing. The host specifies this by setting the anti-aliasing (AA) bit to a 1. Anti-aliasing produces the appearance of smoothing the steps that are the inevitable result of drawing a slanted line into a quantized medium such as a bit map.

Anti-aliasing on the Am95C60 works by selectively suppressing write enables to the bit planes. The value that is written into a bit plane depends on the color bit and SOAXZ field as normal. All the examples in this chapter assume that the SOAXZ field is 101 (graphical SET), that the color bits are 1s and that the figure is being drawn into a blank (BLACK) background.

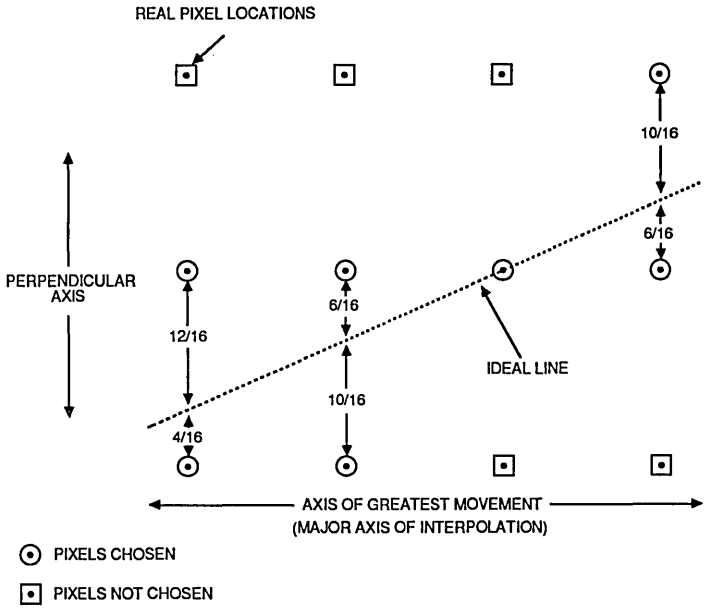
9.1 SELECTION OF CANDIDATE PIXELS

Figure 9-1a shows the candidate pixels for arcs and circles. Figure 9-1b illustrates how the candidate pixels are chosen and their distances calculated. This applies to both the Comparator Method



07785A 9-1A

Figure 9-1a Candidate Pixels for Circles and Arcs



07785A 9-1

Figure 9-1b Selection of Candidate Pixels

and the Inverse-Distance Method. The line-drawing algorithm used in the Am95C60 first determines the axis along which there is the greatest movement. In the case of a straight line, this is the axis whose difference between beginning and end is the greatest. In the case of an arc or circle, this depends on which octant the Am95C60 is currently interpolating. For every pixel along this axis, it finds where the ideal pixel would lie on the perpendicular axis. The location of this ideal pixel is calculated in increments of 1/16 pixel.

The two real pixels lying closest to this ideal pixel along the perpendicular axis are two candidate pixels and are the only two that might be changed (anti-aliasing never affects pixels farther than 15/16 from the ideal pixel). If the ideal pixel exactly corresponds to a real pixel, then only one candidate is chosen. This always occurs for 45 degree and orthogonal lines and for arcs and circles at the 0, 45, and 90 degree points.

9.2 COMPARATOR ANTI-ALIASING

For each of the two candidate pixels, the absolute distance (expressed in 1/16 pixel) is compared to the anti-aliasing distances for each of the four bit planes. These distances will have been previously set using instruction Set Anti-aliasing Distance. If the distance is less than the minimum or greater than the maximum, the plane is not written into. If the distance is equal to or greater than the minimum and equal to or less than the maximum, the bit plane will be written into (if the activity bit is set). The value written into the plane depends on the color bit and SOAXZ field as usual.

Comparator Anti-aliasing is selected by programming all the min values less than their max values.

9.2.1 Comparator Anti-aliasing Without a Look-up Table

Consider a system with four bit planes that drive a 4-bit digital-to-analog converter as shown in Figure 9-2. This example does not use a look-up table (LUT); the bits drive the DAC inputs directly.

In this system (Comparator Anti-aliasing without a LUT), one cannot generate more levels of grey than there are bit planes and still have a monotonic decrease in illumination as one gets further from the ideal pixel.

Bit Plane	DAC Weight	Illumination	Minimum	Maximum
0	MSB	8/15	0000	0111
1	2nd MSB	4/15	0000	1011
2	3rd MSB	2/15	0000	1101
3	LSB	1/15	0000	1111

If the distance between the candidate pixel and the ideal pixel is equal to or less than 7/16, then no write enables will be suppressed. This will result in all 1's that will yield 15/15 output from the DAC.

If the distance is in the range 8/16 through 11/16 inclusive, the write enable to plane 0 will be suppressed. This results in 7/15 output from the DAC.

If the distance is in the range 12/16 or 13/16, the write enables to planes 0 and 1 will be suppressed. This results in 3/15 output from the DAC.

If the distance is in the range 14/16 or 15/16, the write enables to planes 0, 1, and 2 will be suppressed. This results in 1/15 output from the DAC.

This method can be expanded to more levels of grey by adding more Am95C60s and bit planes. In general, the number of unique levels that can be written is 4 • n, where n is the number of Am95C60s.

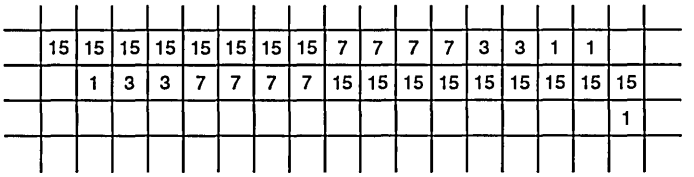
Figures 9-6, 9-7, and 9-8 show the actual pixels resulting from a number of anti-aliasing distances. These are shown as examples of what can be obtained.

9.2.2 Comparator Anti-aliasing With a Look-up Table

If one uses a look-up table to correct for the nonmonotonic decrease in grey level, it is possible to generate more levels of grey. The Am95C60 will write data that is not monotonic into the planes and the look-up table is programmed to compensate. This is shown in Figure 9-3.

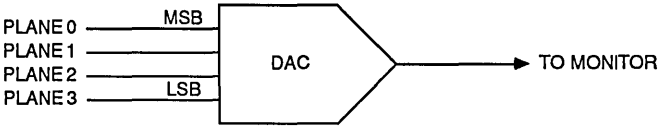
The distance registers are programmed as indicated in the following chart:

Plane	Minimum	Maximum
0	0000	1001
1	0010	1011
2	0101	1110
3	0111	1111

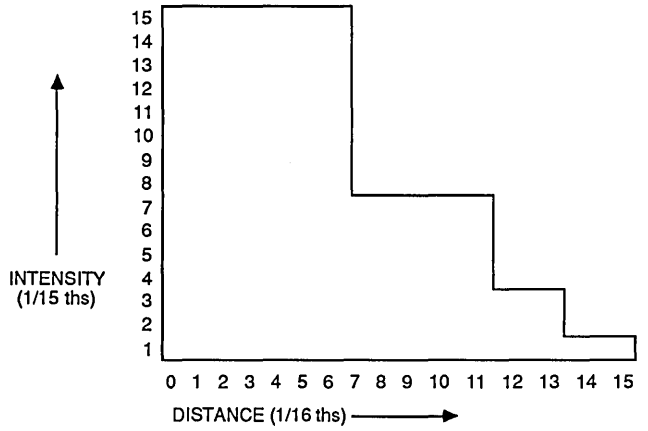



```
aad   Ø   { 0 < 15 , 0 < 13 , 0 < 11 , 0 < 7 }

inst        0037      0B07      0F0D
```



PLANE	MIN	MAX
0	0000	0111
1	0000	1011
2	0000	1101
3	0000	1111



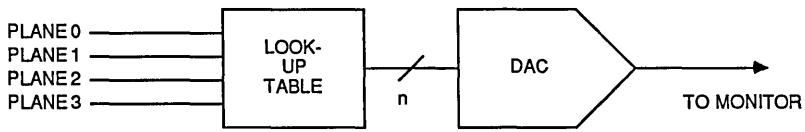
07785A 9-2

Figure 9-2 Comparator Anti-aliasing Without a LUT

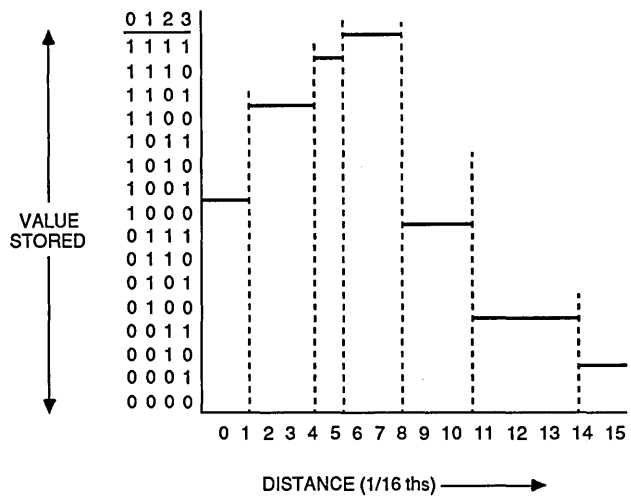
	1	1	3	3	3	7	7	15	15	15	14	14	12	12	12	8		
		12	12	12	8	8	15	15	15	7	7	3	3	3	1	1	1	1
																	8	12

```
aad   Ø   { 0 < 9 , 2 < 11 , 5 < 14 , 7 < 15 }

inst        0037      5E7F      092B
```



PLANE	MIN	MAX
0	0000	1001
1	0010	1011
2	0101	1110
3	0111	1111



07785A 9-3

Figure 9-3 Comparator Anti-aliasing With a LUT

This results in seven zones as shown in the Figure 9-3. Observe that these values cannot be directed to a DAC without correction; zone 4 would be more illuminated than zone 1. Through the nonlinear response of a LUT, this problem can be corrected.

The number of levels of grey can be increased by adding more Am95C60s and bit planes. In general, the number of grey levels that can be written is (8 • n) – 1 where n is the number of Am95C60s. Observe that for n = 3, the number of grey levels obtained in this manner exceeds the number of grey levels available using the Inverse Distance Method.

9.3 Inverse Distance Anti-aliasing

The candidate pixels are chosen as described in Section 9.1. For each of the two pixels, the distance from the ideal pixels is calculated (in 1/16 of a pixel). This value is inverted (1's complement) forming a value in the range 0 (15/16) to 15 (0/16).

The value is taken as four binary digits d3, d2, d1 and d0 where d3 is the most significant. The write enables to the bit planes are enabled according to these four digits and the distance register programming as shown in Figure 9-4.

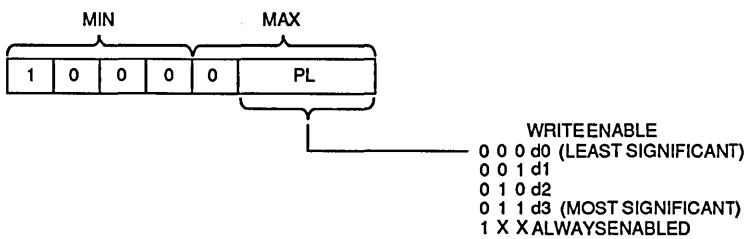
This is illustrated in Figure 9-5. The four bit planes are connected directly to the inputs of a DAC. While there is no look-up table in this example, most applications would require one because a linear decrease in intensity produces the "barber pole effect". Plane 0 is programmed to respond to the MSB and plane 3 is programmed as the LSB. There are 15 levels of grey possible using this method (not counting black).

Using more than one QPDM per color doesn't increase the number of intensities using the Inverse Distance Method.

Inverse Distance Anti-aliasing is selected by programming the most significant bit of each min register to 1 and programming the most significant bit of each max register to 0.

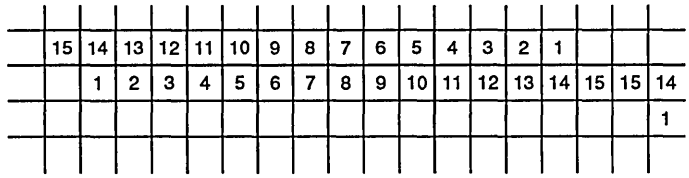
9.4 Anti-aliasing in a Color System

Anti-aliasing in a color system works best when there is an Am95C60 available for each of the three colors (red, green and blue). In this case, the concepts previously discussed hold; there are just three colors rather than one.



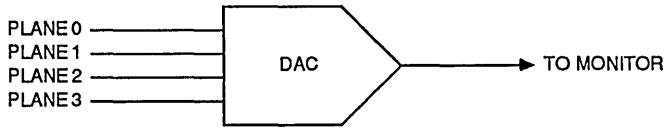
07785A 14-1A

Figure 9-4 Inverse Distance Anti-aliasing Programming

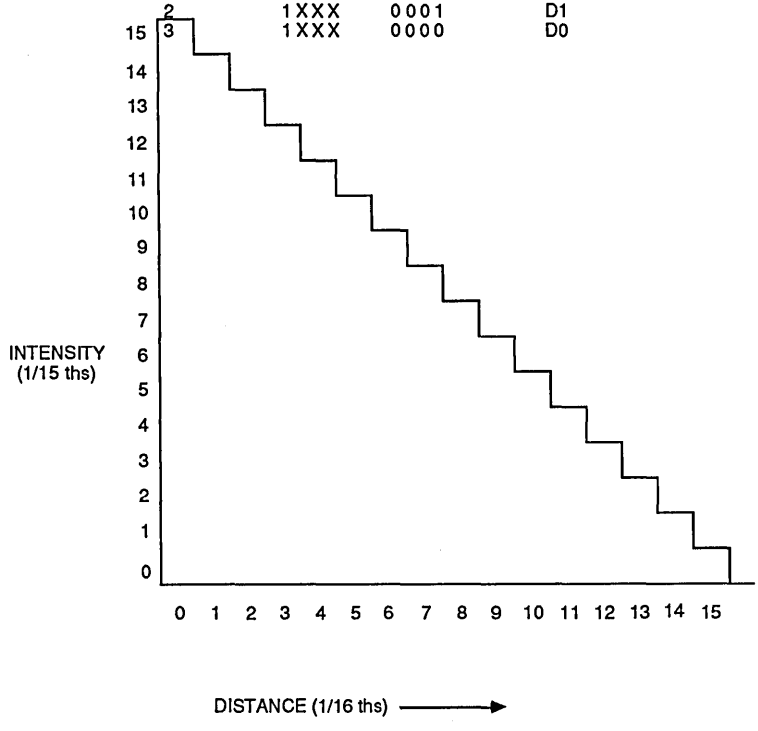



```
aad   Ø   { 0 < 9 , 2 < 11 , 5 < 14 , 7 < 15 }

inst      0037     5E7F     092B
```

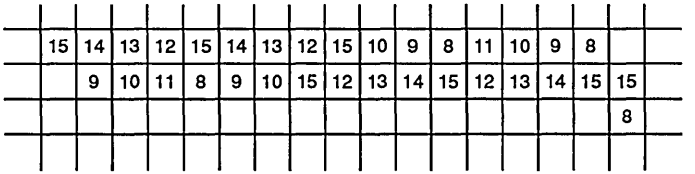


PLANE	MIN	MAX	DISTANCE BIT	WEIGHT
0	1XXX	0011	D3	8
1	1XXX	0010	D2	4
2	1XXX	0001	D1	2
3	1XXX	0000	D0	1



07785A 9-5

Figure 9-5 Inverse Distance Anti-aliasing

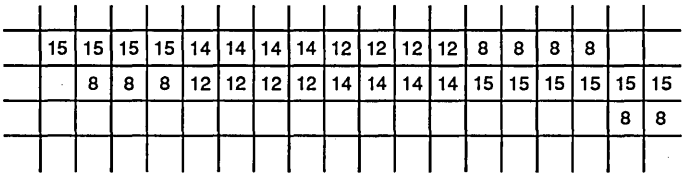


07785A 9A

```
aad   Ø   { 8 < 0 , 8 < 1 , 0 < 8 , 0 < 15 }

inst      0037     080F     8081
```

Figure 9-6

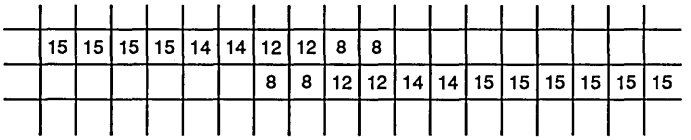


07785A 9B

```
aad   Ø   { 0 < 3 , 0 < 7 , 0 < 11 , 0 < 15 }

inst      0037     0B0F     0307
```

Figure 9-7



07785A 9C

```
aad   Ø   { 0 < 3 , 0 < 5 , 0 < 7 , 0 < 9 }

inst      0037     0709     0305
```

Figure 9-8

CHAPTER 10

STRING OPERATIONS

The purpose of this chapter is to discuss the string capability of the Am95C60. The Am95C60 is capable of creating the image of a text string in display memory very quickly and with a minimum of CPU intervention.

10.1 OVERVIEW OF STRING OPERATIONS

Since the string operations offer a large number of options, this discussion will begin with an overview. Detailed explanations and descriptions will appear later in this chapter.

Two things need to be done to use string operations in the Am95C60. First, the character font has to be stored in some otherwise unused area of the display memory and the Am95C60 must be informed. The character font describes the size of each character and contains its bit pattern. Second, a String instruction is used to specify where a string is to be written into display memory and which characters are to be used. The String instruction copies the patterns from the font to where they are to appear, modifying the Current Pen Position (CPP) following each character.

The characters are not rotated or scaled by the String instruction, but they can be ordered in display memory in any of four directions. If characters must be rotated or scaled, one may make use of subroutines that generate characters using strokes.

10.2 THE CHARACTER FONT

Any number of character fonts can be stored in the display memory and two character fonts can be used concurrently, providing the capability of accessing 8192 characters from a single String instruction. Each character font describes up to 4096 characters and each character can be as large as 63 pixels wide by 60 scan lines high.

A character font may be stored in any single-bit plane. This will result in the same pattern being replicated into all planes whose activity bits are set. Alternatively, a font may be stored in all four planes allowing the pattern to vary from plane to plane. This could be used, for example, to implement anti-aliased characters. The String instructions operate somewhat faster if the font is stored in all planes.

The font(s) must be stored in display memory (by the host) in the format shown in Figure 10-1. The entry for each character consists of an attribute word followed by a list of pattern words. The number of pattern words may vary from character to character within the same font. The Input Block instruction (by plane) with a Y dimension equal to 1 may be conveniently used to store fonts one character at a time.

10.2.1 The Attribute Word

The first word for each character is called the attribute word. Because the two-bit D field has a major influence on the interpretation of the rest of the fields, the attribute word will be described twice. The first description assumes the most significant bit of the D field is a 0 so that the String instruction orders characters horizontally in display memory. The second explanation will assume the most significant bit of the D field is a 1 so that the String instruction orders characters vertically in display memory.

10.2.1.1 Horizontally Ordered Characters

The Figure 10-2 illustrates string operations when the first bit of the D field is a 0.

The format of the attribute word is shown in Figure 10-3.

H is a 4-bit field that specifies the height (number of active scan lines) of the character. H is multiplied by the cell scale factor (either 2 or 4) so that the effective range is 2 through 60. If H is specified as zero, this is a nonprinting character and the S field becomes the ES/RD field that is explained in Section 10.5.

S is a 4-bit field that specifies the number of scan lines that are to be skipped above the character. S is multiplied by the cell scale factor (either 2 or 4) so that the effective range is 2 through 60.

D is a 2-bit field that controls the direction and sense of the H, S and ICO fields. In this description, the most significant bit of the D field is zero. The least significant bit of the D field indicates whether characters are to be written from left-to-right in display memory (D=00) or from right-to-left

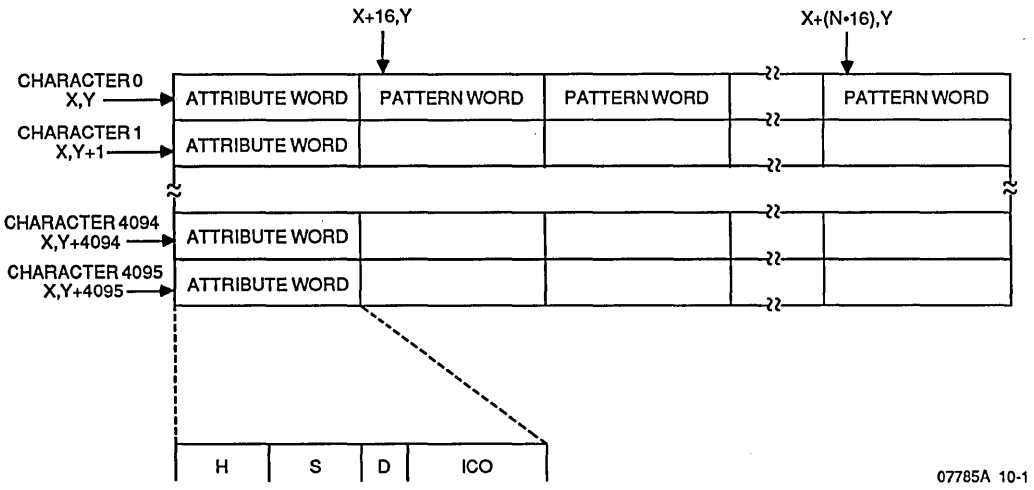


Figure 10-1 Character Font

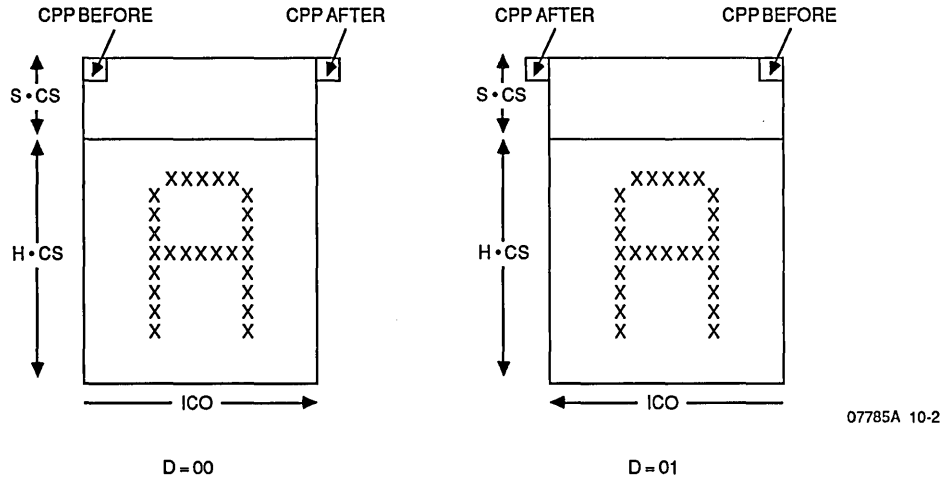


Figure 10-2 Horizontally Ordered Characters

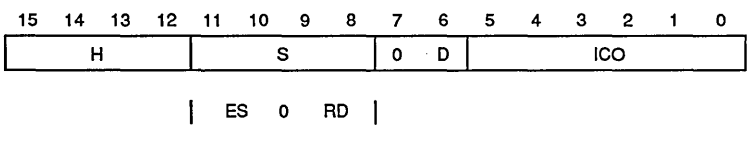


Figure 10-3 Format of the Attribute Word When D = 0X

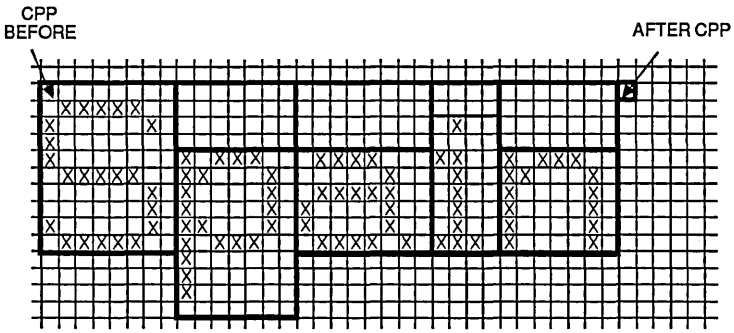
in display memory (D=01). The D field has no influence on how individual characters appear in display memory; it affects only where they are written.

ICO is a 6-bit field that is added to or subtracted from the X component of the Current Pen Position after each character has been copied from the font area. If D=00, ICO will be added after each character (so that characters are arranged from left-to-right). If D=01, ICO will be subtracted (so that characters are arranged from right-to-left). Figure 10-4 shows the results of some combinations of S, H and ICO fields.

The character is copied from the font area in the following manner. First, S cell scan lines are skipped. This provides a means of moving the pattern down (for example for lower case letters). Then words are copied from the pattern portion of the font. As many pattern words are copied from the font as is necessary for one scan line (ICO pixels). Pixels are discarded from the right of the last pattern word if ICO is not an integer multiple of 16 (that is, pattern words do not span scan lines). This process continues for H • cell scale scan lines. Finally, CPP is added to or subtracted from the X component of the CPP. The area in display memory that is written is below and either to the left or the right of CPP according to the low order bit of the D field.

10.2.1.2 Vertically Ordered Characters

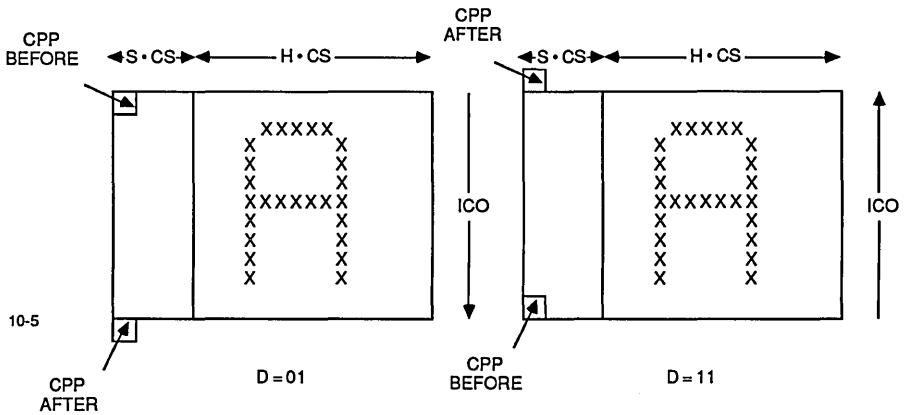
The Figure 10-5 illustrates string operations when the first bit of the D field is a 1.



LETTER	S•CS	H•CS	ICO
S	0	10	8
P	4	10	7
A	4	6	8
I	2	8	4
N	4	6	7

07785A 10-4

Figure 10-4 Example Using D = 00



07785A 10-5

Figure 10-5 Vertically Ordered Characters

The format of the attribute word is shown in Figure 10-6.

H is a 4-bit field that specifies the width (in pixels) of the character. H is multiplied by the cell scale factor (either 2 or 4) so that the effective range is 2 through 60. If H is specified as zero, this is a nonprinting character and the S field becomes the ES/RD field. This is explained in Section 10.5.

S is a 4-bit field that specifies the number of pixels that are skipped to the left of the character. S is multiplied by the cell scale factor (either 2 or 4) so that the effective range is 2 through 60.

D is a 2-bit field that controls the direction and sense of the H, S and ICO fields. In this description, the most significant bit of the D field is one. The least significant bit of the D field indicates whether characters are to be written from top-to-bottom in display memory (D=10) or from bottom-to-top in display memory (D=11). The D field has no influence on how individual characters appear in display memory; it affects only where they are written.

ICO is a 6-bit field that is added to or subtracted from the Y component of the Current Pen Position after each character has been copied from the font area. If D=10, ICO will be added after each character (so that characters are arranged from top-to-bottom). If D=11, ICO will be subtracted (so that characters are arranged from bottom-to-top).

The character is copied from the font area in the following manner. First, S • cell scale pixels are skipped. Then words are copied from the pattern portion of the font. As many pattern words are copied from the font as is necessary for one scan line (H • Cell Scale pixels). Pixels are discarded from the right of the last pattern word if H • Cell Scale is not an integer multiple of 16 (that is, pattern words do not span scan lines). This process continues for ICO scan lines. Finally, ICO is added to or subtracted from the Y component of the CPP. The area in display memory that is written is to the right of and either above or below the CPP according to the low order bit of the D field.

10.3 ESTABLISHING THE CHARACTER FONT

The Am95C60 is given the location in display memory of the character font with the Set Character Font Base instruction. This instruction establishes a number of additional parameters. The format of this instruction is shown in Chapter 14.

FP is a 2-bit field that indicates which bit plane contains the font if it is a single-plane font (which in turn is specified in the String instruction). If the font is a multiple-plane font, FP indicates from which plane the attribute words are to be fetched. One multiple-plane font may have several sets of attribute words associated with it.

CS is a 1-bit field that controls the cell scale. This effects the interpretation of the H and S fields as shown in the following chart:

H or S Value	CS=0	CS=1	Commentary
0000	0	0	(Nonprinting Char)
0001	2	4	
0010	4	8	
0011	6	12	
0100	8	16	
0101	10	20	
0110	12	24	
0111	14	28	
1000	16	32	
1001	18	36	
1010	20	40	
1011	22	44	
1100	24	48	
1101	26	52	
1110	28	56	
1111	30	60	

RC is a 1-bit field that specifies which of the components of the return address will actually be stored. If RC=0, the X component will be stored; if RC=1, the Y component will be stored. This information is used for the Carriage Return characters; see Section 10.5.

RETURN is specified using a standard operand address pair. This specifies either the X or Y address to be used for the Carriage Return

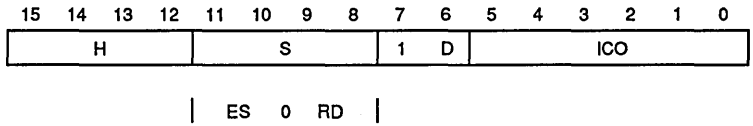


Figure 10-6 Format of the Attribute Word When D = 1X

character; see Section 10.5. If relative addressing is used, RETURN is calculated relative to the Current Pen Position. This MUST be a complete standard operand address pair even though it will not all be used.

FP0x and FP0y specify the address of the attribute word for character 000 of font 0. These are each 8-bit fields; they are multiplied by 16 to form 12-bit absolute addresses. This means that both addresses will have zeroes as the four LSBs.

FP1x and FP1y specify the address of the attribute word for character 000 of font 1. These are each 8-bit fields; they are multiplied by 16 to form 12-bit absolute addresses. This means that both addresses will have zeroes as the four LSBs.

It is not necessary that all 4096 entries of the font(s) be actually implemented, only those entries that will be used.

10.4 MOVING CHARACTERS FROM THE CHARACTER FONT (STRING INSTRUCTION)

Characters are actually copied from the font area with the String or String Current instruction. String specifies the beginning CPP; String Current uses CPP as the beginning CPP. The format of the String instruction is shown in Chapter 14.

SP is a 1-bit field that indicates whether the font is stored in one plane only or is stored on all planes. If SP is a 0, the font is assumed to be stored in all planes whose activity bits are set. The attribute words will be taken from the plane that was specified in the FP field of the Set Character Font Position instruction. If SP is a 1, the font is stored only in the plane that was specified in the FP field of the Set Character Font Position instruction. In this case, the font will be replicated into all planes whose activity bits are set.

SI is a 1-bit field that indicates whether the pattern read from the font area is to be inverted before being applied to the destination.

SOAXZ is a 3-bit field that indicates how the pattern read from the font area is to be applied to the destination.

BEGIN is specified with a standard operand address pair. This is the location to which the CPP will be set before the first character is processed. If relative addressing is specified, BEGIN will be calculated with respect to CPP. In the case of String Current, BEGIN is not present and CPP is used.

LIST is a variable-length string of 13-bit integer operands. Twelve of the bits are an index into one of the two fonts currently in use and the other (the P bit) selects which of the two fonts to use. These operands are processed in the order in which they appear in the instruction stream. Each operand is used as an index into the appropriate font to obtain the attribute word and pattern words. The leftmost bit of each entry in the LIST is a continuation bit. If it is not set, the String instruction terminates after that character has been processed. The String instruction will also terminate if it encounters a nonprinting character (H=0) whose ES bit in the attribute word is set.

10.5 NONPRINTING CHARACTERS

If the H field of the attribute word for any character contains zero, the character is a nonprinting character. In this case, the S field becomes the ES/RD field. If ES is set, this is a termination character. Observe that this bit has the opposite sense as the continue bit in the character list. The least significant two bits are the RD field and are interpreted as indicated below. The return location was specified in the Set Character Font Base instruction.

R D	Effect on CPP
0 0	No change. This character is an effective NOP.
0 1	The X component of CPP is loaded with the return location.
1 0	The Y component of CPP is loaded with the return location.
1 1	Both X and Y are loaded with the (same) return location.

This provides a carriage return facility. If the host is positioning characters from left-to-right, the RD field would be 01 and the return position would be set to the leftmost pixel in the printing area. The CPP is returned to the left of the current line, not the next line.

A "line feed" character can be implemented by specifying a D field of 10 and an ICO field of the standard character height.

Any character code or codes can be used for carriage return. The Am95C60 looks at the H and RD fields of the attribute word, not at the character code.

10.6 AN EXAMPLE

Figure 10-7 can be used to help tie all this together. This assumes that a font has been downloaded into bit plane 0 beginning at 1536, 1024. The font has 256 fixed sized entries each of which are 14 scan lines by 12 pixels.

In this example the character values are the same as ASCII 77. This is arbitrary; it could just as well have been EBCDIC or any other character code.

10.6.1 Set Character Font Base Instruction

The FP field is 00. This means that the font is stored in bit plane 0. The CS field is a 0, meaning that the H and S fields of the attribute words will be multiplied by two (rather than four). The RC field is a 0, meaning that the X component of the return address will be stored.

The Return field is all 0s. This is an absolute address and X is 0. The return address is 0 (that is also the left edge of the screen).

The FONT0 address is 60H,40H. In decimal this is 96,64. Both components are multiplied by 16 in the Am95C60; the result is 1536,1024. This is the leftmost bit of the attribute word for character 0.

The FONT1 address is set to 0,0 because in this example only one font is being used.

10.6.2 String Instruction

SP is set to a 1 indicating that the font is stored only in one bit plane. Since the FP field of the the Set Character Font Base is 00, the font is expected to be in bit plane 0.

SI is a 0 so that the pattern words will not be inverted before being written into the display memory.

The SOAX field is 101 for graphical SET. The color bit of each plane whose activity bit is set will be written everywhere when there is a 1 in the font. This means that the character will be written in the chosen color without changing the background.

If it is necessary to set the background color as well, the procedure is to change the color bits (and perhaps the activity bits) and then reissue the String instruction with the SI bit set. Now the color will be forced everywhere when there is a 0 in the font.

BEGIN specifies absolute addressing mode. The values are 200H,300H. This translates to 512,768 in decimal. CPP will be set to this value before the first character is processed.

10.6.3 Character Processing

The character value for the first character is 41H. This is 65 in decimal. The Am95C60 fetches the attribute word for the first character from 1536, 1089 (1024+65).

Since the H field of the attribute word is not 0, this is a printing character. The D field of the attribute word is 00; the character will be placed below and to the right of CPP.

The S field value is 1. Since the CS field of the Set Font Base is 0, the values in the S and H fields are multiplied by two. Two scan lines are skipped. Now the pattern words are copied into the destination area. Since ICO is 12, one pattern word is needed to define the pattern for each scan line. The four rightmost bits of each pattern word are discarded. A total of 14 pattern words are required for the character.

The font will be written into each bit plane whose activity bit is set. If there are bit planes whose activity bits are not set, they will not be changed at all. This allows for transparent characters. It is not required that the activity bit for the plane containing the font bit be set.

After the character is processed, the X component of the CPP is incremented by ICO (leaving CPP at 524,768). The continue bit is checked and since it is set, the Am95C60 fetches the next character code from the FIFO.

The operation continues until the last character has been processed. In this case, the string "Am95C60" will be written into display memory. The CPP will be left at 596,768.

10.7 BUILDING AND USING VERY LARGE FONTS

The Am95C60 can support two concurrent fonts of up to 4096 entries each (this restriction is due to the maximum Y dimension of display memory). A 24 x 24 (typical for Kanji) font will require two pattern words for each scan line with 8 bits unused. This is a total of 49 words for each font entry or 784 pixels. Since (at least part of) two fonts are required, this could use nearly half of a

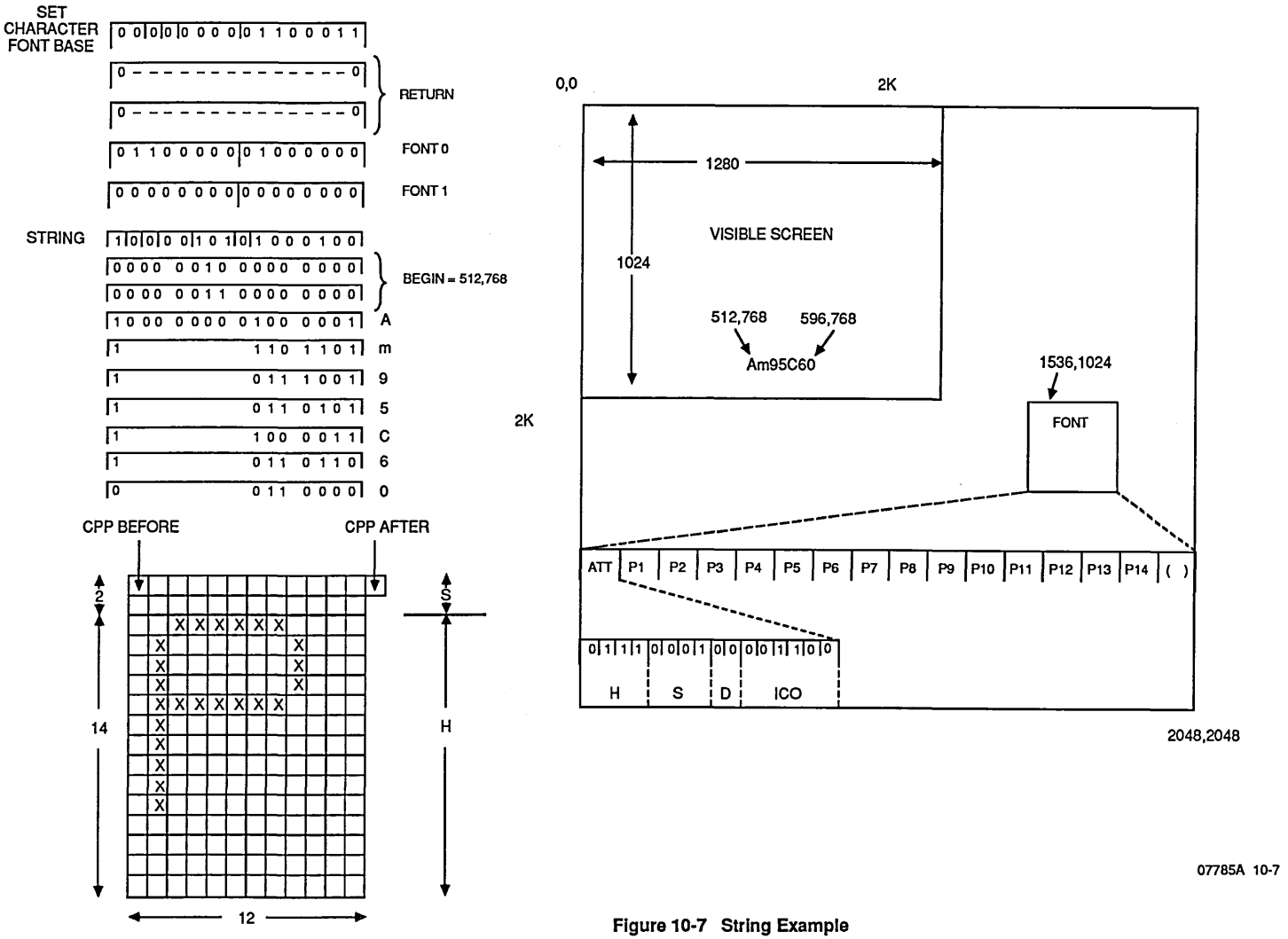


Figure 10-7 String Example

maximum sized display memory. A method of using inexpensive ROMs to contain these fonts is given in the Applications Manual.

10.8 TEXT WITH COLORED BACKGROUND

Some applications require writing text in which the background is forced to a specific color (as well as the characters themselves). By background, we mean that portion of the character cell that is not the character itself. There are two basic approaches that may be used.

The first method is to color the background (using Filled Rectangle) and then writing the text on top with String. This works best if you are dealing with a string of characters in which the character cell size is uniform.

The second method is to execute the string instruction twice. The first iteration, with the drawing color set to the desired background color and with the SI bit set, draws the background of the string. Then, with the drawing color set to the desired character color, the characters themselves are drawn. This works best with strings in which the character cell size is not uniform. It may also be faster than the first method for very short strings.

10.9 OPTIMIZING STRING PERFORMANCE

One can optimize the performance of the String instruction in two ways. These are discussed below.

The first way to minimize the cycles taken by a String instruction is to minimize the number of scan lines per character. A font with a fixed cell size will include characters with some scan lines of all zeroes (consider the lower case vowels and the space character). By using the S and H fields of the attribute word, one can take advantage of these blank scan lines.

Advanced Micro Devices will make available a font that takes these empty scan lines into account. The font is 7*9 + 2 descenders. The table below shows the average number of scan lines per character for three source documents when this font is used.

Document	Average Scan Lines/Character
U.S. Constitution	6.74
C Program	4.91
BASIC Program	7.26

The second way to optimize String performance is to avoid allowing characters to overlap word boundaries. Each scan line of a character that overlaps word boundaries requires two writes into display memory rather than one. An ICO of 8 or 16, in conjunction with beginning strings on word boundaries, will accomplish this.

CHAPTER 11

WINDOWS

The Am95C60 supports a single hardware window and an arbitrary number of software windows.

The Figure 11-1 illustrates the concept of a hardware window. Normally the image on the screen comes from a rectangular, contiguous area of display memory. The hardware window capability allows a rectangle of arbitrary size and location to be replaced dynamically on the screen with an image from another portion of display memory. The contents of display memory are not changed.

The hardware window may be any size and may be positioned on any bit boundary but is always a rectangle.

Software windows are implemented by moving the image to be displayed into the appropriate area of display memory. This is a destructive operation in the sense that the image overwrites the data it replaces. Software windows are discussed later in this section.

11.1 HARDWARE REQUIREMENTS

There are some hardware requirements that must be met in order to have a hardware window. In general, the display memory of the Am95C60 is organized in 16-bit words. As long as no window is being used, this does not present a problem. The image is shifted out of the VRAM in 16-bit words

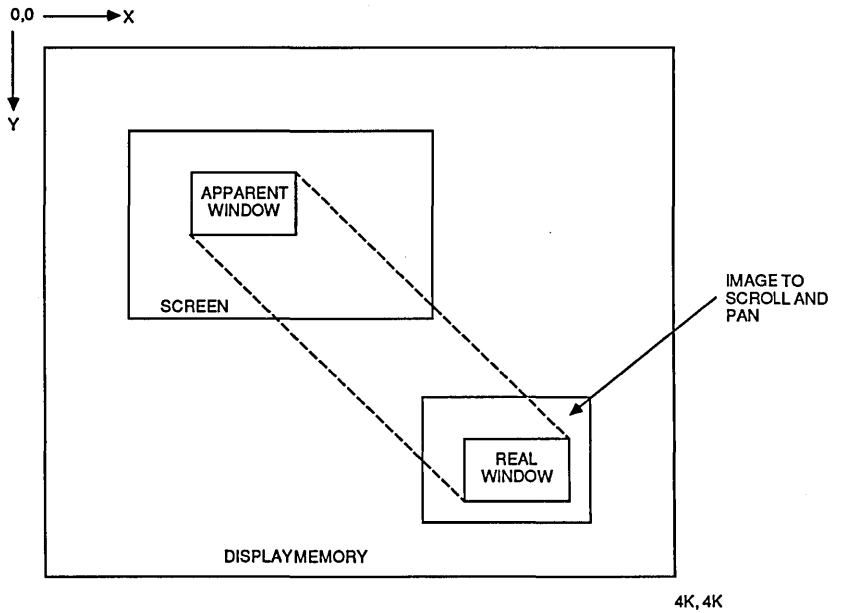
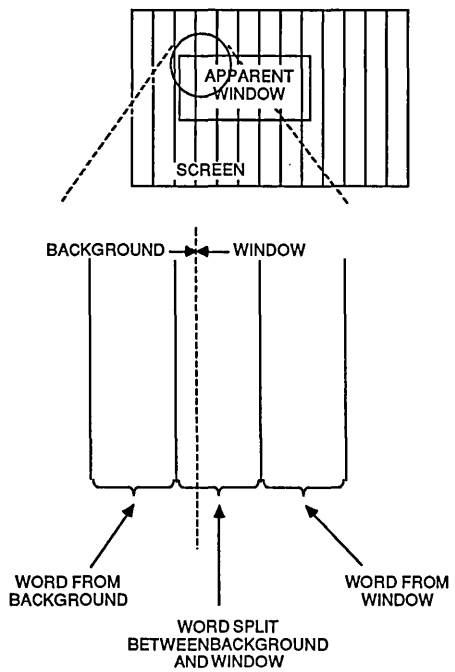


Figure 11-1 Hardware Window

and serialized for presentation on the screen. A single transfer cycle prior to the beginning of the scan line accesses all the information necessary.

Figure 11-2 illustrates one problem that arises when an arbitrarily positioned hardware window is present. The regular vertical lines represent word (16-bit) boundaries in display memory. If the system is implemented using a 16-bit shift register, then 16 bits is the best resolution that can be accomplished for any boundary (whether it be screen or window). Consequently, both the apparent and real window positions would be restricted to 16-bit boundaries.

There is also a timing problem. A transfer cycle has to be executed at the beginning of the window (to access the window image), and another transfer cycle must be executed at the end of the window (to access the background image to the right of the window). Alternatively, it is possible to perform the accesses early and buffer words in a FIFO. The logic that accesses the image gets ahead of the logic that actually puts the image on the screen.



07785A 11-2

Figure 11-2 Concatenation in the Window

The Am8171 Video Data Assembly FIFO is designed to solve both of these problems. During a transfer cycle, the Am95C60 indicates (on CDAT output pins) the position of the first bit that is to be used. Then, as each byte is tranferred from the VRAMs to the VDAF, the Am95C60 indicates (again using the CDAT pins) the number of bits in the byte. This will be 8 (indicating a complete byte) until the very last byte before the window. When the very last byte before the window is transferred to the VDAF, the Am95C60 will indicate how many bits of it are to be used.

When the transfer cycle that accesses the window is done, the Am95C60 again indicates the number of the first bit and as each word is transferred into the VDAF, the Am95C60 indicates how many bits are to be used. Thus the VDAF has sufficient information to pick out the valid bits. These groups of bits are concatenated into bytes and are placed into the VDAF FIFO.

The FIFO in the VDAF makes it possible for the Am95C60 to get ahead of the video stream far enough to allow transfer cycles within the scan line.

11.2 HOW DOES THE HARDWARE WINDOW APPEAR TO THE USER?

Figure 11-1 illustrates the appearance of the hardware window, a rectangular area that can be programmed to appear anywhere on the screen. The image that would otherwise appear in this area does not appear on the screen; it is dynamically replaced with an equal-sized image that can come from any place in display memory. Using the hardware window does not cause any changes whatsoever in the display memory.

For some applications it is desirable to replace only selected bit planes in the window. In a system with a single Am95C60, this is impossible since all four planes are replaced. In a system with multiple Am95C60s, it is possible to program some Am95C60s so that the real window has the same origin as the apparent window. Am95C60s programmed in this manner would have no effective window. In a system with multiple Am95C60s, all must have their apparent window programmed to the same location and size.

11.3 WINDOW CONTROL REGISTERS

The Window Control Registers are adequately described in Chapter 4 of this manual. These registers should be changed only at FRAMEI time.

11.4 TUTORIAL ON HOW TO USE THE WINDOW

The following paragraphs describe some of the ways that the hardware window may be used in an application. There are certainly other ways that are not described here.

11.4.1 Dragging an Object

In design automation systems it is often desirable to drag an object across the screen. Suppose, for example, the user is laying out a circuit diagram and wishes to position an element such as a gate or resistor. The real window would contain the image of the circuit element, and the apparent window would be moved around the screen in response to the user input (joystick or trackball). In this case, two of the apparent window registers would be changed to move the object in one dimension; all four apparent window registers would be changed to move the object simultaneously in both dimensions.

11.4.2 Pop-up Menus

A pop-up menu (or help display) can be easily implemented. The real window would contain the menu that is to appear. The image would be made to appear by programming the apparent window registers to the area in which the menu is to appear. When the user indicates the menu is no longer required, it would be made to disappear by programming the apparent window registers to an area that does not appear on the screen.

11.4.3 Scrolling in a Window

It is easy to scroll the contents of the window by merely changing the real window pointers. Figure 11-1 shows the real window contained within a region labeled "Image to scroll and pan". The apparent window is positioned on the screen, and the real window is positioned within the image. As the real window is moved, the data within the apparent window appears to move while the background display around it remains still.

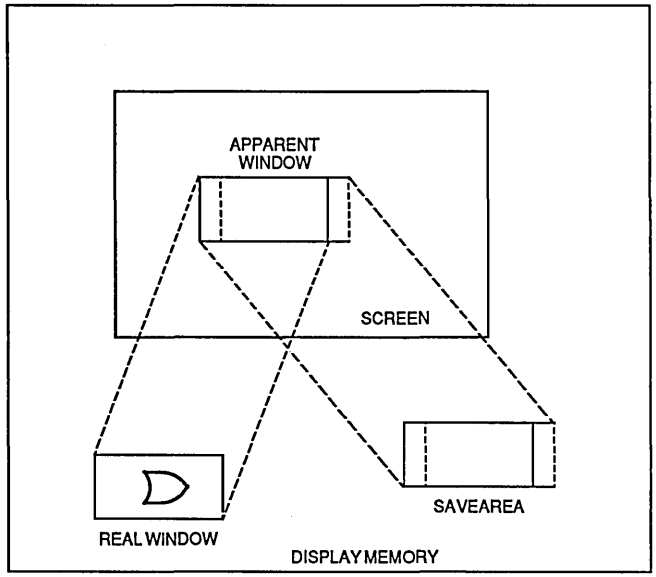
11.5 SOFTWARE WINDOWS

Any number of software windows can be supported by the Am95C60. Software windows are destructive in the sense that the overlayed area in display memory is overwritten.

In the most general case, using a software window is a three-step operation. First, the contents of the apparent window are saved in some unused area of display memory ("Save Area" in Figure 11-3). Second, the contents of the real window are copied into the apparent window. Third, when it is time for the software window to be removed, the contents of the save area are copied back into the apparent window. All of these copy operations are performed using the Copy Block instruction.

It is possible to copy only selected bit planes by programming the active planes. This does not save time, but it does provide for transparent overlays.

In the case where an object is being dragged, the three-step operation described above is done repetitively. If the save area is larger than the apparent window, it is possible to save time by not copying the entire window to the save area and back each iteration. Suppose the object is being moved to the right. When the save area is restored the first time, it is necessary to restore only the vertical strip on the left that will be uncovered by the window. Then to save the new apparent window, it is necessary to save only the vertical strip on the right that is about to be covered. The save area will have moved to the right.



07785A 11-3

Figure 11-3 Software (BITBLT) Window

CHAPTER 12

DISPLAY MEMORY CONFIGURATIONS

The purpose of this chapter is to exhaustively enumerate every possible memory configuration that can be built using the Am95C60. Frequent references will be made to the set of figures called "Memory Configuration A through H". There is one figure for every valid configuration.

The maximum possible size of display memory is 4096 x 4096. This means that twelve address bits are required in each dimension (X and Y) to uniquely specify a pixel (there are up to four bits associated with each pixel). These address bits are named X11, X10 ... X0 and Y11, Y10 ... Y0. Since the Am95C60 accesses display a 16-bit word at a time, the four low order bits of the X address never actually appear on the address bus.

Observe that the Am95C60 doesn't distinguish between memory devices configured as bit-wide and those configured as nibble-wide. It is up to the system designer to guarantee that 16 data bits are available for each plane. In the case of bit-wide devices, this will require 16 devices per bank per plane; in the case of nibble-wide devices, this will require four devices per bank per plane.

12.1 OVERVIEW OF DIAGRAMS

This explanation will use Figure E as an example. This figure is chosen as being especially descriptive as well as applicable.

The top diagram in each of the eight figures indicates which address bit appears on which line of the address bus during row address time (RAD) and column address time (CAD). Diagram E shows that A11 is indeterminate during both row and column time. It also shows that A10, A9 and A8 contain Y11, Y10 and Y9, respectively, during both row and column time. The remaining eight address bus lines contain Y addresses during row address time and X and Y information during column address time.

Address bus lines A10, A9 and A8 may be used for bank selection and A7–A0 are buffered to the array (64 K x n devices have eight multiplexed address inputs).

The center diagram in each figure depicts a 4 K x 4 K memory array. The area that is hatched in each figure cannot be addressed using the particular configuration. In Figure E, the memory width is 2 K and the area from 2 K to 4 K (in the X dimension) is hatched. The dimension lines above the center diagram indicate what X addresses are used to access the memory that is available. This is X10-0 in the figure being discussed, allowing 11 address bits for access to 2 K.

The table to the right of the center diagram shows how the bank select bits are encoded for each bank. In Figure E, there are three bank select bits Y11, Y10 and Y9. There are a maximum of eight banks for this configuration. It is not necessary to implement the maximum number of banks. One could build a 2 K x 2 K memory with this configuration using four banks of memory chips.

THERE ARE ALWAYS SIXTEEN BIT-WIDE CHIPS OR FOUR NIBBLE-WIDE CHIPS IN EACH BANK REGARDLESS OF THE WIDTH OF DISPLAY MEMORY AND REGARDLESS OF THE CHIP SIZE.

The dimension lines on the left of the center diagram indicate which Y address bits are used to select a Y address in each bank.

The table immediately below the center diagram allows the user to puzzle out what happens if a scan line goes past the right edge of addressable memory. This would occur if the screen X terminate register were programmed to a value outside memory (in this case 2K).

Shifter size is the number of bits transferred into the shifter during a transfer cycle for each of the 16 data bits (256 in this instance). Multiplying by 16 yields the total number of bits transferred, and dividing by the memory width yields the number of scan lines transferred (two in this case).

These scan lines are not adjacent in memory but rather are evenly distributed over the extent of each block. Therefore (in this instance), the screen will "wrap" to a scan line offset by 256.

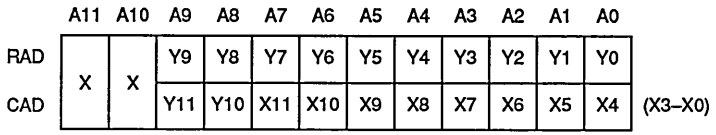
The Mem_Config_Value is the bit pattern that is written into bits 7 through 4 of the Memory Mode Register (see Chapter 4, Section 4.5.1). The table at the bottom of each figure shows the RAM size (for each of the cases X1 and X4), the display memory width and the maximum number of memory devices that a single Am95C60 can control in this configuration.

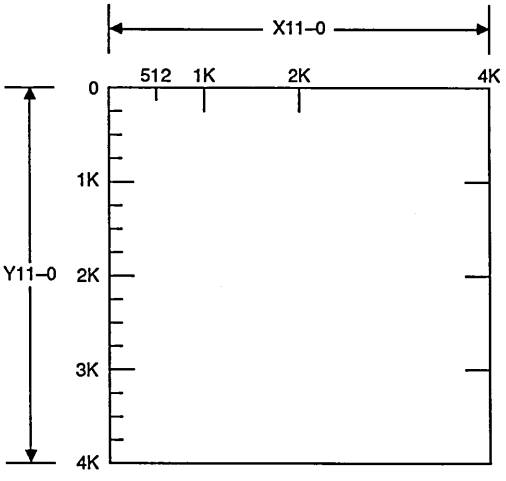
12.2 SUMMARY OF CONFIGURATIONS

Table 12-1 exhaustively lists every possible configuration using every possible memory size.

Table 12-1 Memory Configurations

Memory Size Wide	Deep	RAM Size	Banks	RAMs/Plane	Figure	Practicality
4 K	4 K	1 M • 1	1	16	A	Yes
4 K	4 K	1 M • 4	1	4	A	Prime
4 K	1 K	256 K • 1	1	16	B	Yes
4 K	1 K	256 K • 4	1	4	B	Yes
4 K	2 K	256 K • 1	2	32	B	Barely
4 K	2 K	256 K • 4	2	8	B	Yes
4 K	3 K	256 K • 1	3	48	B	No
4 K	3 K	256 K • 4	3	12	B	Yes
4 K	4 K	256 K • 1	4	64	B	No
4 K	4 K	256 K • 4	4	16	B	Yes
4 K	1 K	64 K • 1	4	64	C	No
4 K	1 K	64 K • 4	4	16	C	Yes
4 K	2 K	64 K • 1	8	128	C	No
4 K	2 K	64 K • 4	8	32	C	Barely
4 K	3 K	64 K • 1	12	192	C	No
4 K	3 K	64 K • 4	12	48	C	Probably No
4 K	4 K	64 K • 1	16	256	C	No
4 K	4 K	64 K • 4	16	64	C	No
2 K	2 K	256 K • 1	1	16	D	Yes
2 K	2 K	256 K • 4	1	4	D	Prime
2 K	4 K	256 K • 1	2	32	D	Barely
2 K	4 K	256 K • 4	2	8	D	Yes
2 K	1 K	64 K • 1	2	32	E	Barely
2 K	1 K	64 K • 4	2	8	E	Yes
2 K	2 K	64 K • 1	4	64	E	No
2 K	2 K	64 K • 4	4	16	E	Yes
2 K	3 K	64 K • 1	6	48	E	No
2 K	3 K	64 K • 4	6	24	E	Barely
2 K	4 K	64 K • 1	8	64	E	No
2 K	4 K	64 K • 4	8	32	E	Barely
1 K	4 K	256 K • 1	1	16	F	Yes
1 K	4 K	256 K • 4	1	4	F	Yes
1 K	1 K	64 K • 1	1	16	G	Yes
1 K	1 K	64 K • 4	1	4	G	Prime
1 K	2 K	64 K • 1	2	32	G	Barely
1 K	2 K	64 K • 4	2	8	G	Yes
1 K	3 K	64 K • 1	3	48	G	No
1 K	3 K	64 K • 4	3	12	G	Yes
1 K	4 K	64 K • 1	4	64	G	No
1 K	4 K	64 K • 1	4	16	G	Yes
512	2 K	64 K • 1	1	16	H	Yes
512	2 K	64 K • 4	1	4	H	Yes
512	4 K	64 K • 1	2	32	H	Barely
512	4 K	64 K • 4	2	8	H	Yes





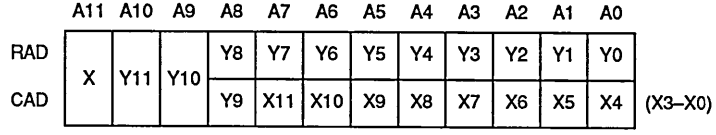
SHIFTER SIZE	1024
TOTAL BITS TRANSFERRED	16384
NUMBER OF ROWS	4

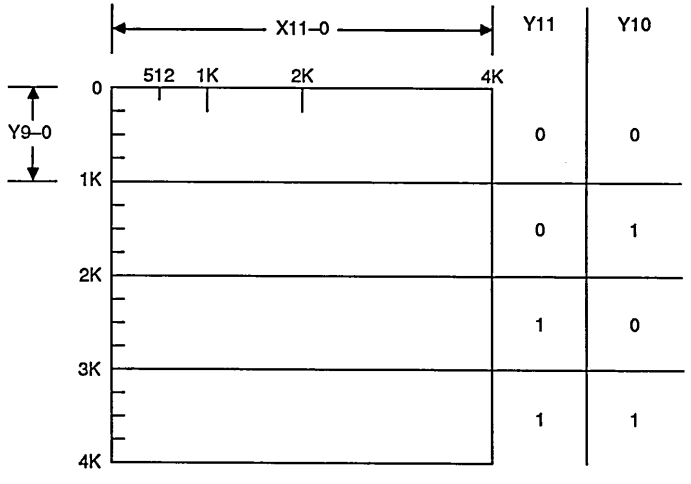
MEM_CONFIG_VALUE : 0000

RAM SIZE	1M • 1	1M • 4
MEMORY WIDTH	4K	4K
MAX DEVICES/PLANE	16	4

07785A 12-1

Figure 12-1 Memory Configuration A





SHIFTER SIZE	512
TOTAL BITS TRANSFERRED	8192
NUMBER OF ROWS	2

MEM_CONFIG_VALUE : 0001

RAM SIZE	256K • 1	256K • 4
MEMORY WIDTH	4K	4K
MAX DEVICES/PLANE	64	16

07785A 12-2

Figure 12-2 Memory Configuration B

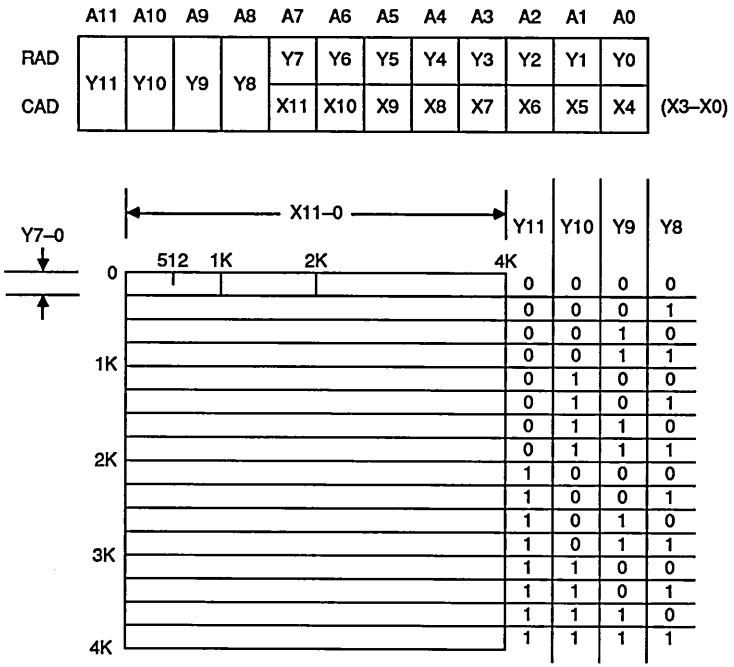


Figure 12-3 Memory Configuration C

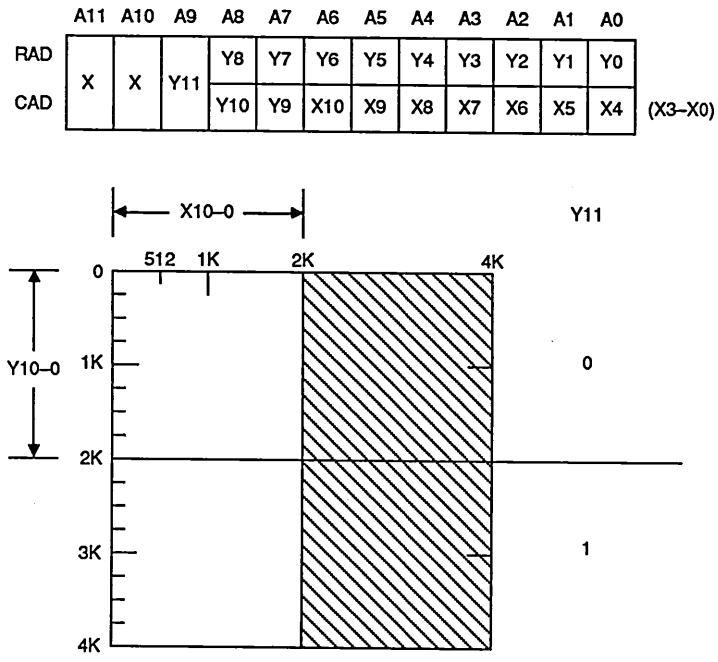


Figure 12-4 Memory Configuration D

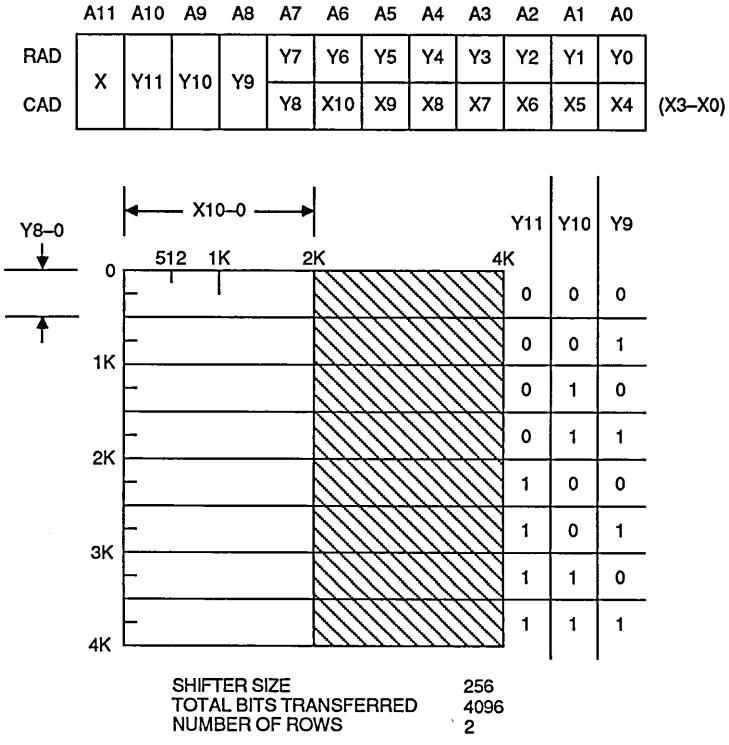


Figure 12-5 Memory Configuration E

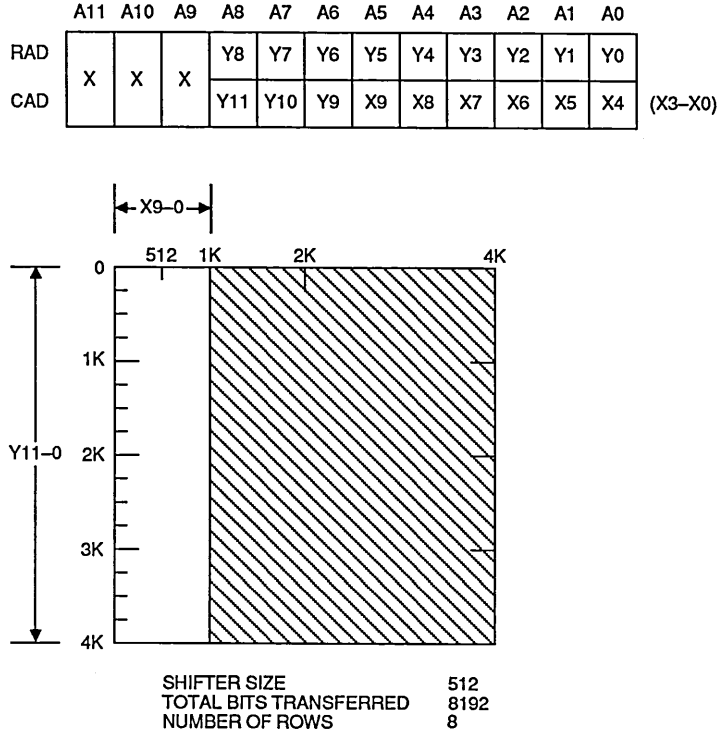


Figure 12-6 Memory Configuration F

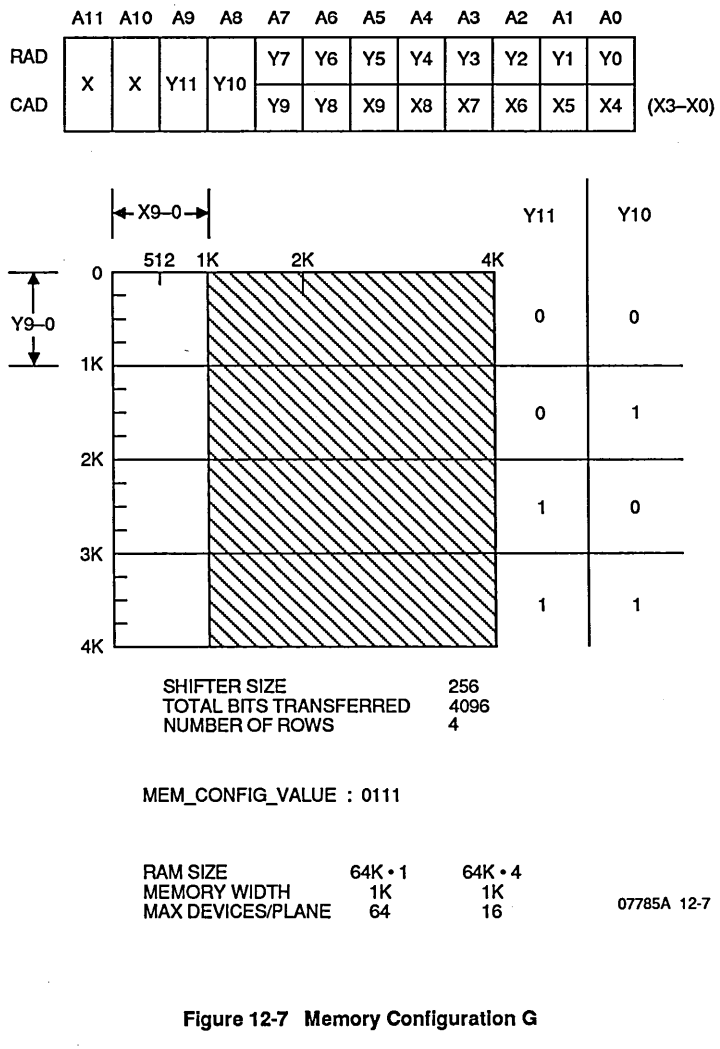


Figure 12-7 Memory Configuration G

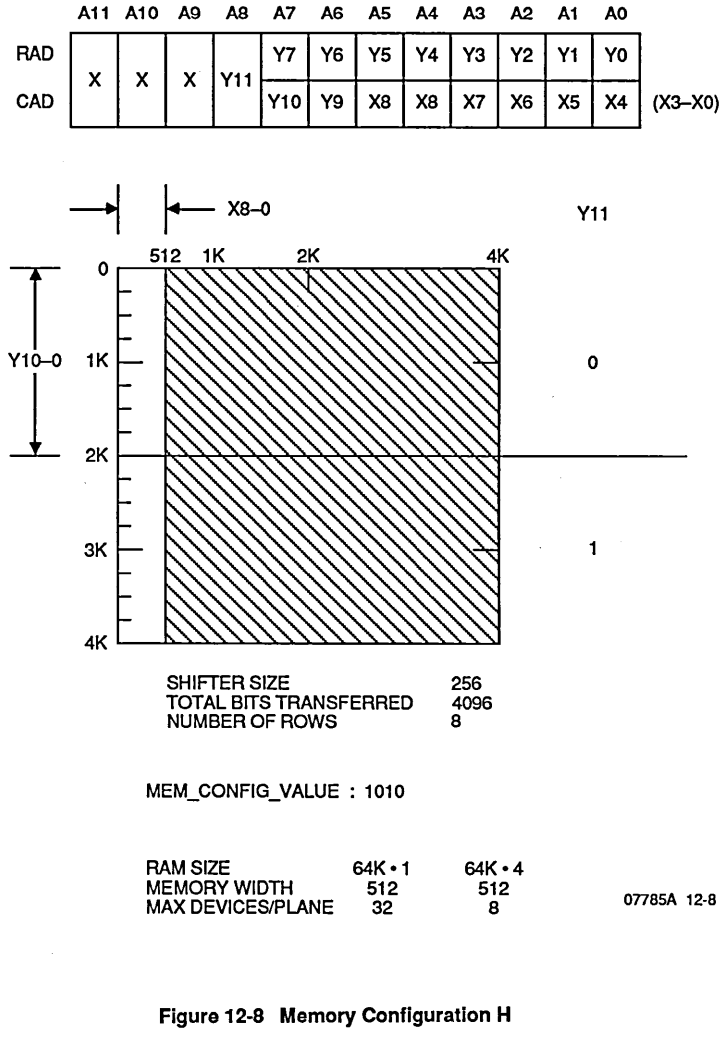


Figure 12-8 Memory Configuration H

CHAPTER 13

MISCELLANEA

13.1 HOW TO CRASH THE Am95C60

The Am95C60 is a microprogrammed processor, and it is possible to force it into an endless loop. The known ways to force the Am95C60 into an endless loop are listed below so that the user may more easily determine into which trap he has fallen.

13.1.1 Jump to Same Address

If the address in a Jump instruction resolves to that of the Jump instruction itself, the Am95C60 will execute that Jump forever. There are obvious variations on this theme such as two Jump instructions that point to each other.

13.1.2 Indirect Addressing Loops

If a standard operand address pair specifies indirect addressing mode and points to itself, the Am95C60 will spend forever repetitively resolving the address.

13.1.3 Invalid Copy Block

If the Block Size is 0,0, the following instructions that use the block size will never terminate. Copy Block, Transform Block, Input Block, and anything that uses the logical PEL.

13.1.4 Illegal Seed Fill

If a Fill Bounded Region or Fill Connected Region (or their Current analogs) is executed with incorrect activity or listen bits, the instruction will never complete. For every plane, the activity bit must be set or the listen bit must be set or both. If any plane is not physically implemented, the corresponding listen bit must be set.

13.2 THE RELATIONSHIP BETWEEN DATA BITS AND PIXELS

The relationship between system data bits (D15-D0) and display memory bus data bits (DM000-DM315) is not explained anywhere else in this manual. It is intended that this section will help clarify this topic.

13.2.1 Relationship Between DM Lines and Pixel Addresses

The naming convention for the DM (Display Memory) data bits is DMzxx where z is the plane number and xx is the data line number. The relationship between these lines and the x addresses of pixels is important because it determines the order in which bits must be serialized to appear on the screen properly. DMz15 is associated with those pixels whose x addresses end in 0000, and DMz00 is associated with those pixels whose x addresses end in 1111. Pixels with larger x addresses must appear further to the right on the screen. DMz15 must be serialized first.

13.2.2 Relationship Between D Bits and DM Bits

The system data bus bits are named D15-D0. The convention adopted in this manual (in the instruction formats, for example) is to show data bit 15 (D15) to the left of data bit 0 (D0). This may or may not be the convention your host processor uses.

There are six instructions that move information between the system data bus and the display memory. These instructions are listed below:

Input Block (by plane)
Input Block Current (by plane)
Output Block (by plane)
Output Block Current (by plane)
Store Immediate
Store Immediate Current

When the above instructions are executed, higher numbered D bits are connected to higher numbered DM bits. In the simplest case, D15 is connected to DMz15, D14 is connected to DMz14 and so on. In the cases of Block I/O by plane, this may be complicated by the ability to begin the operation on a pixel boundary that is not a word boundary. In the case of Store Immediate (Current), this is not a problem since the low order four bits of the x address must resolve to 0s.

13.2.3 Block I/O by Pixel

The relationship between system bus data bits (D15-D0) and display memory bits is shown in the following diagram. This is the data that is read or written during the first cycle of an Input Block or Output Block whose x address ends in 0000. The data will be shifted left for subsequent cycles according to the BIS bits.

D-line	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DM-line	15	15	15	15	14	14	14	14	13	13	13	13	12	12	12	12
X-Adrs	0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3
Plane	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3

13.2.4 Operands in the Set Bit Instructions

The Am95C60s extract bit-wise information from four instructions. These instructions are:

Set Activity Bits
Set Color Bits
Set Listen Bits
Set Search Color

The format of each of these instructions is in Figure 13-1.

The operand word is broken into four fields, one for each of the four Am95C60s that may respond. See the description of Set QPDM Position in Chapter 14. Each field contains four bits, one for each plane controlled by the Am95C60. The arrangement of bits within this field is:

P0 | P1 | P2 | P3

That is, the highest numbered bit controls plane 0 while the lowest numbered bit controls plane 3. This is consistent with the ordering of bits in Block I/O by pixel.

13.3 THE STACK

The Am95C60 maintains a push-down stack in display memory. This is used to contain return addresses for subroutines and is used for temporary storage by Fill Bounded Region and Fill Connected Region (and their Current analogs).

The stack location is specified by the user with Set Stack Boundaries instruction. Normally, the stack would be allocated when display memory is laid out and set when the system is initialized.

The stack is a column in display memory, is exactly one word wide, and is word aligned. It grows from lower Y addresses to higher Y addresses. If the stack grows to above the YT address, stack overflow has occurred and is signaled.

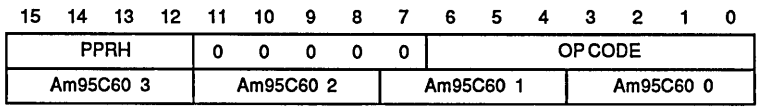
It is safest to allocate as much stack area as possible (one full column).

13.4 FIFO SIZE

The instruction FIFO is 16 words in depth. The Block Input FIFO and the Block Output are four words each.

13.5 TRANSFER CYCLE TIMING

The (first) transfer cycle for each scan line begins between 10 and 24 SYSCLK cycles after the rising edge of HSYNC.



07785A 13-1

Figure 13-1 Set Bit Instructions Format

Chapter 14

INSTRUCTION SET

The Am95C60 is a microprogrammed engine with a rich instruction set oriented toward graphics processing. These instructions are described in detail in the following pages.

14.1 INSTRUCTION SET GROUPED BY CLASSIFICATION

Drawing Primitives

Arc
Arc Current
Circle
Circle Current
Line
Line Current
Move Pen
Point
Point Current
String
String Current

Fill Instructions

Fill Bounded Region
Fill Bounded Region Current
Fill Connected Region
Fill Connected Region Current
Filled Rectangle
Filled Rectangle Current
Filled Triangle
Filled Triangle Current

Block Manipulation

Copy Block
Copy Block Current
Input Block
Input Block Current
Output Block
Output Block Current
Transform Block
Transform Block Current

System Control

Call
Control Clipping
Control Picking
Define Logical PEL
Inquire
Jump
No Operation
Output Current Pen Position
Pop Current Pen Position
Push Current Pen Position
Return
Set Activity Bits
Set Anti-aliasing Distance
Set Listen Bits
Set QPDM Position
Set Stack Boundaries
Signal
Store Current Pen Position

Display Control

Set Block Size
Set Character Font Base
Set Character Font Base Current
Set Clipping Boundary
Set Clipping Boundary Current
Set Color Bits
Set Search Color
Set Line Style
Set Line Style Phase
Set Picking Region
Set Picking Region Current
Set Scale Factor
Set Viewport Location
Set Viewport Location Current
Store Immediate
Store Immediate Current

Table 14-1 Instruction Set Ordered by Operation Code

00	No Operation	20	Set Color Bits	40	Move Pen	60	Reserved
01	Reserved	21	Set Search Color	41	Reserved	61	Reserved
02	Reserved	22	Set Block Size	42	Reserved	62	Reserved
03	Reserved	23	Set Character Font Base Current	43	Reserved	63	Set Chararacter Font Base
04	String Current	24	Set Clipping Boundary Current	44	String	64	Set Clipping Boundary
05	Reserved	25	Set Line Style	45	Reserved	65	Reserved
06	Reserved	26	Set Line Style Phase	46	Reserved	66	Reserved
07	Reserved	27	Set Viewport Location Current	47	Reserved	67	Set Viewport Location
08	Arc Current	28	Set Picking Region Current	48	Arc	68	Set Picking Region
09	Reserved	29	Set Scale Factor	49	Reserved	69	Reserved
0A	Circle Current	2A	Output Current Pen Position	4A	Circle	6A	Reserved
0B	Reserved	2B	Store Current Pen Position	4B	Reserved	6B	Reserved
0C	Line Current	2C	Push Current Pen Position	4C	Line	6C	Reserved
0D	Reserved	2D	Pop Current Pen Position	4D	Reserved	6D	Reserved
0E	Point Current	2E	Reserved	4E	Point	6E	Reserved
0F	Reserved	2F	Reserved	4F	Reserved	6F	Reserved
10	Filled Rectangle Current	30	Set Activity Bits	50	Filled Rectangle	70	Reserved
11	Reserved	31	Set Listen Bits	51	Reserved	71	Reserved
12	Filled Triangle Current	32	Call	52	Filled Triangle	72	Reserved
13	Reserved	33	Return	53	Reserved	73	Reserved
14	Fill Bounded Region Current	34	Control Clipping	54	Fill Bounded Region	74	Reserved
15	Reserved	35	Control Picking	55	Reserved	75	Reserved
16	Fill Connected Region Current	36	Define Logical PEL	56	Fill Connected Region	76	Reserved
17	Reserved	37	Set Anti-aliasing Distance	57	Reserved	77	Reserved
18	Input Block Current	38	Set QPDM Position	58	Input Block	78	Reserved
19	Reserved	39	Set Stack Boundaries	59	Reserved	79	Reserved
1A	Output Block Current	3A	Signal	5A	Output Block	7A	Reserved
1B	Reserved	3B	Store Immediate Current	5B	Reserved	7B	Store Immediate
1C	Copy Block Current	3C	Jump	5C	Copy Block	7C	Reserved
1D	Reserved	3D	Reserved	5D	Reserved	7D	Reserved
1E	Transform Block Current	3E	Reserved	5E	Transform Block	7E	Reserved
1F	Reserved	3F	Inquire	5F	Reserved	7F	Reserved

14.2 ALPHABETICAL LISTING OF INSTRUCTION SET

The instructions are ordered alphabetically; this is the only ordering that is not in some sense subjective. Each instruction begins on a new page.

The format of the instruction definition is the same for all instructions. The items that are covered are:

DESCRIPTION

What does this instruction do?

PARAMETERS

What operands are required for this instruction?

CPP

Where does this instruction leave the pen?

PERFORMANCE

How many SYSCLK cycles are required for this instruction?

The intent is to provide sufficient data for the user to predict the time that will be required for any operation. The Am95C60 is, however, a microprogrammed machine. While it is possible to predict the states it will enter for any given instruction, it is difficult and tedious to enumerate all the possible cases.

For some instructions, it is possible to specify a definite number; Set Color Bits always requires 112 SYSCLK cycles.

For some other instructions, it is possible to specify an overhead, and a number of cycles per instruction iteration. Arc requires 573 cycles overhead, 54 cycles per octant, and 15 cycles per pixel (for a solid line style with no anti-aliasing and no logical PEL).

For some other instructions, it is possible to specify an overhead, but nearly impossible to specify the number of cycles per iteration. An example is Fill Bounded Region; the time per pixel is strongly dependent on the size and shape of the region. In the cases where general prediction is impractical, we give examples.

All the performance numbers are given for absolute addressing. To correct these for other addressing modes, you may use the following adders:

Address Mode	Approximate Overhead
Viewport Unscaled	15 cycles for each address pair
Viewport Scaled	296 cycles for each address pair
Relative Unscaled	20 cycles for each address pair
Relative Scaled	284 cycles for each address pair
Indirect Addressing	34 cycles for each level of indirection

In the case of program mode, add 11 cycles per instruction word.

These numbers are all empirical and were measured on Revision B silicon (5.81) in January 1987.

EXAMPLES

How could one use this instruction?

CROSS-REFERENCES

Where to look in this book for further information.

COMMENTS

Anything that doesn't seem to fit anywhere else.

FORMAT

How does this instruction appear?

Arc
(Drawing Primitive)

Arc creates the image of a circular arc in display memory. Line styles, anti-aliasing, and logical PEL may be specified.

Arc requires 10 parameters:

AA is a 1-bit field that controls anti-aliasing if the Logical PEL is not enabled and controls the PEL source if the logical PEL is enabled.

SI is a 1-bit field that specifies whether the logical PEL is to be inverted. This applies only if the logical PEL is enabled.

M is a 1-bit field that specifies whether the logical PEL is to be stored only where the destination matches the search color. This applies only if the logical PEL is enabled.

LS is a 1-bit field that specifies whether the line style is to be solid or dashed-dotted.

EP is a 1-bit field that specifies whether the termination end point is to be drawn unconditionally or only if it falls within a dash or dot element. The first point (at START) is always drawn.

SOAXZ is a 3-bit field that specifies how the arc is to be drawn over the current contents of the display memory.

START specifies the address of the first pixel in the arc. START is specified with a standard operand address pair. If relative addressing is specified, START is calculated relative to the CPP. The arc is drawn counterclockwise from START to END.

CENTER is specified with a standard operand address pair. If relative addressing is specified, CENTER is calculated with relative to START. The center may lie outside display memory.

RADIUS is specified with a standard operand address pair and is defined as upper-right corner of the square that bounds the circle that contains the arc. This is shown in Figure 14-1. If relative addressing is specified, RADIUS is calculated relative to CENTER. An easy way to specify the radius is to use relative addressing mode, where the X and Y offset are both equal to the desired radius.

END is specified with a standard operand address pair. If relative addressing is specified, END is calculated relative to CENTER. The arc is drawn counterclockwise from START to END.

The Current Pen Position following this instruction is the END point.

PERFORMANCE (in SYSCLK cycles)

Instruction Overhead (PEL disabled)	573
Instruction Overhead (PEL enabled)	641
Octant Overhead	54 cycles
Cycles/Pixel (Plain vanilla arc)	15/pixel
Cycles/Pixel (Anti-aliased)	77/pixel
Cycles/Pixel (Dashed-Dotted)	15/pixel
Cycles/Pixel (1x1 PEL)	195/pixel
Cycles/Pixel (2x2 PEL)	241/pixel
Cycles/Pixel (4x4 PEL)	328/pixel

EXAMPLE

This examples draws an arc with the parameters listed below. This example uses absolute addresses for Start, Center, and End. It uses relative addressing for the radius. The line style is solid and the SOAXZ field specifies Graphical Set.

Start:	534,594
Center:	500,500
Radius:	100
End:	575,566

```
0548    *Arc with SOAXZ=5
0216    *Start is 534,594
0252
01F4    *Center is 500,500
01F4
8064    *Relative Radius 100,100
0064
023F    *End at 575,556
0236
```

CROSS-REFERENCES

Chapter 5: Addressing Modes and Scaling
Chapter 6: Line Texture
Chapter 7: Clipping and Picking
Chapter 8: Graphical Operations

COMMENTS

Arc assumes a pixel aspect ratio of 1 (square pixels). If this is not the case, the appearance on the display surface will be that of an elliptical arc.

More information must be supplied to the Am95C60 than is strictly required to define an arc. Since there is redundant information, it must be self-consistent. The end points may be calculated using the following equations:

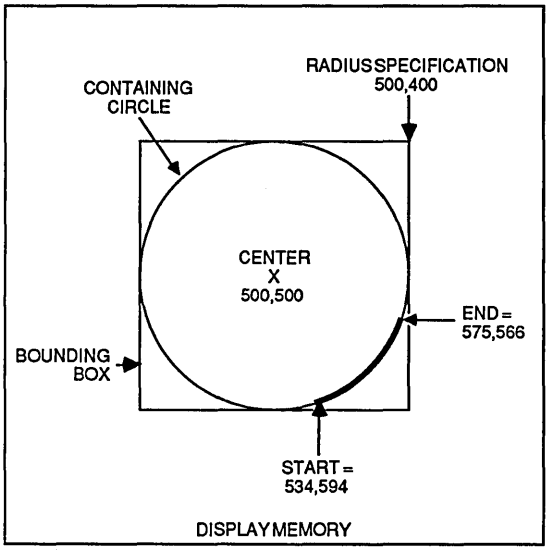
X = Center + Radius • COS Theta
Y = Center + Radius • SIN Theta

A function called ANG is provided in QASM to make it easier to draw arcs when you know the center, radius, and angles.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
AA	SI	M	LS	EP	SOAXZ			0	1	0	0	1	0	0	0	
AM		X/dX														(START)
0	0	Y/dY														
AM		X/dX														(CENTER)
0	0	Y/dY														
AM		X/dX														(RADIUS)
0	0	Y/dY														
AM		X/dX														(END)
0	0	Y/dY														

07785A 14-1

Arc Format



07785A 14-2

Figure 14-1 Arc

Arc Current
(Drawing Primitive)

Arc Current creates the image of a circular arc in display memory. Line styles, anti-aliasing, and logical PEL may be specified. The start point is the Current Pen Position.

Arc Current requires 9 parameters:

AA is a 1-bit field that controls anti-aliasing if the Logical PEL is not enabled and controls the PEL source if the logical PEL is enabled.

SI is a 1-bit field that specifies whether the logical PEL is to be inverted. This applies only if the logical PEL is enabled.

M is a 1-bit field that specifies whether the logical PEL is to be stored only where the destination matches the search color. This applies only if the logical PEL is enabled.

LS is a 1-bit field that specifies whether the line style is to be solid or dashed-dotted.

EP is a 1-bit field that specifies whether the termination end point is to be drawn unconditionally or only if it falls within a dash or dot element. The first point (at START) is always drawn.

SOAXZ is a 3-bit field that specifies how the arc is to be drawn over the current contents of the display memory.

CENTER is specified with a standard operand address pair. If relative addressing is specified, CENTER is calculated with relative to the Current Pen Position. The center may lie outside display memory.

RADIUS is specified with a standard operand address pair and is defined as the upper-right corner of the square that bounds the circle that contains the arc. This is shown in Figure 14-1. If relative addressing is specified, RADIUS is calculated relative to CENTER. An easy way to specify the radius is to use relative addressing mode, where the X and Y offset are both equal to the desired radius.

END is specified with a standard operand address pair. If relative addressing is specified, END is calculated relative to CENTER. The arc is drawn counterclockwise from START to END.

The Current Pen Position following this instruction is the END point.

PERFORMANCE (in SYSCLK cycles)

Instruction Overhead (PEL disabled)	516
Instruction Overhead (PEL enabled)	561
Octant Overhead	54 cycles
Cycles/Pixel (Plain vanilla arc)	15/pixel
Cycles/Pixel (Anti-aliased)	77/pixel
Cycles/Pixel (Dashed-Dotted)	15/pixel
Cycles/Pixel (1x1 PEL)	195/pixel
Cycles/Pixel (2x2 PEL)	241/pixel
Cycles/Pixel (4x4 PEL)	328/pixel

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
AA	SI	M	LS	EP	SOAXZ			0	0	0	0	1	0	0	0	
AM		X/dX														(CENTER)
0	0	Y/dY														
AM		X/dX														(RADIUS)
0	0	Y/dY														
AM		X/dX														(END)
0	0	Y/dY														

07785A 14-4a

Arc Current Format

EXAMPLE

This example draws an arc with the parameters listed below. This example uses absolute addresses for Center and End. It uses relative addressing for the radius. The Current Pen Position is assumed to be 534,594. The line style is solid and the SOAXZ field specifies Graphical Set.

Start: 534,594
Center: 500,500
Radius: 100
End: 575,566

```
0508    *Arc Current with SOAXZ=5
01F4    *Center is 500,500
01F4
8064    *Relative Radius 100,100
0064
023F    *End at 575,556
0236
```

CROSS-REFERENCES

Chapter 5: Addressing Modes and Scaling
Chapter 6: Line Texture
Chapter 7: Clipping and Picking
Chapter 8: Graphical Operations

COMMENTS

Arc Current assumes a pixel aspect ratio of 1 (square pixels). If this is not the case, the appearance on the display surface will be that of an elliptical arc.

More information is available to the Am95C60 than is strictly required to define an arc. Since there is redundant information, it must be self-consistent. The end points may be calculated using the following equations:

$$X = \text{Center} + \text{Radius} \cdot \text{COS Theta}$$
$$Y = \text{Center} + \text{Radius} \cdot \text{SIN Theta}$$

Call
(System Control)

Call causes the Am95C60 to enter Program mode, if it is not currently in Program mode, and pushes a return address of all 1s onto the stack. If the Am95C60 is already in Program mode, a subroutine is called; a return address (the address of the instruction following the call) with bit 12 not set is pushed onto the stack.

In Program mode, the Am95C60 fetches instructions from display memory rather than from the instruction FIFO. This allows the Am95C60 to execute stored programs, thus relieving the host of some burden. Figure 4-2 illustrates a program being called that in turn calls a subprogram.

Program mode is exited using a Return instruction (if not currently within a nested subroutine).

Call requires four parameters, plus an optional list of indices.

PP is a 2-bit field that specifies the bit plane number from which instructions are to be fetched.

I is a one-bit field that indicates whether a list of indices is present. If I is a one, then LOCATION is taken to be the top of a dispatch table, and the indices are pointers into the table. The dispatch table must be in plane zero.

Each index is processed as follows: The index (i) is multiplied by 2 and an X,Y address pair is fetched from LOCATION, LOCATION + (i*2). Execution begins at this X,Y address in the bit plane specified in PP. When an unnested Return is executed, bit 15 (the C bit) of the index is tested. If it is a one, another index is fetched, otherwise the Call terminates.

If I is a zero, LOCATION specifies the address in display memory from which instructions will be fetched and no index list is permitted.

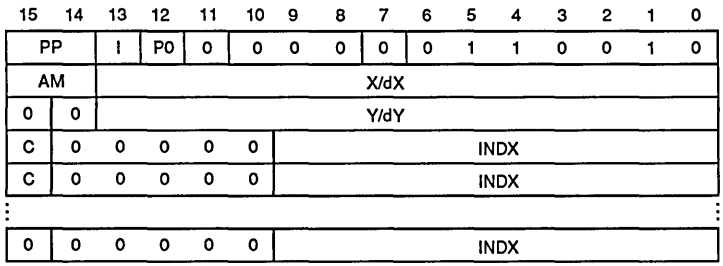
PO is a 1-bit field that makes the execution of the Call conditional. If PO is a one, the Call is executed only if bit 13 of the Y component of the Current Pen Position is a zero (implying CPPY is positive). If PO is a zero, the Call is executed unconditionally. If I is a one, the PO bit is ignored and the instruction is executed unconditionally.

LOCATION is specified as a standard operand address pair. This is the address of the subroutine if I is a zero, or the address of the dispatch table if I is a one. If relative addressing is used, LOCATION is calculated with respect to the Current Pen Position. The four low order bits of the X component of must resolve to all zeroes.

The Current Pen Position is unchanged by the execution of this instruction.

C is present in the list of indices. When each index has been used as a pointer into the dispatch table and an unnested Return has been executed, this bit is tested. If it is a one, the Am95C60 uses the next entry in the index list. If C is a zero, the Call instruction terminates.

INDX is the index into the dispatch table. Only the low order 10 bits are used. Bits 10 through 14 must be zeroes. Bit 15 is the C bit.



07785A 14-5

Call Format

PERFORMANCE

Call requires 140 SYSCLK cycles

EXAMPLE

This example calls a program, the first instruction of which is located at 512,500 in bit plane zero. If the Am95C60 is not in Program mode when this instruction is executed, a return of all 1s is pushed. If the Am95C60 is already in Program mode, the address of the instruction following the Call Program is pushed.

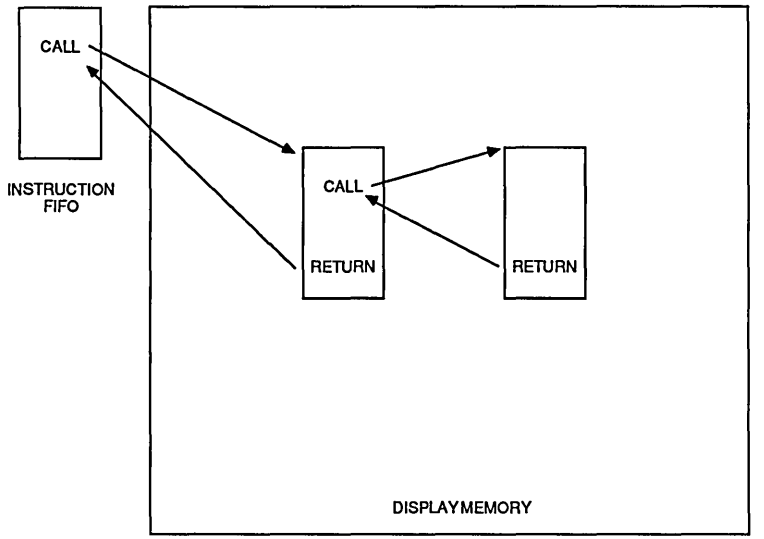
```
0032        *Plane Zero
0200        *Address to call
01F4
```

CROSS-REFERENCES

Return (Instruction)
Set Stack Boundaries (Instruction)

COMMENTS

The Y component of the stack pointer is incremented by 2 following the push. Any instruction may be executed in Program mode.



07785A 14-6

Figure 14-2 Program Mode

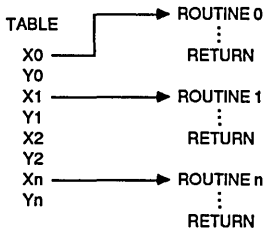


Figure 14-2b Call Table

Circle
(Drawing Primitive)

Circle creates the image of a circle in display memory. Line style, logical PEL and anti-aliasing may be specified.

Circle requires 8 parameters:

AA is a 1-bit field that controls anti-aliasing if the Logical PEL is not enabled and controls the PEL source if the logical PEL is enabled.

SI is a 1-bit field that specifies whether the logical PEL is to be inverted. This applies only if the logical PEL is enabled.

M is a 1-bit field that specifies whether the logical PEL is to be stored only where the destination matches the search color. This applies only if the logical PEL is enabled.

LS is a 1-bit field that specifies whether the line style is to be solid or dashed-dotted.

EP is a 1-bit field that specifies whether the termination end point is to be drawn unconditionally or only if it falls within a dash or dot element. The first point (at the bottom of the circle) is always drawn.

SOAXZ is a 3-bit field that specifies how the arc is to be drawn over the current contents of the display memory.

CENTER is specified with a standard operand address pair. If relative addressing is specified, CENTER is calculated with relative to Current Pen Position. The center may lie outside display memory.

RADIUS is specified with a standard operand address pair and is defined as the upper-right corner of the box that bounds the circle. This is shown in figure 14-3. If relative addressing is used, the radius is calculated with respect to CENTER. The easiest way to specify the radius is with relative addressing where both the X and Y offsets are equal to the desired radius.

The Current Pen Position following this instruction is the CENTER point.

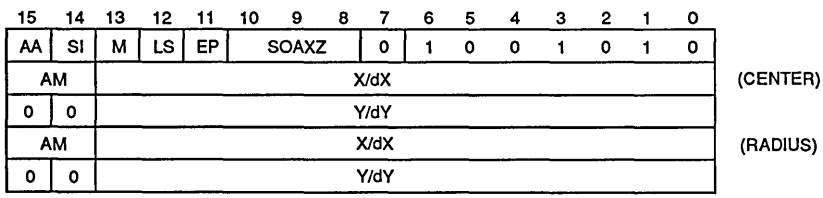
PERFORMANCE

Instruction Overhead (PEL disabled	198
Instruction Overhead (PEL enabled)	306
Octant Overhead	54 cycles
Cycles/Pixel (Plain vanilla circle)	15/pixel
Cycles/Pixel (Anti-aliased)	77/pixel
Cycles/Pixel (Dashed-Dotted)	15/pixel
Cycles/Pixel (1x1 PEL)	193/pixel
Cycles/Pixel (2x2 PEL)	279/pixel
Cycles/Pixel (4x4 PEL)	324/pixel

EXAMPLE

This example draws a circle with the center at 500,500 and a radius of 100. The example uses absolute addressing for the center and relative addressing for the radius. The line style is solid without anti-aliasing. The SOAXZ field is set to 5 to obtain Graphical SET.

```
054A    *Circle
01F4    *Center at 500,500
01F4
8064    *Relative Radius of 100
0064
```



07785A 14-6

Circle Format

CROSS-REFERENCES

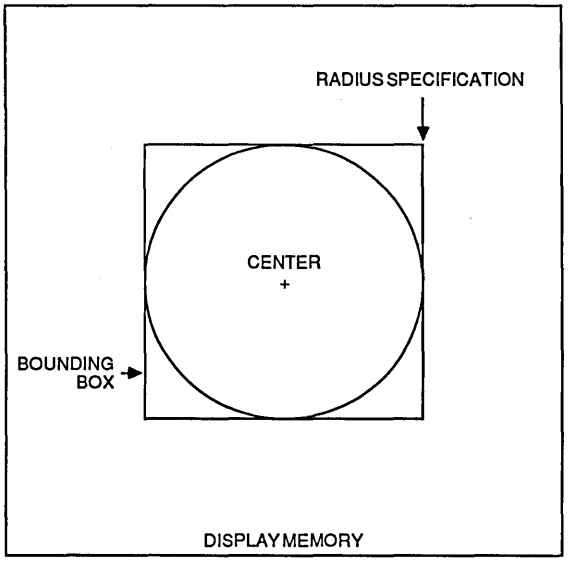
Chapter 5: Addressing Modes and Scaling
Chapter 6: Line Texture
Chapter 7: Clipping and Picking
Chapter 8: Graphical Operations

COMMENTS

Circle assumes pixels with an aspect ratio of 1 (square pixels). If this is not the case, the appearance on the display surface will be that of an ellipse. The X and Y scale factors apply only to the center (and possibly to the radius).

Any addressing mode may be used for either operand. Viewport addressing will probably not produce useful results for the radius.

The circle is drawn beginning at the bottom with a full-length dot or dash going counterclockwise. The last point is immediately to the left of the first point.



07785A 14-11

Figure 14-3 Circle

Circle Current
(Drawing Primitive)

Circle Current creates the image of a circle in display memory. Line style, logical PEL and anti-aliasing may be specified.

Circle Current requires 7 parameters:

AA is a 1-bit field that controls anti-aliasing if the Logical PEL is not enabled and controls the PEL source if the logical PEL is enabled.

SI is a 1-bit field that specifies whether the logical PEL is to be inverted. This applies only if the logical PEL is enabled.

M is a 1-bit field that specifies whether the logical PEL is to be stored only where the destination matches the search color. This applies only if the logical PEL is enabled.

LS is a 1-bit field that specifies whether the line style is to be solid or dashed-dotted.

EP is a 1-bit field that specifies whether the termination end point is to be drawn unconditionally or only if it falls within a dash or dot element. The first point (at the bottom of the circle) is always drawn.

SOAXZ is a 3-bit field that specifies how the arc is to be drawn over the current contents of the display memory.

RADIUS is specified with a standard operand address pair and is defined as the upper-right corner of the box which bounds the circle. This is shown in figure 14-3. If relative addressing is used, the radius is calculated with respect to Current Pen Position. The easiest way to specify the radius is with relative addressing where both the X and Y offsets are equal to the desired radius.

The Current Pen Position following this instruction is unchanged.

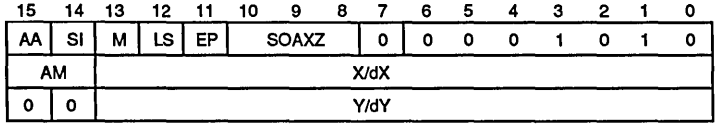
PERFORMANCE

Instruction Overhead (PEL disabled)	132
Instruction Overhead (PEL enabled)	234
Octant Overhead	54 cycles
Cycles/Pixel (Plain vanilla circle)	15/pixel
Cycles/Pixel (Anti-aliased)	77/pixel
Cycles/Pixel (Dashed-Dotted)	15/pixel
Cycles/Pixel (1x1 PEL)	193/pixel
Cycles/Pixel (2x2 PEL)	279/pixel
Cycles/Pixel (4x4 PEL)	324/pixel

EXAMPLE

This example draws a circle with the center at the Current Pen Position and a radius of 100. The example uses relative addressing for the radius. The line style is solid without anti-aliasing. The SOAXZ field is set to 5 to obtain Graphical SET.

```
050A      *Circle
8064      *Relative Radius of 100
0064
```



07785A 14-9

Circle Current Format

CROSS-REFERENCES

Chapter 5: Addressing Modes and Scaling
Chapter 6: Line Texture
Chapter 7: Clipping and Picking
Chapter 8: Graphical Operations

COMMENTS

Circle Current assumes pixels with an aspect ratio of 1 (square pixels). If this in not the case, the appearance on the display surface will be that of an ellipse. Only the radius will be scaled.

Viewport addressing will probably not produce useful results for the radius.

Control Clipping
(System Control)

Control Clipping turns the clipping feature on or off according to the Clipping Enable (CE) bit of the instruction.

Control Clipping requires a single parameter:

CE specifies whether clipping is to be enabled or disabled. If CE is a 1, clipping is enabled. If CE is a 0, clipping is disabled.

The Current Pen Position is not affected by this instruction.

PERFORMANCE

Control Clipping requires 45 SYSCLK cycles.

EXAMPLES

The first example enables clipping; the second example disables clipping.

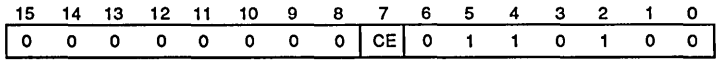
```
00B4        *Enable Clipping
0034        *Disable Clipping
```

CROSS-REFERENCES

Chapter 7: Clipping and Picking
Set Clipping Boundary (Instruction)

COMMENTS

The clipping boundary is unaffected by the execution of this instruction.



Control Clipping Format

07785A 14-10

Control Picking
(System Control)

Control Picking turns picking on or off according to the Picking Enable (PE) bit of the instruction.

Control Picking requires a single parameter:

PE specifies whether picking is to be enabled or disabled. If PE is a **1**, picking will be enabled. If PE is a **0**, picking will be disabled.

The Current Pen Position is not affected by this instruction.

PERFORMANCE

Control Picking requires 40 SYSCLK cycles.

EXAMPLES

Two examples are shown. The first example enables picking, and the second example disables picking.

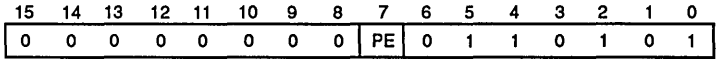
```
00B5        *Enable Picking
0035        *Disable Picking
```

CROSS-REFERENCE

Chapter 7: Clipping and Picking
Set Picking Region (Instruction)
Signal (Instruction)

COMMENTS

When picking is enabled, writes to display memory are disabled. The picking boundary is unaffected by the execution of this instruction.



07785A 14-11

Control Picking Format

Copy Block
(Block Manipulation)

Copy Block copies a block of data from the source image to the destination area. The source image is applied to the destination area under control of the SI, SOAXZ, and M fields.

Copy Block requires six parameters:

Single Plane (SP) controls whether one or all planes are taken to contain the source. If SP is a 1, then a single plane is taken to contain the source image that is replicated for all planes whose activity bits are set. The source plane is specified by the FP field in the Set Character Font Base instruction. If SP is a 0, then all planes whose activity bits are set contain the source image.

Source Invert (SI) controls whether the source image is inverted (1's complement) before processing. If SI is a 1, the image will be inverted. If SI is a 0, the image will not be inverted.

Match (M) controls whether color matching is to take place. If M is a 1, then bits in the destination operand will be modified only if the current destination color matches the search color.

SOAXZ controls the manner in which the source operand is applied to the destination operand.

SOURCE is specified using a standard operand address pair. This specifies the upper-left corner of the source image. Any addressing mode may be used. If relative addressing is specified, SOURCE is calculated with respect to the Current Pen Position. The size of the operands will have been previously specified using Set Block Size.

DESTINATION is specified using a standard operand address pair. This specifies the upper-left corner of the destination area. Any addressing mode may be used. If relative addressing is specified, DESTINATION is calculated with respect to the SOURCE.

Current Pen Position is moved to SOURCE.

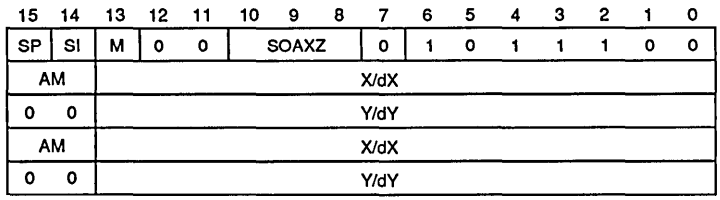
PERFORMANCE (in SYSCLK cycles)

Instruction Setup	218 Cycles
Scan Line Setup (No invert, no match)	36 Cycles
Scan Line Setup (Single Plane Source)	42 Cycles
Scan Line Setup (Source Invert)	40 Cycles
Scan Line Setup (Match Enable)	40 Cycles
Per Destination Word (Logical SET)	19 Cycles
Per Destination Word (Single Plane Source)	25 Cycles
Per Destination Word (Source Invert)	30 Cycles
Per Destination Word (Match Enable)	276 Cycles

(Partial Words at the ends of scan lines require a full word time)

EXAMPLES

The first example illustrates a simple copy. The M bit is not set and SOAXZ field specifies a logical SET. The source image is at 500,500, and the



07785A 14-12

Copy Block Format

destination area is at 1000,1000. The size of the operands have been set with Set Block Size. The operands may overlap arbitrarily.

```
005C
01F4    *Source operand address pair
01F4
03E8    *Destination operand address pair
03E8
```

The second example illustrates a more complex copy. The M bit is set resulting in destination locations being altered only if the current content matches the search color. The source image is at 500,500, and the destination area is at 1000,1000. The size of the operands have been set with Set Block Size.

```
205C    *M bit set
01F4    *Source operand address pair
01F4
03E8    *Destination operand address pair
03E8
```

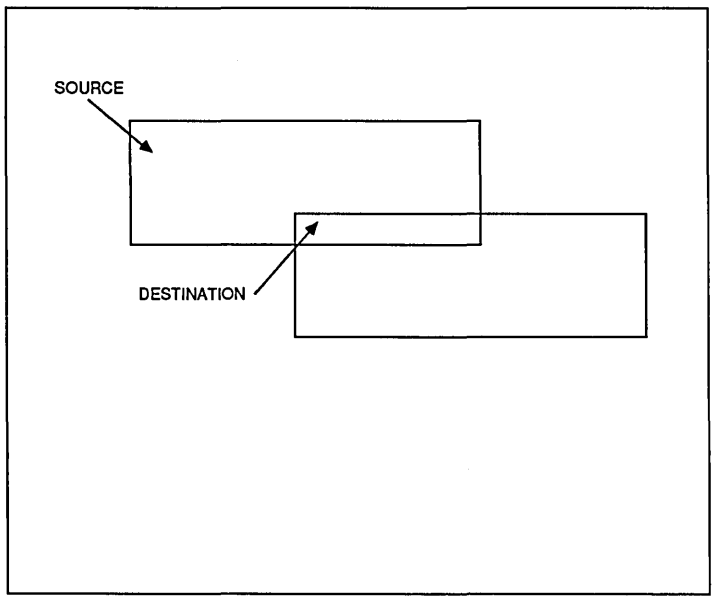
CROSS-REFERENCES

Chapter 8: Graphical Operations
Set Block Size (Instruction)

COMMENTS

Performance is not affected by the relative alignment of the operands.

The M bit may be used to confine the activity of Copy Block to a region that may be irregular. The pixels that are to be modified must first be set to some color and the search color set to the same value. If the M bit is a one, only those pixels that are the same as the search color will be affected.



07785A 14-13

Figure 14-4 Copy Block

Copy Block Current
(Block Manipulation)

Copy Block Current copies a block of data from the source image to the destination area. The source image is applied to the destination operand under control of the SI, SOAXZ, and M fields. The Current Pen Position is the upper-left corner of the source operand.

Copy Block Current requires five operands:

SP controls whether one or all planes are taken to contain the source. If SP is a 1, then a single plane is taken to contain the source image. The plane is specified by the FP field in the Set Character Font Base instruction. If SP is a 0, then all planes whose activity bits are set contain the source image.

SI controls whether the source image is inverted (1's complement) before processing. If SI is a 1 the image will be inverted. If SI is a 0, the image will not be inverted.

M controls whether color matching is to take place. If M is a 1, then bits in the destination operand will be modified only if the current destination color matches the search color.

SOAXZ is a 3-bit field that controls how the source operand is applied to the current contents of display memory.

DESTINATION is specified using a standard operand address pair that specifies the upper-left corner of the destination area. Any addressing mode may be used. If relative addressing is specified, DESTINATION is calculated with respect to the Current Pen Position.

Current Pen Position is not affected by this instruction.

PERFORMANCE (in SYSCLK cycles)

Instruction Setup	166 Cycles
Scan Line Setup (No invert, match)	36 Cycles
Scan Line Setup (Single Plane Source)	42 Cycles
Scan Line Setup (Source Invert)	40 Cycles
Scan Line Setup (Match Enable)	40 Cycles
Per Destination Word (Logical SET)	19 Cycles
Per Destination Word (Single Plane Source)	25 Cycles
Per Destination Word (Source Invert)	30 Cycles
Per Destination Word (Match Enable)	276 Cycles

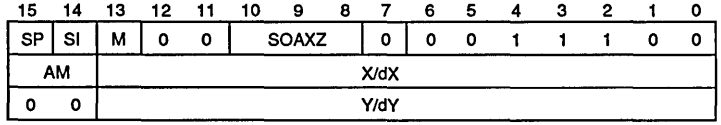
(Partial Words at the ends of scan lines require a full word time)

EXAMPLES

The first example illustrates a simple copy. The M bit is not set and SOAXZ field specifies a logical SET. The source image is at the Current Pen Position, and the destination area is at 1000,1000. The size of the block to be copied will have been set with Set Block Size. The operands may overlap arbitrarily.

```
001C
03E8    *Destination operand address pair
03E8
```

The second example illustrates a more complex copy. The M bit is set, resulting in destination locations being altered only if the current content matches the search color. The source image is at the Current Pen Position, and the destination area



07785A 14-1

Copy Block Current Format

is at 1000,1000. The size of the block to be copied will have been set with Set Block Size.

```
201C    *M bit set
03E8    *Destination operand address pair
03E8
```

CROSS-REFERENCES

Chapter 8: Graphical Operations
Set Block Size (Instruction)

COMMENTS

Performance is not affected by the relative alignment of the operands.

The M bit may be used to confine the activity of Copy Block Current to a region which may be irregular. The pixels that are to be modified must first be set to some color and the search color set to the same value. If the M bit is a one, only those pixels that are the same as the search color will be affected.

Define Logical PEL
(System Control)

Define Logical PEL specifies the pattern to be used in the drawing instructions that can use a logical PEL. These instructions are Arc, Circle, Line and Point. The logical PEL may be any pattern and is stored in display memory.

Two parameters are required:

E is a 1-bit field that indicates whether the logical PEL is to be enabled. A 1 enables the logical PEL; a 0 disables it.

PEL is specified with a nonstandard address pair. This points to the upper-left corner of the area in display memory that contains the logical PEL. The values will be multiplied by 16 prior to being used. This means that the logical PEL must start on a 16 pixel boundary in both dimensions.

The Current Pen Position is not affected by the execution of this instruction.

PERFORMANCE

Define Logical PEL requires 56 SYSCLK cycles.

EXAMPLES

This example disables the logical PEL.

```
0036    *Disable Logical PEL
xxxx    *Ignored
```

This example enables the logical PEL and sets the logical PEL address to 1024,1024.

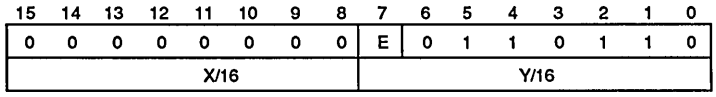
```
00B6    *Enable
4040    *40 hex = 64 dec
```

CROSS-REFERENCES

Chapter 6: Line Texture
Arc (Instruction)
Circle (Instruction)
Line (Instruction)
Point (Instruction)
Set Block Size (Instruction)

COMMENTS

Logical PEL cannot be used with anti-aliasing.



07785A 14-15

Define Logic PEL Format

Fill Bounded Region
(Fill Instruction)

Fill Bounded Region writes the current drawing color into every pixel in the region. The region is defined as that collection of points contained within a boundary. The boundary is a group of pixels, all of the same color that completely surround the region. The boundary color is established using the Set Search Color instruction. The drawing color MUST be set the same as the search color.

One parameter is required by Fill Bounded Region:

SEED is specified with a standard operand address pair. If relative addressing is specified, SEED will be calculated with respect to the Current Pen Position. SEED is the location at which the search begins. Its color is ignored.

CPP is left at SEED.

PERFORMANCE

Instruction Setup Time 200 SYSCLK Cycles

Cycles/Pixel: The actual execution timing of Filled Connected Region is very strongly dependent on the size, shape, and alignment of the region as well as alignment of the seed. The time approaches 5.56 cycles/pixel assuming an infinitely large region. The following table shows some actual measurements. As always, the times are given in SYSCLKS/pixel.

EXAMPLE

This example is illustrated in Figure 14-5. The connected pattern of Xs is the boundary of an area. This pattern could have been drawn using Line instructions for example.

Fill Bounded Region with SEED anywhere inside the pattern will fill the area up to but not including the boundary itself (but since the fill color MUST be the same as the boundary color, this is moot).

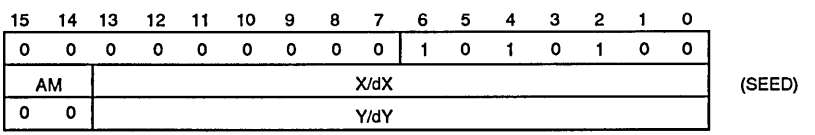
0054
0100 *SEED anywhere inside the boundary
0200

Rectangular Region

Corner	Corner	Area	Seed	C/Pixel	Comment
15,100	992,400	291824	480,200	5.9	
16,100	992,400	291525	480,200	6.7	Edge Alignment
14,100	992,400	292123	480,200	5.9	
15,100	992,400	291824	479,200	7.2	Seed Alignment
15,100	992,400	291824	481,200	7.2	

Circular Region

Center	Radius	Area	Seed	C/Pixel	Comment
480,480	400	500145	480,480	8.6	
481,480	400	500145	480,480	8.6	
479,480	400	500145	480,480	8.6	
480,480	400	500145	479,480	10.4	Seed Alignment
480,480	400	500145	481,480	10.4	



07785A 14-16

Fill Bounded Region Format

CROSS-REFERENCES

Set Search Color (Instruction)
Set Color Bits (Instruction)
Fill Connected Region (Instruction)

COMMENTS

A single-pixel wide diagonal line will behave as a boundary.

If clipping is enabled, no pixels outside the clipping region will be affected.

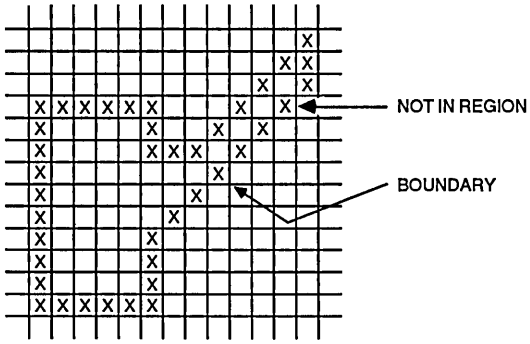
Multiple Am95C60s compare colors over the entire system (assuming MATIN and MATOUT are properly connected).

Planes whose listen bits are a 1 do not participate in color comparisons but may be written into.

Planes whose activity bits are a 0 may participate in color comparisons but will not be written into.

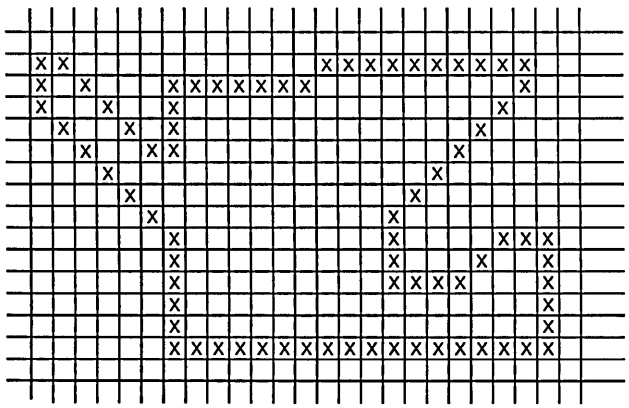
The stack usage can be estimated as follows. A simple convex polygon will require four entries total. If the polygon is concave, two entries will be required per concavity. Four entries will be required per island.

The activity bits and listen bits MUST be such that every plane in the system is either active or ignored (or both). This means that the activity bit or the listen bit must be set. If this condition is not met, Fill Connected Region will never complete.



07785A 14-71

The pixels contained in the diagonal area to the right of the main area are not in the Bounded Region. There is a boundary as indicated.



07785A 14-17

Figure 14-5 Fill Bounded Region

Fill Bounded Region Current
(Fill Instruction)

Fill Bounded Region Current writes the current drawing color into every pixel in the region. The region is defined as that collection of points contained within a boundary. The boundary is a group of pixels, all of the same color, that completely surrounds the region. The boundary color is established using the Set Search Color instruction. The drawing color MUST be set the same as the search color.

No parameters are required by Fill Bounded Region Current.

CPP is unchanged.

PERFORMANCE

Instruction Setup 140 SYSCLK Cycles

Cycles/Pixel: See the timing notes under Fill Bounded Region for time per pixel numbers.

EXAMPLES

This example is illustrated in Figure 14-5. The connected pattern of Xs is the boundary of an area. This pattern could have been drawn using Line instructions for example.

Fill Bounded Region Current with the CPP anywhere inside the pattern will fill the area up to but not including the boundary (but since the fill color MUST be the same as the boundary color, this is moot).

0014

CROSS-REFERENCES

Set Search Color (Instruction)
Set Color Bits (Instruction)
Fill Connected Region (Instruction)

COMMENTS

A single-pixel wide diagonal line will behave as a boundary.

If clipping is enabled, no pixels outside the clipping region will be affected.

Multiple Am95C60s compare colors over the entire system (assuming MATIN and MATOUT are properly connected).

Planes whose listen bits are a 1 do not participate in color comparisons but may be written into.

Planes whose activity bits are a 0 do participate in color comparisons but will not be written into.

The activity bits and listen bits MUST be such that every plane in the system is either active or ignored (or both). This means that the activity bit or the listen bit must be set. If this condition is not met, Fill Bounded Region Current will never complete.

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

07785A 14-18

Fill Bounded Region Current Format

Fill Connected Region
(Fill Instruction)

Fill Connected Region fills a region with the current drawing color. The region is defined as any connected group of pixels that are the same color as the seed point.

A single parameter is required by Fill Connected Region:

SEED is specified with a standard operand address pair. If relative addressing is used, SEED is calculated with respect to CPP. SEED is the address of a pixel anywhere within the connected region.

CPP is left at the SEED point.

PERFORMANCE

Instruction Setup Time 200 SYSCLK Cycles

The actual execution timing of Filled Connected Region is very strongly dependent on the size, shape, and alignment of the region as well as alignment of the seed. The time approaches 5.56 cycles/pixel assuming an infinitely large region. The following table shows some actual measurements. As always, the times are given in SYSCLKS/pixel.

EXAMPLE

Fill Connected Region is illustrated in Figure 14-6. The irregular shape marked with Xs is two groups of pixels that are all the same color. If Fill Connected Region is executed and the SEED point is any of the pixels marked with a X, all of the pixels in that group will be filled with the current drawing color.

```
0056
0100    *SEED
0200
```

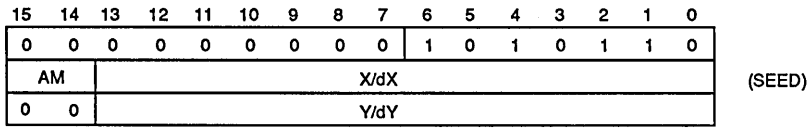
CROSS-REFERENCES

Fill Bounded Region (Instruction)
Set Color Bits (Instruction)

COMMENTS

The search color is changed to the same color as the SEED pixel (that is, the drawing color) following the execution of this instruction.

Rectangular Region Corner	Corner	Area	Seed	C/Pixel	Comment
15,100	992,400	294378	480,200	6.1	
16,100	992,400	294077	480,200	6.1	
14,100	992,400	294679	480,200	6.1	
15,100	992,400	294378	479,200	7.3	Seed Alignment
15,100	992,400	294378	481,200	7.3	



07785A 14-24

Fill Connected Region Format

If clipping is enabled, no pixels outside the clipping region will be written into.

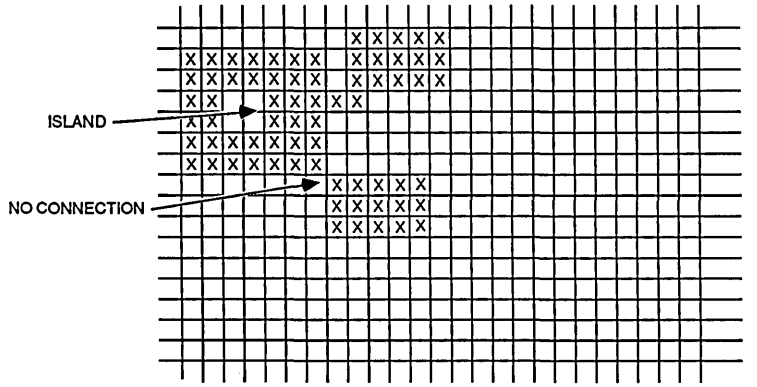
Planes whose listen bits are a 1 do not participate in color matches but may be written into.

Planes whose activity bits are a 0 may participate in color matches but will not be written into.

Connectivity is 4-way only. Pixels that are displaced in both X and Y are not connected to each other (but may be both connected to a common pixel).

Stack usage may be estimated as follows. A simple convex polygon will require four entries. If the polygon is concave, two entries will be required for each concavity. Four entries will be required for each island.

The activity bits and listen bits MUST be such that every plane in the system is either active or ignored (or both). This means that the activity bit or the listen bit must be set. If this condition is not met, Fill Connected Region will never complete.



07785A 14-20

Figure 14-6 Fill Connected Region

Fill Connected Region Current
(Fill Instruction)

Fill Connected Region Current fills a region with the current drawing color. The region is defined as any connected group of pixels that are the same color as the SEED point.

No parameters are required by Fill Connected Region Current.

CPP is unchanged.

PERFORMANCE

Instruction Setup 140 SYSCLK Cycles

For time/pixel measurements, see Fill Connected Region.

EXAMPLE

Fill Connected Region Current is illustrated in Figure 14-6. The irregular shape marked with Xs is two groups of pixels that are all the same color. If Fill Connected Region is executed and the CPP is any of the pixels marked with a X, all of the pixels in that group will be filled with the current drawing color.

0016

CROSS-REFERENCES

Fill Bounded Region (Instruction)
Set Color Bits (Instruction)

COMMENTS

The search color is changed to the same color as the SEED pixel (the drawing color) following the execution of this instruction.

If clipping is enabled, no pixels outside the clipping region will be written into.

Planes whose listen bits are a 1 do not participate in color matches but may be written into.

Planes whose activity bits are a 0 may participate in color matches but will not be written into.

Connectivty is 4-way only. Pixels that are displaced in both X and Y are not connected to each other (but may be both connected to a pixel).

Stack usage may be estimated as follows. A simple convex polygon will require four entries. If the polygon is concave two entries will be required for each concavity. Four entries will be required for each island.

The activity bits and listen bits MUST be such that every plane in the system is either active or ignored (or both). This means that the activity bit or the listen bit must be set. If this condition is not met, Fill Connected Region Current will never complete.

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

07785A 14-21

Fill Connected Region Current Format

Filled Rectangle
(Fill Instruction)

Filled Rectangle creates the image of a rectangle in display memory and fills it using the current drawing color. The current drawing color is applied to the background under control of the SOAXZ field.

Three parameters are required by this instruction:

SOAXZ is a 3-bit field that indicates how the drawing color is to be applied to the current contents of the region.

START is specified with a standard operand address pair. This is the upper-left corner of the rectangle. If relative addressing is used, START will be calculated with respect to CPP.

END is specified with a standard operand address pair. This is the lower-right corner of the rectangle. If relative addressing is specified, END will be calculated with respect to START.

The CPP is set to START.

PERFORMANCE (in SYSCLK cycles)

Instruction Setup Time 238 SYSCLK Cycles

The time per pixel very strongly depends on the size and horizontal alignment of the rectangle being filled. It also depends on the SOAXZ operation. For an infinitely large rectangle, the time approaches 0.375 SYSCLK cycles per pixel. The following table shows some measured numbers.

SOAXZ	Corner	Corner	Area	C/Pixel
SET	16,100	96,200	8181	1.01
SET	15,100	96,200	8282	1.09
SET	17,100	96,200	8080	1.02
SET	16,100	496,200	48682	0.49
XOR	16,100	496,200	48682	0.89

EXAMPLE

The following example creates a filled rectangle with opposite corners at 500,500 and 1000,1000. Since the SOAXZ field is zero, the current drawing color is used to fill the rectangle without regard to the current contents of the area affected.

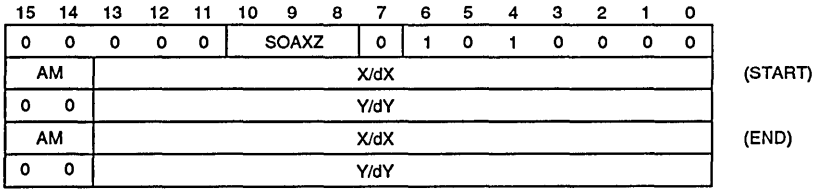
```
0050
01F4    *Start
01F4
03E8    *End
03E8
```

CROSS-REFERENCES

Chapter 8: Graphical Operations

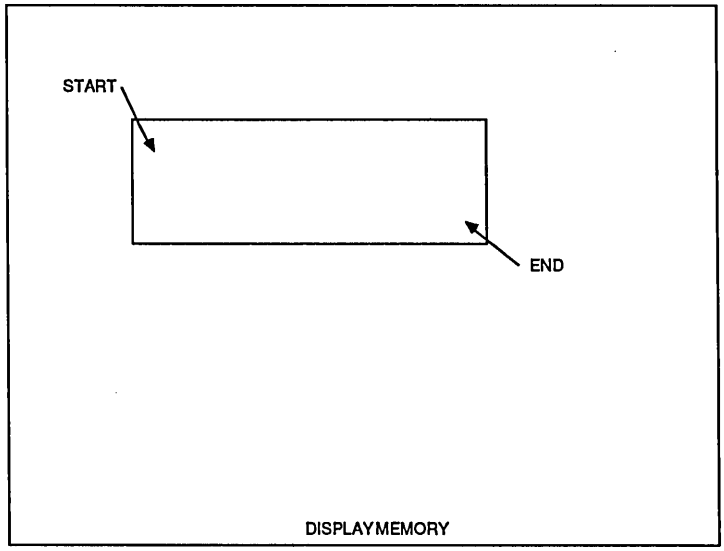
COMMENTS

Both the X and Y components of the END address must resolve to a value greater than the respective components of the START address.



07785A 14-22

Filled Rectangle Format



07785A 14-23

Figure 14-7 Filled Rectangle

Filled Rectangle Current
(Fill Instruction)

Filled Rectangle Current creates the image of a rectangle in display memory and fills it using the current drawing color. The current drawing color is applied to the background under control of the SOAXZ field. The upper-left corner of the rectangle is the Current Pen Position.

Two parameters are required by this instruction:

SOAXZ is a 3-bit field that indicates how the drawing color is to be applied to the current contents of the region.

END is specified with a standard operand address pair. This is the lower-right corner of the rectangle. If relative addressing is specified, END will be calculated with respect to the Current Pen Position.

The CPP is unchanged.

PERFORMANCE (in SYSCLK cycles)

Instruction Setup Time 135 SYSCLK Cycles

For time per pixel measurements, see Filled Rectangle.

EXAMPLE

The following example creates a filled rectangle with one corner at CPP and the other corner displaced by 100,100. Relative addressing is used. Since the SOAXZ field is five, the current drawing color is used to fill the rectangle without regard to the current color of the area affected.

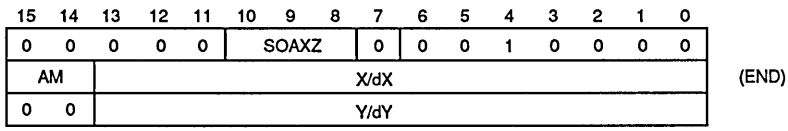
```
0510
8064      *END
3F9C      *Observe negative relative Y
          component
```

CROSS-REFERENCES

Chapter 8: Graphical Operations

COMMENTS

The END address must resolve to greater than the Current Pen Position in both components. If not, nothing useful will happen.



07785A 14-1

Filled Rectangle Current Format

Filled Triangle
(Fill Instruction)

Filled Triangle creates the image of a triangle in display memory and fills it with the current drawing color. The SOAXZ field is used to determine how the present contents of the area are combined with the current color.

Four parameters are required by Filled Triangle:

SOAXZ is a 3-bit field that indicates how the drawing color is to be applied to the current contents of the region.

START is specified with a standard operand address pair. This is the first vertex of the triangle. If relative addressing is used, START will be calculated with respect to CPP.

MID is specified with a standard operand address pair. This is the second vertex of the triangle. If relative addressing is used, MID will be calculated with respect to START.

END is specified with a standard operand address pair. This is the third vertex of the triangle. If relative addressing is used, END will be calculated with respect to START.

CPP is set to START following this instruction.

PERFORMANCE (in SYSCLK cycles)

Instruction Set-up Time 1021 SYSCLK Cycles

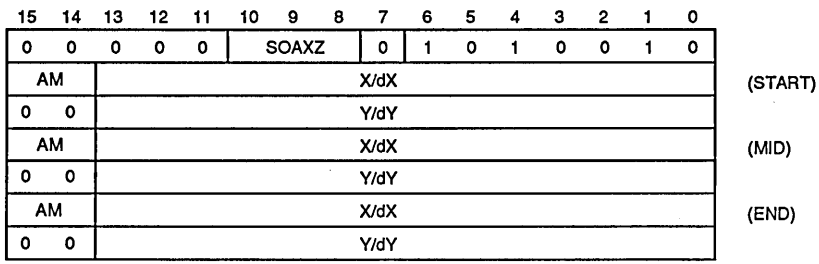
The time per pixel depends very strongly on the size and shape of the Triangle being filled. Since Filled Triangle proceeds one scan line at a time, wide triangles are filled more quickly than tall, narrow triangles of the same area. The table below shows some typical measurements.

SOAXZ	V1	V2	V3	Area	C/Pixel
SET	0,50	0,497	447,497	100K	1.26
SET	0,50	0,497	447,50	100K	1.26
SET	250,50	0,497	447,50	100K	1.40
SET	0,50	447,50	250,447	100K	1.15
SET	0,50	0,250	0,1050	100K	0.85
SET	0,50	0,1050	200,50	100K	2.34
SET	0,50	1020,50	500,1020	500K	0.91
SET	0,220	447,50	447,497	100K	1.28
SET	0,50	0,497	447,250	100K	1.29
SET	0,50	0,250	1000,1000	100K	2.34
SET	0,50	0,250	1000,800	100K	2.34
SET	0,50	0,250	1000,600	100K	1.60
SET	0,50	0,250	1000,400	100K	1.22
SET	0,50	0,250	1000,200	100K	0.94

EXAMPLE

The following example draws a filled triangle with vertices at 500,500 and 1000,500 and 800,900. Since the SOAXZ field is five, the current drawing color is applied without regard to the original contents of the field.

```
0052
01F4      *Start
01F4
03E8      *Mid
01F4
0320      *End
0385
```



07785A 14-30

Filled Triangle Format

CROSS-REFERENCES

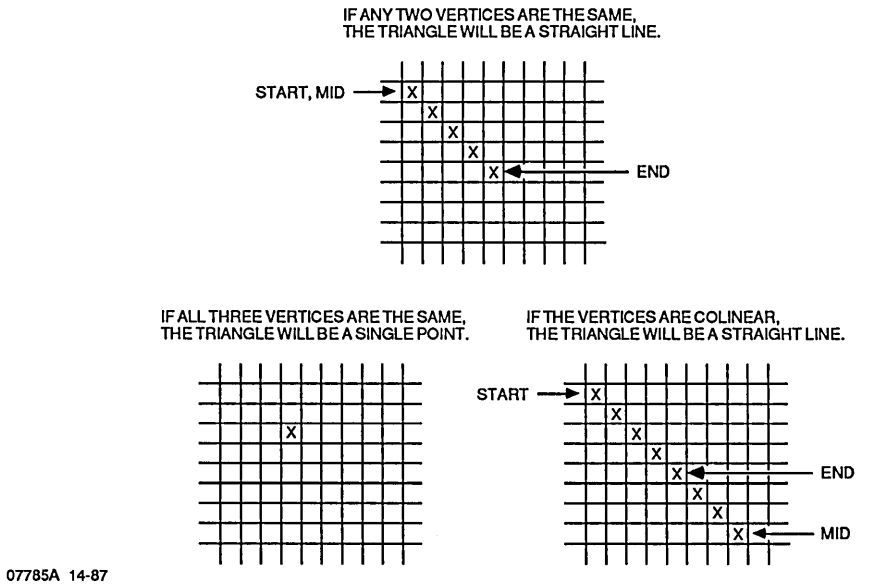
Chapter 8: Graphical Operations

COMMENTS

The purpose of Filled Triangle is to provide an analytical means of filling arbitrary regions (as opposed to Fill Connected Region and Fill Bounded Region that rely on the contents of display memory prior to the execution of the instruction).

Any polygon can be decomposed into some number of triangular regions. The notion is that the user would break the polygon into triangles and then issue the required number of Filled Triangle instructions to complete the polygon.

The vertices may have any relationship.



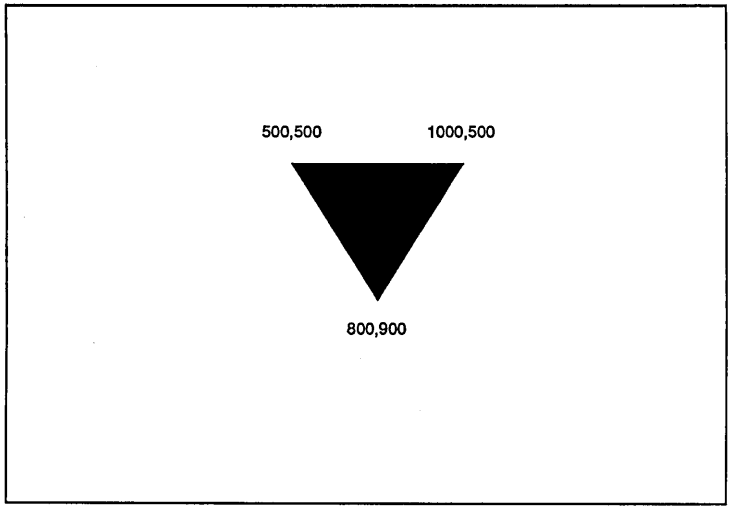


Figure 14-8 Filled Triangle

Filled Triangle Current
(Fill Instruction)

Filled Triangle Current creates the image of a triangle in display memory and fills it with the current drawing color. The SOAXZ field is used to determine how the present contents of the area are combined with the current color. The Current Pen Position is the first vertex.

Three parameters are required by Filled Triangle Current:

SOAXZ is a 3-bit field that indicates how the drawing color is to be applied to the current contents of the region.

MID is specified with a standard operand address pair. This is the second vertex of the triangle. If relative addressing is used, MID will be calculated with respect to the Current Pen Position.

END is specified with a standard operand address pair. This is the third vertex of the triangle. If relative addressing is used, END will be calculated with respect to Current Pen Position.

CPP is unaffected by this instruction.

PERFORMANCE

Instruction Setup 970 SYSCLK Cycles

For per pixel timing, see Filled Triangle.

EXAMPLE

The following example draws a filled triangle with vertices at the Current Pen Position and 1000,500 and 800,900. Since the SOAXZ field is five, the current drawing color is applied without regard to the original contents of the field.

```
0512
03E8      *Mid
01F4
0320      *End
0384
```

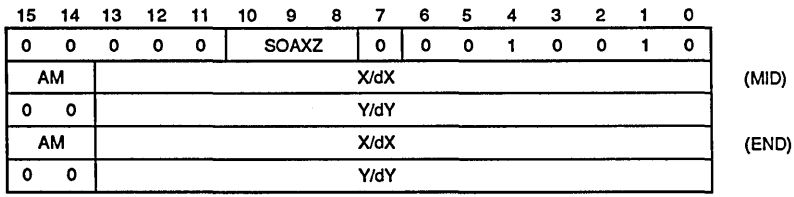
CROSS-REFERENCES

Chapter 8: Graphic Operations

COMMENTS

The purpose of Filled Triangle Current is to provide an analytical means of filling arbitrary regions (as opposed to Fill Connected Region and Fill Bounded Region that rely on the contents of display memory prior to the execution of the instruction).

Any polygon can be decomposed into some number of triangular regions. The notion is that the user would break the polygon into triangles and then issue the required number of Filled Triangle Current instructions to complete the polygon.



07785A 14-27

Filled Triangle Current Format

Input Block
(Block Manipulation)

Input Block transfers information from the host directly into display memory. The size of the (rectangular) block will have been previously specified with a Set Block Size instruction. The host is expected to write the data to be transferred into the Block Input FIFO (BIF).

Three parameters are required by Input Block:

BIS is a 3-bit field that indicates to the Am95C60 how the X portion of the address is to be incremented following each input word. BIS is determined by the number of Am95C60s in the system and will always be the same for a given system.

Number of Am95C60s	Required Value of BIS
1	4 (100)
2	2 (010)
3 or more	1 (001)

The Z bit indicates whether the data transferred are to be stored by plane (Z=0) or by pixel (Z=1).

BEGIN is specified with a standard operand address pair. This is the upper-left corner of the rectangular area into which data is to be transferred. If relative addressing is used, BEGIN is calculated with respect to the Current Pen Position.

CPP is not affected by this instruction.

PERFORMANCE (In SYSCLK cycles)

	Input By Plane	Input By Pixel
Overhead:	63	69
Per Pixel:	~0.31 per plane	~5 (BIS =4) ~9 (BIS =2) ~17 (BIS =1
Per Scan Line:	24	107

Note: The numbers for Input Block and Output Block are predictions.

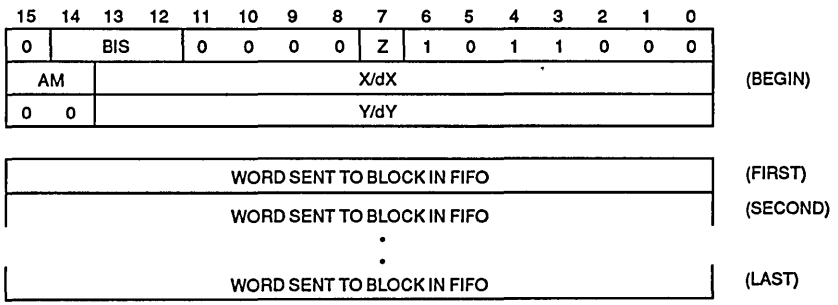
EXAMPLE

The following example starts the Am95C60 accepting data from the host. The data will be stored by pixel beginning at location 500,500. This example is for a single-Am95C60 system. The size of the block will have been specified by Set Block Size.

```
40D8    *BIS=4, Z=1
01F4    *Address of destination
01F4
```

CROSS-REFERENCES

Set Block Size (Instruction)
Input Block (Instruction)



07785A 14-28

Input Block Format

COMMENTS

Clipping is ignored.

Data will be written to all planes whose activity bits are set.

The data words to be stored in display memory are transmitted to the Block Input FIFO (rather than the instruction FIFO).

An integer number of operand words must be transferred for each scan line in the block.

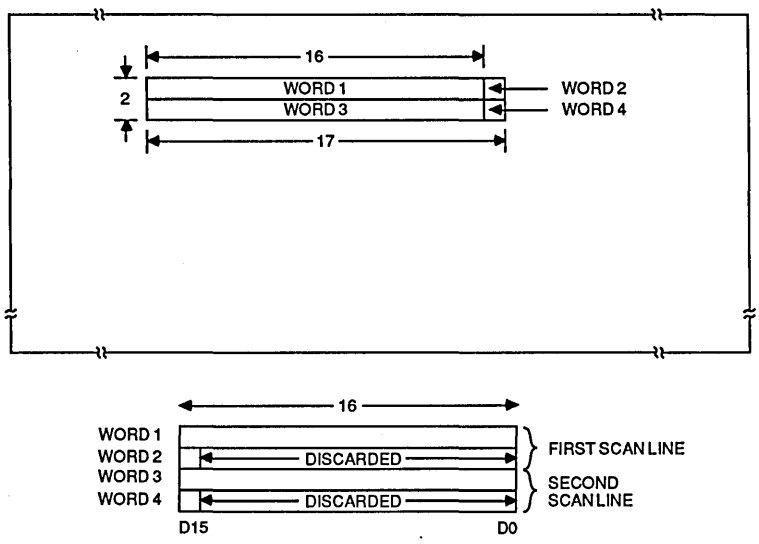
Figure 14-9 illustrates how words from the FIFO are used for input by plane. The block size has been set to 17 pixels by two scan lines. The first word into the FIFO is entered into the leftmost 16 bits of the first scan line. The leftmost bit of the second word is entered into the 17th pixel of the first scan line and the remaining 15 bits are discarded. The third and fourth words are entered into the second scan line in a similar manner.

Figure 14-10A shows how bits from the FIFO are used for input by pixel where BIS is set to 1. This is appropriate for 9 to 16 bit planes. The information transmitted on D15–D12 is used for each pixel; the information on D11–D0 is discarded. For a block size of 17 pixels by two scan lines, 34 writes are required, one for each pixel. The data transferred on the first write are entered into the leftmost pixels of the first scan line. The first scan line is filled from left-to-right and then the second scan line is filled from left-to-right.

Figure 14-10B shows the recommended connection for a four-Am95C60 system. Each Am95C60 has its own (16-bit) data buffer. The fifth data buffer is used for Block I/O by pixel. The 16 bits from the host data bus are distributed to the four Am95C60s as indicated in the table in Figure 14-10B. The system interface must make provisions to enable this extra buffer; it usually has its own I/O address.

Figure 14-11A shows how bits from the FIFO are used for input by pixel where BIS is set to 2. This is correct for five to eight bit planes. The information is transferred on D15-D8. The leftmost pixel is transmitted on D15-D12, and the rightmost pixel is transmitted on D11-D8. For a block size of 17 pixels by two scan lines, 18 writes are required. Two pixels are transmitted on each write except the 9th and 18th. Only one pixel is transmitted on each of these two writes.

Figure 14-11B shows the recommended connection for a two-Am95C60 system. Each Am95C60 has its own (16-bit) data buffer. The third data buffer is used for Block I/O by pixel. The 16 bits from the host data bus are distributed to the two Am95C60s as indicated in the table in the figure. Each pixel is transferred in one byte, two pixels per word. If the buffer is connected as recommended, the bits in the pixel will be consistent with the bits in the Set Activity Bits operand. The system interface must make provisions to enable this extra buffer; it usually has its own I/O address.

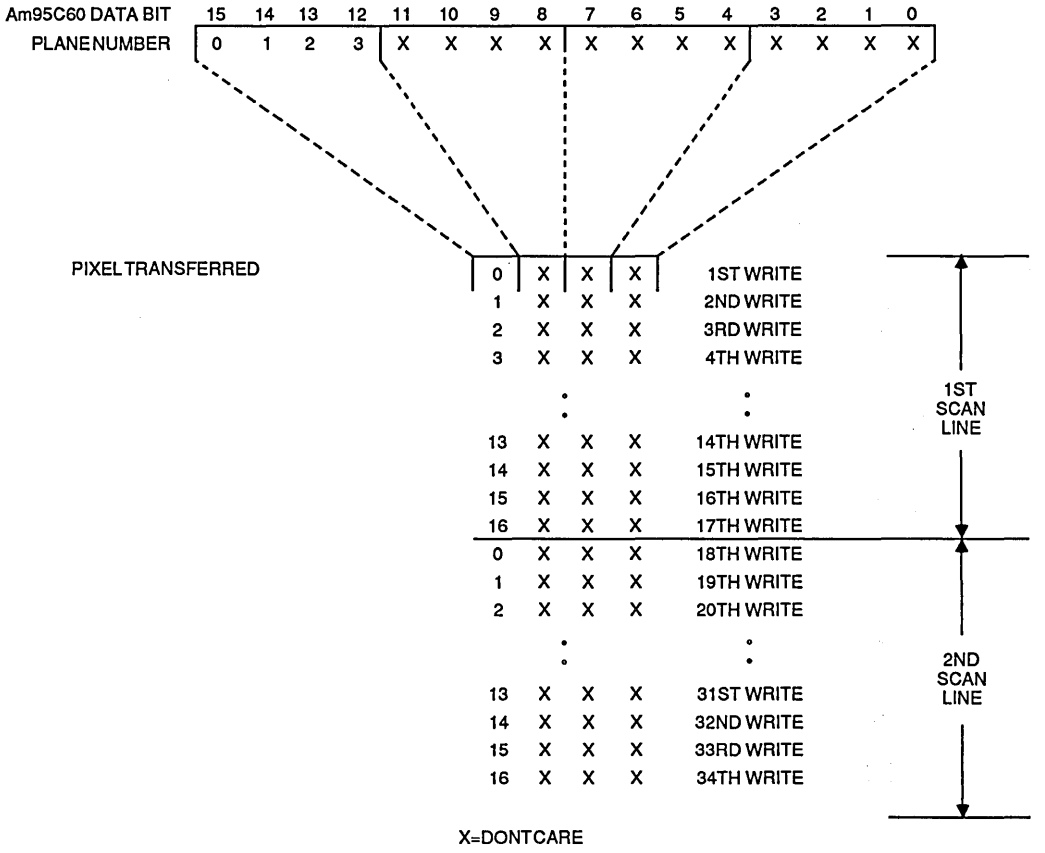


07785A 14-74

Figure 14-9 Input Block by Plane

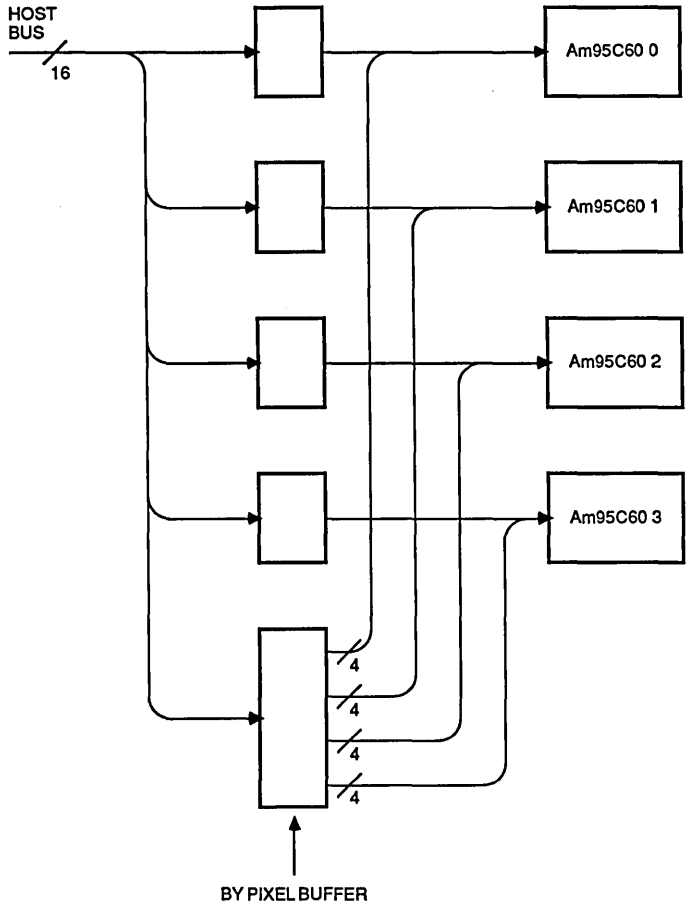
Figure 14-12 shows how bits from the FIFO are used in input by pixel where BIS is set to 4. This is correct for one to four bit planes. The information is transmitted on all 16 data lines. The leftmost pixel is transmitted on D15-D12 and the rightmost pixel is transmitted on D3-D0. For a block size of 17 pixels by two scan lines, ten writes are required.

Four pixels are transmitted on each of the writes except the fifth and tenth. Only one pixel is transmitted on each of these. The ordering of pixels within the nibbles is consistent with the ordering of plane information in the Set Color Bits operand. No special buffer is required.



07785A 14-35

Figure 14-10A Input Block by Pixel for BIS=1 As Seen by Each Am95C60



HOST BUS	Am95C60	PLANE	Am95C60 BUS
15	3	0	15
14	3	1	14
13	3	2	13
12	3	3	12
11	2	0	15
10	2	1	14
9	2	2	13
8	2	3	12
7	1	0	15
6	1	1	14
5	1	2	13
4	1	3	12
3	0	0	15
2	0	1	14
1	0	2	13
0	0	3	12

07785A 14-36

Figure 14-10B Input Block by Pixel for BIS=1 Connections

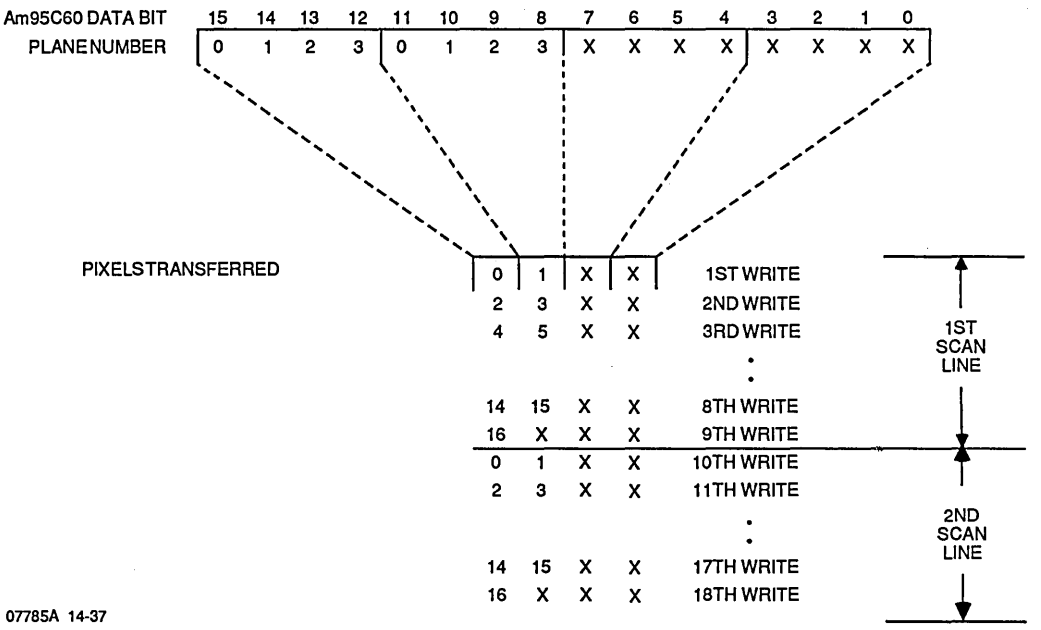
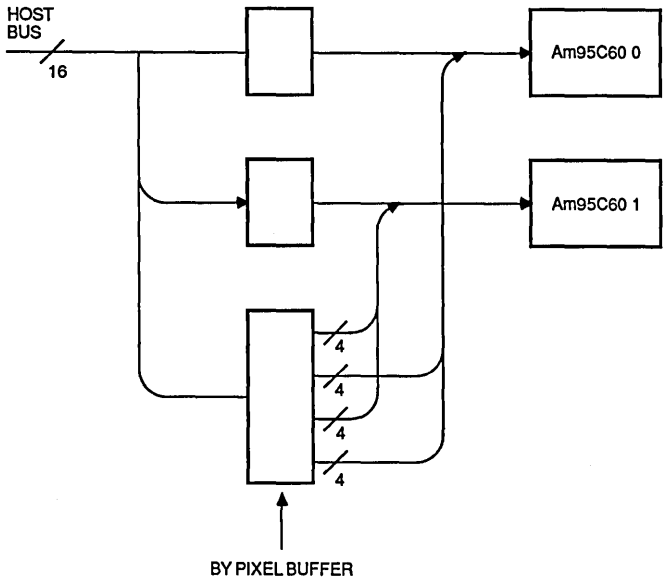


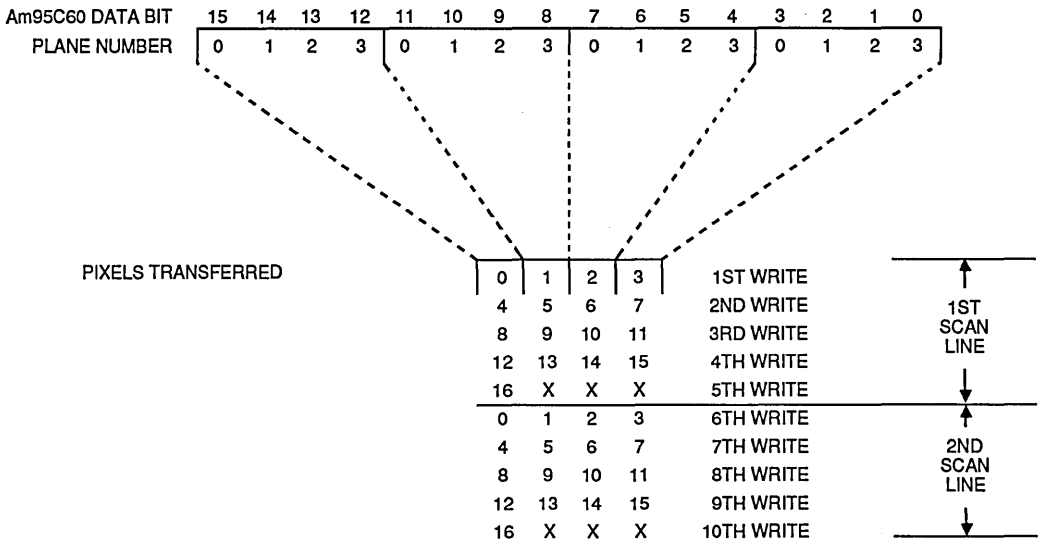
Figure 14-11A Input Block by Pixel for BIS=2 As Seen by Each Am95C60



HOST BUS	Am95C60	PLANE	Am95C60 BUS
15	1	0	15
14	1	1	14
13	1	2	13
12	1	3	12
11	0	0	15
10	0	1	14
9	0	2	13
8	0	3	12
7	1	0	11
6	1	1	10
5	1	2	9
4	1	3	8
3	0	0	11
2	0	1	10
1	0	2	9
0	0	3	8

07785A 14-78

Figure 14-11B Input Block by Pixel for BIS=2 Connections



07785A 14-79

Figure 14-12 Input Block by Pixel for BIS=4 As Seen by Each Am95C60

Input Block Current
(Block Manipulation)

Input Block Current transfers information from the host directly into display memory. The size of the (rectangular) block will have been previously specified with a Set Block Size instruction. The host is expected to write the data to be transferred into the Block Input FIFO (BIF). The upper-left corner of the destination operand is the Current Pen Position.

Two parameters are required by Input Block Current:

BIS is a 3-bit field that indicates to the Am95C60 how the X portion of the address is to be incremented following each input word. BIS is determined by the number of Am95C60s in the system and will always be the same for a given system.

Number of Am95C60s	Required Value of BIS
1	4 (100)
2	2 (010)
3 or more	1 (001)

The Z bit indicates whether the data transferred are to be stored by plane (Z=0) or by pixel (Z=1).

CPP is not affected by this instruction.

PERFORMANCE (in SYSCLK cycles)

	Input By Plane	Input By Pixel
Overhead:	63	69
Per Pixel:	~0.31 per plane	~5 (BIS=4) ~9 (BIS=2) ~17 (BIS=1)
Per Scan Line:	24	107

Note: The numbers for Input Block and Output Block are predictions.

EXAMPLE

The following example starts the Am95C60 accepting data from the host. The data will be stored by pixel beginning at the Current Pen Position. This example is for a single-Am95C60 system. The size of the block will have been specified by Set Block Size.

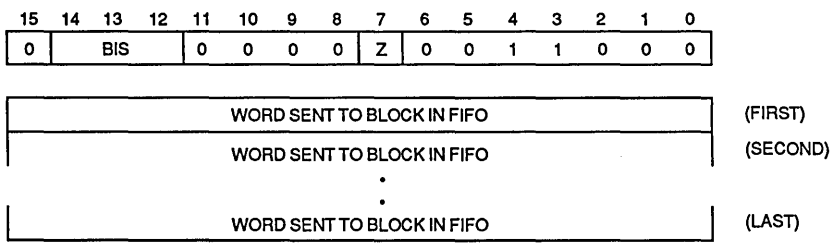
```
4098        *BIS = 4, Z=1
```

CROSS-REFERENCES

Set Block Size (Instruction)
Input Block (Instruction)

COMMENTS

See Input Block.



07785A 14-40

Input Block Current Format

Inquire
(System Control)

Inquire causes the Am95C60 to write the microcode version into the Block Output FIFO (BOF). If the BOF is full, the Am95C60 waits until there is room.

No parameters are required by Inquire.

PERFORMANCE:

Inquire requires 60 SYSCLK cycles plus any time spent waiting for room in the BOF.

EXAMPLE:

This example causes the Am95C60 to write the microcode version number into the BOF.

003F

CROSS-REFERENCES:

COMMENTS:

FORMAT OF THE INFORMATION WRITTEN TO BOF:

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

INSTRUCTION FORMAT:

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

07785A 14-90

Inquire Format

Jump
(System Control)

Jump causes a transfer to a new location in display memory when the Am95C60 is in Program mode. This instruction must not be executed from the FIFO.

Jump requires four parameters, plus an optional index:

PP is a 2-bit field that specifies the bit plane number from which instructions are to be fetched.

I is a 1-bit field that indicates whether an index is present. If I is a one, then LOCATION is taken to be the top of a dispatch table, and the index is a pointer into the table. The dispatch table is assumed to be in plane zero. The index (i) is multiplied by 2 and a X,Y address pair is fetched from LOCATION, LOCATION + (i *2). Execution begins at this X,Y address in the plane specified in PP.

If I is a zero, LOCATION specifies the address in display memory from which instructions will be fetched and no index is permitted.

PO is a 1-bit field that makes the execution of the Call conditional. If PO is a one, the Jump is executed only if bit 13 of the Y component of the Current Pen Position is a zero (implying CPPY is positive). If PO is a zero, the Jump is executed unconditionally. If I is a one, the PO bit is ignored and the instruction is executed unconditionally.

LOCATION is specified as a standard operand address pair. This is the address of the subroutine if I is a zero, or the address of the dispatch table if I is a one. If relative addressing is used, LOCATION is calculated with respect to the Current Pen Position. The four low order bits of the X component must resolve to all zeroes.

The Current Pen Position is unchanged by the execution of this instruction.

INDX is the index into the dispatch table. Only the low order 10 bits are used. Bits 10 through 15 must be zeroes.

PERFORMANCE

Jump requires 96 SYSCLK cycles.

EXAMPLE

The following example will cause a transfer to location 512,256 of bit plane 3. Absolute addressing mode is used. No scaling is assumed.

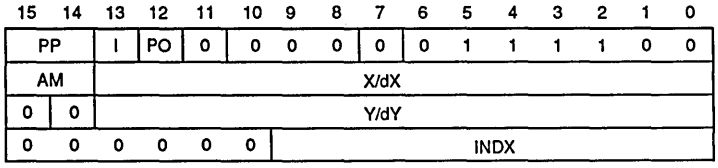
```
C03C
0200      *Address
0100
```

CROSS-REFERENCES

Call (Instruction)
Return (Instruction)

COMMENTS

Jump may be used to connect fragments of programs.



07785A 14-30

Jump Format

Line creates the image of a vector in display memory. Line style, logical PEL and anti-aliasing may be specified.

Line requires 9 parameters:

AA is a 1-bit field that controls anti-aliasing if the Logical PEL is not enabled and controls the PEL source if the logical PEL is enabled.

SI is a 1-bit field that specifies whether the logical PEL is to be inverted. This applies only if the logical PEL is enabled.

M is a 1-bit field that specifies whether the logical PEL is to be stored only where the destination matches the search color. This applies only if the logical PEL is enabled.

LS is a 1-bit field that specifies whether the line style is to be solid or dashed-dotted.

EP is a 1-bit field that specifies whether the termination end point is to be drawn unconditionally or only if it falls within a dash or dot element. The first point (at START) is always drawn.

SOAXZ is a 3-bit field that specifies how the vector is to be drawn over the current contents of the display memory.

START is specified with a standard operand address pair. If relative addressing is specified, START is calculated with relative to the Current Pen Position.

END is specified with a standard operand address pair. If relative addressing is specified, END is calculated with respect to START.

C (bit 15 of the second word of the second standard operand address pair) is a continuation bit. If this bit is a one, the Am95C60 will fetch another two standard operand address pairs and draw another line. The option bits from the opcode word will not be changed.

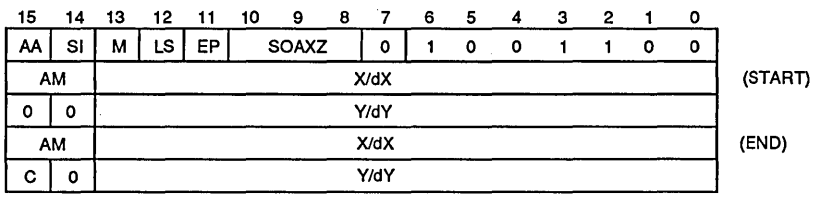
The Current Pen Position following this instruction is END.

PERFORMANCE

Instruction Overhead (Single Segment)	259 Cycles
Instruction Overhead (First Segment)	249 Cycles
Instruction Overhead (Middle Segments)	159 Cycles
Instruction Overhead (Last Segment)	179 Cycles

Drawing Time (Per Pixel)	
SOAXZ=5 (Graphical SET)	6 Cycles/Pixel
SOAXZ=7 (Graphical XOR)	12 Cycles/Pixel
Anti-aliased	95 Cycles/Pixel
Dashed-Dotted	14 Cycles/Pixel
Logical PEL (1x1)	200 Cycles/Pixel
Logical PEL (2x2)	240 Cycles/Pixel
Logical PEL (4x4)	329 Cycles/Pixel
Dashed-dotted Line (SET)	

In some cases, a horizontal line is drawn 16 pixels per write (rather than the normal 1). The line must have a solid line style, the logical PEL must not be enabled and the line must not be anti-aliased.



07785A 14-31

Line Format

EXAMPLES

This example is a non-anti-aliased solid line from 500,500 to 507,510. The SOAXZ field is set to five to obtain a graphical SET.

```
054C
01F4      *Begin
01F4
01FB      *End
01FE
```

This example draws two disjunct lines. The first line is from 0,0 to 100,100 and the second is from 200,200 to 200,400. The attributes of the two lines are identical. Neither is anti-aliased or uses a PEL. Each is drawn with a solid line style.

```
054C      *Line with Graphical SET
0000      *Address of 0,0
0000
0064      *Address of 100,100
8064      *With Continue bit set
00C8      *Address of 200,200
00C8      *
00C8      *Address of 200,400
0190      *Without Continue bit
```

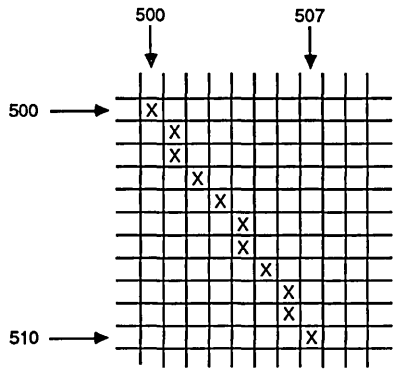
CROSS-REFERENCES

Chapter 5: Addressing Modes and Scaling
Chapter 6: Line Texture
Chapter 7: Clipping and Picking
Chapter 8: Graphical Operations
Chapter 9: Anti-aliasing

COMMENTS:

QASM provides a construct called DPL that makes it somewhat more natural to generate multiple-segment lines.

The drawing time of 6 SYSCLK cycles per pixel requires that the QPDM have been programmed to enable masked writes. See the Set QPDM Position instruction.



07785A 14-32

Figure 14-13 Line Example

Line Current
(Drawing Primitives)

Line Current creates the image of a vector in display memory. Line style, logical PEL and anti-aliasing may be specified.

Line Current requires 8 parameters:

AA is a 1-bit field that controls anti-aliasing if the Logical PEL is not enabled and controls the PEL source if the logical PEL is enabled.

SI is a 1-bit field that specifies whether the logical PEL is to be inverted. This applies only if the logical PEL is enabled.

M is a 1-bit field that specifies whether the logical PEL is to be stored only where the destination matches the search color. This applies only if the logical PEL is enabled.

LS is a 1-bit field that specifies whether the line style is to be solid or dashed-dotted.

EP is a 1-bit field that specifies whether the termination end point is to be drawn unconditionally or only if it falls within a dash or dot element. The first point (at the Current Pen Position) is never drawn.

SOAXZ is a 3-bit field that specifies how the vector is to be drawn over the current contents of the display memory.

END is specified with a standard operand address pair. If relative addressing is specified, END is calculated with respect to the Current Pen Position.

C (bit 15 of the second word of the standard operand address pair) is a continuation bit. If this bit is a one, the Am95C60 will fetch another standard operand address pair and draw another line beginning at the END of the last line. The option bits from the opcode word will not be changed.

The Current Pen Position following this instruction is END.

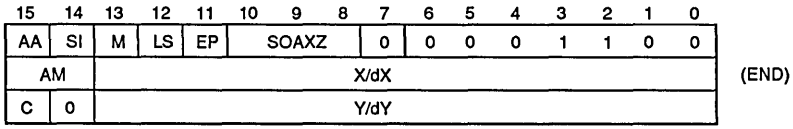
PERFORMANCE:

Instruction Overhead (Single Segment)	202 Cycles
Instruction Overhead (First Segment)	191 Cycles
Instruction Overhead (Middle Segments)	96 Cycles
Instruction Overhead (Last Segment)	119 Cycles

Drawing Time (Per Pixel)

SOAXZ=5 (Graphical SET)	6 Cycles/Pixel
SOAXZ=7 (Graphical XOR)	12 Cycles/Pixel
Anti-aliased	95 Cycles/Pixel
Dashed-Dotted	14 Cycles/Pixel
Logical PEL (1x1)	200 Cycles/Pixel
Logical PEL (2x2)	240 Cycles/Pixel
Logical PEL (4x4)	329 Cycles/Pixel
Dashed-dotted Line (SET)	

In some cases, a horizontal line is drawn 16 pixels per write (rather than the normal 1). The line must have a solid line style, the logical PEL must not be enabled and the line must not be anti-aliased.



07785A 14-33

Line Current Format

EXAMPLES

This example draws a non-anti-aliased line from the Current Pen Position to the Current Pen Position offset by +10,+10. The operand address pair specifies relative addressing mode. The scale factors should have been set to 0s to prevent scaling. The SOAXZ field is set to five for graphical SET.

```
050C
800A      *Relative address, dX=10
000A      *dY=10
```

This example draws a box 10 pixels square. The upper-left corner is at the Current Pen Position. The pen (as it turns out) is unchanged by this example. The line is drawn using a solid line style.

```
050C      *Line Current with Graphical SET
8000      *Relative: 0,-10
BFF6      *Continue bit is set
800A      *Relative: +10,0
8000      *Continue bit is set
8000      *Relative: 0,10
800A      *Continue
BFF6      *Relative: -10,0
0000      *No continue bit
```

CROSS-REFERENCES

Chapter 5: Addressing Modes and Scaling
Chapter 6: Line Texture
Chapter 7: Clipping and Picking
Chapter 8: Graphical Operations
Chapter 9: Anti-aliasing

COMMENTS

QASM includes a construct called CPL (Connected PolyLine) to help with drawing continuous lines.

Move Pen
(Drawing Primitive)

Move Pen sets the Current Pen Position.

One parameter is required by Move Pen:

LOCATION is specified with a standard operand address pair. If relative addressing mode is used, LOCATION will be calculated with respect to the Current Pen Position.

CPP is set to the operand address.

PERFORMANCE

Move Pen requires 84 SYSCLK cycles.

EXAMPLES

This example sets the Current Pen Position to 500,500. Absolute addressing is used.

```
0040
01F4      *Location
01F4
```

This example moves the Current Pen Position with an offset of +10,+10. Relative addressing is used. This assumes that the scale factors have been set to 0.

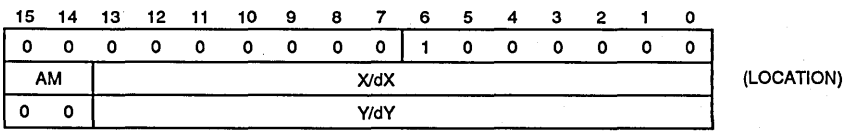
```
0040
800A      *Relative, dX=10
000A      *dY=10
```

CROSS-REFERENCES

Chapter 5: Addressing Modes and Scaling

COMMENTS

Move Pen doesn't draw anything.



07785A 14-34

Move Pen Format

No Operation
(System Control)

No Operation ensures that no operation is performed.

No parameters are required.

CPP is not affected by this instruction.

PERFORMANCE

No Operation requires 34 SYSCLK cycles.

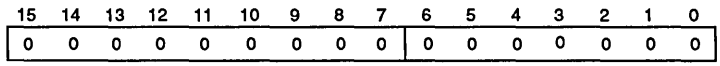
EXAMPLE

This example does nothing.

```
0000
```

CROSS-REFERENCES

COMMENTS



07785A 14-21

No Operation Format

Output Block
(Block Manipulation)

Output Block sets up the Am95C60 to transfer a rectangular block of display memory to the Block Output FIFO (BOF). The Am95C60 will transfer the contents of the block to the BOF. The size of the block will have been set by Set Block Size.

Three parameters are required by Output Block:

BOS is a 3-bit field that specifies to the Am95C60 how to increment the X address following each output operation. This value is selected according to the number of Am95C60s in the system and will always be the same for any given system.

Number of Am95C60s	Required BOS Value
1	4 (100)
2	2 (010)
3 or more	1 (001)

Z is a 1-bit field that specifies whether the output is to be done by plane (Z=0) or by pixel (Z=1).

LOCATION is specified as a standard operand address pair. Any addressing mode may be specified. If relative addressing is specified, LOCATION will be calculated with respect to the Current Pen Position. This is the address of the upper-left corner in display memory of the block to be transferred to the BOF. The size of the block to be transferred will have been set by Set Block Size.

The Current Pen Position is not affected by this instruction.

PERFORMANCE (in SYSCLK cycles)

	Output By Plane	Output By Pixel
Overhead:	71	74
Per Pixel:	~0.6 per bit plane	~2 (BOS=4) ~4 (BOS=2) ~6 (BOS=1)
Per Scan Line:	28	34

Note: These numbers are predictions.

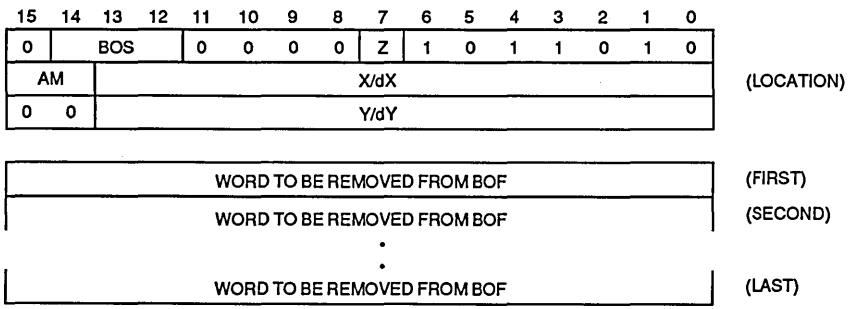
EXAMPLE

The following example sets the Am95C60 up to transfer a block of data from 500,500. The transfer will be by pixel and assumes a single-Am95C60 system. The size of the block to be transferred will have been set using Set Block Size.

```
40DA
01F4
01F4
```

CROSS-REFERENCES

Set Block Size (Instruction)
Input Block (Instruction)



07785A 14-36

Output Block Format

COMMENTS

If Z=0 (output by plane), only a single plane will be read. Only one activity bit may be set. If no plane is active, 1s will be returned. In a system with multiple Am95C60s, only one Am95C60 should have activity bits set. If multiple Am95C60s try to transmit, bus contention will result.

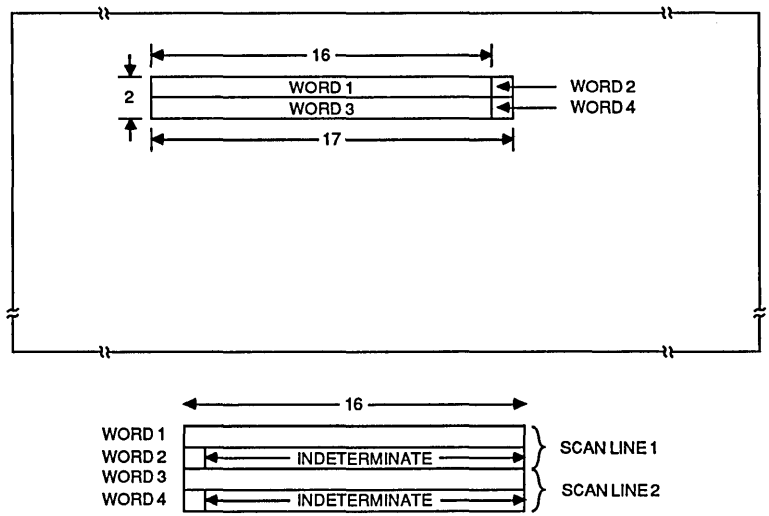
If Z=1 (output by pixel), 1s will be returned for any planes that are not active.

Figure 14-14 shows how the data are placed on the system bus data lines (D15-D0) for output by plane. The leftmost pixel is placed on D15. To transmit a block of 17 pixels by two scan lines requires four reads. The leftmost 16 pixels of the first scan line are transmitted in the first word. The 17th pixel of the first scan line is transmitted on D15 of the second word; the other 15 bits are indeterminate. The 17 pixels of the second scan line are transmitted in the third and fourth words.

Figure 14-15A shows how the data are placed on the system bus data lines for output by pixel when BOS is set to 1. This is appropriate for a system with 9 to 16 bit planes. The leftmost pixel is transmitted on D15-D12. If the recommended connection is used, the information on D11-D0 will be ignored by the system. After each word is read from the Block Output FIFO, the next pixel is shifted into the leftmost nibble of the data bus. On the 15th read, pixel number 14 has shifted to the left and the rightmost nibble is indeterminate. By the 17th read, all but the leftmost nibble is indeterminate. The first pixel of the second scan line appears on D15-D12 of the 18th read, and the last pixel of the second scan line comes out on the 34th read.

Figure 14-15B shows the recommended system bus connection for Block I/O by pixel in a four-Am95C60 system. Each Am95C60 has its own data buffer. The fifth buffer is for Block I/O by pixel. It takes four bits from each Am95C60 and concatenates them into a 16-bit word. The table in the figure shows how the data bus bits should be connected. This buffer needs to have a separate enable, usually decoded from a separate address.

Figure 14-16A shows how the data are placed on the system bus data lines for output by pixel when BOS is set to 2. This is appropriate for a system with five to eight bit planes. The leftmost pixel is transmitted on D15–D12, and the adjacent pixel is transmitted on D11–D8. The system should ignore the data on D7–D0. For a block size of 17 pixels by two scan lines, 18 reads are required. Each read transmits two pixels except for the 9th and 18th reads. Observe that the data are shifted



07785A 14-80

Figure 14-14 Output Block by Plane

by two pixels rather than one as is the case for BOS=1. Figure 14-16B shows the recommended connection for a two-Am95C60 system. The third data buffer takes four bits of each of two pixels from each of the two Am95C60s and concatenates them into two, 8-bit pixels. One pixel appears as the left byte and the other appears as the right byte. This buffer needs to have its own enable; this is usually decoded from a separate I/O address.

Figure 14-17 shows how the data are transmitted for BOS=4. This is appropriate for a single-Am95C60 system. Leftmost pixels are transmitted on higher-order D lines. For a block size of 17 pixels by two scan lines, ten reads are required. Each read transmits four pixels except the fifth and tenth that move one pixel each. No special buffer is required.

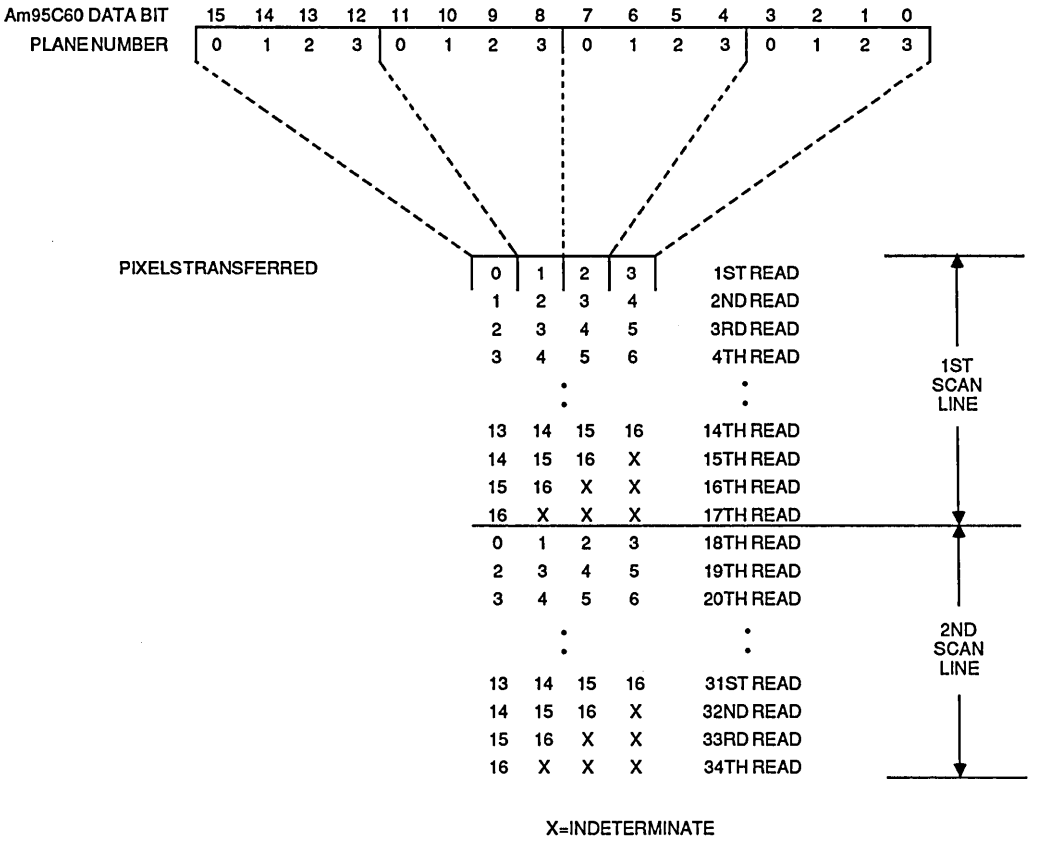
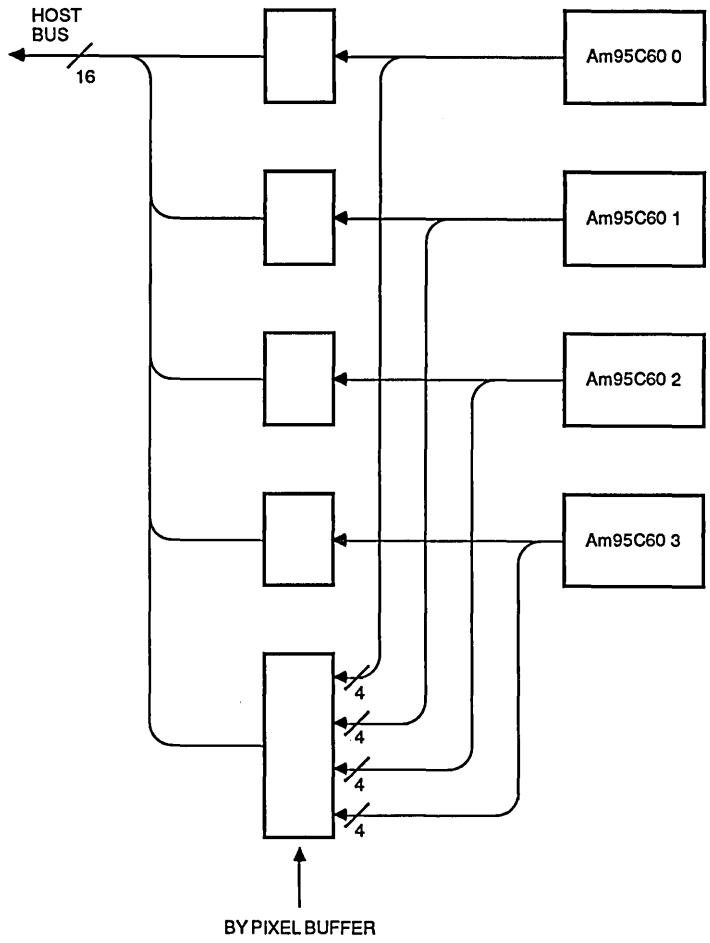


Figure 14-15A Output Block by Pixel for BOS=1 As Seen by Each Am95C60



HOST BUS	Am95C60	PLANE	Am95C60 BUS
15	3	0	15
14	3	1	14
13	3	2	13
12	3	3	12
11	2	0	15
10	2	1	14
9	2	2	13
8	2	3	12
7	1	0	15
6	1	1	14
5	1	2	13
4	1	3	12
3	0	0	15
2	0	1	14
1	0	2	13
0	0	3	12

07785A 14-82

Figure 14-15B Output Block by Pixel for BOS=1 Connections

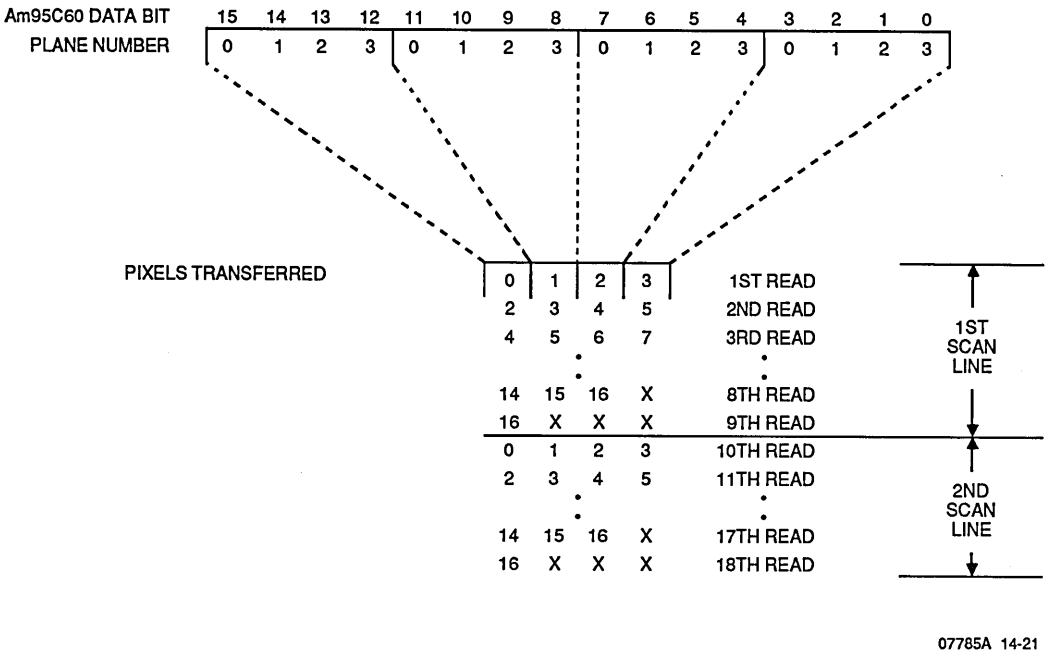
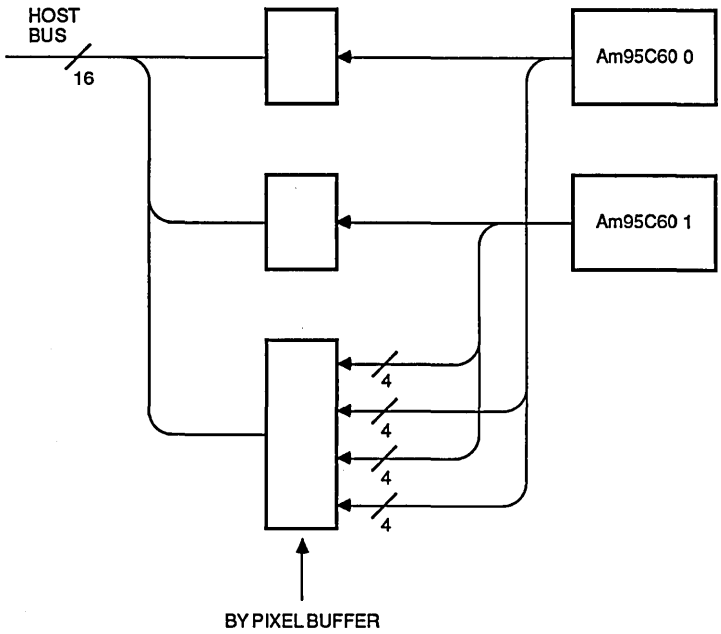


Figure 14-16A Output Block by Pixel for BOS=2 As Seen by Each Am95C60



HOST BUS	Am95C60	PLANE	Am95C60 BUS
15	1	0	15
14	1	1	14
13	1	2	13
12	1	3	12
11	0	0	15
10	0	1	14
9	0	2	13
8	0	3	12
7	1	0	11
6	1	1	10
5	1	2	9
4	1	3	8
3	0	0	11
2	0	1	10
1	0	2	9
0	0	3	8

07785A 14-83

Figure 14-16B Output Block by Pixel for BOS=2 Connections

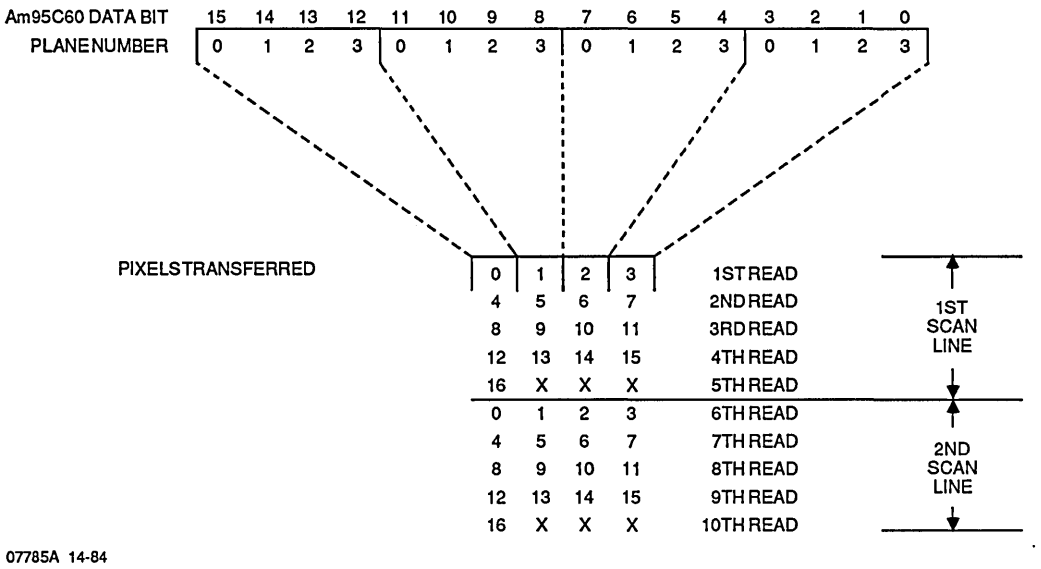


Figure 14-17 Output Block by Pixel for BOS=4 As Seen by Each Am95C60

Output Block Current
(Block Manipulation)

Output Block Current sets up the Am95C60 to transfer a rectangular block of display memory to the Block Output FIFO (BOF). The Am95C60 will transfer the contents of the block to the BOF. The size of the block will have been set by Set Block Size. The upper-left corner of the block will be the Current Pen Position.

Two parameters are required by Output Block Current:

BOS is a 3-bit field that specifies to the Am95C60 how to increment the X address following each output operation. This value is selected according to the number of Am95C60s in the system and will always be the same for any given system.

Number of Am95C60s	Required BOS Value
1	4 (100)
2	2 (010)
3 or more	1 (001)

Z is a 1-bit field that specifies whether the output is to be done by plane (Z=0) or by pixel (Z=1).

The Current Pen Position is not affected by this instruction.

PERFORMANCE (in SYSCLK cycles)

	Output By Plane	Output By Pixel
Overhead:	71	74
Per Pixel:	~0.6 per bit plane	~2 (Bos=4) ~4 (Bos=2) ~6 (Bos=1)
Per Scan Line:	28	34

Note: These numbers are predictions.

EXAMPLE

The following example sets the Am95C60 up to transfer a block of data from the Current Pen Position. The transfer will be by pixel and assumes a single-Am95C60 system. The size of the block to be transferred will have been set using Set Block Size.

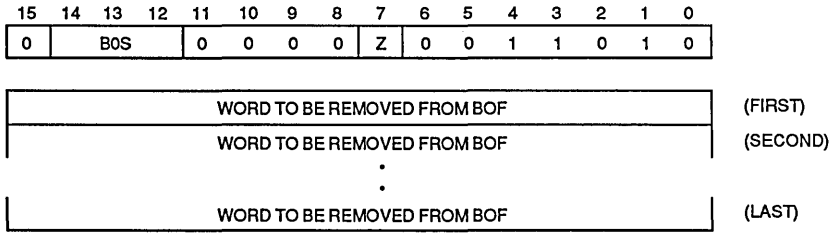
```
409A
```

CROSS-REFERENCES

Set Block Size (Instruction)
Output Block (Instruction)
Input Block (Instruction)

COMMENTS

See Output Block.



07785A 14-37

Output Block Current Format

Output Current Pen Position
(System Control)

Output Current Pen Position may be used to read the CPP from the Am95C60. When the Am95C60 encounters this instruction, the CPP is written into the Block Output FIFO (BOF).

No parameters are required.

The Current Pen Position is unchanged by the execution of this instruction.

PERFORMANCE

Output Current Pen Posiion requires 60 SYSCLK cycles.

EXAMPLE

The following example will cause the CPP to be written to the BOF.

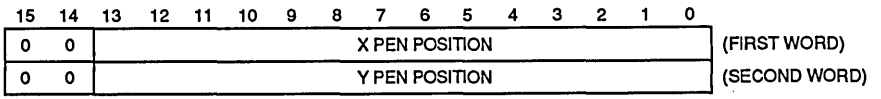
002A

CROSS-REFERENCES

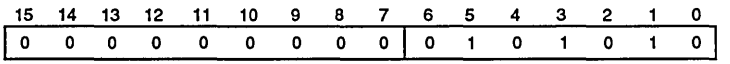
Chapter 7: Clipping and Picking

COMMENTS

Format of information written to BOF:



07785A 14-89



07785A 14-38

Output Current Pen Position Format

Point
(Drawing Primitive)

Point creates the image of the logical PEL in display memory. The image of the logical PEL is applied to the current contents of display memory under control of the AA, SI, M, and SOAXZ fields. If the logical PEL is disabled, a single point will be drawn.

Point requires 6 parameters:

AA is a 1-bit field that is unused if the Logical PEL is not enabled and controls the PEL source if the logical PEL is enabled.

SI is a 1-bit field that specifies whether the logical PEL is to be inverted. This applies only if the logical PEL is enabled.

M is a 1-bit field that specifies whether the logical PEL is to be stored only where the destination matches the search color. This applies only if the logical PEL is enabled.

SOAXZ is a 3-bit field that specifies how the PEL is to be drawn over the current contents of the display memory.

LOCATION is specified with a standard operand address pair. Location indicates the address at which the point is to be written.

C is the continue bit. If this bit is set, the Am95C60 will fetch another standard operand address pair and will draw another point. The attributes of the next point will be identical to those of the first point. If this bit is a zero, the instruction will terminate.

The Current Pen Position following this instruction is LOCATION.

PERFORMANCE

Logical PEL Disabled	99 SYSCLK Cycles
(1x1) PEL	350 SYSCLK Cycles
(2x2) PEL	393 SYSCLK Cycles
(4x4) PEL	488 SYSCLK Cycles

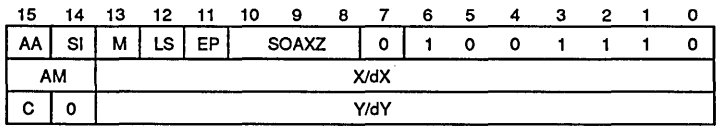
EXAMPLE

This example draws the logical PEL at 500,500. The SOAXZ field is five (graphical SET), so the PEL is deposited without regard to the current contents of display memory.

```
054E
01F4      *500,500
01F4
```

CROSS-REFERENCES

Chapter 6: Line Texture and Logical PEL



07785A 14-39

Point Format

Point Current
(Drawing Primitive)

Point Current creates the image of the logical PEL in display memory at the Current Pen Position. The image of the logical PEL is applied to the current contents of display memory under control of the AA, SI, M, and SOAXZ fields. If the logical PEL is disabled, a single point will be drawn.

Point Current requires 4 parameters:

AA is a 1-bit field that is unused if the Logical PEL is not enabled and controls the PEL source if the logical PEL is enabled.

SI is a 1-bit field that specifies whether the logical PEL is to be inverted. This applies only if the logical PEL is enabled.

M is a 1-bit field that specifies whether the logical PEL is to be stored only where the destination matches the search color. This applies only if the logical PEL is enabled.

SOAXZ is a 3-bit field that specifies how the PEL is to be drawn over the current contents of the display memory.

The Current Pen Position is unchanged following this instruction.

PERFORMANCE

Logical PEL Disabled	49 SYSCLK Cycles
(1x1) PEL	256 SYSCLK Cycles
(2x2) PEL	334 SYSCLK Cycles
(4x4) PEL	421 SYSCLK Cycles

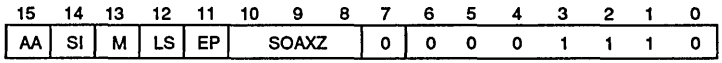
EXAMPLE

This example draws the logical PEL at the Current Pen Position. The SOAXZ field is five (graphical SET), so the PEL is deposited without regard to the current contents of display memory.

```
050E
```

CROSS-REFERENCES

Chapter 6: Line Texture and Logical PEL



07785A 14-40

Point Current Format

Pop Current Pen Position
(System Control)

Pop Current Pen Position causes the Current Pen Position and the fractional error to be retrieved from the stack. The Y component of the stack address is decremented by four.

The four words that are loaded from the stack are CPP (Y), CPP (X), FERR (Y), and FERR (X).

This is the opposite of Push Current Pen Position.

CPP is loaded to the value from the stack.

PERFORMANCE

Pop Current Pen Position requires 114 SYSCLK cycles.

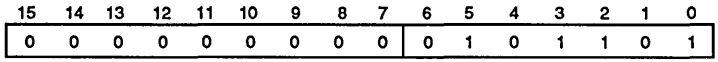
EXAMPLE

This example pops the Current Pen Position and fractional error terms from the stack. The stack address is decremented by four.

```
002D
```

CROSS-REFERENCES

Push Current Pen Position



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

07785A 14-91

Pop Current Pen Position

Push Current Pen Position
(System Control)

Push Current Pen Position causes the Current Pen Position and the fractional error terms to be stored onto the stack. The Y component of the stack pointer is incremented by four after the push takes place. The quantities that are stored are (in order of increasing Y addresses), FERR (X), FERR (Y), CPP (X) and CPP (Y).

The Current Pen Position is unchanged by the execution of this instruction.

PERFORMANCE

Push Current Pen Position requires 114 SYSCLK Cycles.

CROSS-REFERENCES

Pop Current Pen Position

EXAMPLE

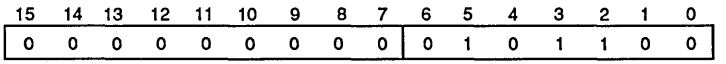
The following example pushes the CPP and Fractional Error.

```
2C
```

COMMENTS

The words on the stack look like:

```
FERR (X)
FERR (Y)
CPP (X)
CPP (Y)
```



07785A 14-92

Push Current Pen Position

Return
(System Control)

Return causes the Am95C60 to fetch a two-word return address from the stack. If bit 12 of the first word is a 1, Program mode is exited and the Am95C60 resumes executing instructions from the FIFO. If bit 12 is a 0, the two words from the stack are loaded into the Program Counter. The stack pointer is decremented by 2 before the return address is fetched.

No parameters are required.

CPP is not changed by this instruction.

PERFORMANCE

Return requires 88 SYSCLK cycles.

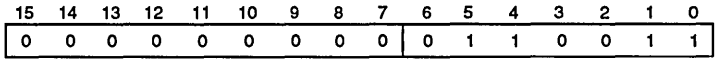
EXAMPLE

The following example fetches two words from the top of the stack. If bit 12 of the first word is a 1, program execution is continued by fetching instructions from the FIFO. If bit 12 of the first word is a 0, program execution continues from the location specified by the stack entry.

```
0033
```

CROSS-REFERENCES

Call (Instruction)
Jump (Instruction)



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

Return Format

07785A 14-41

Set Activity Bits
(System Control)

Set Activity Bits indicates which of the bit planes connected to an Am95C60 are to be treated as active. This instruction can be directed to up to four Am95C60s at once.

Two parameters are required:

PPRH is a 4-bit integer that selects a group of four Am95C60s in a multiple-Am95C60 system. Each Am95C60 will have had specified to it a certain Am95C60 number. If PPRH doesn't match, this is an effective NOP.

BITS is a 16-bit binary field that contains activity bits for up to four Am95C60s. This is broken into four subfields, one for each of the four Am95C60s. Each of the four subfields contain one bit for each bit plane. The high order bit in the field controls bit plane 0; the low order bit in the field controls bit plane 3.

CPP is not changed by this instruction.

PERFORMANCE

Set Activity Bits requires 112 SYSCLK cycles.

EXAMPLES

This example sets the planes of the Am95C60 in a four-plane system to all active. The QPDM position of the Am95C60 must have been set to 000000 by a Set QPDM Position instruction.

```
0030    *PPRH is 0
XXXF    *12 don't care bits
```

This example sets the active planes of a group of four Am95C60s as indicated in the chart below. A 1 implies the plane will be active; a 0 implies the plane will be not active. The four Am95C60s must have had their QPDM positions set to 000100, 000101, 000110 and 000111 with Set QPDM Position instructions.

Am95C60:	Three	Two	One	Zero
Plane 3:	1	1	0	1
Plane 2:	0	0	0	1
Plane 1:	1	1	1	0
Plane 0:	1	0	0	0

```
1030
D543
```

*PPRH is 0001

CROSS-REFERENCES

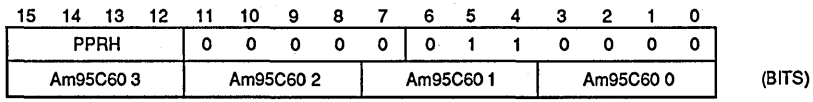
Chapter 13, Section 13.2
Set QPDM Position (Instruction)

COMMENTS

If the activity bit for a plane is set to a 1, the plane will operate normally. Write operations will conditionally write into the plane.

If the activity bit for a plane is set to a 0, the plane will not be written into (the Am95C60 will suppress the WE pin to that plane). If an Am95C60 is supporting less than four planes, the activity bits of the missing planes MUST be reset.

The high order bit in the field controls bit plane 0; the low order bit in the field controls bit plane 3.



07785A 14-42

Set Activity Bits Format

Set Anti-aliasing Distance
(System Control)

Set Anti-aliasing Distance is used to program the minimum and maximum anti-aliasing distances for each of the four bit planes. If the QPDM POSITION does not match the value programmed with Set QPDM Position, this will be an effective NOP.

Nine Parameters are required:

QPDM POSITION specifies which Am95C60 should respond to this instruction. If this value does not match the value programmed, the Am95C60 will ignore this instruction. In a system containing multiple Am95C60s, this instruction will have to be issued for each Am95C60 even if the minimum and maximum values programmed are to be the same.

Px MIN and MAX are used to indicate whether the Comparator Method or Inverse-Distance Method is to be used. In addition, the MINs and MAXs are used to specify the distances or data bits for each of the four planes. The four PXs must specify the same method for all four planes or the results will be indeterminate.

If the high order bit of the MIN value is a 0, the Comparator Method is chosen. If the high order bit of the MIN value is a 1, the Inverse-Distance Method is chosen. These methods are described in detail in Chapter 9.

CPP is not affected.

PERFORMANCE

Set Anti-aliasing Distance requires 64 SYSCLK cycles.

EXAMPLES

This example programs the Am95C60 for comparator anti-aliasing as described in Section 9.2.1.

The MINs and MAXs of QPDM zero are programmed as indicated below.

Bit Plane	Minimum	Maximum
0	0000	0111
1	0000	1011
2	0000	1101
3	0000	1111

0037	*QPDM Position Zero
0B07	*Plane 1 and Plane 0
0F0D	*Plane 3 and Plane 2

This example programs the Am95C60 for inverse distance anti-aliasing as described in Section 9.3. The digits of QPDM zero are assigned as indicated below.

Bit Plane	Digit
0	3
1	2
2	1
3	0

0037	*QPDM Position Zero
8283	*Plane 1 and Plane 0
8180	*Plane 3 and Plane 2

CROSS-REFERENCES

Chapter 9: Anti-aliasing

COMMENTS

All planes of the Am95C60 must be programmed for the same method of anti-aliasing. If not, the results will be indeterminate and may not be consistent or repeatable.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QPDM POSITION						0	0	0	0	1	1	0	1	1	1
P1 MIN				P1 MAX					P0 MIN				P0 MAX		
P3 MIN				P2 MAX					P2 MIN				P2 MAX		

07785A 14-43

Set Anti-aliasing Distance Format

Set Block Size
(Display Control)

Set Block Size is used to specify the size of the rectangle to be used by Input Block, Output Block, Copy Block, and Transform Block. It is also used to set the size of the logical PEL.

One parameter is required by Set Block Size:

SIZE is specified using a standard operand address pair. This pair is evaluated in the normal manner. The resultant X,Y pair is the block size rather than the address of an operand. Any addressing mode may be used. If relative addressing is specified SIZE is calculated with respect to the Current Pen Position.

CPP in not changed by this instruction.

PERFORMANCE

Set Block Size requires 82 SYSCLK cycles.

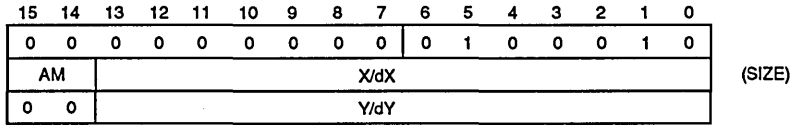
EXAMPLE

This example sets the block size to 100 (X) by 200 (Y). Absolute addressing is used. No scaling is assumed.

```
0022
0064      *X Size
00C8      *Y Size
```

CROSS-REFERENCES

Chapter 6: Line Texture
Input Block (Instruction)
Output Block (Instruction)
Copy Block (Instruction)
Transform Block (Instruction)



07785A 14-44

Set Block Size Format

Set Character Font Base
(Display Control)

Set Character Font Base specifies the base addresses of the two character fonts and the address to be used for carriage return operations. It also sets three miscellaneous flags that control how the String instruction works.

Six parameters are required by Set Character Font Base:

FP is a 2-bit field that indicates which bit plane contains the font when the String instruction specifies that it, the font, is to be treated as a single plane. For a multiple-plane font, FP indicates from which plane the attribute words are to be fetched.

CS is a 1-bit field that controls the cell scale. This modifies the interpretation of the H and S fields in attribute words as discussed in Chapter 10, Section 10.3. The table is not repeated here.

RC is a 1-bit field that specifies which of the components of the return address will actually be kept. If RC=0, the X component will be kept; if RC=1, the Y component will be kept. This information is used for the carriage return character; see Chapter 10, Section 10.5.

RETURN is specified using a standard operand address pair. This specifies either the X or Y address to be used for the Carriage Return character; see Chapter 10, Section 10.5. If relative addressing is used, RETURN is calculated relative to the Current Pen Position. This MUST be a complete standard operand address pair even though it will not all be used.

FP0x and FP0y specify the address of the attribute word for character 000 of font 0. These are each 8-bit fields; they are multiplied by 16 to form 12-bit absolute addresses. Consequently, both addresses will have 0s as the four LSBs.

FP1x and FP1y specify the address of the attribute word for character 000 of font 1. These are each 8-bit fields; they are multiplied by 16 to form 12-bit absolute addresses. As a result, both addresses will have 0s as the four LSBs.

CPP is not affected by this instruction.

PERFORMANCE

Set Character Font Base requires 136 SYSCLK cycles.

EXAMPLES

See Chapter 10, Section 6.

CROSS-REFERENCES

Chapter 10: String Operations
String (Instruction)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
FP		CS	RC	0	0	0	0	0	1	1	0	0	0	1	1	
AM		X/dX														(RETURN)
0	0	Y/dY														
FP0x									FP0y							
FP1x									FP1y							

07785A 14-45

Set Character Font Base Format

Set Character Font Base Current
(Display Control)

Set Character Font Base Current specifies the base addresses of the two character fonts and the address to be used for carriage return operations. It also sets three miscellaneous flags that affect how the String instruction works. The Current Pen Position is used for the return address.

Five parameters are required by Set Character Font Base Current:

FP is a 2-bit field that indicates which bit plane contains the font if it is a single-plane font (which in turn is specified in the String instruction). If the font is a multiple-plane font, FP indicates from which plane the attribute words are to be fetched.

CS is a 1-bit field that controls the cell scale. This modifies the interpretation of the H and S fields of attribute words as discussed in Chapter 10, Section 10.3. The table is not repeated here.

RC is a 1-bit field that specifies which of the components of the return address will actually be kept. If RC=0, the X component will be kept; if RC=1, the Y component will be kept. This information is used for the carriage return operation; see Chapter 10, Section 10.5. CPP is taken to be the return address.

FP0x and FP0y specify the address of the attribute word for character 000 of font 0. These are each 8-bit fields; they are multiplied by 16 to form 12-bit absolute addresses. Therefore, both addresses will have 0s as the four LSBs.

FP1x and FP1y specify the address of the attribute word for character 000 of font 1. These are each 8-bit fields; they are multiplied by 16 to form 12-bit absolute addresses. Hence, both addresses will have 0s as the four LSBs.

CPP is not affected by this instruction.

PERFORMANCE

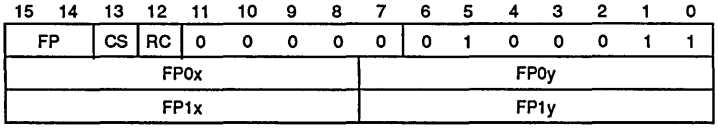
Set Character Font Base Current requires 86 SYSCLK cycles.

EXAMPLES

See Chapter 10, Section 6.

CROSS-REFERENCES

Chapter 10: String Operations
String (Instruction)



07785A 14-46

Set Character Font Base Current Format

Set Clipping Boundary
(Display Control)

Set Clipping Boundary establishes the clipping window. If clipping is enabled, then drawing instructions will perform writes only inside the clipping window. The window is rectangular and is defined by two opposite corners. The pixels that lie on the boundary are considered to be within the clipping window. Clipping may also be enabled or disabled with this instruction.

Three parameters are required by Set Clipping Boundary:

CE may be used to enable or disable clipping. If CE is a 1, clipping will be enabled. If CE is a 0, clipping will be disabled.

BEGIN is specified using a standard operand address pair. Any addressing mode may be specified. If relative addressing is used, BEGIN will be calculated with respect to the Current Pen Position. The relative values of BEGIN and END are important; BEGIN must be the upper-left corner of the window.

END is specified using a standard operand address pair. Any addressing mode may be specified. If relative addressing is used, END will be calculated with respect to BEGIN. The relative values of BEGIN and END are important; END must be the lower-right corner of the window.

CPP is changed to BEGIN.

PERFORMANCE

Set Clipping Boundary requires 146 SYSCLK cycles.

EXAMPLES

This example sets the clipping boundaries to 500,500 and 1000,1000. Absolute addressing is used. The CE bit is set so that clipping is enabled.

```
00E4
01F4      *Begin
01F4
03E8      *End
03E8
```

This example (incorrectly) sets the clipping boundaries to 1000,1000 and 500,500. In this example, the Am95C60 will treat all of display memory as being OUTSIDE the clipping window. The CE bit is set so that clipping is enabled.

```
00E4
03E8      *Begin (is bigger than End)
03E8
01F4      *End
01F4
```

CROSS-REFERENCES

Chapter 7: Clipping and Picking
Control Clipping (Instruction)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	CE	1	1	0	0	1	0	0	
AM		X/dX														(BEGIN)
0	0	Y/dY														
AM		X/dX														(END)
0	0	Y/dY														

07785A 14-47

Set Clipping Boundary Format

Set Clipping Boundary Current
(Display Control)

Set Clipping Boundary Current establishes the clipping window. If clipping is enabled, then drawing instructions will perform writes only inside the clipping window. The window is rectangular and is defined by two opposite corners. The pixels that lie on the boundary are considered to be within the clipping window. The upper-left corner of the clipping window is CPP.

Two parameters are required:

CE may be used to enable or disable clipping. If CE is a 1, clipping will be enabled. If CE is a 0, clipping will be disabled.

END is specified using a standard operand address pair. Any addressing mode may be specified. If relative addressing is used, END will be calculated with respect to CPP. The relative values of CPP and END are important; END must be the lower-right corner of the window.

CPP is unchanged by this instruction.

PERFORMANCE

Set Clipping Boundary Current requires 96 SYSCLK cycles.

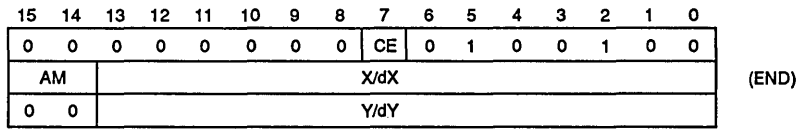
EXAMPLE

This example sets the clipping boundaries to Current Pen Position and CPP +10,10. Relative addressing mode is used.

```
00A4      *Opcode for current
800A      *End
3FF6
```

CROSS-REFERENCES

Chapter 7: Clipping and Picking
Control Clipping (Instruction)



07785A 14-48

Set Clipping Boundary Current Format

Set Color Bits
(Display Control)

Set Color Bits indicates—by bit plane—whether a 1 or a 0 is to be written. This instruction can be directed to up to four Am95C60s at once.

Two parameters are required:

PPRH is a 4-bit integer that selects a group of four Am95C60s in a multiple-Am95C60 system. Each Am95C60 will have had specified to it its QPDM position.

BITS is a 16-bit binary field that contains drawing color bits for up to four each Am95C60s. This is broken into four subfields, one for each of the four Am95C60s. Each of the four subfields contain one bit for each bit plane. The high order bit in the field controls bit plane 0; the low order bit in the field controls bit plane 3.

CPP is unaffected by this instruction.

PERFORMANCE

Set Color Bits requires 112 SYSCLK cycles.

EXAMPLES

This example sets the drawing color in a single-Am95C60 system to all 1s. The QPDM position of the Am95C60 must have been set to 000000 by a Set QPDM Position instruction.

```
0020
XXXF      *12 don't care bits
```

This example sets the drawing color in a four-Am95C60 system as indicated in the following chart. The Am95C60s must previously have had their QPDM position numbers set to 000100, 000101, 000110 and 000111 with Set QPDM Position instructions.

Am95C60:	Three	Two	One	Zero
Plane 3:	1	1	0	1
Plane 2:	0	0	0	1
Plane 1:	1	1	1	0
Plane 0:	1	0	0	0

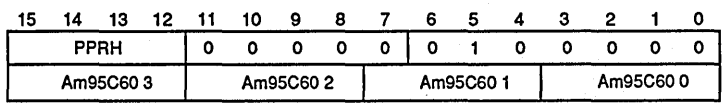
```
1020
D543      *Color Bits
```

CROSS-REFERENCES

Chapter 8: Graphical Operations
Chapter 13, Section 13.2
Set QPDM Position (Instruction)

COMMENTS

This instruction determines the sense of the bits that are written into the bit planes (by plane). These bits are used in conjunction with the SOAXZ field to determine precisely what is to happen when a pixel is written into display memory.



07785A 14-49

Set Color Bits Format

Set Line Style
(Display Control)

Set Line Style specifies the lengths of the elements for use in nonsolid lines, circles and arcs.

Three parameters are required by Set Line Style:

DASH specifies the length of the dash element. This field is a 12-bit positive integer.

SPACE specifies the length of the space element. This field is a 12-bit positive integer.

DOT specifies the length of the dot element. This field is a 12-bit positive integer.

CPP is not affected by this instruction.

PERFORMANCE

Set Line Style requires 126 SYSCLK cycles.

EXAMPLE

This example sets the line style elements as follows:

DASH is set to 10;
SPACE is set to 5;
DOT is set to 2.

```
0025  *Opcode
000A  *Dash
0005  *Space
0002  *Dot
```

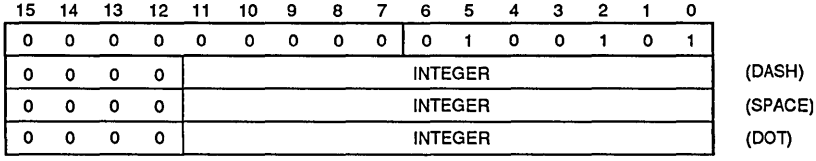
CROSS-REFERENCES

Chapter 6: Line Texture
Set Line Style Phase (Instruction)

COMMENTS

The INTEGERS are all multiplied by the X scale factor prior to being stored. If the X scale factor is zero the INTEGERS are stored as they appear in the instruction.

If scaling is going to be used, it should be set prior to executing the Set Line Style instruction.



07785A 14-50

Set Line Style Format

Set Line Style Phase
(Drawing Control)

Set Line Style Phase sets the Line Style State and the Line Style Counter. This allows lines and arcs to begin at arbitrary places in the line style.

Three input parameters are required by Set Line Style Phase:

LSS is a 2-bit field that specifies the Line Style State (which element is to be drawn next).

LSS	BEGINNING ELEMENT
0 0	Space preceding a DASH
0 1	DASH
1 0	Space preceding a DOT
1 1	DOT

LSC specifies the Line Style Counter. This field is a 12-bit number that specifies the number of pixels remaining in the current element.

CPP is unaffected by this instruction.

PERFORMANCE

This instruction requires 71 SYSCLK cycles.

EXAMPLE

This example sets the Line Style Phase to 2 and the Line Style Counter to 25. The next drawing instruction will begin with a space preceding a DOT with 25 pixels remaining.

```
2026      *LSS = 10
0019      * = 25 D
```

CROSS-REFERENCES

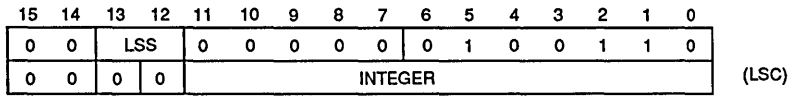
Chapter 6: Line Textures
Set Line Style (Instruction)

COMMENTS

If the scaling factor is not set to zero, the integer is multiplied by the scale factor prior to being stored. If scaling is to be used, the scale factor should be set prior to executing Set Line Style Phase.

If LSC is greater than the number of pixels, the line will be drawn entirely in the beginning element.

If LSC is set to 0, a maximum length element will be drawn.



07785A 14-51

Set Line Style Phase Format

Set Listen Bits
(System Control)

Set Listen Bits indicates—by bit plane—whether or not the bit plane is to participate in color matches. This instruction can be directed to up to four Am95C60s at once.

Two parameters are required:

PPRH is a 4-bit integer that selects a group of four Am95C60s in a multiple-Am95C60 system. Each Am95C60 will have had specified to it its QPDM Position.

BITS is a 16-bit binary field that contains listen bits for up to four Am95C60s. This is broken into four subfields, one for each of the four Am95C60s. Each of the four subfields contain one bit for each bit plane. The high order bit in the field controls bit plane 0; the low order bit in the field controls bit plane 3.

CPP is unaffected by this instruction.

PERFORMANCE

Set Listen Bits requires 112 SYSCLK cycles.

EXAMPLES

This example sets the listen bits in a single Am95C60 system to all 0s. The QPDM position of the Am95C60 must have been set to 000000 by a Set QPDM Position instruction.

```
0031
XXX0        *12 don't care bits
```

This example sets the listen bits in a four-Am95C60 system as indicated in the following table. The QPDM position numbers must have been set to 000000, 000001, 000010, and 000011 using Set QPDM Position instructions.

Am95C60:	Three	Two	One	Zero
Plane 3:	1	1	0	1
Plane 2:	0	0	0	1
Plane 1:	1	1	1	0
Plane 0:	1	0	0	0

```
0031
D543
```

CROSS-REFERENCES

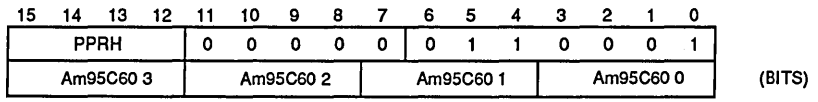
Chapter 13, Section 13.2
Set QPDM Position (Instruction)
Fill Bounded Region (Instruction)

COMMENTS

The listen bits (one per bit plane) are used to indicate whether the contents of a bit plane are to be considered in operations involving color matching. Examples of such operations are Copy Block (with the M bit set) and Fill Bounded Region.

If the listen bit for a plane is a 0, the sense of the pixel at a given location must match the search color in order for *that plane* to report a match. If the listen bit for a plane is a 1, that plane will always report a match.

If a plane is not physically implemented, the listen bit must be a 1.



07785A 14-52

Set Listen Bits Format

Set Picking Region
(Display Control)

Set Picking Region establishes the picking region. If picking is enabled, any drawing instruction that would generate a write into this region will cause the Pick Detect (PCKD) bit in the status register to be set. The region is rectangular and is defined by two corners. The pixels that lie on the boundary are considered to be within the picking region.

Three parameters are required:

PE is used to enable or disable picking. If PE is a 1, picking is enabled. If PE is a 0, picking is disabled.

BEGIN is specified using a standard operand address pair. Any addressing mode may be used. If relative addressing is specified, BEGIN is calculated with respect to the Current Pen Position. BEGIN must specify the upper-left corner of the picking region.

END is specified using a standard operand address pair. Any addressing mode may be used. If relative addressing is specified, END is calculated with respect to BEGIN. END must specify the lower-right corner of the picking region.

CPP is set to BEGIN.

PERFORMANCE

Set Picking Region requires 138 SYSCLK cycles.

EXAMPLES

This example sets the picking region corners to 500,500 and 1000,1000. Absolute addressing is used.

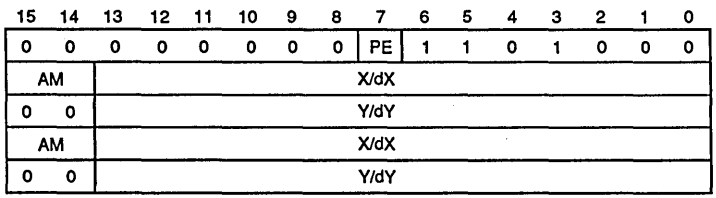
```
00E8
01F4      *Begin
01F4
03E8      *End
03E8
```

This example (incorrectly) sets the picking region corners to 1000,1000 and 500,500. In this example, the Am95C60 will treat all of display memory as being OUTSIDE the picking region.

```
00E8
03E8      *Begin (is bigger than End)
03E8
01F4      *End
01F4
```

CROSS-REFERENCES

Chapter 7: Clipping and Picking
Control Picking (Instruction)



07785A 14-53

Set Picking Region Format

Set Picking Region Current
(Display Control)

Set Picking Region Current establishes the picking region. If picking is enabled, any drawing instruction that would generate a write into this region will cause the Pick Detect (PCKD) bit in the status register to be set. The region is rectangular and is defined by two corners. The pixels that lie on the boundary are considered to be within the picking region.

Two parameters are required:

PE is used to enable or disable picking. If PE is a 1, picking is enabled. If PE is a 0, picking is disabled.

END is specified using a standard operand address pair. Any addressing mode may be used. If relative addressing is specified, END is calculated with respect to CPP. END must specify the lower-right corner of the picking region. The Current Pen Position is the upper-left corner.

CPP is unchanged.

PERFORMANCE

Set Picking Region Current requires 88 SYSCLK cycles.

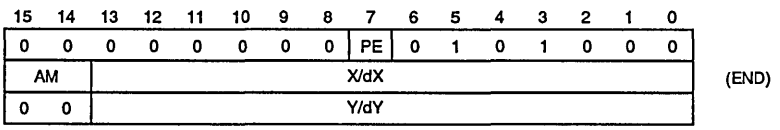
EXAMPLE

This example sets the picking region to the Current Pen Position and CPP +10,10. Relative addressing is used.

```
0028
800A      *Relative, 10
3FF6
```

CROSS-REFERENCES

Chapter 7: Clipping and Picking
Control Picking (Instruction)



07785A 14-54

Set Picking Region Current Format

Set QPDM Position
(System Control)

Set QPDM Position is used to specify to an Am95C60 its position. This instruction must be issued to each Am95C60 individually.

Two parameters are required:

PPR is a 6-bit integer that assigns the position of a particular Am95C60. This instruction limits the number of Am95C60s in a system to 64; this limits the number of planes in a system to 256.

ME is a 1-bit field that enables or disables masked writes to VRAMs. A 1 enables masked writes; a 0 disables masked writes.

CPP is unaffected by Set QPDM position.

PERFORMANCE

Set QPDM Position requires 36 SYSCLK cycles.

EXAMPLE

This example sets the QPDM position of an Am95C60 to 000000.

```
0038
```

CROSS-REFERENCES

Set Activity Bits (Instruction)
Set Color Bits (Instruction)
Set Search Color (Instruction)
Set Listen Bits (Instruction)

COMMENTS

Set QPDM Position is used to condition a specific Am95C60 as to which bit-wise instructions it is to recognize and which bits it is to extract from bit-wise instruction operands.

The QPDM position is a 6-bit field that can be divided into two fields as shown below.

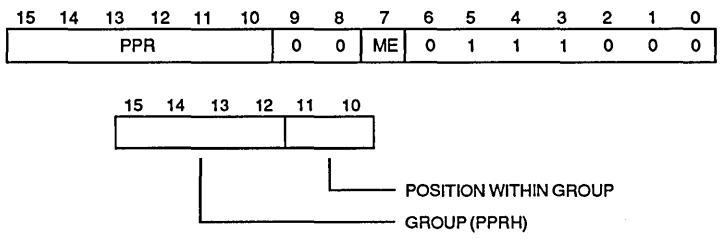
The Am95C60 will execute the bit-wise instructions only if the PPRH bits match the group bits that the Am95C60 was programmed to during the Set QPDM Position instruction. If the bits do not match, the Am95C60 will ignore that bit-wise instruction.

If the Am95C60 does recognize the PPRH bits, it will extract four bits from the operand according to the low order two bits of its QPDM position. The bits are indicated in the chart below.

Position	Bits Extracted
0 0	3–0
0 1	7–4
1 0	11–8
1 1	15–12

This instruction unconditionally disables picking.

Line and Line Current are the only instructions that take advantage of Masked Writes.



Set QPDM Position Format

Set Scale Factor
(Drawing Control)

Set Scale Factor specifies the X and Y scale factors that are used in absolute, viewport and relative addressing modes. If the scale factor for an axis is 0, then scaling for that axis is inhibited.

Four parameters are required:

SFX is a 15-bit number that is used as the X scale factor. The high order bit must be zero. If the address is taken as an integer, 7FFF yields scaling of unity -2^{-15}.

SFY is a 15-bit number that is used as the Y scale factor. The high order bit must be zero. If the address is taken as an integer, 7FFF yields scaling of unity -2^{-15}.

P indicates whether the PLD is present. A one indicates it is present; a zero indicates it is not present.

PLD is optional. If it is present, it modifies the diagonal count for element lengths as discussed in Section 6.1.3.

PERFORMANCE

Set Scale Factor requires 57 SYSCLK cycles.

EXAMPLES

This example sets both scale factors to unity (no scaling is done).

```
0029
0000
0000
```

This example sets the X scale factor to 0.5 and the Y scale factor 0.75. These numbers assume the user thinks of the addresses as being integers.

```
0029
4000      *1/2
6000      *1/2 + 1/4
```

This example sets the scale factors to 0.5 and sets the PLD to 22/16.

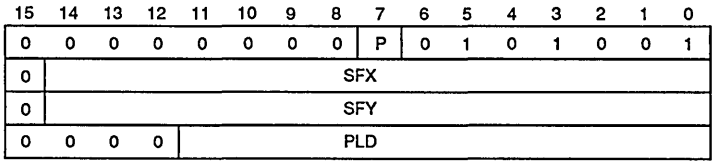
```
00A9      *PLD is present
4000      *.5
4000      *.5
0016      *22 decimal = 16 hex
```

CROSS-REFERENCES

Chapter 5: Addressing Modes and Scaling
Chapter 6: Line Styles

COMMENTS

Setting the scale factors differently will compensate for nonsquare pixels except that circles will appear as ellipses.



07785A 14-57

Set Scale Factor Format

Set Search Color
(Display Control)

Set Search Color indicates–by bit plane–whether the search color is to be a 1 or a 0. This instruction can be directed to up to four Am95C60s at once.

Two parameters are required:

PPRH is a 4-bit integer that selects a group of four Am95C60s in a multiple-Am95C60 system. Each Am95C60 will have had specified to it its QPDM position.

BITS is a 16-bit binary field that contains search color bits for up to four Am95C60s. This is broken into four subfields, one for each of the four Am95C60s. Each of the four subfields contain one bit for each bit plane. The high order bit in the field controls bit plane 0; the low order bit in the field controls bit plane 3.

CPP is unaffected by this instruction.

PERFORMANCE

Set Search Color requires 112 SYSCLK cycles.

EXAMPLES

This example sets the search color bits in a single-Am95C60 system to all 1s. The QPDM position of the Am95C60 must have been set to 000000 by a Set QPDM Position instruction.

```
0021
XXXF      *12 don't care bits
```

This example sets the search color in a four-Am95C60 system as indicated in the following chart. The QPDM positions must have been set to 000100, 000101, 000110 and 000111 using Set QPDM Position instructions.

Am95C60:	Three	Two	One	Zero
Plane 3:	1	1	0	1
Plane 2:	0	0	0	1
Plane 1:	1	1	1	0
Plane 0:	1	0	0	0

```
1021      *PPRH is 0001
D543
```

CROSS-REFERENCES

Chapter 13, Section 13.2
Set QPDM Position (Instruction)
Fill Bounded Region (Instruction)
Copy Block (Instruction)

COMMENTS

The search color is used for two purposes:

One use of the search color is as the "boundary" in a Fill Bounded Region.

Another use of the search color is as the color to search for in Copy Block (M bit).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPRH				0	0	0	0	0	0	1	0	0	0	0	1
Am95C60 3				Am95C60 2				Am95C60 1				Am95C60 0			

07785A 14-58

Set Search Color Format

Set Stack Boundaries
(System Control)

Set Stack Boundaries is used to specify the upper and lower boundaries of the stack in display memory. The stack is stored in plane 0 of each Am95C60 (implying that bit plane 0 must be implemented).

Three parameters are required:

X is the X coordinate of the stack. This is a 12-bit integer and the four low order bits are required to be 0s (the stack is word aligned).

YS is the lowest Y address of the region set aside for the stack. This is a 12-bit integer. This is the top of the stack.

YT is the highest Y address of the region set aside for the stack. This is a 12-bit integer. This is the bottom of the stack.

CPP is unaffected by this instruction.

PERFORMANCE

Set Stack Boundaries requires 90 SYSCLK cycles.

EXAMPLE

This example sets the stack boundaries to 3F0, 0, 3FC.

```
0039
03F0      *X address
0         *YS
03FC      *YT
```

CROSS-REFERENCES

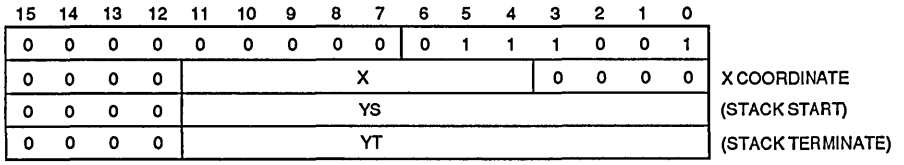
Chapter 13, Section 13.2
Call (Instruction)
Fill Bounded Region (Instruction)
Fill Connected Region (Instruction)
Jump (Instruction)
Return (Instruction)

COMMENTS

The stack begins at YS and grows upward in memory. If the stack pointer passes YT, stack overflow has occurred. When elements are removed from the stack, the pointer moves toward YS. Stack underflow cannot occur.

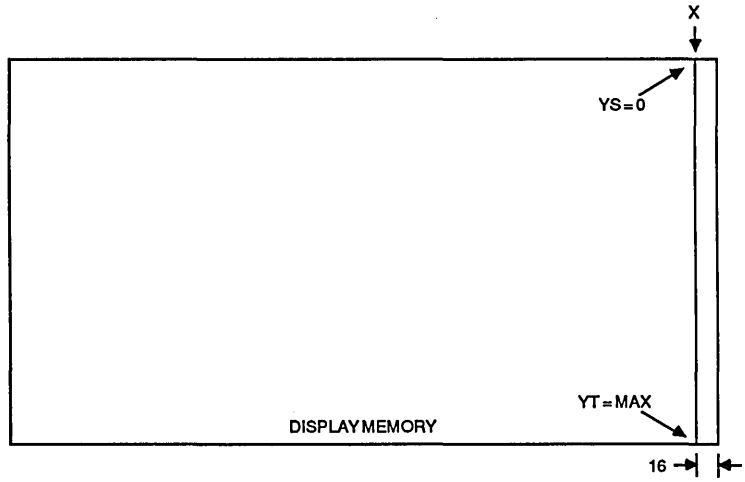
Stack is depicted in Figure 14-18.

We recommend that the stack be as large as possible. This will minimize the likelihood of stack overflow.



07785A 14-59

Set Stack Boundaries Format



07785A 14-60

Figure 14-18 Set Stack Boundaries

Set Viewport Location
(Display Control)

Set Viewport Location is used to specify the location in display memory to be treated as the base for viewport addressing.

One parameter is required:

BASE is specified using a standard operand address pair. BASE specifies the base address for viewport addressing. Any addressing mode may be specified. If relative addressing is used, BASE is calculated with respect to the Current Pen Position.

CPP is left at BASE

PERFORMANCE

Set Viewport Location requires 85 SYSCLK cycles.

EXAMPLE

This example places the viewport base at 500,500. Absolute addressing mode is used.

```
0067
01F4        *Begin
01F4
```

CROSS-REFERENCES

Chapter 5: Addressing Modes and Scaling

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	1	
AM		X/dX														(BASE)
0	0	Y/dY														

07785A 14-61

Set Viewport Location Format

Set Viewport Location Current
(Display Control)

Set Viewport Location Current is used to specify the location in display to be treated as the base address for viewport address calculations.

No parameters are required.

CPP is not changed by this instruction.

PERFORMANCE

Set Viewport Location Current requires 35 SYSCLK cycles.

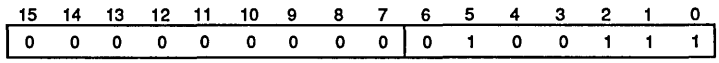
EXAMPLE

This example programs the viewport base to the Current Pen Position.

```
0027
```

CROSS-REFERENCES

Chapter 5: Addressing Modes and Scaling



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

Set Viewport Location Current Format

07785A 14-53

Signal
(System Control)

Signal may be used to indicate when a particular point in the instruction stream has been reached. It is also used in conjunction with picking.

Two parameters are required:

VALUE is a 8-bit integer that identifies the Signal instruction. It is ignored by the Am95C60 but may be returned to the host.

C is a 1 bit field that controls the exact operation of Signal. If C=1 or if Pick Enable is set, then the Signal instruction is written to the BOF and SWI is set. The Am95C60 pauses until a word is available in BIF. It removes and discards the word and continues with the next instruction. If C=0 and Pick Enable is not set, Signal is an effective NOP.

CPP is unaffected by the execution of Signal.

PERFORMANCE

SIGNAL requires 120 SYSCLK cycles.

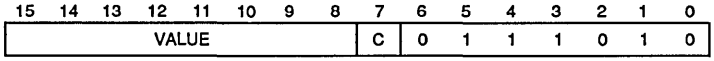
EXAMPLE

CROSS-REFERENCES

Chapter 7: Clipping and Picking
Output Current Pen Position (Instruction)

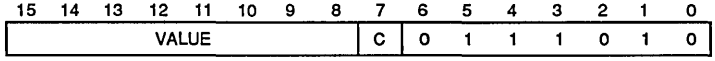
COMMENTS

The format of the word written to the BOF is shown in Figure 14-19.



07785A 14-64

Figure 14-19 Word Written by the Signal Instruction



07785A 14-63

Signal Format

Store Current Pen Position
(System Control)

Store Current Pen Position causes the Am95C60 to write the integer part of the Current Pen Position into display memory.

A single parameter is required by this instruction:

LOCATION is specified with a standard operand address pair. LOCATION indicates the address at which the CPP is to be written. If relative addressing is specified, LOCATION is calculated with respect to CPP. The low order four bits of the X address must evaluate to zeroes.

CPP is unchanged by the execution of this instruction.

PERFORMANCE:

Store Current Pen Position requires 102 SYSCLK Cycles.

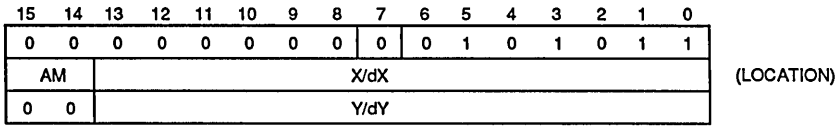
EXAMPLE

This example stores the CPP at 256,256.

```
002B      *Store CPP
0100      *256, 256
0100
```

COMMENT

The fractional error is not stored.



07785A 14-93

Store Current Pen Position Format

Store Immediate
(Display Control)

Store Immediate is used to store up to 128 words from the instruction stream into display memory. The words are stored beginning with the location specified in the instruction and increasing Y addresses.

Three parameters plus a variable length list are required:

COUNT is a 7-bit field that specifies the number of words that follow in the instruction stream and are to be stored. The actual number of words is COUNT+1.

Z indicates the planes into which the data will be stored. If Z=0, the data will be stored only into plane 0, regardless of the activity bits. If Z=1, the data will be stored into all planes whose activity bits are set.

BEGIN is specified with a standard operand address pair. This specifies the address into which the first word will be stored. Any addressing mode may be used. If relative addressing mode is used, BEGIN will be calculated with respect to the Current Pen Position. The low order four bits of the X address must resolve to 0s.

Current Pen Position in X is left at BEGIN. Current Pen Position in Y is left at the last word stored + 1.

PERFORMANCE

Store Immediate requires 71 SYSCLK cycles plus 39 SYSCLK cycles per word stored.

EXAMPLE

This example writes the values 0,1,2,3,4 beginning at 400,500. In particular the following values are written into the indicated locations.

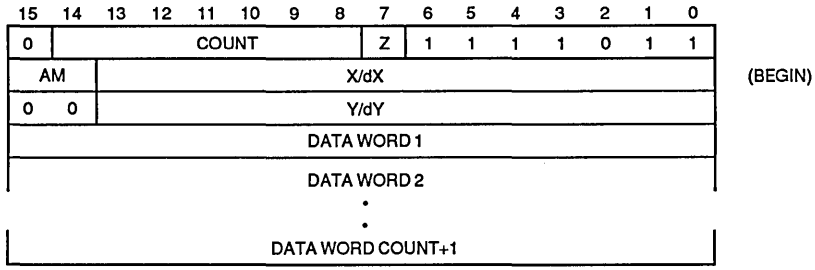
Value	Location
0000	400,500
0001	400,501
0002	400,502
0003	400,503
0004	400,504

```
047B
0190
01F4
0000
0001
0002
0003
0004
```

CROSS-REFERENCES

COMMENTS

Clipping is ignored.



07785A 14-65

Store Immediate Format

Store Immediate Current
(Display Control)

Store Immediate Current is used to store up to 128 words from the instruction stream into display memory. The words are stored beginning at the Current Pen Position and increasing Y addresses.

Two parameters plus a variable length list be required:

COUNT is a 7-bit field that specifies the number of words that follow in the instruction stream and are to be stored. The actual number of words is COUNT+1.

Z indicates the planes into which the data will be stored. If Z=0, the data will be stored only into plane 0, regardless of the activity bits. If Z=1, the data will be stored into all planes whose activity bits are set.

Current Pen Position in X is unchanged. Current Pen Position in Y is left at the last word stored + 1.

PERFORMANCE

Store Immediate Current requires 21 SYSCLK cycles plus 39 SYSCLK cycle per word stored.

EXAMPLE

This example writes the values 0,1,2,3,4 beginning at the Current Pen Position. In particular the following values are written into the indicated locations.

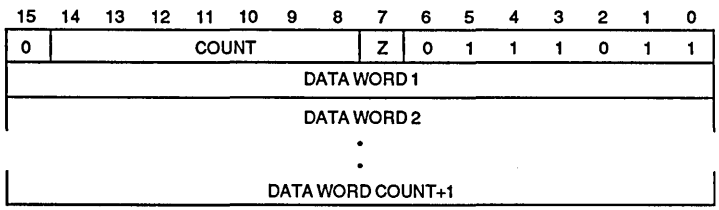
Value	Location
0000	CPP
0001	CPP,CPP+1
0002	CPP,CPP+2
0003	CPP,CPP+3
0004	CPP,CPP+4

```
043B
0000
0001
0002
0003
0004
```

CROSS-REFERENCES

COMMENTS

Clipping is ignored.



07785A 14-93

Store Immediate Current Format

String
(Drawing Primitive)

String is used to write text or other character-oriented patterns into display memory.

Six parameters are required by String:

SP is a 1-bit field that is used to indicate whether the font is stored in one plane only or is stored on all planes. If SP is a 0, the font is assumed to be stored in all planes whose activity bits are set. The attribute words will be taken from the plane that was specified in the FP field of the Set Character Font Position instruction. If SP is a 1, the font is stored only in the plane that was specified in the FP field of the Set Character Font Position instruction. In this case, the character will be replicated into all planes whose activity bits are set.

SI is a 1-bit field that is used to indicate whether the pattern words from the font area is to be inverted before being applied to the destination.

M is a 1-bit field that determines whether matching is to take place. If M is a 1, only pixels that match the search color will be changed.

SOAXZ is a 3-bit field that is used to indicate how the pattern words from the font area is to be applied to the destination.

BEGIN is specified with a standard operand address pair. This is the location to which CPP will be set before the first character is processed. If relative addressing is specified, BEGIN will be calculated with respect to CPP.

LIST is a variable-length string of 13-bit integer operands. Twelve of the bits are an index into one of the two fonts currently in use, and the 13th bit (the P bit) selects which of the two fonts to use. The C field, the leftmost bit of each entry in the LIST, is a continue bit. If it is set, the String instruction fetches another LIST entry. If it is a zero, the String instruction terminates after the entry has been processed.

PERFORMANCE (in SYSCLK cycles)

Instruction Setup	100
Per Character Setup	198
Per Scan Line (One Word)	44
Per Scan Line (Two Words)	74
Per Scan Line (Three Words)	100

EXAMPLE

See the example in Chapter 10, Section 10.3.

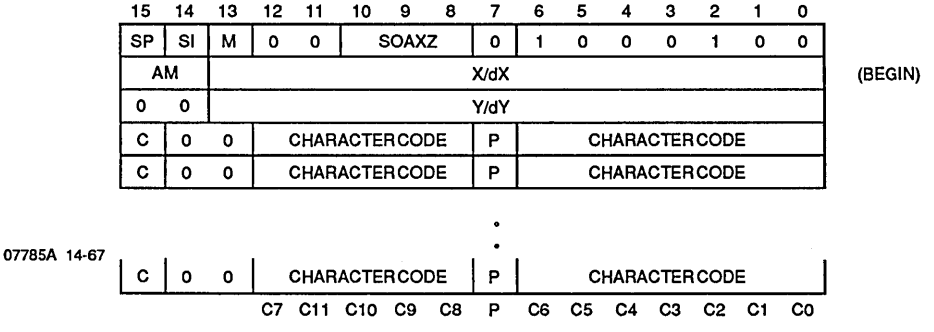
CROSS-REFERENCES

Chapter 8: Graphics Operations
Chapter 10: String Operations
Set Character Font Base (Instruction)

COMMENTS

While the examples given in this manual assume that the characters are encoded according to ASCII 77, the Am95C60 has no such restriction.

Observe that the weighting of the Character Code bits is not monotonic with bit position. In particular, bit 12 has the weight 128, while bit 8 has the weight 256. The character code weighting is indicated under the character code in the format diagram.



07785A 14-67

String Format

String Current
(Drawing Primitive)

String Current is used to write text or other character-oriented patterns into display memory. The text is written beginning at the Current Pen Position.

Five parameters are required by String Current:

SP is a 1-bit field that is used to indicate whether the font is stored in one plane only or is stored on all planes. If SP is a 0, the font is assumed to be stored in all planes whose activity bits are set. The attribute words will be taken from the plane that was specified in the FP field of the Set Character Font Position instruction. If SP is a 1, the font is stored only in the plane that was specified in the FP field of the Set Character Font Position instruction. In this case, the character will be replicated into all planes whose activity bits are set.

SI is a 1-bit field that is used to indicate whether the pattern words from the font area is to be inverted before being applied to the destination.

M is a 1-bit field that determines whether matching is to take place. If M is a 1, only pixels that match the search color will be changed.

SOAXZ is a 3-bit field that is used to indicate how the pattern read from the font area is to be applied to the destination.

LIST is a variable-length string of 13-bit integer operands. Twelve of the bits are an index into one of the two fonts currently in use, and the 13th bit (the P bit) selects which of the two fonts to use.

The C field, the leftmost bit of each entry in the LIST, is a continue bit. If it is set, the String Current instruction fetches another LIST entry. If it is a zero, the String Current instruction terminates after the entry has bee processed.

PERFORMANCE (in SYSCLK cycles)

Instruction Setup	50
Per Character Setup	198
Per Scan Line (One Word)	44
Per Scan Line (Two Words)	74
Per Scan Line (Three Words)	100

EXAMPLES

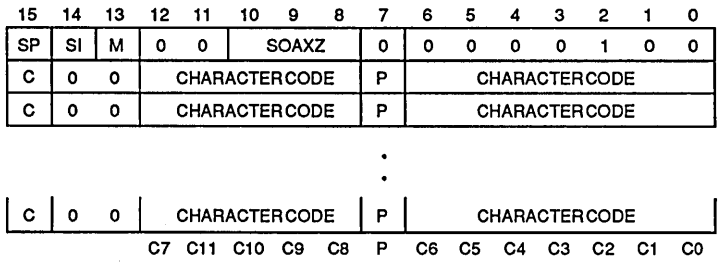
See the example in Chapter 10, Section 10.3.

CROSS-REFERENCES

Chapter 8: Graphic Operations
Chapter 10: String Operations
Set Character Font Base (Instruction)

COMMENTS

Observe that the weighting of the Character Code bits is not monotonic with bit position. In particular, bit 12 has the weight 128, while bit 8 has the weight 256. The character code weighting is indicated under the character code in the format diagram.



07785A 14-68

String Current Format

Transform Block
(Block Manipulation)

Transform Block is used to copy a rectangular block of data from one portion of display memory to another. Certain operations (rotate, zoom and mirror) can be performed on the image operand before it is stored into the destination operand area. The source image (after the operations) will be applied to the destination area under control of the SOAXZ field. The size of the source image prior to rotation and zooming will have been specified with Set Block Size.

Six parameters are required:

ROT is a 2-bit field that specifies whether the source operand is to be rotated before being stored. ROT is interpreted according to the following chart:

ROT	Rotation that is applied
0 0	None
0 1	90 degrees Counterclockwise
1 0	180 degrees
1 1	270 degrees Counterclockwise

MI is a 1-bit field that indicates whether the source image is to be mirrored before being stored. A 0 specifies the image is not to be mirrored; a 1 specifies the operand is to be mirrored. The source image is mirrored vertically about the left edge. Mirroring takes place before rotation and zooming.

SOAXZ is a 3-bit field that specifies how the image is to be applied to the current contents of the display memory.

SOURCE is specified with a standard operand address pair. This operand specifies the upper-left corner of the source image. Any addressing mode may be used. If relative addressing is used, SOURCE is calculated with respect to the Current Pen Position.

DESTINATION is specified with a standard operand address pair. This operand specifies the upper-left corner of the rectangular region into which the operand is to be stored. Any addressing mode may be specified. If relative addressing is specified, DESTINATION is calculated with respect to SOURCE.

ZOOM is specified as two 16-bit unsigned integers, allowing a zoom factor of 1 to 65,536. X is the first word, Y is the second. A value of 1 results in a zoom factor of 1.

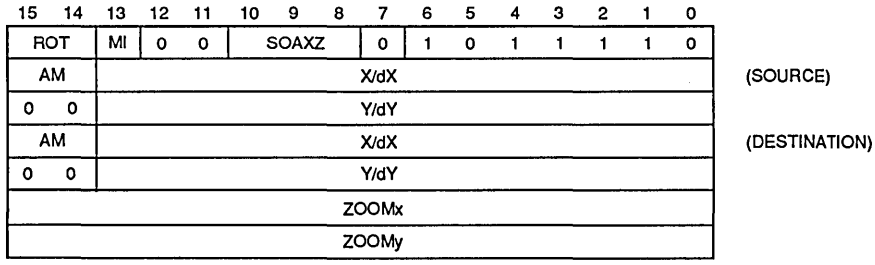
The CPP is left at the DESTINATION.

PERFORMANCE (in SYSCLK cycles)

Instruction Setup	153
Per Destination Pixel (Copy)	37
Per Destination Pixel (Mirror)	37
Per Destination Pixel (No Zoom)	37
Per Destination Pixel (4x4 Zoom)	24

EXAMPLE

This example transforms a block at 500,500 and writes it into a block at 1000,1000. The source operand is rotated 90 degrees counterclockwise



07785A 14-69

Transform Block Format

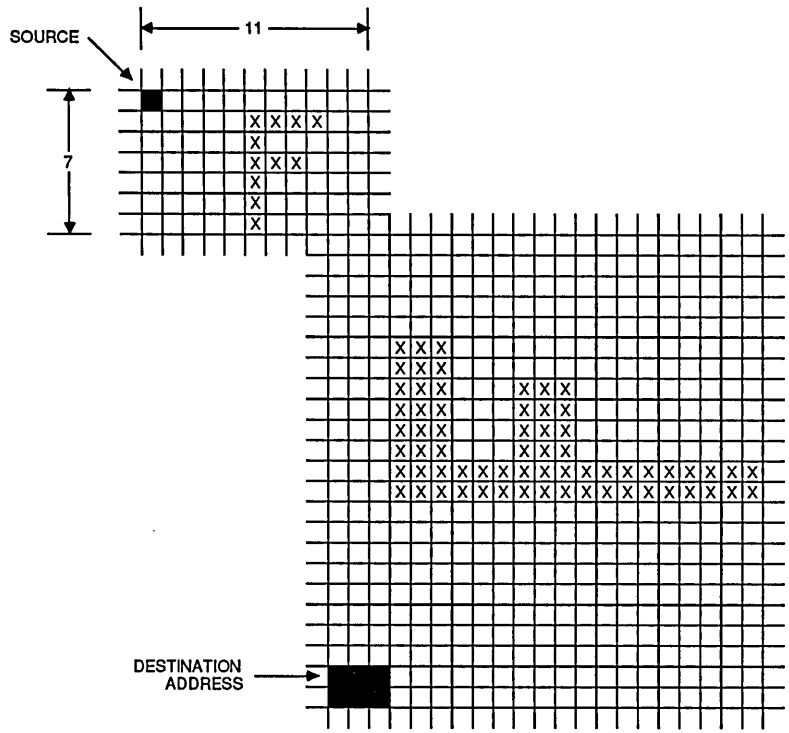
and is zoomed by a factor of 2 in the X dimension and 3 in the Y dimension.

```
405E      *Rotated by 90 degrees
01F4      *Source Operand
01F4
03E8      *Destination Operand
03E8
0002
0003      *ZOOOOOOOOOOOOOOOM
```

CROSS-REFERENCES

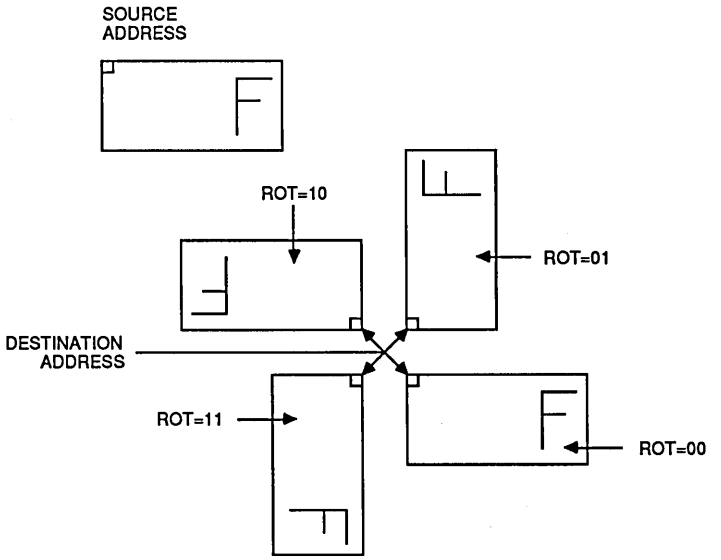
Chapter 5: Addressing Modes and Scaling
Chapter 8: Graphical Operations

TRANSFORM BLOCK EXAMPLE



07785A 14-73

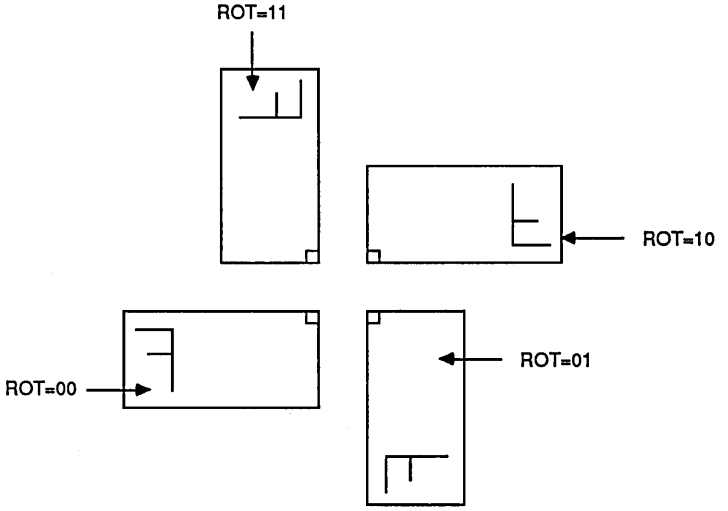
TRANSFORM BLOCK: ROTATION WITHOUT MIRRORING



07785A 14-85

MIRROR = 0

TRANSFORM BLOCK: ROTATION WITH MIRRORING



07785A 14-86

MIRROR=1

Transform Block Current
(Block Manipulation)

Transform Block Current is used to copy a rectangular block of data from one portion of display memory to another. Certain operations (rotate, zoom and mirror) can be performed on the source image before it is stored into the destination operand area. The source image (after the operations) will be applied to the destination area under control of the SOAXZ field. The size of the source image prior to rotation and zooming will have been specified with Set Block Size.

Five parameters are required:

ROT is a 2-bit field that specifies whether the source image is to be rotated before being stored. ROT is interpreted according to the following chart:

ROT	Rotation that is applied
0 0	None
0 1	90 degrees Counterclockwise
1 0	180 degrees
1 1	270 degrees Counterclockwise

MI is a 1-bit field that indicates whether the source image is to be mirrored before being stored. A 0 specifies the image is not to be mirrored; a 1 specifies the operand is to be mirrored. The source image is mirrored vertically about the left edge. Mirroring takes place before rotation and zooming.

SOAXZ is a 3-bit field that specifies how the image is to be applied to the current contents of the display memory.

DESTINATION is specified with a standard operand address pair. This operand specifies the upper-left corner of the rectangular region into which the operand is to be stored. Any addressing mode may be specified. If relative addressing is specified, DESTINATION is calculated with respect to the Current Pen Position.

ZOOM is specified as two 16-bit unsigned integers, allowing a zoom factor of 1 to 65,536. X is the first word, Y is the second. A value of 1 results in a zoom factor of 1.

The CPP is left at the DESTINATION.

PERFORMANCE (in SYSCLK cycles)

Instruction Setup	103
Per Destination Pixel (Mirror)	37
Per Destination Pixel (No Zoom)	37
Per Destination Pixel (4x4 Zoom)	24

EXAMPLE

This example transforms a block at the Current Pen Position and writes it into a block at 1000,1000. The source image is rotated 90 degrees counterclockwise and is zoomed by a factor of 3 in the X dimension and 5 in the Y dimension.

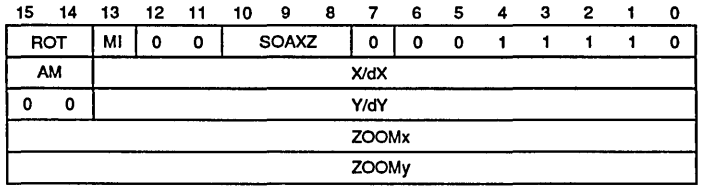
```
401E      *Rotated by 90 degrees
03E8      *Destination Operand
03E8
0003
0005      *ZOOOOOOOOOOOOOOOOM
```

CROSS-REFERENCES

Chapter 5: Addressing Modes and Scaling

COMMENTS

See Transform Block.



07785A 14-70

Transform Block Current Format

Appendix A
FURTHER READING

This list of titles of books may be of interest to the user of this manual.

Conrac Division/Conrac Corporation, *Raster Graphics Handbood*, 2nd Edition. New York: Van Nostrand Reinhold Company, 1985. ISBN 0-442-21608-4

Enderle, G., et al, *Computer Graphics Programming GKS*. New York: Springer-Verlag, 1984. ISNB 0-387-11525-0

Foley, J. D., and A. Van Dam, *Fundamentals of Interactive Computer Graphics*. Menlo Park: Addison-Wesley, 1983. ISBN 0-201-14468-9

Hopgood, F.R.A., et al, *Introduction to the Graphical Kernel System (GKS)*. London: Adademic Press, 1983. 0-12-355570-1

Newman, W. M., and R. F. Sproull, *Principles of Interactive Computer Graphics, 2nd Edition*. New York: McGraw-Hill, 1979. ISBN 0-07-046338-7

Van Deusen, Edmund, *Graphics Standards Handbook*. Laguna Beach: CC Exchange, 1985. ISBN 0-939078-01-5

ANSI X3.124-1985
American National Standards Institute, Inc.
1430 Broadway
New York, NY 10018

GAWAIN Document

QASM Document
AMD Order No. 08737B
January 1987

Am95C60 Data Sheet
AMD Order No. 07013B
April 1987

Appendix B
GLOSSARY

The glossary comes in two parts. The first part is a list of abbreviations and what they mean; the second part is a list of words and how they are used in this manual.

ABBREVIATIONS

AA	Anti-alias Anti-aliasing Anti-aliased
AM	Address Mode – 2-bit field
BIF	Block Input FIFO
BIS	Block Input Step – 2-bit field
BOF	Block Output FIFO
BOS	Block Output Step – 2-bit field
C	Continuation - 1-bit field
CE	Clipping Enable – 1-bit field
CPP	Current Pen Position
CRT	Cathode Ray Tube
CS	Cell Scale – 1-bit field
D	Direction (Char. Path) – 2-bit field
DAC	Digital-to-Analog Converter
E	Enable Logical PEL – 1-bit field
EP	End-Point Option – 3-bit field
FIFO	First-In First-Out
FP	Font Plane – 2-bit field
H	Height – 4-bit field
I	Index Present - 1-bit field
ICO	Inter-Character Space – 6-bit field
LS	Line Style – 2-bit field
LSB	Least Significant Bit
LSS	Line Style State – 2-bit field
M	Match – 1-bit field
ME	Masked write Enable – 1-bit field
MI	MIrror – 1-bit field
MSB	Most Significant Bit
NOP	No OPeration
PE	Picking Enable – 1-bit field
PO	Positive - 1-bit field
PP	Program Plane – 2-bit field
PPR	Plane Position Register – 6-bit field
PPRH	PlanePosition Reg HI – 4-bit field
QPDM	Quad Pixel Dataflow Manager (Am95C60)
RC	Return Code – 1-bit field
RD	Return Direction – 2-bit field
RGB	Red Green Blue
ROT	ROTation – 2-bit field
S	Skip – 4-bit field
SI	Source Invert – 1-bit field
SOAP	Standard Operand Address Pair
SOAXZ	SetOrAndXorZero – 3-bit field
SP	Single Plane – 1-bit field
VDAF	Video Data Assembly FIFO (Am8171)
VRAM	Video Random Access Memory
X/dX	XAddress or X Displacement field
Y/dY	YAddress or Y Displacement field
Z	Block I/O in depth – 1-bit field

TERMS

Addressing Mode (AM) – One of the four possible ways an operand address pair may be translated.

Back Porch – The portion of a composite video signal which occupies the time between the trailing edge of the sync pulse and the trailing edge of the corresponding blanking pulse.

Black Level – The video signal level which causes zero beam intensity at the CRT.

Blank Level – The video signal level which is more negative than black. The blank level is used in monitors with AC video signals to establish the proper DC levels.

Blanking – The process of decreasing the video signal level so that the screen is blank outside the active region.

Clipping – Not drawing parts of display elements that lie outside an area called the clipping region.

Column – Vertical groupings of pixels. Orthogonal with rows.

Current Pen Position (CPP) – An X,Y coordinate pair which "remembers" the drawing position between instructions. The Current Pen Position is used with some instructions and some addressing modes.

Display Element – A graphic element or primitive that can be used to construct a display image. Examples of display elements are lines, arcs and circles.

Display Memory – The memory used to contain images. This memory is organized into an array with X and Y coordinates with the origin at the upper-left corner. In addition to the area which is reproduced on the screen, the display memory contains invisible portions.

Display Surface – The portion of a graphics system upon which images actually appear. The term "screen" is used interchangeably.

Field – One of two equal parts into which a display frame is divided in an interlaced system.

Flicker – A perceivable periodic variation in the amount of light from the screen. Flicker is no longer perceived when the frequency of the variation exceeds a rate called the critical flicker frequency.

Fractional Error (FE) – The fractional part resulting from the last address translation. This is added into the real number following scaling in some address modes.

Frame – The total amount of information (as perceived by the viewer) presented on the screen. In an interlaced display, there are two fields (called even field and odd field) per frame.

Front Porch – The portion of a composite video signal which occupies the time between the leading edge of a blanking pulse and the leading edge of the corresponding sync pulse.

Horizontal Resolution – The number of addressable locations in the X (horizontal) dimension. The Am95C60 can support display memories with up to 4096 locations. The actual horizontal resolution in a real system is typically determined by the CRT monitor.

Horizontal Frequency – The rate at which scan lines are written onto the CRT monitor.

Image – A pattern written into display memory, typically for the purpose of being displayed on the screen. This word is often used in this book in the phrase "create the image".

Interlaced Scanning – A raster-scanning process in which two fields (the odd field and the even field) are combined on the screen to form a single frame.

Line Style (LS) – The fashion in which a line or arc is drawn. Examples of line styles are solid, dashed, and dotted.

Logical PEL – A multiplane "paint brush" which can be used for points, lines, arcs, and circles.

Pan – To move the image on the screen horizontally.

PEL – See Picture Element.

Picture Element (PEL) – The smallest unit of display memory which can be individually controlled. Often called pixel or pel.

Pixel – See Picture Element.

Pixel Aspect Ratio – The ratio of width to height of pixels as they appear on the screen. By providing nonuniform numbers of rows and columns of pixels, it is possible to compensate for screen aspect ratios other than 1:1.

Raster Graphics – A graphics system in which a display image is made up of an array of pixels arranged in rows and columns.

Row – Horizontal grouping of pixels. Orthogonal to columns.

Scaling – Enlarging or reducing all or part of a display image by multiplying the coordinates of display elements by a constant value.

Scroll – Moving an image vertically on the screen.

Scan Lines – See Vertical Resolution.

Screen – The portion of a graphics system upon which the image actually appears.

Screen Aspect Ratio – The ratio of width to height of a display. Typical landscape CRTs have an aspect ratio of 4:3 while page CRTs have an aspect ratio of 3:4.

Standard Operand Address Pair (SOAP) – Two words in the instruction stream which are translated into the absolute X-Y address of an operand in display memory.

Vertical Resolution – The number of pixels in the vertical direction. The Am95C60 can support vertical resolutions of up to 4096 pixels. Typically, system vertical resolution is determined by the CRT.

Vertical Frequency – The rate at which fields (not necessarily frames) are written onto the screen.

REVISION HISTORY

The following is a list of the changes that were made to the Am95C60 QPDM (Rev. A) Technical Manual (#07785A) to produce the Rev. B version of the technical manual (#07785B). This is not a formal *delta document*, but is rather a list of places where changes were made and the nature of the changes.

Table of Contents
The TOC was updated (mostly page number changes) to reflect the revised document. The format and outline of the document itself is unchanged except where necessary.

Chapter 1 Introduction
A block diagram of the QPDM was added. Table 1-1 (Display Memory Update Time) was completely revised to reflect the Rev. B timing.

Chapter 2 Hardware Interface
Table 2-1 (pinouts) was made somewhat easier to use and understand by making it correspond closer to the physical pinouts. The pin names and assignments have not changed.

The device characteristics, timing diagrams, and A.C. Parameters were removed. The data sheet (#07013B, April 87) contains this information. If you did not receive a copy of the data sheet with this manual and would like a copy, please contact Advanced Micro Devices. Please observe that some of the A.C. Parameters were changed; both functionally and parametrically.

Chapter 3 Host-Am95C60 Communications
The description of RESET was changed.

Chapter 4 Register Set
The Mnemonics and Addresses were added to the pictorial representation of the registers in Section 4.7.

Chapter 5 Addressing Modes and Scaling
Material was added and changed in the definition of the scale factors.
All references to scaling in the description of Absolute Addressing were removed.

Chapter 6 Line Texture
This chapter was completely rewritten. Line Styles and End Points are defined differently in Rev. B.

Chapter 7 Clipping and Picking
No changes.

Chapter 8 Graphical Operations
The operation that was called Graphical ZERO is now called Logical ZERO.
New material was added to Figures 8-1 and 8-2.
The explanations of the SI and SP bits were rewritten.

Chapter 9 Anti-aliasing
New material was added to almost every diagram in this chapter.
In addition, several examples of Comparator Anti-aliasing were added.

Chapter 10 String Operations
The chapter reflects the inverted sense of the high order bit of the LIST.

Chapter 11 Windows
No changes.

Chapter 12 Display Memory Configurations
Fixed a couple of typos.

Chapter 13 Miscellanea
Some changes to Section 13.1. Added two new sections.

Chapter 14 Instruction Set
This chapter was revised to reflect the instructions that were modified as well as the new instructions:

Arc Current:
The instruction word is different.
The operands are reordered.
The definition of RADIUS was changed.

Call:
Added the I and PO fields (and capability).
Added the index list.

Circle Current:
The instruction word is different.
The definition of RADIUS was changed.

Inquire:
This is a new instruction.

Jump:
Added the I and PO fields (and capability).

Line Current:
The instruction word is different.
Added the C field to the second operand pair.

Point:
The instruction word is different.
Added the C field to the address.

Point Current:
The instruction word is different.

Pop Current Pen Position:
New instruction

Push Current Pen Position:
New instruction

Set Line Style Phase:
Removed X bit.

Set Scale Factor:
Added fourth word and P bit.

Signal:
Changed Description.

Store Current Pen Position:
New instruction

String Current:
Changed Sense of continue bit.

Appendix A Further Reading
Added a couple of AMD documents.

Appendix B Glossary
Minor changes and corrections.

ADVANCED
MICRO
DEVICES, INC.
901 Thompson Place
P.O. Box 3453
Sunnyvale,
California 94088
(408) 732-2400
TWX: 910-339-9280
TELEX: 34-6306
TOLL FREE
(800) 538-8450

Order No. 07785B

IH-CP-10M-6/87-0