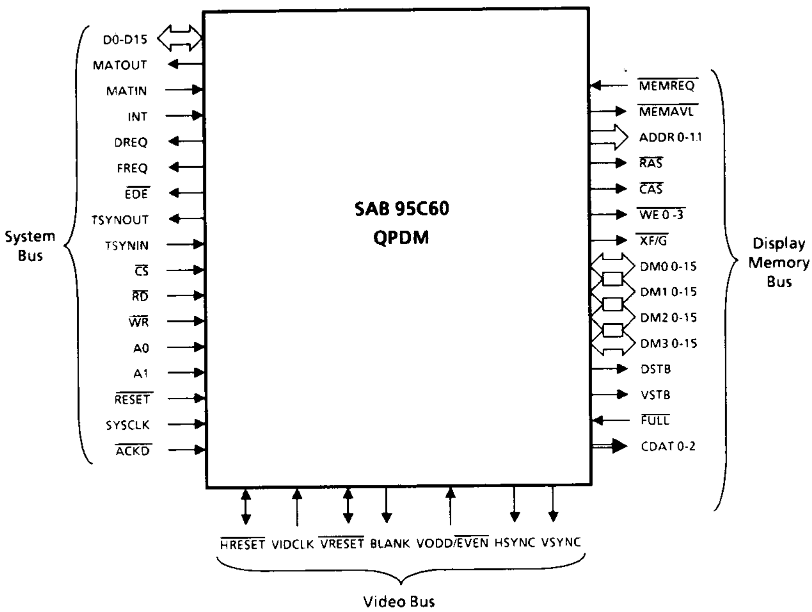
Advance Information

# SAB 95C60 Quad Pixel Dataflow Manager (QPDM)

- Generates mixed text and graphics within display memory
- Draws vectors up to 3.3 million pixels per second, or places text at 45,000 characters per second
- One chip handles four display memory planes of any size up to 4K × 4K bits and screen sizes up to 2K × 2K pixels
- Capable of cascading to handle multiple memory planes without system perfomance degradation
- Reflects GKS, VDI and NAPLPS software standards
- Supports windowing, panning and scrolling
- Supports drawing of anti-aliased vectors, circles and arcs with various user-definable line styles
- Fills arbitrary polygons
- Supports dual-port video DRAMs
- CMOS technology
- Provides memory and video refresh at user-definable rates
- Interfaces to any 8 or 16-bit system bus
- Comprehensive instruction set
- 145-pin PGA package

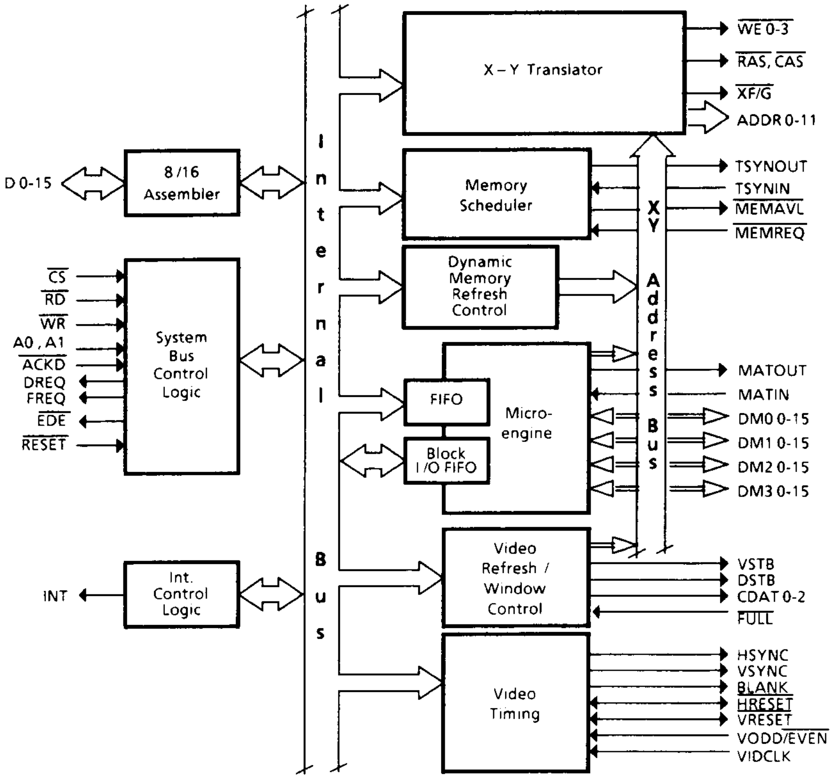
**Logic Symbol**




The Siemens SAB 95C60 Quad Pixel Dataflow Manager (QPDM) is a CMOS graphics processor which contains the necessary circuitry and control functions for driving four bit-mapped memory arrays. Featuring a maximum system clock speed of 20 MHz, the SAB 95C60 can interface to any 8 or 16-bit system bus, and can draw vectors at a speed of up to 3.3 million pixels per second or place text at a rate of 50,000 characters per second. Such performance allows the user to efficiently mix text and graphics within the bit map. The SAB 95C60 QPDM also contains graphics primitives which smoothly interface with the GKS, VDI and NAPLPS software standards.

9.87

**Block Diagram**

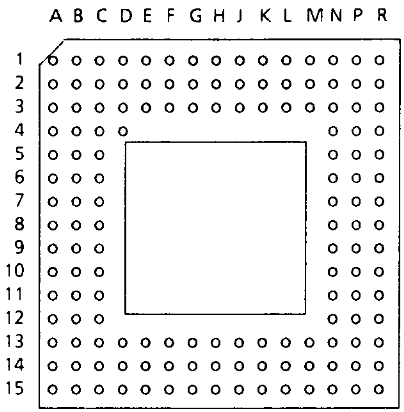


## Pin Configuration

**Pin Grid Array Package**
**Bottom view**

| | A | B | C | D | E | F | G | H | J | K | L | M | N | P | R | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | DM3 14 | DM3 13 | DM3 11 | DM3 10 | DM3 7 | DM3 5 | DM3 3 | DM3 1 | TEST | DM0 12 | DM0 10 | DM0 9 | DM0 7 | DM0 5 | DM0 1 | 1 |
| 2 | DM2 0 | DM3 15 | DM3 12 | DM3 8 | DM3 4 | DM3 2 | DM3 0 | MATOUT | DM0 14 | DM0 13 | DM0 11 | DM0 8 | DM0 2 | DM0 3 | DM0 0 | 2 |
| 3 | DM2 2 | DM2 4 | DM2 1 | DM3 9 | GND | DM3 6 | +5V | MATIN | +5V | DM0 15 | GND | DM0 6 | DM0 4 | DM1 4 | DM1 0 | 3 |
| 4 | DM2 6 | DM2 7 | DM2 5 | NC | | | | | | | | | DM1 1 | DM1 3 | DM1 2 | 4 |
| 5 | DM2 9 | DM2 11 | DM2 3 | | | | | | | | | | DM1 7 | DM1 6 | DM1 5 | 5 |
| 6 | DM2 13 | DM2 8 | GND | | | | | | | | | | GND | DM1 8 | DM1 9 | 6 |
| 7 | DM2 15 | DM2 12 | DM2 10 | | | | | | | | | | DM1 14 | DM1 10 | DM1 11 | 7 |
| 8 | DM2 14 | ADDR10 | ADDR11 | | | | | | | | | | DM1 12 | DM1 13 | DM1 15 | 8 |
| 9 | ADDR 8 | ADDR 9 | GND | | | | | | | | | | GND | D 14 | D 15 | 9 |
| 10 | ADDR 6 | ADDR 5 | +5V | | | | | | | | | | D 13 | D 12 | D 11 | 10 |
| 11 | ADDR 4 | ADDR 7 | ADDR 1 | | | | | | | | | | GND | D 10 | D 9 | 11 |
| 12 | ADDR 3 | ADDR 2 | GND | | | | | | | | | | D 8 | D 7 | D 6 | 12 |
| 13 | $\overline{FULL}$ | ADDR 0 | $\overline{RAS}$ | $\overline{WE0}$ | $\overline{CAS}$ | $\overline{XF/G}$ | +5V | VODD /$\overline{EVEN}$ | DREQ | INT | +5V | $\overline{WR}$ | D 5 | D 3 | D 4 | 13 |
| 14 | VSTB | CDAT 2 | CDAT 0 | $\overline{WE1}$ | TSYNIN | $\overline{MEMAVL}$ | VSYNC | BLANK | $\overline{HRESET}$ | SYSCLK | $\overline{RESET}$ | $\overline{RD}$ | $\overline{ACKD}$ | D 1 | D 2 | 14 |
| 15 | DSTB | CDAT 1 | $\overline{WE3}$ | WE 2 | TSYNOUT | $\overline{MEMREQ}$ | HSYNC | $\overline{VRESET}$ | VIDCLK | FREQ | A 0 | A 1 | $\overline{CS}$ | $\overline{EDE}$ | D 0 | 15 |
| | A | B | C | D | E | F | G | H | J | K | L | M | N | P | R | |

**PGA Package**
**Bottom view**



## Pin Names in Alphabetical Order

| Pin Name/ Function | Pin | Pin Name/ Function | Pin | Pin Name/ Function | Pin |
|---|---|---|---|---|---|
| A0 | L15 | D13 | N10 | DM2 00 | A2 |
| A1 | M15 | D14 | P9 | DM2 01 | C3 |
| ADDR 00 | B13 | D15 | R9 | DM2 02 | A3 |
| ADDR 01 | C11 | | | DM2 03 | C5 |
| ADDR 02 | B12 | DM0 00 | R2 | DM2 04 | B3 |
| ADDR 03 | A12 | DM0 01 | R1 | DM2 05 | C4 |
| ADDR 04 | A11 | DM0 02 | N2 | DM2 06 | A4 |
| ADDR 05 | B10 | DM0 03 | P2 | DM2 07 | B4 |
| ADDR 06 | A10 | DM0 04 | N3 | DM2 08 | B6 |
| ADDR 07 | B11 | DM0 05 | P1 | DM2 09 | A5 |
| ADDR 08 | A9 | DM0 06 | M3 | DM2 10 | C7 |
| ADDR 09 | B9 | DM0 07 | N1 | DM2 11 | B5 |
| ADDR 10 | B8 | DM0 08 | M2 | DM2 12 | B7 |
| ADDR 11 | C8 | DM0 09 | M1 | DM2 13 | A6 |
| $\overline{\text{ACKD}}$ | N14 | DM0 10 | L1 | DM2 14 | A8 |
| | | DM0 11 | L2 | DM2 15 | A7 |
| BLANK | H14 | DM0 12 | K1 | | |
| | | DM0 13 | K2 | DM3 00 | G2 |
| $\overline{\text{CAS}}$ | E13 | DM0 14 | J2 | DM3 01 | H1 |
| CDAT0 | C14 | DM0 15 | K3 | DM3 02 | F2 |
| CDAT1 | B15 | | | DM3 03 | G1 |
| CDAT2 | B14 | DM1 00 | R3 | DM3 04 | E2 |
| $\overline{\text{CS}}$ | N15 | DM1 01 | N4 | DM3 05 | F1 |
| | | DM1 02 | R4 | DM3 06 | F3 |
| D00 | R15 | DM1 03 | P4 | DM3 07 | E1 |
| D01 | P14 | DM1 04 | P3 | DM3 08 | D2 |
| D02 | R14 | DM1 05 | R5 | DM3 09 | D3 |
| D03 | P13 | DM1 06 | P5 | DM3 10 | D1 |
| D04 | R13 | DM1 07 | N5 | DM3 11 | C1 |
| D05 | N13 | DM1 08 | P6 | DM3 12 | C2 |
| D06 | R12 | DM1 09 | R6 | DM3 13 | B1 |
| D07 | P12 | DM1 10 | P7 | DM3 14 | A1 |
| D08 | N12 | DM1 11 | R7 | DM3 15 | B2 |
| D09 | R11 | DM1 12 | N8 | | |
| D10 | P11 | DM1 13 | P8 | DSTB | A15 |
| D11 | R10 | DM1 14 | N7 | DREQ | J13 |
| D12 | P10 | DM1 15 | R8 | | |

## Pin Names in Alphabetical Order (cont'd)

| Pin Name/ Function | Pin |
|---|---|
| $\overline{\text{EDE}}$ | P15 |
| | |
| FREQ | K15 |
| $\overline{\text{FULL}}$ | A13 |
| | |
| GND | C9 |
| GND | C12 |
| GND | N9 |
| GND | L3 |
| GND | N6 |
| GND | E3 |
| GND | C6 |
| GND | N11 |
| | |
| $\overline{\text{HRESET}}$ | J14 |
| HSYNC | G15 |
| | |
| INT | K13 |
| | |
| MATIN | H3 |
| MATOUT | H2 |
| $\overline{\text{MEMAVAL}}$ | F14 |
| $\overline{\text{MEMREQ}}$ | F15 |
| | |
| NC | D4 |

| Pin Name/ Function | Pin |
|---|---|
| $\overline{\text{RAS}}$ | C13 |
| $\overline{\text{RD}}$ | M14 |
| $\overline{\text{RESET}}$ | L14 |
| | |
| SYSCLK | K14 |
| TEST | J1 |
| TSYNIN | E14 |
| TSYNOUT | E15 |
| | |
| VIDCLK | J15 |
| VODD/$\overline{\text{EVEN}}$ | H13 |
| $\overline{\text{VRESET}}$ | H15 |
| VSTB | A14 |
| VSYNC | G14 |
| | |
| $V_{CC}$ | G3 |
| $V_{CC}$ | G13 |
| $V_{CC}$ | J3 |
| $V_{CC}$ | C10 |
| $V_{CC}$ | L13 |
| | |
| $\overline{\text{WE0}}$ | D13 |
| $\overline{\text{WE1}}$ | D14 |
| $\overline{\text{WE2}}$ | D15 |
| $\overline{\text{WE3}}$ | C15 |
| $\overline{\text{WR}}$ | M13 |
| | |
| $\overline{\text{XF}}/\overline{\text{G}}$ | F13 |

## Pin Definitions and Functions

**System Bus**

| Symbol | Pin | Input (I)<br>Output O) | Function |
|---|---|---|---|
| D0-D15 | N10<br>N12, N13<br>P9-P14<br>R9-R15 | I/O | COMMAND/DATA/STATUS (tristate, bidirectional)<br>These sixteen lines are used for transferring commands/ data/status on the system bus. The nature of the information transferred on the D0-D15 lines is established in conjunction with the port address pins A0, A1. |
| $\overline{RD}$ | M14 | I | $\overline{READ}$ (input, active low)<br>Signal used for reading information (data/status) from the SAB 95C60 QPDM on the D0-D15 lines by a bus master. |
| $\overline{WR}$ | M13 | I | $\overline{WRITE}$ (input, active low)<br>Signal used for strobing information (commands/data) into the SAB 95C60 on the D0-D15 lines. |
| $\overline{CS}$ | N15 | I | $\overline{CHIP\ SELECT}$ (input, active low)<br>Signal used for selecting the SAB 95C60 from several peripherals connected to the same system bus. |
| A0, A1 | L15, M15 | I | PORT ADDRESS (input, active high)<br>These two inputs are used for selecting the appropriate port to be read or written. |
| SYSCLK | K15 | I | SYSTEM CLOCK (input, active high)<br>20 MHz maximum frequency clock. Controls the SAB 95C60 QPDM internal timing exept for video timing. |
| INT | K13 | O | INTERRUPT (output, active high)<br>High-level interrupt output used to signal that an exception has occured. The nature of the exception can be determined by reading the status register. Execution of a write to the interrupt acknowledge register clears the INT output. |
| DREQ | J13 | O | DATA FIFO REQUEST (output, open drain, active high)<br>Signal used to start and suspend a transfer of data between the system memory and the display memory. |
| $\overline{ACKD}$ | N14 | I | $\overline{ACKNOWLEDGE\ DMA}$ (input, active low)<br>The external DMA device may drive this pin low in response to a DMA request to strobe in or read out data in fly-by DMA transfer format. |
| FREQ | K15 | O | INSTRUCTION FIFO REQUEST (output, open drain, active high)<br>This signal is used to start and suspend a transfer of instructions from the system memory into the QPDM instruction FIFO. |

## Pin Definitions and Functions (cont'd)

**System Bus** (cont'd)

| Symbol | Pin | Input (I)<br>Output (O) | Function |
|---|---|---|---|
| $\overline{EDE}$ | P15 | O | $\overline{\text{EXTERNAL DRIVER ENABLE}}$ (output, active low)<br>This pin is used to enable external data bus drivers on the system bus and is inactive (high) during an output block operation on those SAB 95C60 devices that do not participate in the output. This signal eliminates contention on the system bus. |
| MATOUT | H2 | O | MATCH OUT (output, active high)<br>This pin is used in a multiple QPDM application to search for a matching pattern. As long as the pattern is not found, this pin stays low. When the matching pattern is found, the pin is driven high. Since all the MATOUT outputs are logically ANDed externally, a match in a multiple QPDM environment is visible on MATIN when all the MATOUT outputs are high. This pin is also used for instruction execution synchronization by re-aligning the SAB 95C60 devices in a system at the beginning of each instruction execution and at the beginning of each word transfer in a block I/O instruction. |
| MATIN | H3 | I | MATCH IN (input, active high)<br>This pin is connected to the output of the AND gate connected to the MATOUT outputs. |
| TSYNOUT | E15 | O | TIMING SYNCHRONIZATION (output, active high)<br>In conjunction with TSYNIN, this pin is used to synchronize display memory bus activities. The TSYNOUT pins of all SAB 95C60 devices in a system are ANDed together and connected (optionally buffered) to the TSYNIN input pins of all SAB 95C60s. User-transparent information signals all SAB 95C60 devices about display memory bus activities. |
| TSYNIN | E14 | I | TIMING SYNCHRONIZATION (input, active high)<br>All TSYNIN input pins are connected to the TSYNOUT junction, or to the output of the optional buffer. |
| $\overline{RESET}$ | L14 | I | $\overline{\text{SYSTEM RESET}}$ (input, active low)<br>The $\overline{RESET}$ signal brings all SAB 95C60s in the system to the same initial state. All the outputs are brought into the inactive state. If this pin is activated while the SAB 95C60 is active, all activities will be suspended. |
| TEST | J1 | I | TEST INPUT<br>Must be grounded. |

## Pin Definitions and Functions (cont'd)

### Display Memory Bus

| Symbol | Pin | Input (I)<br>Output (O) | Function |
|---|---|---|---|
| ADDR 0–11 | A9–A12<br>B8–B13<br>C8, C11 | O | ADDRESS (output, active high)<br>The twelve lines of ADDR are used for addressing bit-map planes each up to 4 K × 4 K bits. The addresses are multiplexed and contain row and column addresses and bank-select bits. |
| DM0 0–15<br>DM1 0–15<br>DM2 0–15<br>DM3 0–15 | | I/O | DISPLAY MEMORY BUS (bidirectional)<br>These 64 lines are used for transferring data between the SAB 95C60 and the display memory. There are sixteen lines for each of the four planes. |
| $\overline{\text{RAS}}$ | C13 | O | $\overline{\text{ROW ADDRESS STROBE}}$ (output, active low)<br>This line is used for strobing the memory planes to be read or written to at the address on ADDR (row address). |
| $\overline{\text{CAS}}$ | E13 | O | $\overline{\text{COLUMN ADDRESS STROBE}}$ (output, active low)<br>This line is used for strobing the memory planes to be read or written to at the address on ADDR (column address). |
| $\overline{\text{XF/G}}$ | F13 | O | $\overline{\text{TRANSFER/OUTPUT ENABLE}}$ (output, active low)<br>This pin interfaces directly to the video DRAM. During a transfer cycle this pin indicates a transfer to the video DRAMs. During a random read cycle, this pin enables the output buffer of the video DRAMs. |
| $\overline{\text{WE 0}}$ – $\overline{\text{WE 3}}$ | D13–D15<br>C15 | O | $\overline{\text{WRITE ENABLE}}$ (output, active low)<br>The $\overline{\text{WE}}$, when active, signifies that the current transaction on the display memory bus is a write to the corresponding bit plane. |
| $\overline{\text{MEMREQ}}$ | F15 | I | $\overline{\text{MEMORY REQUEST}}$ (input, active low)<br>This asynchronous signal is used by an external device to request access to the display memory bus. |
| $\overline{\text{MEMAVL}}$ | F14 | O | $\overline{\text{MEMORY BUS AVAILABLE}}$ (output, active low)<br>This line is used to inform external devices requesting the bus that the SAB 95C60 is not driving the data lines. |

## Pin Definitions and Functions (cont'd)

### Video Control Bus

| Symbol | Pin | Input (I) Output (O) | Function |
|---|---|---|---|
| VODD/$\overline{\text{EVEN}}$ | H13 | I | VERTICAL ODD/$\overline{\text{EVEN}}$ (input)<br>This input is optionally used in the interlaced display mode to distinguish between the even frame and the odd frame produced by an external device synchronized with the SAB 95C60. |
| VSTB | A14 | O | VIDEO STROBE (output, active high)<br>This strobe signal is used in a system with video dynamic RAMs (VRAMs) and an external VDAF (AM8171/8172 discrete components) to shift video data out of the video memory. With every strobe, a 16-bit word is shifted out of the video memory. |
| DSTB | A15 | O | DATA STROBE (output, active high)<br>This strobe signal loads 8 bits of video data into the VDAF. DSTB is synchronous to SYSCLK and has twice the frequency of VSTB. |
| CDAT 0–2 | B14, B15<br>C14 | O | CONTROL DATA (output, active high)<br>These lines are used to output three bits of information to the VDAF. Information is sent to the VDAF during the transfer cycle. |
| $\overline{\text{FULL}}$ | A13 | I | $\overline{\text{FULL}}$ (input, active low)<br>This input alerts the SAB 95C60 that the VDAF cannot accept more video data. If the $\overline{\text{FULL}}$ signal is active, the SAB 95C60 stops generating the VSTB and DSTB signals. |
| VIDCLK | J15 | I | VIDEO CLOCK (input)<br>15 MHz maximum frequency clock. This clock signal is used for generating video synchronization signals and for loading the video serialization registers. |
| HSYNC | G15 | O | HORIZONTAL SYNC (output, active high)<br>HSYNC is an active high output used to cause horizontal retrace of the CRT's electron beam. This output is held low when the SAB 95C60 is reset to prevent any uncontrolled synchronization to the CRT which may cause damage of the tube. |
| VSYNC | G14 | O | VERTICAL SYNC (output, active high)<br>VSYNC is an active high output used to cause vertical retrace of the CRT's electron beam. This output is held low when the SAB 95C60 is reset. |
| BLANK | H14 | O | BLANK VIDEO (output, active high)<br>BLANK is an active high output which serves to blank out inactive display areas of the CRT. This output is held high when the SAB 95C60 is reset. |

## Pin Definitions and Functions (cont'd)

**Video Control Bus** (cont'd)

| Symbol | Pin | Input (I) Output (O) | Function |
|---|---|---|---|
| $\overline{\text{HRESET}}$ | J14 | I/O | $\overline{\text{HORIZONTAL RESET}}$ (bidirectional, active low)<br>This pin is an output for horizontal video masters, and an input for horizontal video slaves. It is used for horizontal video synchronization to other SAB 95C60 devices or to an external video source. |
| $\overline{\text{VRESET}}$ | H15 | I/O | $\overline{\text{VERTICAL RESET}}$ (bidirectional, active low)<br>This pin is an output for vertical video masters and an input for vertical video slaves. It is used for vertical video synchronization to other SAB 95C60 devices or to one external video source. |

**Power Connections**

| Symbol | Pin | Function |
|---|---|---|
| $V_{CC}$ | C10, G3<br>G13, I3<br>L13 | POWER SUPPLY<br>Each $V_{CC}$ must be connected to + 5 V power supply. |
| GND | C6, C9<br>C12, E3<br>L3, N6<br>N9, N11 | GROUND<br>Each GND pin must be connected to the power supply ground. |

## Functional Description

The SAB 95C60 quad pixel dataflow manager is a graphics processor which maintains, updates and displays information on four bit-mapped video planes. As depicted in the figure below, the system interface communicates with either an 8 or 16-bit host CPU while the display memory interface controls four bit-mapped memory planes. These planes consist of dual-port video dynamic memories (VRAMs). The SAB 95C60 connects VRAMs to the random ports while the serial ports are used as access to video information for the screen refresh. When used in conjunction with a video data assembly FIFO, the SAB 95C60 is capable of displaying a hardware window with the associated features of smooth pan and soft scroll.

The SAB 95C60 QPDM performs three fundamental functions described as follows:

### Video Refresh

The SAB 95C60 QPDM manages the screen (display) refresh function by generating the addresses to the bit-mapped VRAMs as required to access data for display on the screen. The data from the VRAMs is serialized externally to the SAB 95C60.

The video refresh operation is fully programmable allowing the user to tailor the system as required. The screen display can be aligned on any pixel boundary and can also include one hardware window overlay. Additionally, the total video process can be externally synchronized to any external source at the horizontal or vertical synchronization rate. Video refresh can be disabled for operation as a slave device.

### Dynamic Memory Refresh

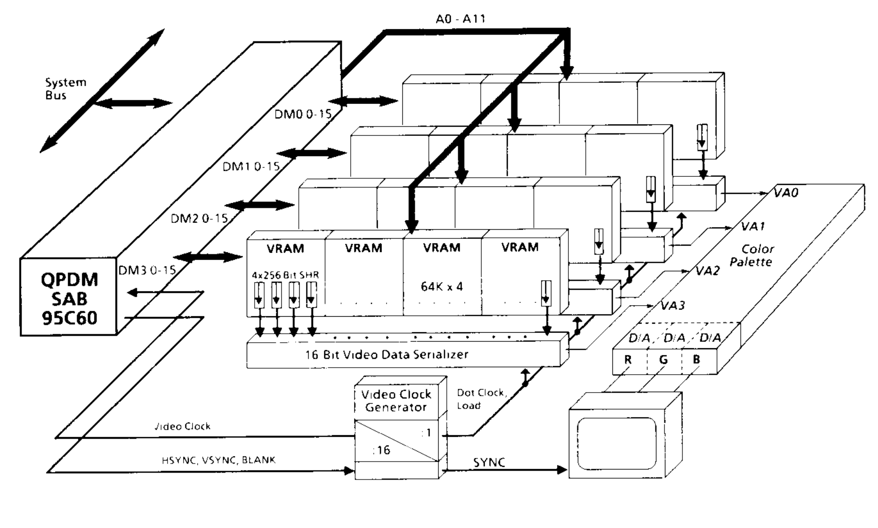
The SAB 95C60 performs the dynamic memory refresh function for the display memory. The dynamic memory refresh process is interleaved with the video refresh and with the display memory updating. The refresh rate can be programmed by loading the 10-bit dynamic memory refresh rate register.

This register holds the number of SYSCLK cycles between two refresh cycles.

The dynamic memory refresh rate register is loaded into a counter, which is counted down by the SYSCLK. When the counter reaches its zero state, it sends a refresh request to the memory scheduler. As soon as the memory scheduler arbitrates a time slice for the dynamic memory refresh block, a refresh cycle is initiated.

The SAB 95C60 executes $\overline{CAS}$ before $\overline{RAS}$ refresh cycles.

**Typical QPDM Graphics System**



### Display Memory Update

The SAB 95C60 has access to the display memory bus for updating purposes, except for those times when it is performing video or dynamic memory refresh.

Placing an instruction in the execution FIFO (see block diagram) signals the microengine to begin operation. If the execution of the instruction requires access to the display memory bus (this is the case in most instructions), the SAB 95C60 continues the instruction execution unless the display memory bus is occupied by the video refresh or the dynamic memory refresh process. If the display memory bus cannot be accessed momentarily, the execution is suspended until the display memory is available.

The drawing instruction set includes line, circle, fill polygon, string and many others which will be briefly described in the following.

## Feature Description

### Window Display Mechanism

The SAB 95C60 QPDM, in conjunction with a video data assembly FIFO (VDAF) or an equivalent device, can support a single non-destructive hardware window. The image to appear in the window is located in some other area of the display memory than that visible on the screen. The size and position of the window is programmed into a set of registers on the SAB 95C60. Since the window position is dynamically programmable, it is easy to 'drag' a rectangular area containing an object. It is also easy to perform soft scrolling and smooth panning of either background or foreground.

The window may be positioned arbitrarily on the screen and it may be of any size. Final serialization cannot occur until data from the background has window. Five signals are provided to control a data assembler.

$\overline{\text{MEMREQ}}$ and $\overline{\text{MEMAVL}}$ are used to allow another processor direct access to the display memory.

### Clipping

The clipping feature on the SAB 95C60 allows a rectangular region to be defined outside of which vectors and arcs will not be drawn, blocks will not be moved or modified, and polygons will not be filled. The clipping window is specified by the user, and remains in effect until changed or disabled.

### Picking

If a drawing consists of a large number of objects and each object is defined by a number of drawing primitives, any object can be identified by the following picking process:

First, the picking area is defined as a rectangular region in display memory. Whenever a drawing intersects the picking area, a "pick detect" bit in the status register is set. Objects to be displayed are labeled by using the signal instruction which will return the label of the object that intersected the picking area.

### String Mode

The SAB 95C60 QPDM has the capability of generating alphanumerics and so-called "character graphics" efficiently and with a minimum of user intervention. This capability is referred to as "string mode".

In string mode, the user issues a string of 16-bit characters which are then written into the bit map. Before using this mode, the user must load a "font" into the display memory. All mixing of text and graphics is accomplished within the display memory. Character patterns generated in this manner can be of arbitrary size and alignment.

## Multiple SAB 95C60 Operation

In order to accommodate systems requiring access to more than four bit planes, the SAB 95C60 is designed to be fully cascadable with no performance degradation. Multiple SAB 95C60 devices can communicate with each other to share timing information for synchronization purposes, and status information for color comparisons in depth.

Instructions that are to be executed are transmitted to all SAB 95C60 devices at once. Each plane will use the instruction in conjunction with its own activity and color bits to decide whether to execute or to ignore the instruction. Each plane may execute the instruction differently, depending on the contents of the individual memory plane and the contents of plane-specific status information.

Transmission is accomplished by selecting all SAB 95C60 QPDMs simultaneously and writing to port 0. (Write to the instruction FIFO.)

TSYNOUT and TSYNIN are used to synchronize display memory operations in a multiple SAB 95C60 system.

MATOUT and MATIN are used to exchange color searching information in a multiple SAB 95C60 device system and to synchronize instruction execution.

## Interface Description

### System Bus Interface

The host is connected to the system side of the SAB 95C60 as depicted in the logic symbol diagram. There is a 16-bit data path, and the 8 or 16-bit option allows the SAB 95C60 to be connected to an 8 or 16-bit host processor.

The normal bus interface control lines are supported through $\overline{CS}$, $\overline{RD}$, $\overline{WR}$ and two address bits. The address bits are decoded to select one of four ports:

| A1 | A0 | Write Function | Read Function |
|---|---|---|---|
| 0 | 0 | Write Instruction FIFO | Read Status |
| 0 | 1 | Write Block Input FIFO | Read Block Output FIFO |
| 1 | 0 | Write Register Address | Read Register Address |
| 1 | 1 | Write Register | Read Register |

The SAB 95C60 QPDM can be supported with a DMA controller allowing blocks of information or instructions to be transferred without tying up the host. In addition to the normal flow-through DMA operations, the SAB 95C60 also supports fly-by operations.

The SAB 95C60 can also use interrupts to signal the occurrence of certain events. Some events are repetitive (e.g. frame), some indicate error conditions (e.g. stack overflow), and some merely report status (e.g. idle). There are registers for masking interrupts, reading interrupt requests and acknowledging interrupts.

### Display Memory Bus Interface

On the display memory side, the SAB 95C60 is capable of controlling four bit planes. Addresses, $\overline{RAS}$ and $\overline{CAS}$ are common signals to all bit planes while each plane has its own set of data lines and write enable ($\overline{WE}$). Typically, eight or nine address lines (multiplexed row/column) go to the VRAM devices while the others may be used for bank select. If multiple banks of memory are used, row addresses are decoded to select the proper bank.

Each bit plane has a 16-bit data bus used for the display memory update function. Typically, to write a single pixel (one bit in each plane), the SAB 95C60 would perform simultaneous 16-bit reads from all planes, followed by simultaneous 16-bit writes to all planes. Logic is also provided in the SAB 95C60 to perform individual pixel writes.

The SAB 95C60 is intended to be used with a variety of dual-ported video RAMs. Prior to the beginning of each scan line, the SAB 95C60 executes a transfer cycle which copies the contents of the scan line into shift registers on the VRAM devices. The scan line image is then shifted out of the VRAM devices 16 bits at a time and further serialized at the dot clock rate for display purposes. The primary VRAM port is available during this time for display memory update.

### Display Memory/RAM Size Examples

| Bit Map Size | VRAM Size | VRAMs/ Plane | VRAMs/ SAB 95C60 |
|---|---|---|---|
| 1024 × 1024 | 64K × 4 | 4 | 16 |
| 1024 × 2048 | 64K × 4 | 8 | 32 |
| 2048 × 2048 | 64K × 4 | 16 | 64 |
| 4096 × 4096 | 256K × 4 | 16 | 64 |

### Video Bus Interface

The SAB 95C60 generates video timing. Horizontal timing is programmed in terms of VIDCLK cycles and vertical timing is programmed in terms of scan lines.

$\overline{HRESET}$ and $\overline{VRESET}$ are either inputs or outputs depending on the SAB 95C60 master/slave status. In the case where multiple SAB 95C60s control the display, one SAB 95C60 would be programmed as the timing master and the others would be programmed as timing slaves.

## Performance Figures

Working with VRAMs, the SAB 95C60 can easily manage high-resolution screen formats requiring dot clock rates up to 240 MHz.

Moreover, it performs the bit-map update function as indicated in the table below:
The SAB 95C60 provides extremely fast access at the host interface. A host write requires as little as 80 ns, while a register may be read in as little as 120 ns.

**Performance Figures of SAB 95C60**

| Instruction | Instruction Overhead | Intermediate Overhead | Execution Time | Comments |
|---|---|---|---|---|
| Line | 12.9 μs | (not applicable) | 300 ns/Pixel | |
| Line | 12.9 μs | (not applicable) | 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 | 55 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 | |

## Register Description

The SAB 95C60 QPDM contains a number of registers which are programmable by the host. These registers are listed below:

**Video Control** uses eight registers which define video operation parameters:

- Horizontal Sync Pulse Width (HSYNC)
- Horizontal Scan Delay (HDEL)
- Horizontal Active Pixels (HACT)
- Horizontal Total Count (HTOT)
- Vertical Sync Pulse Width (VSYNC)
- Vertical Scan Delay Odd (VDELODD)
- Vertical Scan Delay Even (VDELEVEN)
- Vertical Active Lines (VACT)
- Vertical Total Lines (VTOT)

**Visible Screen Coordinates** use four registers which contain the (x,y) address in real memory of the top-left and bottom-right corner of the visible screen in display memory:

- Screen X Start
- Screen Y Start
- Screen X Terminate
- Screen Y Terminate

**Window Control** uses six registers which specify where the window is located on the screen (the apparent window) and where it begins in memory (the real window):

- Window Apparent X Start
- Window Apparent Y Start
- Window Apparent X Terminate
- Window Apparent Y Terminate
- Window Real X
- Window Real Y

**Video Mode Register** is used to indicate to each SAB 95C60 whether it is a timing master or slave, and whether or not interlaced display mode is to be used.

**Memory Mode Register** specifies display memory configuration in terms of memory width and device size (e.g. 4K wide/256 K devices).

**Dynamic Memory Refresh Rate Register** specifies the number of SYSCLK clock cycles between row refreshes in dynamic memory refresh.

**Interrupt Mask Register** indicates which conditions are allowed to cause interrupts to the host.

**Video Timing Enable** is a 1-bit register used to enable and disable video sync and output.

**Video Refresh Enable** is a 1-bit register used to enable and disable the collection of the video information from the display RAM.

**Interrupt Acknowledge Register** indicates that a specific interrupt condition is known to the host and that the request is to be cancelled in the SAB 95C60.

**System Bus Width Register** configures the SAB 95C60 to 8-bit or 16-bit-system bus modes.

## Instruction Set

The SAB 95C60 QPDM is a graphics processor with a powerful instruction set. Most of the graphics instructions can be specified with respect to several addressing modes, relative to screen position, pen position, and absolute memory. The SAB 95C60 QPDM instructions are briefly described in the following:

**Arc** draws the image of a circular arc in display memory. The input parameters are the center of the arc, the radius of the arc and the two end-points. The image may be drawn using anti-aliasing.

**Arc Current** draws the image of a circular arc in display memory. It is similar to "arc" except the center is taken to be the current pen position (rather than being specified).

**Call** begins fetching instructions from display memory rather than from the instruction FIFO. The parameters specify the location of the program to be called. A stack in display memory is used to contain the return location. The call instruction may also be indexed or conditional.

**Circle** draws the image of a circle in display memory. The input parameters are the circle's center and radius. The image can be drawn using anti-aliasing.

**Circle Current** draws the image of a circle in display memory. It is similar to "circle", except that the center is taken to be the current pen position.

**Control Clipping** enables or disables the clipping function. When clipping is enabled, all drawing primitives will change only that portion of display memory which lies within the rectangular clipping region.

**Control Picking** enables or disables the picking function. When picking is enabled, drawing primitives will not execute any writes to the display memory. The pick detect status bit is set whenever a drawing primitive intersects the picking area.

**Copy Block** moves a block of data within display memory and may optionally perform logical operations on the data. The size of the block will have been determined by a "set block size" instruction. The location of each block, source and destination, is determined by the instruction.

**Copy Block Current** is identical to the "copy block" instruction except that the source operand is the current pen position.

**Define Logical Pel** specifies the logical pel (pen size) used by the drawing primitives. This can be used to draw thick lines.

**Fill Bounded Region** fills an arbitrary polygon with a specific color. The polygon is defined as the group of dots connected to the seed point and completely contained within a boundary of pixels of the edge color. All pixels connected to the seed point will be changed to the current drawing color. The location of the seed is specified in the instruction.

**Fill Bounded Region Current** fills an arbitrary polygon with a specific color. The polygon is defined as the group of dots connected to the seed point and completely contained within a boundary of pixels of the edge color. All pixels connected to the seed point will be changed to the current drawing color. The location of the seed is the current pen position.

**Fill Connected Region** fills an arbitrary polygon with a specific color. The polygon is defined as any group of connected dots of the seed's color. All pixels connected to the seed point having the same color will be changed to the current drawing color. The location of the seed is specified in the instruction.

**Fill Connected Region Current** fills an arbitrary polygon with a specific color. The polygon is defined as any group of connected dots of the seed's color. All pixels connected to the seed point having the same color will be changed to the current drawing color. The location of the seed is the current pen position.

**Filled Rectangle** creates the image of a rectangle and fills it. The parameters specify two opposite corners of the rectangle. The color of the filled rectangle is the current drawing color.

**Filled Rectangle Current** is similar to "fill rectangle" except the current pen position is taken to be the starting corner of the rectangle. The other corner is specified by the instruction.

**Filled Triangle** creates the image of a triangle and fills it. The parameters specify the three vertices of the triangle. The color of the filled triangle is the current drawing color.

**Filled Triangle Current** creates the image of a triangle and fills it. The parameters specify two vertices of the triangle. The current pen position is taken to be the initial vertex of the triangle. The color of the filled triangle is the current drawing color.

**Input Block** transfers a rectangular block of data from the host to display memory. The size of the block will have been determined by a "set block size" instruction. The destination address in the display memory is specified in the instruction. The data to be stored in display memory is written into the data FIFO.

**Input Block Current** transfers a rectangular block of data from the host to display memory. The size of the block will have been determined by a "set block size" instruction. The destination address in the display memory is the current pen position. The data to be stored in display memory is written into the data FIFO.

**Jump** unconditionally changes the location counter when executing instructions from display memory. The jump instruction may also be indexed or conditional.

**Line (Polyline)** draws the image of a line in display memory. The input parameters are the two ends of the line. The image may be drawn using anti-aliasing.
Polylines may be drawn, setting a special bit within the end point parameters. In this way, the QPDM is forced to fetch more parameters to continue line drawing.

**Line Current** is similar to "line", except the current drawing position is taken to be the starting point of the line.

**Move Pen** sets the current pen position.

**No Operation** ensures that no operation is performed.

**Output Block** transfers a rectangular block of data from display memory to the host. The size of the block will have been determined by a "set block size" instruction. The source address in the display memory is specified in the instruction. The host is expected to remove the data from the data FIFO.

**Output Block Current** transfers a rectangular block of data from display memory to the host. The size of the block will have been determined by a "set block size" instruction. The source address in the display memory is the current pen position. The host is expected to remove the data from the data FIFO.

**Point (Polypoint)** draws the image of the current logical pel at the location specified in the instruction. Polypoints may be drawn, setting a special bit within the point parameters. In this way, the QPDM is forced to fetch more parameters to continue point drawing.

**Point Current** draws the image of the current logical pel at the current pen position.

**Pop Current Pen Position** causes the internal current pen position to be popped from the display memory stack.

**Push Current Pen Position** causes the internal current pen position to be pushed on the display memory stack.

**Return** exits from a subroutine or from program mode when executing instructions from display memory.

**Set Activity Bits** indicates which of the four display memory planes, controlled by the SAB 95C60, are to be written into.

**Set Anti-Aliasing Distance** programs the anti-aliasing distance deviation from the ideal line.

**Set Block Size** specifies the number of pixels moved in a block move operation. This is used for input block, output block, copy block, and transform block.

**Set Character Font Base** specifies the character font address in the display memory. The character font contains the patterns of letters and numbers used in the "string" instruction.

**Set Clipping Boundary** specifies where the clipping region is in display memory. When clipping is enabled, all drawing primitives will change only that portion of display memory which lies within the rectangular clipping region. The parameters are the addresses of two opposite corners of the clipping rectangle.

**Set Clipping Boundary Current** specifies where the clipping region is in display memory. When clipping is enabled, all drawing primitives will change only that portion of display memory which lies within the rectangular clipping region. The current pen position is taken to be the start corner of the clipping rectangle.

**Set Color Bits** defines the current drawing color.

**Set Search Color** specifies the edge color used in "fill" instructions.

**Set Line Style** specifies the line style. This defines the dash length, the interspace length, and the dot length.

**Set Line Style Phase** indicates where the line style begins within the line.

**Set Listen Bits** indicates which planes take part in polygon and color change operations. If reset, the corresponding plane doesn't participate in the color matching.

**Set Picking Region** specifies the rectangular area to be picked. The parameters are two opposite corners of the picking rectangle. When picking is enabled, drawing primitives that intersect the picking region will cause the "pick detect" bit in the status register to be set and no writes will be executed to display memory.

**Set Picking Region Current** specifies the rectangular area to be picked. The parameters are two opposite corners of the picking rectangle. The current pen position is taken to be the start corner of the picking region. When picking is enabled, drawing primitives that intersect the picking region will cause the "pick detect" bit in the status register to be set and no writes will be executed to display memory.

**Set QPDM Position** specifies the logical addresses for the display memory planes in systems with multiple QPDMs.

**Set Scale Factor** loads the scale factor register with values used to multiply the operands of instructions which address the bit map.

**Set Stack Boundaries** specifies to the SAB 95C60 which area of display memory has been set aside for the stack. Stack overflow is detected and signaled to the host with an interrupt.

**Signal** is used to indicate to the host when a particular point in the instruction stream has been reached, to delimit objects during picking, or to pause operation waiting for a signal from the host.

**Store Immediate** deposits a specified number of 16-bit words in the display memory. The pointer may be used by one of the indirect addressing modes.

**Store Current Pen Position** stores the internal current pen position at the location specified in the instruction.

**String** is used to create the image of a string of text in the display memory. The parameters are the address at which the string should begin, followed by a variable length list of 16-bit pointers. Each pointer is used to look up a pattern in the character font table.

**String Current** is similar to "string" except the address at which the string should begin is set to the current pen position.

**Transform Block** allows a block of data to be taken from display memory, operated on and written to a different area of display memory. The operations which may be performed are rotate (90 degree increments), zoom (by pixel replication), and mirror. The zoom in X-direction and zoom in Y-direction are independently specified. The size of the source block (prior to rotation and zooming) will have been specified as "set block size" instruction.

**Transform Block Current** is similar to "transform block" except the source operand is at the current pen position.

## Absolute Maximum Ratings

| | |
|---|---|
| Storage temperature | −65°C to + 125°C |
| Ambient operating temperature | −55°C to + 125°C |
| Maximum $V_{CC}$ relative to $V_{SS}$ | −0.3V to + 7.0V |
| DC voltage applied to any input with respect to $V_{SS}$ | −1.0V to $V_{CC}$ + 0.3V |
| DC voltage applied to any output pin with respect to $V_{SS}$ | −0.5V to $V_{CC}$ + 0.3V |

Note:
Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. Exposure to absolute maximum ratings for extended periods may affect device reliability.

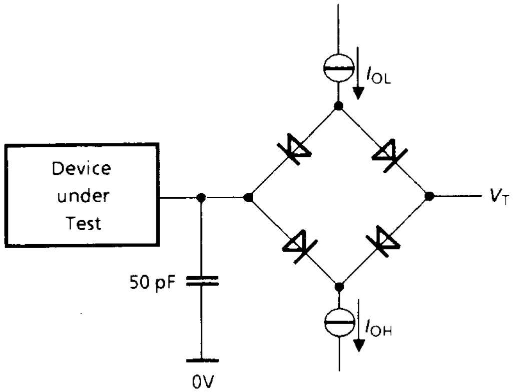
## DC Characteristics

$T_A$ = 0 to 70°C; $V_{CC}$ = 5 V ± 5%

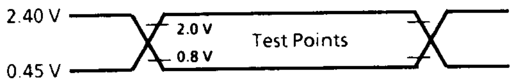
| Symbol | Parameter | Limit values | | Unit | Test condition |
|---|---|---|---|---|---|
| | | min. | max. | | |
| $V_{IL}$ | Input low voltage | −0.3 | +0.8 | V | – |
| $V_{IH}$ | Input high voltage | +2.0 | $V_{CC}$+0.3 | V | – |
| $V_{OL}$ | Output low voltage | – | 0.4 | V | $I_{OL}$ = 2.0 mA |
| | ($\overline{CAS}$ and $\overline{XF/G}$) | – | 0.4 | V | $I_{OL}$ = 4.0 mA |
| $V_{OH}$ | Output high voltage | 2.4 | – | V | $I_{OH}$ = 250 µA |
| $I_{OL}$ | Output leakage current | – | ±10 | µA | $0.4 < V_{OUT} < V_{CC}$ |
| $I_I$ | Input current | – | ±10 | µA | $0.4 < V_{IN} < V_{CC}$ |
| $C_{IN}$ | Input capacitance | – | 20 | pF | – |
| $C_{IO}$ | I/O capacitance | – | 20 | pF | – |
| $C_{OUT}$ | Output capacitance | – | 15 | pF | |

## AC Testing

**Standard Test Load**




**I/O Waveforms**



## AC Characteristics

$T_A$ = 0 to 70 °C; $V_{CC}$ = 5 V ± 5%

**System Bus Timing**

| Symbol | Parameter | Limit values | | | | | | Unit |
|---|---|---|---|---|---|---|---|---|
| | | 20 MHz | | 16 MHz | | 12 MHz | | |
| | | min. | max. | min. | max. | min. | max. | |
| $t_1$ | Requires that the addresses be valid a minimum of {} ns before $\overline{CS}$ begins to fall. | 0 | – | 0 | – | 0 | – | ns |
| $t_2$ | Requires that both $\overline{CS}$ and $\overline{ACKD}$ be not active a minimum of {} ns before either can go active. 1) | 65 | – | 95 | – | 125 | – | ns |
| $t_3$ | Guarantees that DREQ will not become active a maximum of {} ns after $\overline{RD}$ or $\overline{WR}$ becomes active. | – | 50 | – | 60 | – | 70 | ns |
| $t_4$ | Guarantees that $\overline{EDE}$ will become active a maximum of {} ns after $\overline{CS}$ becomes active. Also guarantees that $\overline{EDE}$ will become active a maximum of {} ns after $\overline{ACKD}$ becomes active. 1) | – | 50 | – | 60 | – | 70 | ns |
| $t_5$ | Guarantees that $\overline{EDE}$ will remain active a minimum of {} ns after $\overline{CS}$ has become inactive. Also guarantees that $\overline{EDE}$ will remain active a minimum of {} ns after $\overline{ACKD}$ has become inactive. 1) 3) | 10 | – | 10 | – | 10 | – | ns |
| $t_6$ | Guarantees that $\overline{EDE}$ will have gone inactive no more than {} ns after $\overline{CS}$ has become inactive. Also guarantees that $\overline{EDE}$ will have gone inactive no more than {} ns after $\overline{ACKD}$ has become inactive. | – | 65 | – | 70 | – | 75 | ns |
| $t_7$ | Requires that $\overline{CS}$ be valid a minimum of {} ns before $\overline{RD}$ can begin to go active. Also requires that $\overline{ACKD}$ be valid a minimum of {} ns before $\overline{WR}$ can begin to go active. 1) 2) | 0 | – | 0 | – | 0 | – | ns |
| $t_8$ | Requires that the address remains valid a minimum of {} ns after $\overline{RD}$ has gone inactive. | 10 | – | 10 | – | 10 | – | ns |
| $t_9$ | Requires that $\overline{CS}$ remains active a minimum of {} ns after $\overline{RD}$ has gone inactive. | 0 | – | 0 | – | 0 | – | ns |

For notes refer to page 374.

**System Bus Timing** (cont'd)

| Symbol | Parameter | Limit values | | | | | | Unit |
|---|---|---|---|---|---|---|---|---|
| | | 20 MHz | | 16 MHz | | 12 MHz | | |
| | | min. | max. | min. | max. | min. | max. | |
| $t_{10}$ | Guarantees that the read data will be valid within {} ns of $\overline{RD}$ becoming active. Also guarantees that the read data will be valid within {} ns of $\overline{WR}$ becoming active in a fly-by read cycle. 2) | – | 110 | – | 110 | – | 120 | ns |
| $t_{11}$ | Guarantees that the read data will remain valid a minimum of {} ns after $\overline{RD}$ has gone inactive. Also guarantees that the read data will remain valid a minimum of {} ns after $\overline{WR}$ has gone inactive in a fly-by read cycle. 2) 3) | 10 | – | 10 | – | 10 | – | ns |
| $t_{12}$ | Guarantees that the read buffers will begin to enter high impedance within {} ns of $\overline{RD}$ having gone inactive. Also guarantees that the read buffers will begin to enter high impedance within {} ns of $\overline{WR}$ having gone inactive in a fly-by read cycle. 2) 3) | – | 35 | – | 40 | – | 45 | ns |
| $t_{14}$ | Requires that the address remains valid a minimum of {} ns after $\overline{WR}$ has gone inactive. | 10 | – | 20 | – | 20 | – | ns |
| $t_{15}$ | Requires that $\overline{CS}$ remains active a minimum of {} ns after $\overline{WR}$ has gone inactive. | 10 | – | 20 | – | 20 | – | ns |
| $t_{16}$ | Requires that $\overline{CS}$ is active for a minimum of {} ns before $\overline{WR}$ can begin to go active. Also requires that $\overline{ACKD}$ is active for a minimum of {} ns before $\overline{RD}$ can begin to go active in a fly-by write cycle. 1) 2) | 0 | – | 0 | – | 0 | – | ns |
| $t_{17}$ | Requires that $\overline{WR}$ is active for a minimum of {} ns. Also requires that $\overline{RD}$ is active for a minimum of {} ns in the case of a fly-by write cycle. Also requires that $\overline{ACKD}$ remains active for a minimum of {} ns after $\overline{RD}$ has gone active in a fly-by write cycle. | 70 | – | 90 | – | 110 | – | ns |

For notes refer to page 374.

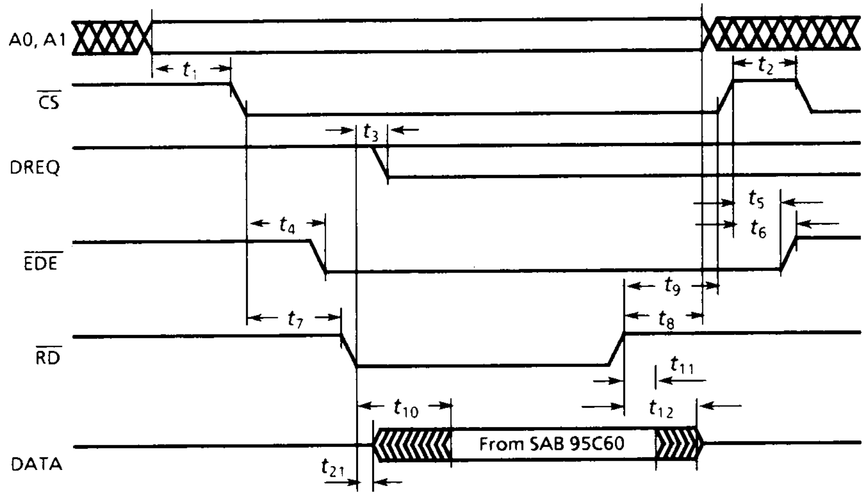
**System Bus Timing** (cont'd)

| Symbol | Parameter | | Limit values | | | | | | Unit |
|---|---|---|---|---|---|---|---|---|---|
| | | | 20 MHz | | 16 MHz | | 12 MHz | | |
| | | | min. | max. | min. | max. | min. | max. | |
| $t_{18}$ | Requires that the write data is active for a minimum of { } ns before $\overline{WR}$ begins to go inactive. Also requires that the write data is active for a minimum of { } ns before $\overline{RD}$ or $\overline{ACKD}$ (whichever is first) begins to go inactive in a fly-by write cycle. | | 50 | – | 75 | – | 100 | – | ns |
| $t_{19}$ | Requires that the write data is kept valid for a minimum of { } ns after $\overline{WR}$ has gone inactive. Also requires that the write data is kept valid for a minimum of { } ns after $\overline{RD}$ or $\overline{ACKD}$ (whichever is first) has gone inactive in a fly-by write cycle. | Word mode | 0 | – | 0 | – | 0 | – | ns |
| | | Byte mode | 15 | – | 25 | – | 25 | – | ns |
| $t_{20}$ | Guarantees that the INT line will become inactive no more than { } ns after $\overline{WR}$ has gone inactive. | | – | 120 | – | 150 | – | 180 | ns |
| $t_{21}$ | Guarantees that the data buffers will not become active before $\overline{RD}$ goes active. Also guarantees that the data buffers will not become active before $\overline{WR}$ goes active in a fly-by read cycle. | | 0 | – | 0 | – | 0 | – | ns |
| $t_{22}$ | Requires that $\overline{ACKD}$ is active for a minimum of { } ns after $\overline{RD}$ or $\overline{WR}$ has gone inactive | | 0 | – | 0 | – | 0 | – | ns |

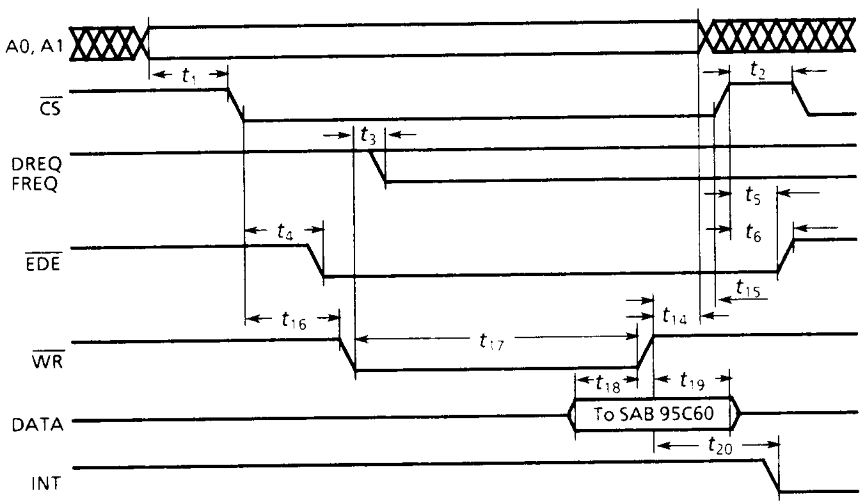
For notes refer to page 374.

## System Bus Timing Waveforms

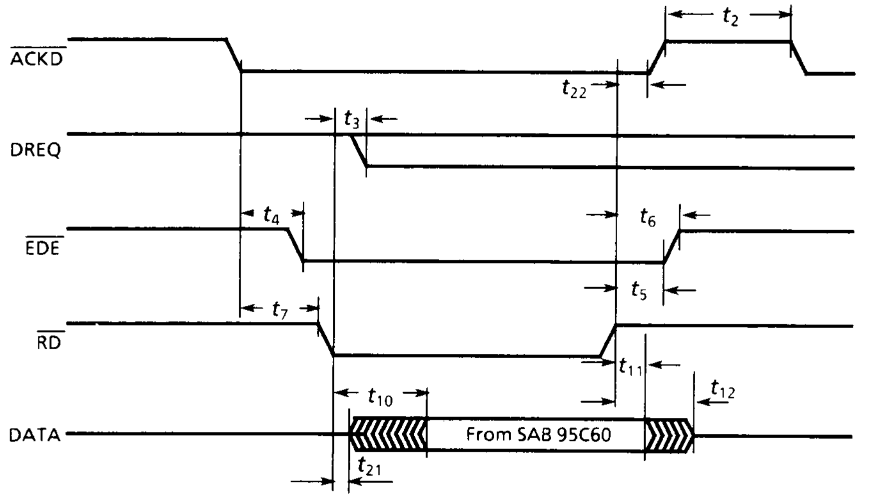
**Processor/DMA Flow-Through Read Cycle**



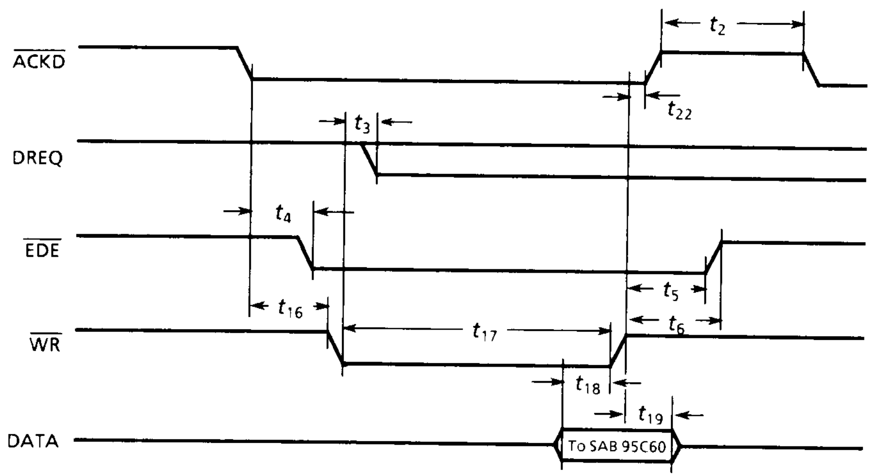

**Processor/DMA Flow-Through Write Cycle**



**DMA Fly-by Read Cycle**




**DMA Fly-by Write Cycle**



## AC Characteristics (Notes see page 374)

$T_A$ = 0 to 70°C; $V_{CC}$ = 5V ± 5%

**Display Memory Interface Timing**

| Symbol | Parameter | Limit values | | | | | | Unit |
|---|---|---|---|---|---|---|---|---|
| | | 20 MHz | | 16 MHz | | 12 MHz | | |
| | | min. | max. | min. | max. | min. | max. | |
| $t_{30}$ | Guarantees the row address will be stable (valid) for a minimum of {} ns before $\overline{RAS}$ begins to go active. 4) | 15 | – | 16 | – | 21.5 | – | ns |
| $t_{31}$ | Guarantees the row address will remain valid for a minimum of {} ns after $\overline{RAS}$ has gone active. 4) | 35 | – | 45 | – | 63 | – | ns |
| $t_{32}$ | Guarantees that $\overline{XF/G}$ will not go active until a minimum of {} ns after $\overline{RAS}$ has gone active. 4) | 160 | – | 202 | – | 270.5 | – | ns |
| $t_{33}$ | Guarantees that $\overline{RAS}$ will be active for a minimum of {} ns. 4) | 180 | – | 225 | – | 307 | – | ns |
| $t_{34}$ | Guarantees that $\overline{RAS}$ will remain active for a minimum of {} ns after $\overline{XF/G}$ has gone inactive. 4) | 14 | – | 15 | – | 21.5 | – | ns |
| $t_{35}$ | Guarantees that $\overline{RAS}$ will remain inactive for a minimum of {} ns. 4) | 95 | – | 109 | – | 146 | – | ns |
| $t_{36}$ | Guarantees that $\overline{CAS}$ will not become active until a minimum of {} ns after $\overline{RAS}$ has gone active. See parameter $t_{56}$ for write cycles. 4) | 65 | – | 78 | – | 104.5 | – | ns |
| $t_{37}$ | Guarantees that the column address will be valid and stable for a minimum of {} ns before $\overline{CAS}$ will go active. 4) | 13 | – | 15 | – | 21.5 | – | ns |
| $t_{38}$ | Guarantees that the column address will remain valid and stable for a minimum of {} ns after $\overline{CAS}$ has gone active. 4) | 80 | – | 104 | – | 144 | – | ns |
| $t_{39}$ | Guarantees that $\overline{CAS}$ will be active for a minimum of {} ns in case of a read cycle or a transfer cycle. See parameter $t_{57}$ for write cycles. 4) | 100 | – | 130 | – | 180 | – | ns |
| $t_{40}$ | Guarantees that $\overline{CAS}$ will remain inactive for a minimum of {} ns. This is important when a refresh cycle follows any cycle. 4) | 40 | – | 47 | – | 63 | – | ns |

**Display Memory Interface Timing** (cont'd)

| Symbol | Parameter | Limit values | | | | | | Unit |
|---|---|---|---|---|---|---|---|---|
| | | 20 MHz | | 16 MHz | | 12 MHz | | |
| | | min. | max. | min. | max. | min. | max. | |
| $t_{41}$ | Guarantees that $\overline{\text{XF/G}}$ will not have gone inactive until a minimum of {} ns after $\overline{\text{CAS}}$ has gone active 4) | 80 | – | 100 | – | 140 | – | ns |
| $t_{42}$ | Guarantees that $\overline{\text{XF/G}}$ will not be active until a minimum of {} ns after $\overline{\text{RAS}}$ has gone active. 4) | 39 | – | 46 | – | 60 | – | ns |
| $t_{43}$ | Guarantees that $\overline{\text{CAS}}$ will remain active until a minimum of {} ns after $\overline{\text{XF/G}}$ has gone inactive. 4) | 13 | – | 16 | – | 21.5 | – | ns |
| $t_{44}$ | Guarantees that $\overline{\text{XF/G}}$ will be active for a minimum of {} ns for a read cycle. 4) | 110 | – | 125 | – | 180 | – | ns |
| $t_{45}$ | Requires that the read data is valid for a minimum of {} ns before $\overline{\text{XF/G}}$ begins to go inactive. | 20 | – | 30 | – | 45 | – | ns |
| $t_{46}$ | Requires that the read data remains valid for a minimum of {} ns after $\overline{\text{XF/G}}$ has gone inactive. | 0 | – | 0 | – | 0 | – | ns |
| $t_{47}$ | Guarantees that $\overline{\text{RAS}}$ will not begin to go active until a minimum of {} ns after $\overline{\text{CAS}}$ has become active in a refresh cycle. 4) | 37 | – | 45 | – | 63 | – | ns |
| $t_{48}$ | Guarantees that $\overline{\text{CAS}}$ will not begin to go inactive until a minimum of {} ns after $\overline{\text{RAS}}$ has gone active in a refresh cycle. 4) | 185 | – | 230 | – | 307 | – | ns |
| $t_{49}$ | Guarantees that $\overline{\text{RAS}}$ will not begin to go active until a minimum of {} ns after $\overline{\text{XF/G}}$ is active in a transfer cycle. 4) | 12 | – | 13 | – | 21.5 | – | ns |
| $t_{50}$ | Guarantees that the start offset on CDAT will be valid and stable for a minimum of {} ns before $\overline{\text{RAS}}$ begins to go active in a transfer cycle. 4) | 10 | – | 11 | – | 21.5 | – | ns |
| $t_{51}$ | Guarantees that the start offset on CDAT will remain valid and stable for a minimum of {} ns after $\overline{\text{RAS}}$ has gone active in a transfer cycle. 4) | 65 | – | 78 | – | 104.5 | – | ns |

For notes refer to page 374.

**Display Memory Interface Timing** (cont'd)

| Symbol | Parameter | Limit values | | | | | | Unit |
|---|---|---|---|---|---|---|---|---|
| | | 20 MHz | | 16 MHz | | 12 MHz | | |
| | | min. | max. | min. | max. | min. | max. | |
| $t_{52}$ | Guarantees that VSTB will be high for a minimum of { } ns before $\overline{RAS}$ begins to go active in a transfer cycle. 4) | 90 | – | 109 | – | 146 | – | ns |
| $t_{53}$ | Guarantees that VSTB will remain high for a minimum of { } ns after $\overline{RAS}$ has gone inactive in a transfer cycle. 4) | 40 | – | 47 | – | 63 | – | ns |
| $t_{54}$ | Guarantees that DSTB will be low for a minimum of { } ns before $\overline{RAS}$ begins to go active in a transfer cycle. 4) | 90 | – | 109 | – | 146 | – | ns |
| $t_{55}$ | Guarantees that DSTB will remain low for a minimum of { } ns after $\overline{RAS}$ has gone inactive in a transfer cycle. | 40 | – | 47 | – | 63 | – | ns |
| $t_{56}$ | Guarantees that $\overline{CAS}$ will not become active until a minimum of { } ns after $\overline{RAS}$ has gone active. This is for a write cycle. See parameter $t_{36}$ for read and transfer cycles. 4) | 90 | – | 109 | – | 140 | – | ns |
| $t_{57}$ | Guarantees that $\overline{CAS}$ will be active for a minimum of { } ns for a write cycle. See parameter $t_{39}$ for read and transfer cycles. 4) | 80 | – | 100 | – | 146 | – | ns |
| $t_{58}$ | Guarantees that $\overline{WE0\text{-}WE3}$ will be active for a minimum of { } ns in the case of a masked write. 4) | 180 | – | 225 | – | 312 | – | ns |
| $t_{59}$ | Guarantees that $\overline{WE0\text{-}WE3}$ will be active for a minimum of { } ns before $\overline{RAS}$ begins to fall in the case of a masked write. 4) 5) | 11 | – | 14 | – | 21.5 | – | ns |
| $t_{60}$ | Guarantees that $\overline{WE0\text{-}WE3}$ will be active for a minimum of { } ns before $\overline{CAS}$ begins to fall in the case of an unmasked write. 4) | 13 | – | 14 | – | 21.5 | – | ns |
| $t_{61}$ | Guarantees that $\overline{WE0\text{-}WE3}$ will be active for a minimum of { } ns in the case of an unmasked write. 4) | 78 | – | 100 | – | 140 | – | ns |

For notes refer to page 34



For notes refer to page 374.

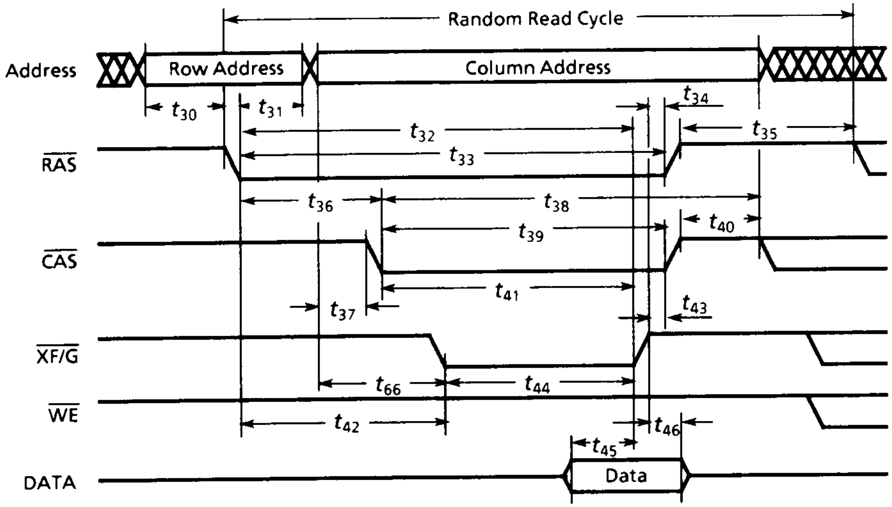
**Display Memory Interface Timing** (cont'd)

| Symbol | Parameter | Limit values | | | | | | Unit |
|---|---|---|---|---|---|---|---|---|
| | | 20 MHz | | 16 MHz | | 12 MHz | | |
| | | min. | max. | min. | max. | min. | max. | |
| $t_{62}$ | Guarantees that the write mask will be valid and stable on the DM pins {} ns before $\overline{RAS}$ begins to go active. | 2 | – | 7 | – | 15 | – | ns |
| $t_{63}$ | Guarantees that the write mask will remain active {} ns after $\overline{RAS}$ has gone active. 4) | 60 | – | 73 | – | 104.5 | – | ns |
| $t_{64}$ | Guarantees that the write data will be valid and stable on the DM pins {} ns before $\overline{CAS}$ begins to go active. | 2 | – | 7 | – | 15 | – | ns |
| $t_{65}$ | Guarantees that the write data will remain active {} ns after $\overline{CAS}$ has gone active. 4) | 60 | – | 73 | – | 104.5 | – | ns |
| $t_{66}$ | Guarantees that the column address will be valid {} ns before $\overline{XF/G}$ goes active. | –8 | – | –12 | – | –16 | – | ns |

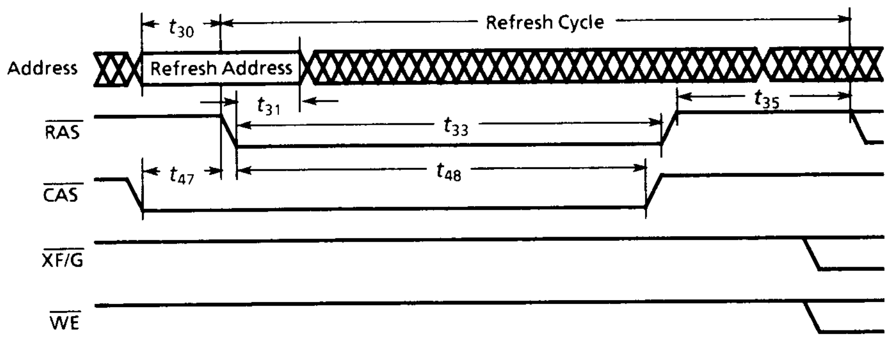
For notes refer to page 374.

## Display Memory Interface Timing Waveforms

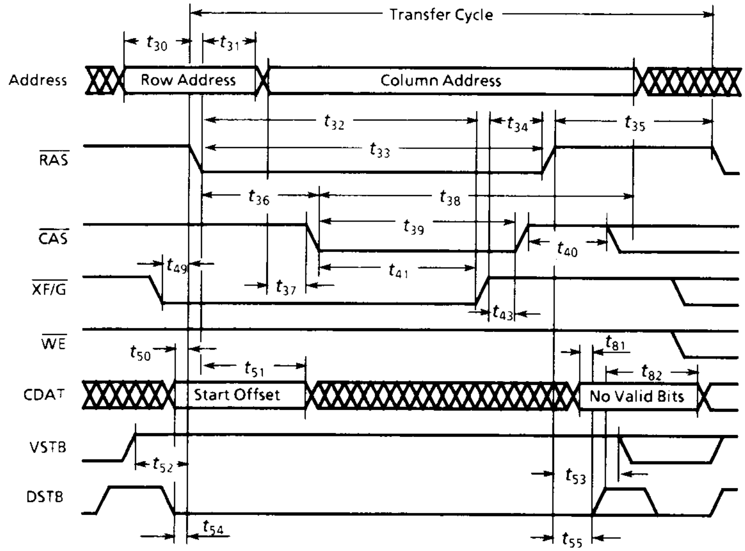
**Display Memory Read Cycle**



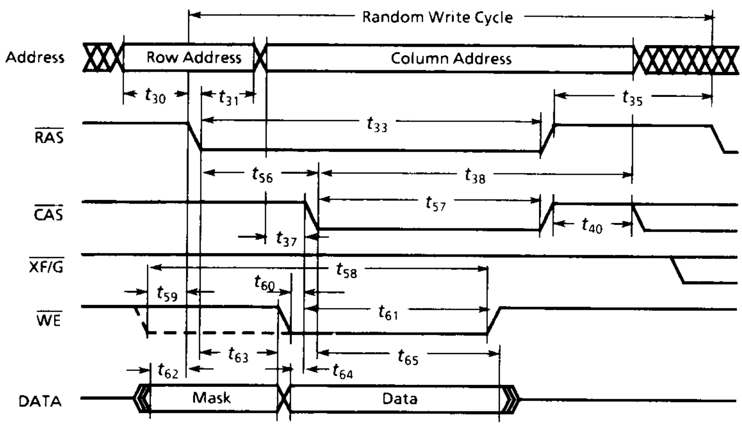

**Display Memory Dynamic RAM Refresh Cycle**



**Display Memory Transfer Cycle**




**Display Memory Write Cycle**



## AC Characteristics

$T_A$ = 0 to 70°C; $V_{CC}$ = 5 V ± 5%

**VDAF Interface Timing**

| Symbol | Parameter | Limit values | | | | | | Unit |
|---|---|---|---|---|---|---|---|---|
| | | 20 MHz | | 16 MHz | | 12 MHz | | |
| | | min. | max. | min. | max. | min. | max. | |
| $t_{80}$ | Guarantees that VSTB will change to the new state within 0-{} ns following the positive edge of DSTB. | 0 | 10 | 0 | 10 | 0 | 10 | ns |
| $t_{81}$ | Guarantees that the value on CDAT will be valid and stable for a minimum of {} ns before DSTB begins to rise. | 0 | – | 0 | – | 5 | – | ns |
| $t_{82}$ | Guarantees that the value on CDAT will remain valid for a minimum of {} ns after the rising edge of DSTB. | 20 | – | 30 | – | 40 | – | ns |
| $t_{83}$ | Guarantees that the positive edges on DSTB will occur with the same period as SYSCLK. 4) | 50 | – | 62 | – | 83 | – | ns |
| $t_{84}$ | Guarantees that $\overline{FULL}$ will be recognized at the current SYSCLK cycle if it is valid for a minimum of {} ns before the edge. This is not an operating parameter; if this setup time is not met, the part will not go metastable. | – | 25 | – | 50 | – | 75 | ns |
| $t_{85}$ | Requires that $\overline{FULL}$ remains active for at least one SYSCLK period. 4) | 50 | – | 62 | – | 83 | – | ns |
| $t_{86}$ | Guarantees that a positive edge will not occur on DSTB until a minimum of {} ns following $\overline{FULL}$ going inactive. 4) | 150 | – | 186 | – | 249 | – | ns |

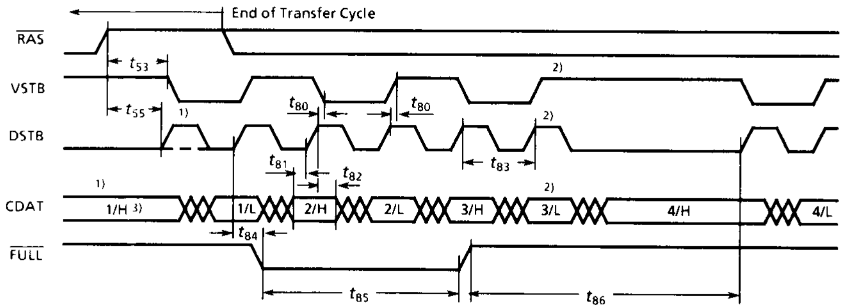
**Memory Bus Arbitration Timing**

| Symbol | Parameter | Limit values | | | | | | Unit |
|---|---|---|---|---|---|---|---|---|
| | | 20 MHz | | 16 MHz | | 12 MHz | | |
| | | min. | max. | min. | max. | min. | max. | |
| $t_{90}$ | Guarantees that the SAB 95C60 will set $\overline{MEMAVL}$ inactive within {} ns of $\overline{MEMREQ}$ going inactive. 4) | – | 160 | – | 190 | – | 245.5 | ns |
| $t_{91}$ | Guarantees that the SAB 95C60 will not begin a memory cycle for a minimum of {} ns following the rising edge of $\overline{MEMAVL}$. 4) 6) | 450 | – | 558 | – | 747 | – | ns |

For notes refer to page 374.

## VDAF Interface Timing Waveforms

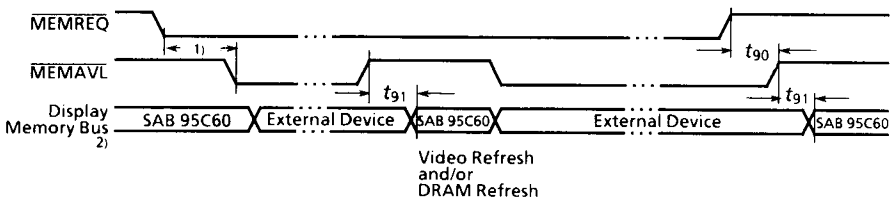
**VDAF Interface**




1) DSTB and CDAT for (1/H) only occurs on even boundaries.
2) One less byte may be strobed after FULL active.
3) CDAT 1/H = CDAT for 1st word high byte.

## Memory Bus Arbitration Timing Waveforms

**Display Memory Arbitration**




1) MEMREQ to MEMAVL delay = Minimum 6 CLKs for arbitration cycle
= Maximum 12 CLKs if DRAM refresh lower priority
= Maximum 18 CLKs DRAM refresh higher priority
} Not an operating parameter provided for reference only

2) Display memory bus = ADDR, RAS, CAS, XF/G, WE, DATA
$t_{91}$ is referred to RAS falling edge, which defines the start of the QPDM cycle

## AC Characteristics

$T_A$ = 0 to 70°C; $V_{CC}$ = 5V ± 5%

**Miscellaneous Timings**

| Symbol | Parameter | Limit values | | | | | | Unit |
|---|---|---|---|---|---|---|---|---|
| | | 20 MHz | | 16 MHz | | 12 MHz | | |
| | | min. | max. | min. | max. | min. | max. | |
| $t_{100}$ | Guarantees that transitions on $\overline{\text{HRESET}}$ and $\overline{\text{VRESET}}$ will occur for a maximum of {} ns following the high-to-low transition of VIDCLK. This applies only if $\overline{\text{HRESET}}$ and $\overline{\text{VRESET}}$ are programmed as output(s). | – | 25 | – | 31 | – | 41 | ns |
| $t_{101}$ | Requires that $\overline{\text{HRESET}}$ and $\overline{\text{VRESET}}$ are valid for a minimum of {} ns prior to the rising edge of CIDCLK. This is not an operational parameter. If this setup time is not met, the SAB 95C60 will not become metastable. | 0 | – | 0 | – | 0 | – | ns |
| $t_{102}$ | Requires that $\overline{\text{HRESET}}$ and $\overline{\text{VRESET}}$ remain valid for a minimum of {} ns after VIDCLK has fallen low. This is not an operational parameter. If this hold time is not met, the SAB 95C60 will not become metastable. | 15 | – | 20 | – | 25 | – | ns |
| $t_{103}$ | Guarantees that transitions on BLANK, HSYNC and VSYNC will occur within a maximum of {} ns after the rising edge of VIDCLK. | – | 30 | – | 40 | – | 50 | ns |
| $t_{104}$ | Requires that VODD/$\overline{\text{EVEN}}$ is valid for a minimum of {} ns prior to the rising edge of VIDCLK. | 15 | – | 20 | – | 25 | – | ns |
| $t_{105}$ | Requires that VODD/$\overline{\text{EVEN}}$ remain valid for a minimum of {} ns after VIDCLK has risen high. | 15 | – | 20 | – | 25 | – | ns |
| $t_{106}$ | Requires that the period of SYSCLK lies between {} and {} ns. | 50 | 500 | 62 | 500 | 83 | 500 | ns |
| $t_{107}$ | Requires that SYSCLK transition times lie within a maximum of {} ns. | – | 5 | – | 5 | – | 5 | ns |
| $t_{108}$ | Requires that the SYSCLK low time is a minimum of {} ns. | 18 | – | 23 | – | 32 | – | ns |
| $t_{109}$ | Requires that the SYSCLK high time is a minimum of {} ns. | 18 | – | 23 | – | 32 | – | ns |



For notes refer to page 374.

**Miscellaneous Timings** (cont'd)

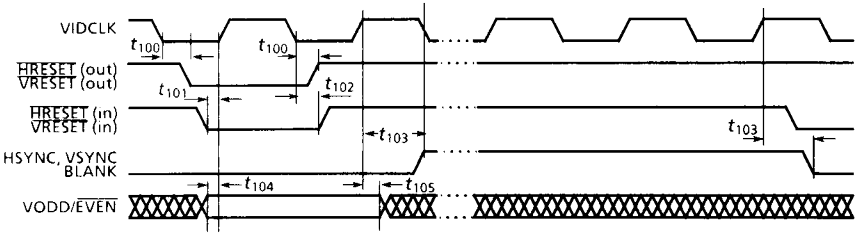
| Symbol | Parameter | Limit values | | | | | | Unit |
|---|---|---|---|---|---|---|---|---|
| | | 20 MHz | | 16 MHz | | 12 MHz | | |
| | | min. | max. | min. | max. | min. | max. | |
| $t_{110}$ | Requires that the period of VIDCLK is minimum of {} ns and a maximum of {} µs. | 66 | 4 µs | 72 | 4 µs | 83 | 4 µs | ns |
| $t_{111}$ | Requires that the VIDCLK transition times are a maximum of {} ns. | – | 5 | – | 5 | – | 5 | ns |
| $t_{112}$ | Requires that the VIDCLK low time is a minimum of {} ns. | 25 | – | 27 | – | 32 | – | ns |
| $t_{113}$ | Requires that the VIDCLK high time is a minimum of {} ns. | 25 | – | 27 | – | 32 | – | ns |
| $t_{114}$ | Requires that $\overline{\text{RESET}}$ remains active for a minimum of {} ns. 4) | 200 | – | 248 | – | 332 | – | ns |
| $t_{115}$ | Requires that the external delay from TSYNOUT to TSYNIN and the external delay from MATOUT to MATIN is a maximum of {} ns. | – | 20 | – | 25 | – | 30 | ns |

**Notes:**

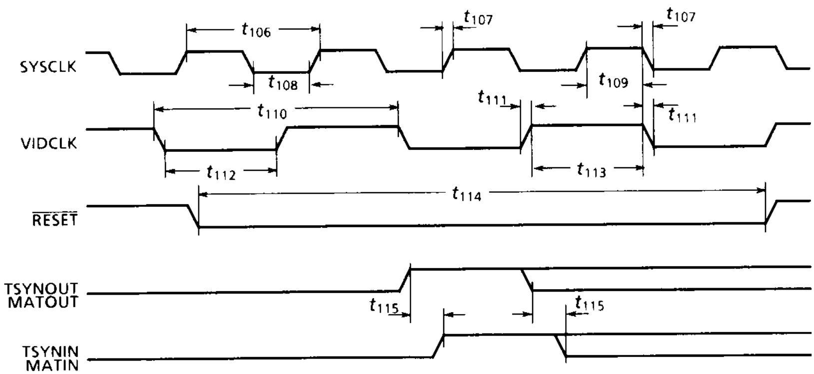
1) Timings are referred to $\overline{\text{CS}}$ or $\overline{\text{ACKD}}$.
2) $\overline{\text{RD}}$ and $\overline{\text{WR}}$ reverse operations in fly-by DMA cycles.
3) Timings are referred to $\overline{\text{RD}}$/$\overline{\text{WR}}$ or $\overline{\text{ACKD}}$ rising edge, whichever occurs first.
4) Require a SYSCLK symmetry of no worse than 45/55%.
5) This timing applies to masked writes.
6) All display memory cycles are exactly six SYSCLK cycles.

## Miscellaneous Waveforms

**Video Control**




**Clocks, Reset and SAB 95C60 Synchronization**



## Ordering Information

| Type | Ordering code | Description |
|---|---|---|
| SAB 95C60-12-A | Q67120-P267 | Quad pixel dataflow manager (QPDM) |
| SAB 95C60-16-A | Q67120-P268 | Quad pixel dataflow manager (QPDM) |
| SAB 95C60-20-A | Q67120-P269 | Quad pixel dataflow manger (QPDM) |