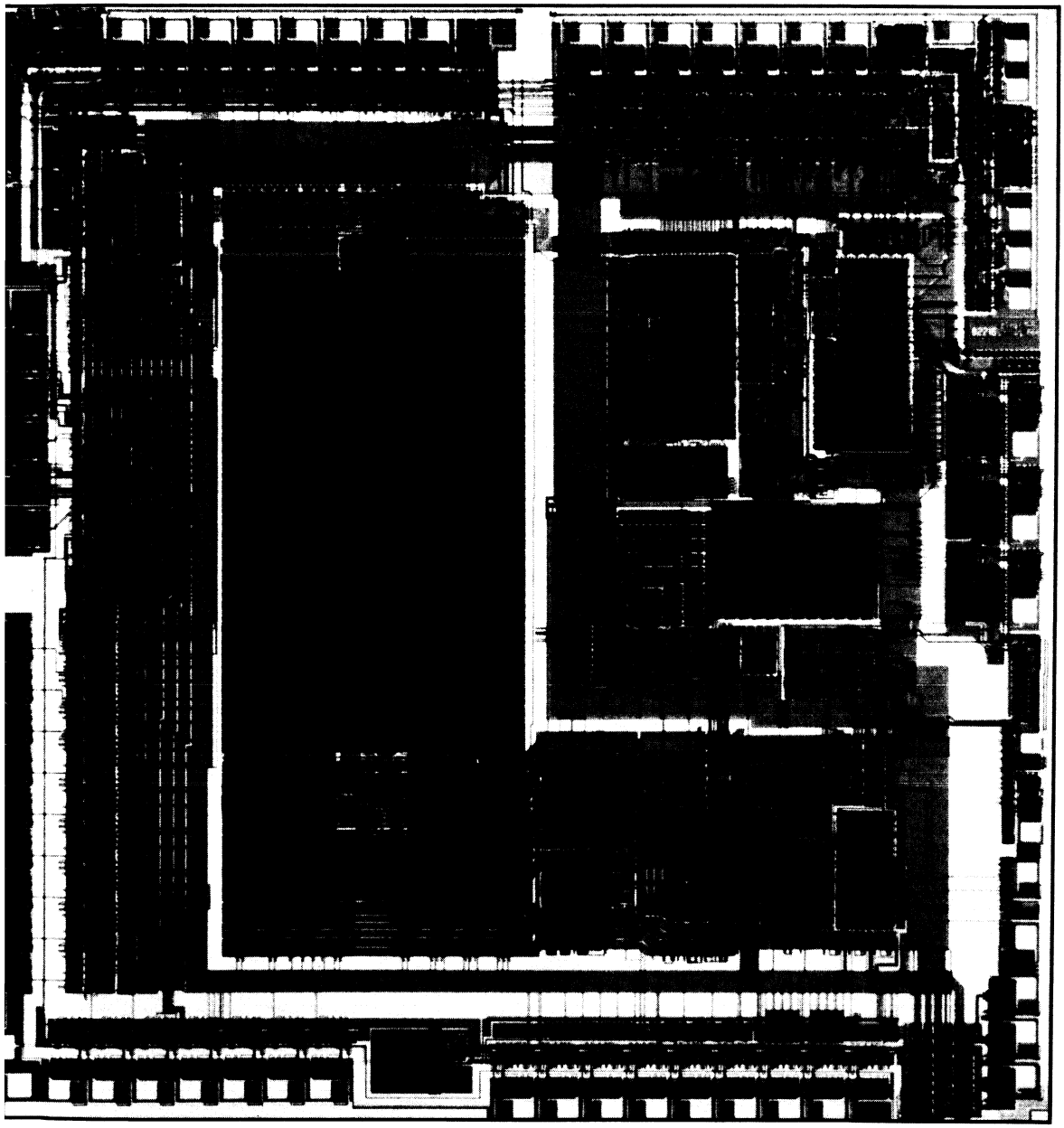
82716
User's Manual



Order Number: 231678-002

intel

82716
User's Manual
VSDD

August 1986
Order Number: 231678-002

Intel Corporation makes no warranty for the use of its products and assumes no responsibility for any errors which may appear in this document nor does it make a commitment to update the information contained herein.

Intel retains the right to make changes to these specifications at any time, without notice.

Contact your local sales office to obtain the latest specifications before placing your order.

The following are trademarks of Intel Corporation and may only be used to identify Intel Products:

Above, BITBUS, COMMputer, CREDIT, Data Pipeline, FASTPATH, Genius, i, $\hat{\text{i}}$, ICE, iCEL, iCS, iDBP, iDIS, I²ICE, iLBX, i_m, iMDDX, iMMX, Inboard, Insite, Intel, int_el, int_elBOS, Intelevision, $int_eligent$ Identifier, $int_eligent$ Programming, Intellec, Intellink, iOSP, iPDS, iPSC, iRMX, iSBC, iSBX, iSDM, iSXM, KEPROM, Library Manager, MAP-NET, MCS, Megachassis, MICROMAINFRAME, MULTIBUS, MULTICHANNEL, MULTIMODULE, MultiSERVER, ONCE, OpenNET, OTP, PC-BUBBLE, Plug-A-Bubble, PROMPT, Promware, QUEST, QueX, Quick-Pulse Programming, Ripplemode, RMX/80, RUPI, Seamless, SLD, UPI, and VLSiCEL, and the combination of ICE, iCS, iRMX, iSBC, iSBX, MCS, or UPI and a numerical suffix, 4-SITE.

MDS is an ordering code only and is not used as a product name or trademark. MDS® is a registered trademark of Mohawk Data Sciences Corporation.

*MULTIBUS is a patented Intel bus.

Additional copies of this manual or other Intel literature may be obtained from:

Intel Corporation
Literature Distribution
Mail Stop SC6-59
3065 Bowers Avenue
Santa Clara, CA 95051

CG-10/86

82716 USER'S MANUAL

CONTENTS

PAGE

82716: VIDEO STORAGE & DISPLAY DEVICE

PART I: AN OVERVIEW

The 82716 is a video storage and display device (VSDD) which performs the functions of a dynamic RAM controller and a CRT or RGB monitor display controller.

A block diagram of a video display system which uses the device is shown in Figure 1. As shown, the VSDD interfaces with three blocks in the system: the CPU, the display memory (DRAM), and a CRT monitor. The CPU generates the display data and stores it in the dynamic RAM through the 82716. The 82716 performs the functions of a dynamic RAM controller, providing a window through which the CPU can access the DRAM, while generating the necessary signals to address, refresh, and directly drive the DRAM.

The 82716 also controls the CRT display. It synchronizes the raster and generates analog RGB signals for the individual scan lines. In this function it accesses the DRAM independently of the CPU.

Scan lines are generated as shown in Figure 2. Pixel data for one scan line is written into one of two line buffers. Pixel data is stored in the scan line buffers at either 4 bits/pixel or 8 bits/pixel. Four-bit pixels can be converted to analog RGB signals using the on-chip color table and digital-to-analog converters. Eight-bit pixels require external circuitry to convert them to analog RGB signals.

Each line buffer can hold enough information to generate one scan line for the monitor, up to 640 pixels at 4 bits/pixel or up to 320 pixels at 8 bits/pixel. The line buffers are used in an alternating manner: While the data in one buffer is being displayed, data for the next line is being loaded into the other buffer.

To convert line buffer data to RGB signals, the pixel data is shifted out one pixel at a time from the line buffer to the color lookup table. Each pixel code selects one of 16 colors from the lookup table. The outputs from the lookup table are converted by the on-chip DACs to analog RGB output signals. External buffers are required to interface these analog signals to the CRT.

Alternatively, the VSDD can be programmed to bypass the internal color table and DACs, and emit the pixel codes digitally.

A complete display image consists of an arrangement of individual objects, as shown in Figure 3. Each object is rectangular and upright. Information regarding height, width, on-screen location, and priority of each object is generated by the CPU and stored in DRAM.

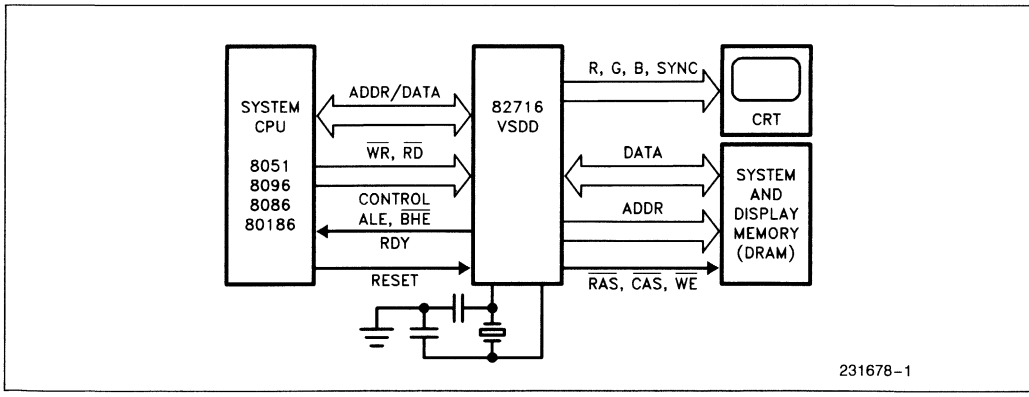


Figure 1. VSDD in Application

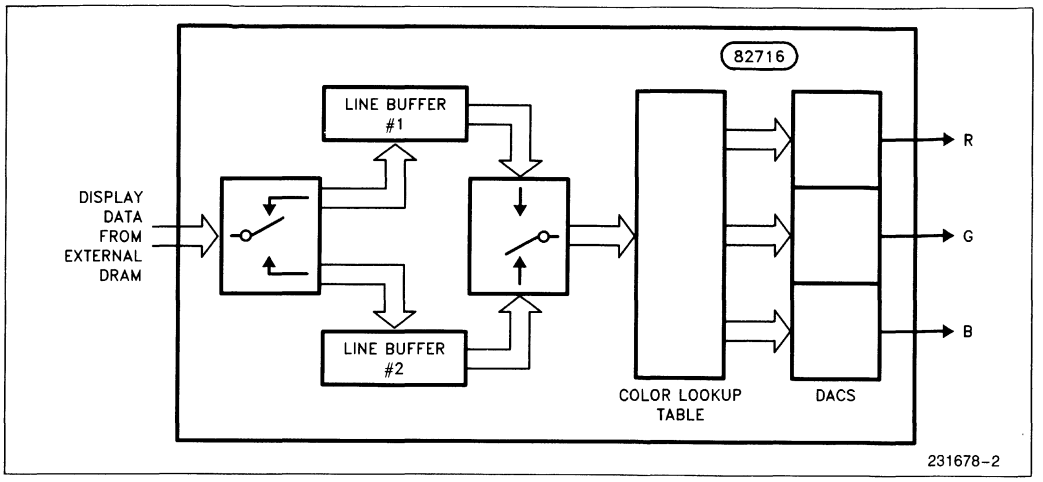


Figure 2. Scan Line Generation in the 82716. One line buffer is being loaded as the other is being read out.

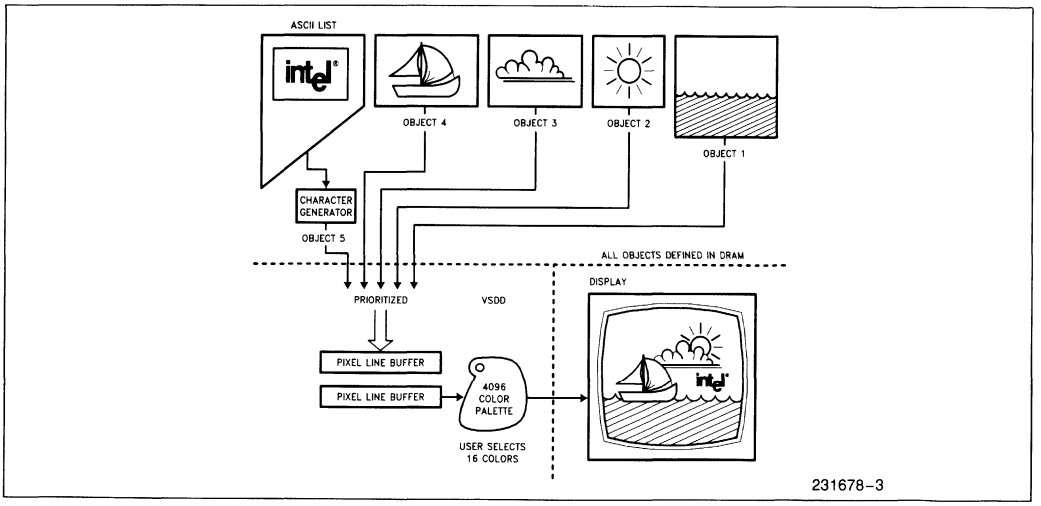


Figure 3. Building an Animation Scene

The "priority" of an object determines which objects are in front of or behind which other objects in the display. The object farthest in the background has priority 0. Any other object that overlaps this one on the screen will cover it, except where the overlapping object's pixels are transparent. Pixel code 0 is used to indicate transparency, if desired.

The 82716 is capable of handling up to 16 objects, having priorities 0 through 15. An object with priority 15 will be "in front of" any other object that it overlaps.

For data storage purposes, objects can be of two types: bit-mapped and character.

Bit-mapped objects are stored in DRAM as lists of pixel specifications. For each scan line in which a slice of the object is to appear, a certain number of these pixel specifications (depending on the object's width) are read into the line buffer.

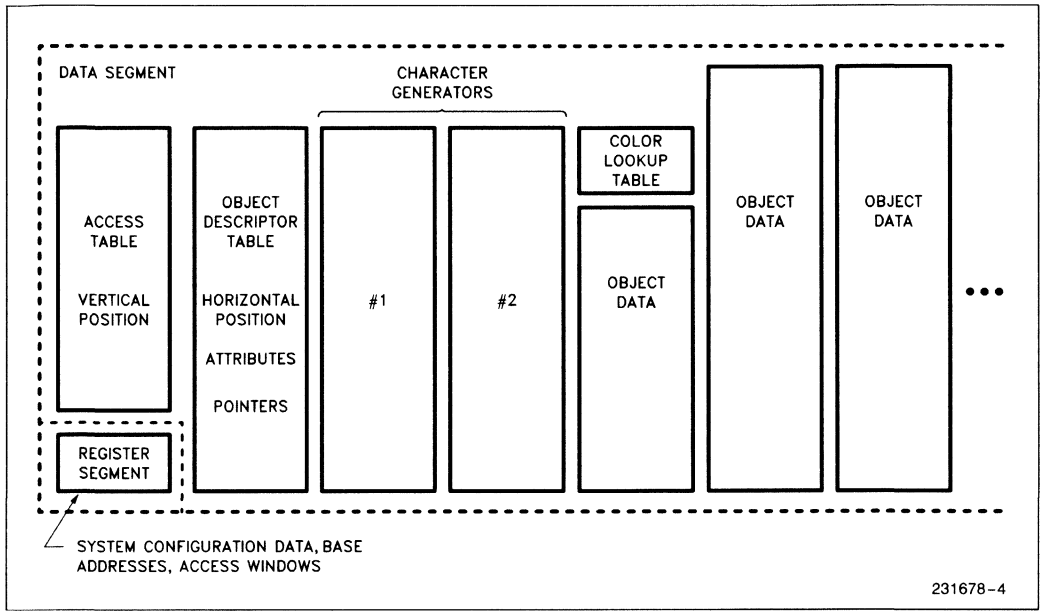


Figure 4. Data Blocks in Display Memory (External DRAM)

A character object is a sequence of text characters. Character objects are stored in DRAM as sequences of ASCII codes. Pixel specifications for this kind of object come from a character generator which is driven by the ASCII codes. The 82716 provides for two 256-character dynamically reconfigurable character generators to be simultaneously available.

Data Blocks

The 82716 divides the memory space into two segments as shown in Figure 4. These are the register segment and the data segment. The register segment consists of 32 bytes that contain system configuration data and base addresses for data blocks and CPU access windows.

The data segment contains the actual display data. This segment can be up to 512K bytes long (less the 32 bytes occupied by the register segment).

Included in the data segment are:

The **Access Table** contains the vertical position information and the priority number for each object. It consists of one 16-bit word for each line in the raster. Within each word, bit number **i** is the Access Flag for object number **i**, and object number **i** has priority **i**.

The **Object Descriptor Table** contains each object's horizontal position, width, and some other attributes, and pointers to the object's data block.

The two **Character Generators** control the pixel data for character objects. Each character generator is indexed by 8-bit codes.

The **Color Lookup Table** contains sixteen 12-bit entries specifying RGB intensities, at 4 bits per signal. Thus 4096 different colors can be specified, but only 16 are available at a time. The table is stored in DRAM, but is copied into on-chip registers for actual use. The base address of the Lookup Table in DRAM is stored in the register segment. A different color table can be called up, if desired, by writing a new base address to the register segment. This would cause a new Lookup Table to be read into the on-chip table at the end of the current frame.

The **Object Data** blocks contain pixel specifications (bit-mapped objects) or index/ASCII codes (character objects).

SCAN LINE CONSTRUCTION

The first thing the 82716 does to construct a scan line is fill the line buffer with the specified background color. Simultaneous to this operation, the VSDD reads the Access Table word for the line that is to be constructed. This is to determine which objects are present on the line and what their priorities are.

Then, if an object is determined to be present, its Object Descriptor data is read. This is to determine its width, horizontal position, and where to find the display data for this line of the object.

Then the object data itself is read, and the pixel codes are written into the line buffer at the appropriate horizontal positions. This data overwrites the background pixels that were previously written into the buffer.

The procedure is repeated for each object that is to be present on the line. No time is wasted processing objects that are not present on the line. If objects overlap, the new data overwrites the old data. Thus the priority number of an object is just the order in which the object is written into the buffer. Pixels that are transparent are blocked from being written into the buffer, so a lower priority object can still be visible behind the transparent parts of a higher priority object.

The construction process results in more pixels being read from the DRAM than are actually displayed on the line. Since only a finite time exists for line construction, it is important that the amount of vertical overlap between objects be considered when examining display performance.

When construction of the scan line is complete the VSDD enters an idling state to wait until the previously constructed line finishes being displayed. If display of the previously constructed line ends before construction of the new line is complete, the remainder of the line construction algorithm is aborted, and the VSDD's Construction Time Overflow signal is activated to indicate this condition to the CPU.

Construction time overflow can result when there are more pixels on the line than the VSDD has time to process, or when CPU-generated accesses to DRAM take up too much of the VSDD's time.

To avoid construction time overflows due to CPU-generated accesses, the VSDD can be programmed to allow only a certain (programmable) number of such accesses during line construction. Construction time overflow due to too many pixels on a line represents a more fundamental limitation on the display performance, which will be discussed presently.

Raster Control

The 82716 contains a Programmable Sync Generator which allows the raster parameters shown in Figure 5 to be defined by the user.

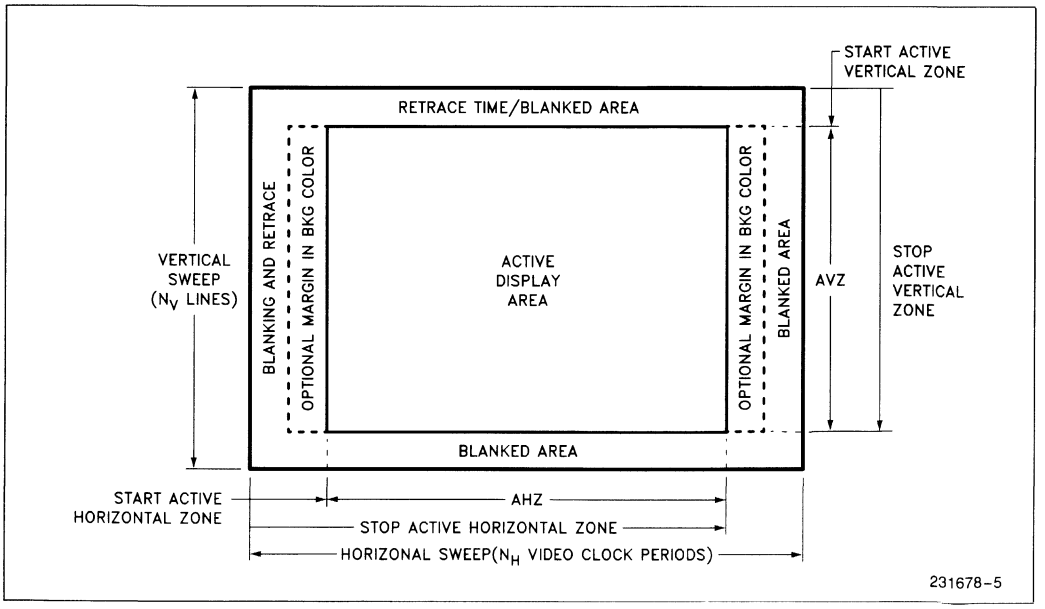


Figure 5. Programmable Raster Parameters

The number of lines in the vertical period is programmed into the register segment in the DRAM. The width of the field reserved for this number limits the vertical period to a maximum of 1024 lines. The same limitation applies also to the AVZ (Active Vertical Zone), but in practice the AVZ would have to be less than the vertical period to allow for vertical retrace time.

The horizontal timings are programmable from 1 to 64 units of time. The user can program the time step to be either 16 or 32 video clock periods, and the video clock frequency can range from 5 MHz to 25 MHz. For example, if a 14.5 MHz video clock is used, the horizontal timings can be programmed from 1.1 to 70.6 μs in 1.1 μs steps.

Programming the horizontal frequency and the number of lines per frame also defines the vertical frequency.

In most CRT applications the vertical frequency is the AC line frequency and the horizontal frequency is 15750 Hz $\pm$500 Hz. This corresponds to a line time of 61.5 to 65.5 μs. The number of lines in the vertical period is then the horizontal frequency divided by the vertical frequency, 254 to 271 lines, using 60 Hz for the vertical frequency. These requirements are easily met by the 82716 sync generator.

In addition to programmable timings, the raster can also be programmed to be interlaced or noninterlaced, and the vertical and horizontal snyc signals can be programmed, to be separate or composite.

The VSDD can also be programmed to lock onto externally generated sync signals.

Display Performance

The maximum amount of time that is available for the VSDD in single mode to construct a scan line is one horizontal period. In standard CRT displays this time is approximately 63.5 μs. During this time the VSDD must determine which objects are to appear on the line, what types of objects they are, and where their display data is. Then the data must be fetched from DRAM and written into the line buffer.

Since each object that appears on the line requires an additional series of DRAM accesses, the more objects there are on the line, the more time it will take to construct the line. One measure of display performance is the number of objects that can be displayed on the same scan line.

Because the number of DRAM accesses needed for each object depends also on the object width, a more general measure is just the total number of pixels that can be processed during line construction.

Performance is also affected by the VSDD system clock frequency and the DRAM speed. A system clock of 14 to 15 MHz is appropriate in most applications, and for the examples in this discussion we'll assume a system clock period of 70 ns.

Dynamic RAM access times are generally quoted in three ways: from $\overline{RAS}$, from $\overline{CAS}$, and in terms of the minimum cycle time in page mode. The VSDD accesses object data in DRAM using the page mode. The cycle time is programmable to be either 2 or 3 system clock periods, thus either 140 or 210 ns. DRAMs that have a minimum page mode cycle time of 140 ns or less can use the faster access mode of 2 oscillator periods per memory cycle. DRAMs that can't make the 140 ns cycle time must use the slower access mode of 3 oscillator periods per memory cycle.

The total time available for the VSDD to process pixels is as follows:

total pixel access time = line time − line overhead − (no. of objects × object overhead)

We will assume the line time to be 63.5 μs. The line overhead (during which the VSDD does a burst refresh on the DRAM, and a few other chores) is 86 system clock periods, approximately 6 μs, if fast DRAM is used, and 96 clock periods, 6.7 μs, if slower DRAM is used. The object overhead is the time it takes to read the object descriptor data for an object that is to be displayed, and compute the DRAM start and stop addresses for the object data that is to be read. This is 44 periods for fast DRAM and 48 for slow DRAM, amounting respectively to 3.1 μs or 3.4 μs per object.

For example if 4 objects have to be processed for a given line, then the total available pixel access time is

$$\text{total pixel access time} = 63.5\ \mu s - 6\ \mu s - (4 \times 3.1\ \mu s)$$
$$= 45.1\ \mu s$$

for fast DRAM and 43.2 μs for slow DRAM.

The number of DRAM accesses that can be made by the VSDD during that amount of time is

$$\text{number of DRAM accesses} = \frac{\text{total pixel access time}}{140\ \text{ns}}$$

for fast DRAMs, and

$$\text{number of DRAM accesses} = \frac{\text{total pixel access time}}{210\ \text{ns}}$$

for slow DRAMs. Thus for the example given above, the number of accesses that can be made is 322 for fast DRAMs and 205 for slow DRAMs.

How many pixels does each access fetch? Pixels can be stored in memory at 8 bits/pixel, 4 bits/pixel, or 2 bits/pixel. Memory accesses for line construction are always 16-bit accesses. Therefore each access will bring in 2, 4, or 8 pixels, depending on how the pixels are encoded.

Thus the total number of pixels that can be fetched during construction of this line that has 4 objects on it is 644, 1288, or 2526 (at 8, 4, or 2 bits/pixel) for fast DRAMs.

Notice that the total number of pixels that the VSDD is capable of fetching during line construction may be larger than the total number of pixels that can be displayed on a line. This is useful, because not all the pixels fetched are necessarily displayed. For example, if objects overlap, the VSDD still has to fetch the pixels "underneath" the overlap area, even though they will soon be overwritten by pixels from the overlapping object.

It should be noted that the 2 bits/pixel mode reduces to 4 the number of colors that the object can have. Any object that can be drawn with 4 colors or less can take advantage of the 2 bits/pixel mode to increase the display performance. Objects that require the full range of colors in the lookup table must be stored at 4 bits/pixel. Both types of objects (i.e., those stored at 2 bits/pixel and those stored at 4 bits/pixel) can be mixed on the screen.

Up to 256 colors can be obtained in digital mode using external color look-up table and DACs.

For character objects, the performance numbers will change as the VSDD has to access both the object data and the character generator. The 82716 accesses the object data and character generator alternately.

The VSDD needs 12 system clock cycles for fetching two (2) characters in 1 byte/character mode and 14 cycles for 3 bytes/character mode.

Assuming 63.6 μs line time and 70 ns system clock:

$$\text{Number of cycles/line} = \frac{63.6 \times 10^3}{70} = 908 \text{ cycles}$$

Line overhead = 86 cycles for fast DRAMS

= 96 cycles for slow DRAMS

Object overhead = 44 cycles for fast DRAMS

= 48 cycles for slow DRAMS

Assume that there are four (4) objects on each line:

Number of cycles available for character processing = $908 - 96 - 48 \times 4$ = 620 for slow DRAMs

= $980 - 86 - 44 \times 4$ = 656 for fast DRAMs

For 1 byte/character mode, VSDD can fetch 103 ($620/12 \times 2$) characters for slow DRAM and 109 characters for fast DRAM during one line time.

Similarly, for 3 bytes/character, 88 characters can be fetched using slow DRAM and 93 characters, using fast DRAM.

PART II: VSDD CIRCUITRY

A block diagram of the 82716 is shown in Figure 6. Most of the blocks in this diagram will be discussed in depth in subsequent sections. By way of introduction their functions are as follows:

The BIU, or Bus Interface Unit, is the window through which the CPU accesses the display memory. It consists of a 16-bit multiplexed address/data bus plus one programmable chip select (A16), and associated control inputs: ALE, $\overline{RD}$, $\overline{WR}$, and $\overline{BHE}$. It also provides a ready output, RDY, and construction time overflow, CTO flag.

The MIU, or Memory Interface Unit, is the circuitry associated with accessing and maintenance of the external dynamic RAM. It consists of a 9-bit multiplexed row and column address bus, a 16-bit data bus, and associated control outputs: $\overline{RAS}$, $\overline{CAS}$, and $\overline{WE}$.

The Pixel Unit is the block that converts raw display data from external memory to analog RGB signals. It contains two Scan Line Buffers, the Color Lookup Table, and the Digital-to-Analog Converters whose outputs are the analog Red, Green, and Blue signals to the CRT monitor. It also provides an Overlay signal. V_{REF} is an input for the reference voltage (normally about 1.6V) required by the DACs.

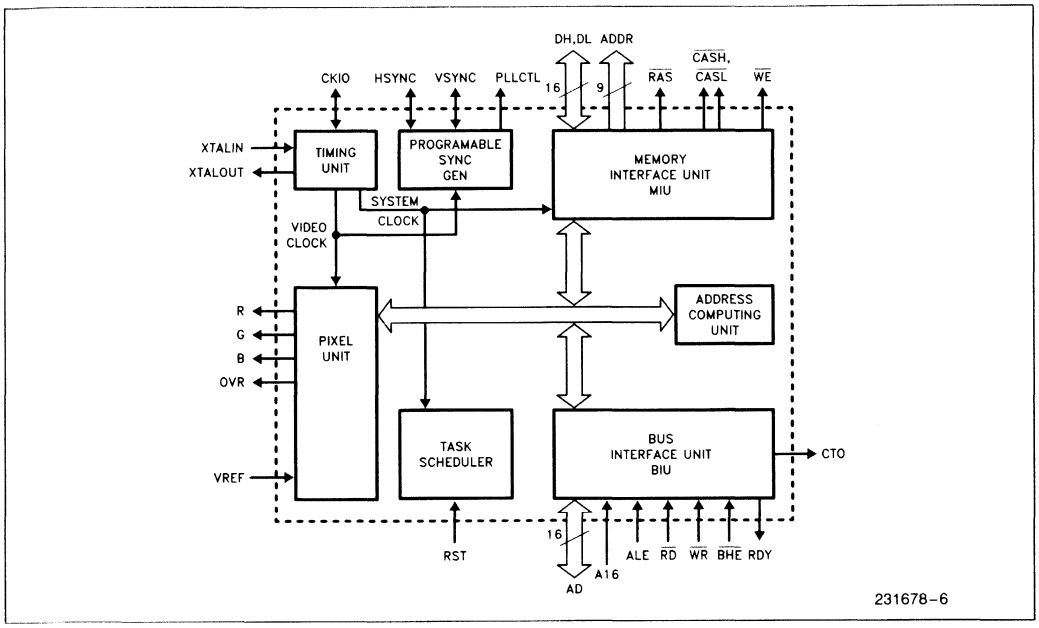


Figure 6. Block Diagram of the 82716

The Programmable Sync Generator provides vertical and horizontal sync pulses to the CRT. The number of lines in the raster, the type of raster (interlaced or non-interlaced), the period and width of the sync pulses, etc., are all programmable in the 82716. The same block is also capable of locking the 82716 onto externally generated sync signals. Separate pins are provided for vertical and horizontal sync signals (VSYNC and HSYNC). The PLLCTL output is provided to control the video clock oscillator frequency in phase-locked loop operation.

The Timing Unit contains the inverting amplifier (between the XTALIN and XTALOUT pins) for a crystal controlled oscillator, and the clocking circuits for the logic and video sections of the VSDD. The CKIO pin is software programmable to be either a clock output signal or an external clock input for the video section of the VSDD.

Some of the most important blocks in the system are largely transparent to the user, in the sense that they are neither programmable nor accessible, and so will not be discussed to the same depth as the blocks that the user must deal with. Among these are the Task Scheduler and the Address Computing Unit.

The Address Computing Unit is a dedicated ALU, capable of incrementing or decrementing counters, computing start and stop addresses for VSDD-generated DMA sequences, etc.

The Task Scheduler is the main control circuit of the VSSD. It is a ROM-based state machine which oversees the sequences which the VSDD must execute. The sequences are, for example, read/write sequences, the scan line construction sequence, etc.

Timing Unit

The Timing Unit, Figure 7, consists of the oscillator and clock generators.

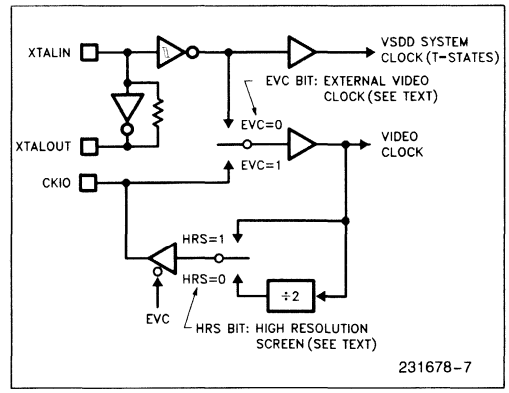
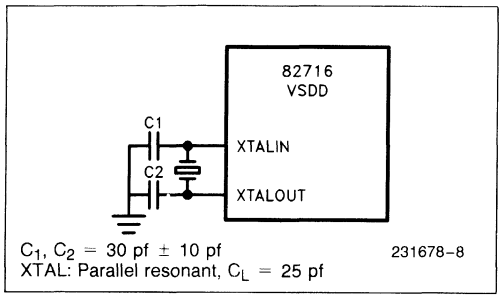


Figure 7. 82716 Timing Unit

The oscillator is a single stage inverting amplifier, which may be used as a crystal-controlled, positive reactance (or "Pierce") oscillator. In this application the crystal is operated in its fundamental response mode as an inductive reactance in parallel resonance with capacitance external to the crystal. Figure 8(a) shows the external components that may be used. A 14 MHz–15 MHz crystal (parallel resonant, 25 pF load capacitance) can be used, to facilitate programming of the sync generator for standard TV timings.



C_1, C_2 = 30 pf ± 10 pf
XTAL: Parallel resonant, C_L = 25 pf

Figure 8(a). Using the VSDD Oscillator

An external oscillator can also be used. In that case the external oscillator signal would be connected to XTALIN, and XTALOUT would be left unconnected, as shown in Figure 8(b).

NOTE:
XTALOUT is to be used only as the feedback loop in the XTAL oscillator circuit and should not be used for any other purpose.

The external oscillator signal should nominally have a 50% duty cycle. In either case the signal at XTALIN drives the VSDD system clock.

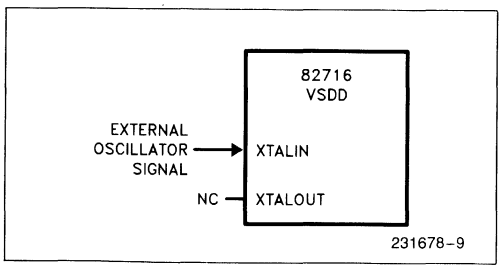


Figure 8(b). Driving the System Clock with an External Oscillator Signal

The video clock, which is the time base for the Pixel Unit and the Programmable Sync Generator, can either be the XTALIN signal or a separate clock signal applied to the CKIO pin. This selection is made by clearing or setting the control bit EVC (External Video Clock). If EVC = 0, the video clock is taken from XTALIN. If EVC = 1, the video clock is taken from CKIO.

When EVC is 0, the CKIO pin emits a clocking signal whose frequency is either the same as the video clock or ½ the frequency of the video clock. This selection is made by clearing or setting the HRS (High Resolution Screen) bit. If HRS = 0, the CKIO output frequency is ½ the video clock frequency. If HRS = 1, the CKIO output frequency is the same as the video clock frequency. The CKIO signal is normally used when the VSDD is in a mode of bypassing its on-chip DACs, and is emitting video information digitally. In that case the CKIO signal is used to indicate when the digital information is valid. These options are discussed more fully in the section covering the Pixel Unit.

MIU: Memory Interface Unit

The 82716 can access 512K bytes of dynamic RAM, which it organizes as 256K words, using one 18-bit address for each 16-bit word. Address information is sent to the DRAM 9 bits at a time through the ADDR bus. When the first 9 bits are valid on the bus, $\overline{RAS}$ is activated. When the second 9 bits are valid, $\overline{CAS}$ is activated. Two $\overline{CAS}$ pins are provided: $\overline{CASH}$ and $\overline{CASL}$. Both are activated for word accesses, and either $\overline{CASH}$ or $\overline{CASL}$ is activated for byte accesses. If the access is a write cycle, $\overline{WE}$ is also activated.

Timing diagrams for the basic write and read cycles are shown in Figures 9 and 10. The T-states are derived from the oscillator signal at XTALIN. Each T-state lasts for one oscillator period. CK1 and CK2 are the first and second halves of each T-state. Their exact duration depends on the duty cycle of the signal at XTALIN.

For both read and write cycles, the row address is emitted during CK2 of T1. $\overline{RAS}$ is activated ½ oscillator period later, at the beginning of T2. The ADDR pins switch to column address during the second half of T2, and $\overline{CAS}$ is activated at the beginning of T3.

For write cycles, write data begins during the CK2 half of T1, and is held till the CK2 half of T5. As shown in Figure 9, $\overline{WE}$ is activated for write cycles ½ oscillator period before $\overline{CAS}$, and deactivated ½ oscillator period after $\overline{CAS}$. Write cycles are therefore of the "Early Write" or "$\overline{CAS}$ Controlled" type.

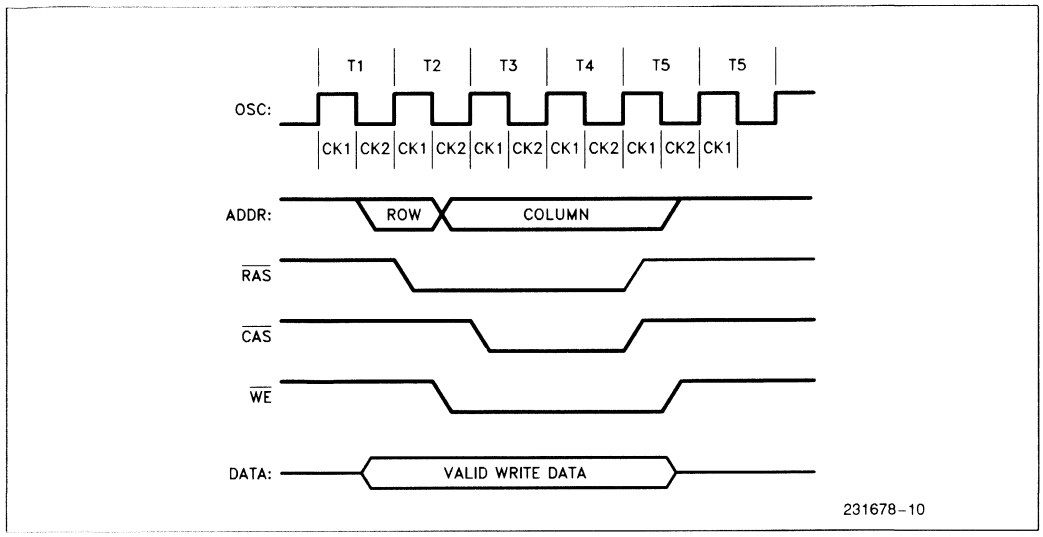


Figure 9. MIU Write Cycle

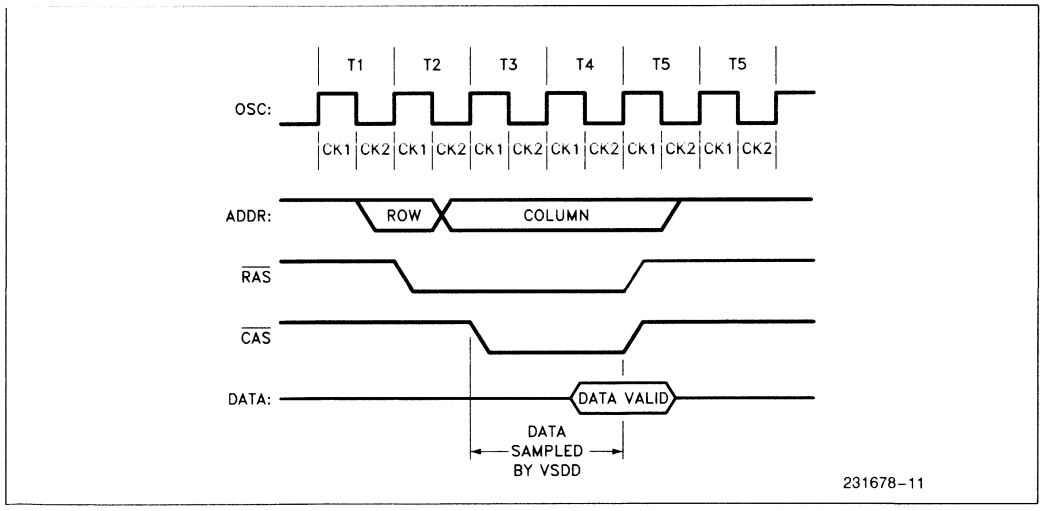


Figure 10(a). MIU Random Read Cycle

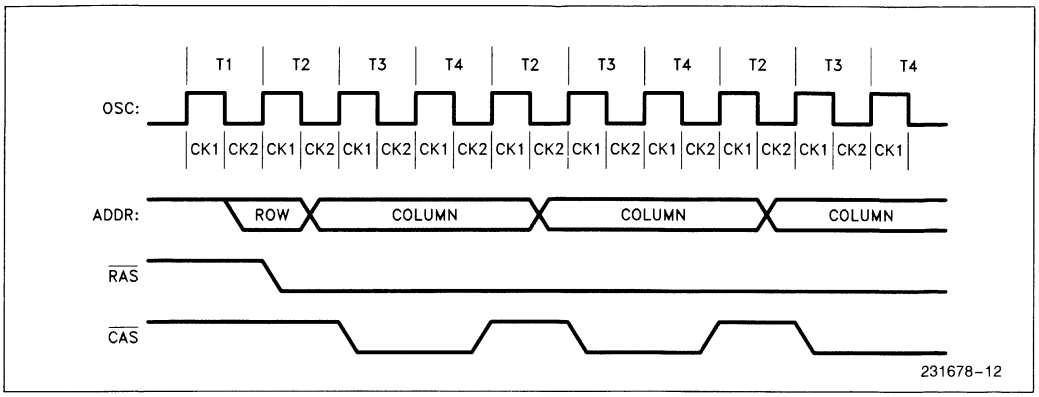


Figure 10(b). MIU Page Mode Read Cycles

For read cycles, data is sampled by the VSDD during the entire time that $\overline{CAS}$ is active. Read cycles that are initiated by the VSDD (as opposed to those requested by the CPU) support page mode operation when sequential reads use the same row address, which is quite often, during line construction. Figure 10(a) shows a random read cycle (requested by the CPU), and Figure 10(b) shows the beginning of a series of page mode read cycles as would be initiated by the VSDD during line construction. Note that the sequence of T-states in page mode operation is (T2, T3, T4) for each read—thus each access takes 3 oscillator periods.

In fact, the T4 state is optional. Its inclusion in the memory cycles depends on the state of control bit SAB (Slow Access Bit). If SAB = 1, then T4 is included. If SAB = 0, T4 is skipped, thus shortening the cycle times for both read and write operations by one oscillator period.

SAB can be used to increase the performance of the VSDD when fast DRAMs are used, or to make the VSDD compatible with slower DRAMs in lower cost systems. If one assumes a 70 ns clock period (14.3 MHz at XTALIN), then SAB can be used to set the read cycle time in page mode to either 140 ns or 210 ns. DRAMs that have a minimum page mode cycle time of 140 ns or less can use the faster access mode to obtain optimum performance in terms of the total number of pixels that can be fetched out of memory during line construction.

DRAM Configuration

As previously noted, the DRAM is organized by the VSDD in 16-bit words. Both types of DRAMs, x1 and x4 are supported by the VSDD. Three configuration bits, DS0, DS1, and DOF, are written to the VSDD by the CPU during system initialization to establish the DRAM size and configuration. DS0 and DS1 are used to indicate the DRAM configuration size and not the total amount of memory available to the VSDD. See table below. For 16K DRAMs, bit DOF (DRAM Organization Flag) is used to indicate if the DRAM is bit-wide (DOF = 0) or nibble-wide (DOF = 1). The VSDD uses the ADDR lines in a slightly different way for each DRAM configuration, as shown in the table below:

DS1	DS0	DOF	DRAM Configuration	Byte Capacity	ADDR Pins Used		
					Row	Col	Bank Select
0	0	0	16K x 1	32K	0–6	0–6	(none)
0	0	1	16K x 4	128K	0–7	0–5	6, 7 at CAS
0	1	0	64K x 1	128K	0–7	0–7	8 at $\overline{RAS}$, $\overline{CAS}$
0	1	1	64K x 4	512K	0–7	0–7	8 at $\overline{RAS}$, $\overline{CAS}$
1	0	0	256K x 1	512K	0–8	0–8	(none)

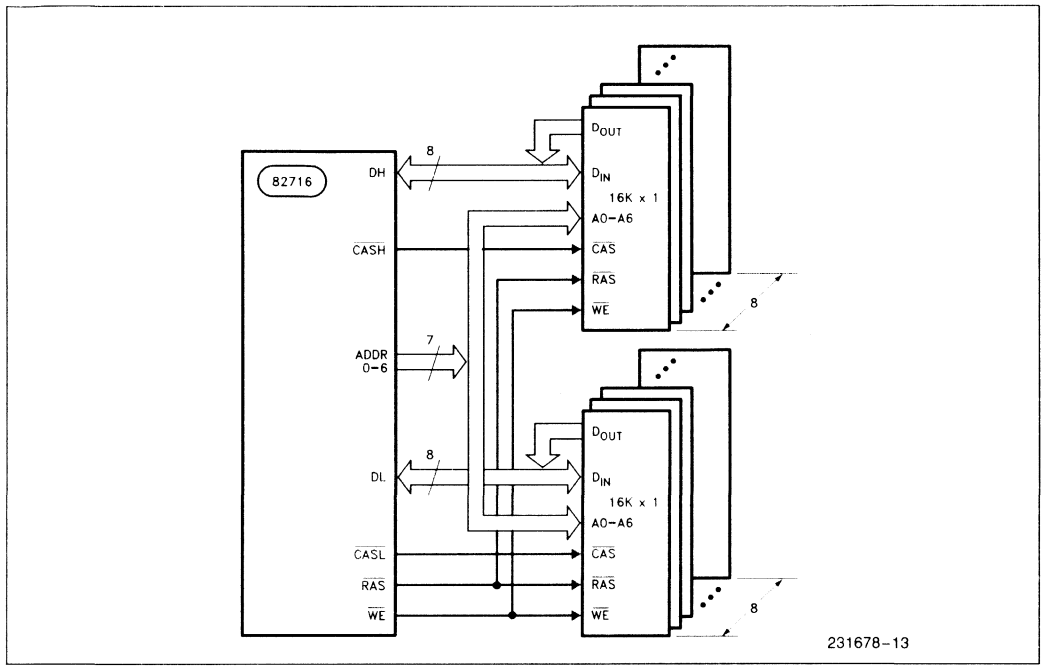


Figure 11. VSDD/DRAM Interface (16K x 1)

Thus if the VSDD is configured for 16K x 1 DRAMs, it generates 7-bit row and column addresses ADDR0 through ADDR6. ADDR8 and ADDR7 aren't used. Since no bank select bits are emitted, 16K words (32K bytes) is the only memory size that can be implemented with these devices. A memory interface of this type is shown in Figure 11.

Configured for 16K x 4 DRAMs, the 82716 emits 8-bit row address on ADDR0 through ADDR7, and then a 6-bit column address on ADDR0 through ADDR5. ADDR6 and ADDR7 emit bank select bits (ADDR7 = MSB, ADDR6 = LSB) with the column address. At least four 16K x 4 DRAMs must be used, amounting to 16K words. In addition, up to four such banks can be selected by decoding the bank select bits, as shown in Figure 12. In this manner up to 64K words can be implemented.

In Figure 12 a dual 2-to-4 line decoder is used to generate $\overline{CAS}$ signals to the selected bank when the VSDD activates $\overline{CASL}$ and/or $\overline{CASH}$. Note that the $\overline{OE}$ inputs to the DRAMs are hard-wired to ground. Since the VSDD always activates $\overline{WE}$ before $\overline{CAS}$, and deactivates $\overline{CAS}$ before $\overline{WE}$, the state of $\overline{OE}$ during write operations is a don't-care.

Note too that standard 16K x 4 DRAMs want column addresses on address inputs 1 through 6, whereas the VSDD emits column addresses on lines 0 through 5. For that reason address outputs ADDR0 through ADDR6 are connected to DRAM inputs A1 through A7, and ADDR7 is connected to A0.

Figure 13 shows how to connect the VSDD to 64K x 4 DRAMs. When the VSDD is configured for 64K x 4 DRAMs, it emits 8-bit row and column addresses on ADDR0 through ADDR7. Two bank select bits come out on ADDR8: one with the row address (MSB) and one with the column address (LSB). At least four such DRAMs must be used, amounting to 64K words (128K bytes). One or both bank select bits can be used to direct $\overline{CASL}$ and $\overline{CASH}$ to a selected bank in 128K-word and 256K-word systems. Figure 13 shows both bank select bits being used in a four-bank system. In a two-bank system the flip-flop used to capture the first bank select bit at $\overline{RAS}$ can be eliminated and the B inputs to the address decoder should be tied low.

For 64K x 1 DRAMs, the VSDD emits 8 bit row and column addresses on ADDR0 through ADDR7. Bank select bits are also emitted as in 64K x 4. But the VSDD does not have the capability to drive four banks of 64K x 1.

9 row and column address bits are emitted on ADDR0 through ADDR8 for 256K x 1 DRAMs. 512K bytes of memory can be implemented using 16 such DRAMs.

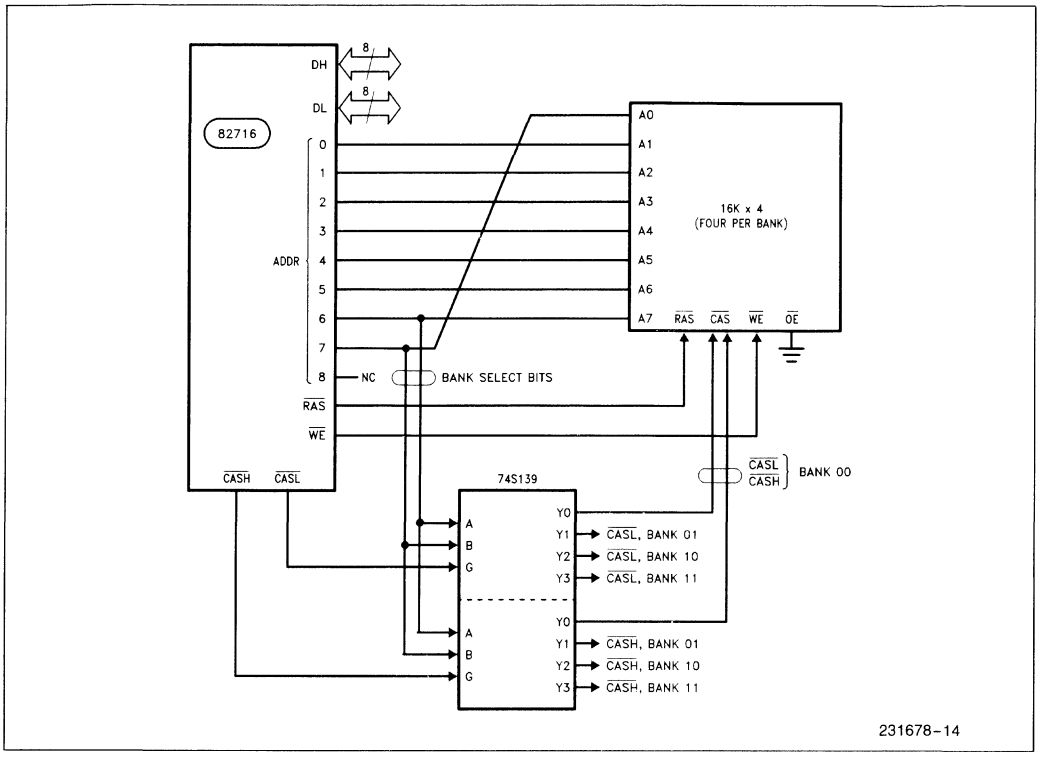


Figure 12. VSDD/DRAM Interface (16K x 4)

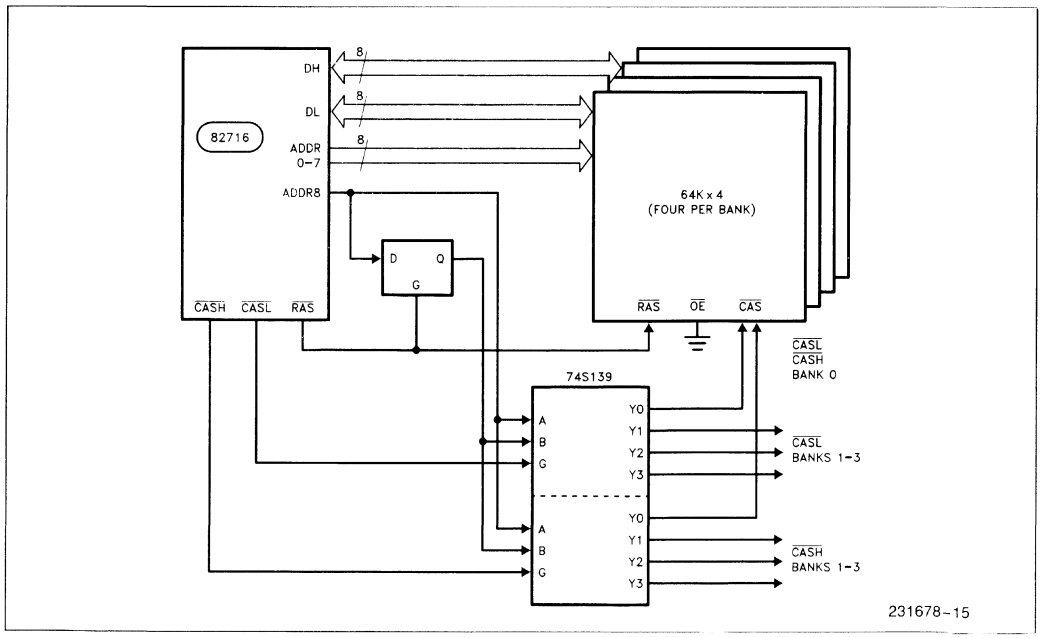


Figure 13. VSDD/DRAM Interface 64K x 4

DRAM Refresh

Data retention in dynamic RAMs requires that every row in the RAM array be accessed (refreshed) at least once during a specified time interval. Row access for refresh requires only that $\overline{RAS}$ be activated while the address of the selected row is valid on the DRAM address lines.

Standard DRAMs generally fall into one of two categories regarding refresh requirements. Some require 128 accesses (to all combinations of the low 7 bits of row addresses) within 2 ms, and some require 256 accesses (to all combinations of the low 8 bits of row addresses) within 4 ms.

The 82716 refreshes 12 rows at the beginning of each line construction sequence. Thus it steps through all 128 combinations of the low 7 bits of row addresses in 11 line-times, and through all 256 combinations of the low 8 bits of row addresses in 22 line-times.

At the end of each frame the VSDD enters into a sequence of housekeeping tasks which requires 740 oscillator periods (at XTALIN) to complete. While it's executing these tasks, no refreshes are generated. If the CPU is simultaneously accessing the DRAM during this period, then the length of time it takes to complete the housekeeping tasks is lengthened. The tasks continue to be executed, however, and the increased length of time will not exceed 8880 oscillator periods. This is the maximum length of time that no refreshes are generated.

If a 70 ns clock is used (14.3 MHz), the 8880 oscillator periods take 622 μs. This leaves 1378 μs in which to complete a refresh cycle of 128 rows within 2 ms, or 3378 μs in which to complete a refresh cycle of 256 rows within 4 ms.

Since it takes 11 line-times to cycle through 128 rows, and this must be done in 1378 μs, there is a maximum allowable line-time of 1378/11 = 125 μs in systems that use 128 cycle/2 ms DRAMs. Since it takes 22 line-times to cycle through 256 rows, and this must be done in 3378 μs, there is a maximum allowable line-time of 3378/22 = 153 μs in systems that use 256 cycle/4 ms DRAMs.

Most graphics displays use a line time of about 64 μs. Consequently the limiting values on line times to ensure data retention in the DRAMs do not represent a barrier. Line times generated by the 82716 are programmable in the manner described in the section covering the Programmable Sync Generator.

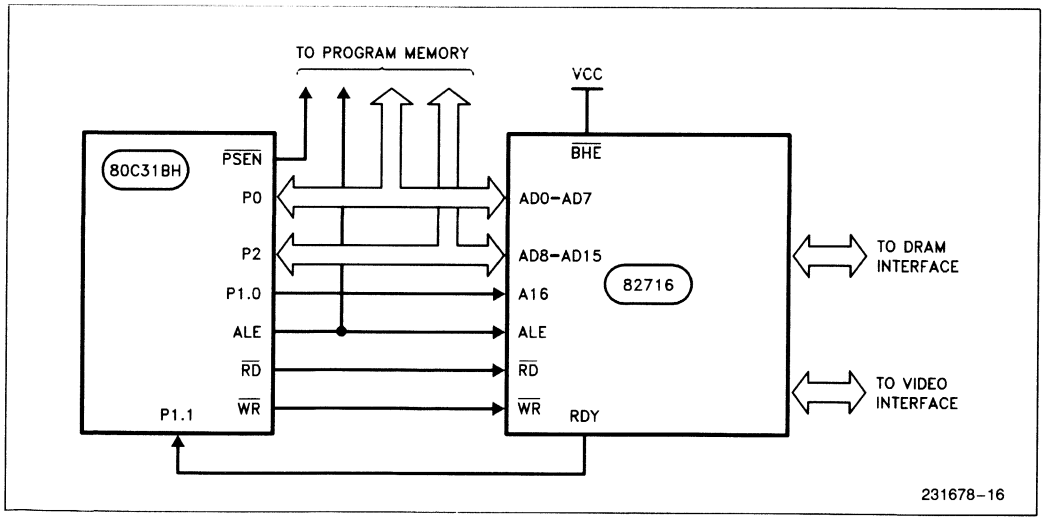


Figure 14. Interfacing the 82716 to an 8-bit CPU. RDY is being used in its "Free Access" function (see text).

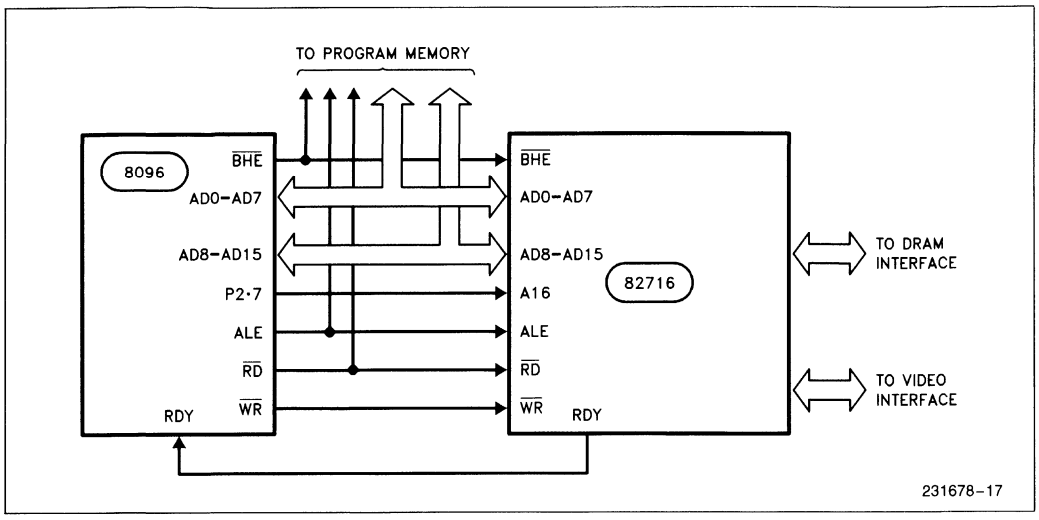


Figure 15(a). Interfacing the 82716 to the 8096

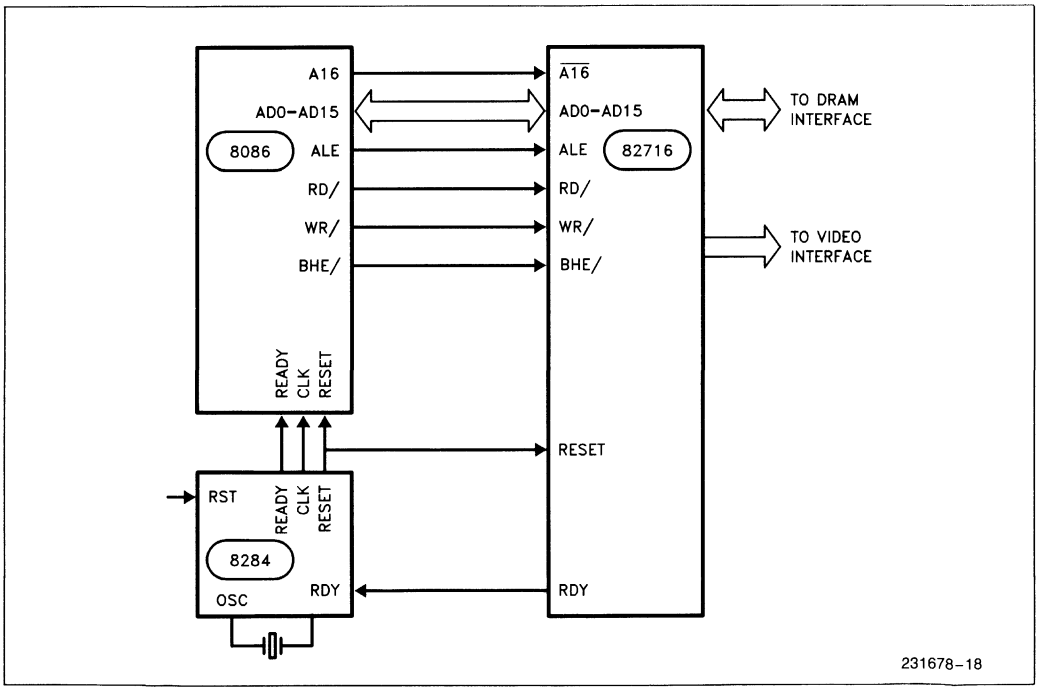


Figure 15(b). Interfacing the 82716 to the 8086

BIU: Bus Interface Unit

The CPU accesses the DRAM through the 82716's Bus Interface Unit. The low 16 address bits are multiplexed with data on the AD pins of the VSDD. Figure 14 shows a typical interface between an 8-bit CPU (an 80C31BH, in this case), and Figure 15(a) shows a typical interface to a 16-bit CPU. Figure 15(b) shows the 8086 interface. A16 acts as chip select.

The 82716 has a $\overline{BHE}$ (bus high enable) input, which enables the upper 8 bits of the AD bus to be used for data. If the CPU's data bus is 8 bits wide, $\overline{BHE}$ must be tied high, as shown in Figure 14. For a 16-bit data bus, $\overline{BHE}$ should be connected to the CPU's $\overline{BHE}$ output pin, as shown in Figure 15(a).

The VSDD generates a Ready signal (RDY) which can be used to wait-state the CPU during memory accesses, as shown in Figure 15(a). However, the CPU can also access the DRAM without reference to RDY, as shown in Figure 14. Here RDY pin acts as Free Access Enable indicator to the CPU.

A CPU request to access the DRAM begins with the CPU emitting an address to the VSDD that is within one of two pre-defined windows in the CPU's address space. The way these windows are defined and used is described in **Part III: Data Structures**. Here we consider only the mechanics of the VSDD's interface to the CPU.

While ALE is high, address information flows into the VSDD. When ALE falls the address is held. Then, if the address is within a VSDD window, $\overline{RD}$ or $\overline{WR}$ can be activated to request access to the DRAM.

Write Cycle

$\overline{WR}$ going low activates a write request to the 82716 Task Scheduler. While $\overline{WR}$ is in a low state, the data to be written flows into a transparent latch in the BIU.

To service the request the VSDD must transfer the write information internally from the BIU to the MIU, and then generate the MIU write cycle. Depending on what the VSDD was doing at the time the write request was activated, 10 to 16 oscillator periods will elapse before the MIU write cycle begins, and the MIU write cycle (Figure 9) will take either 4 or 5 oscillator periods, depending on whether SAB is 0 or 1. With a 70 ns clock at XTALIN, these times are: 0.7 to 1.12 μs from $\overline{WR}$ low to MIU write, plus either 0.28 or 0.35 μs for the MIU write cycle.

Use of RDY to generate wait states in the CPU during the write cycle is optional, since the BIU holds the write data after $\overline{WR}$ goes high. If RDY is used for that purpose, then RDY = 0 forces the CPU to hold $\overline{WR}$ low. The VSDD brings RDY up when the internal transfer of write data from the BIU to the MIU has been completed. Waveforms and appropriate timing parameters are shown in Figure 16.

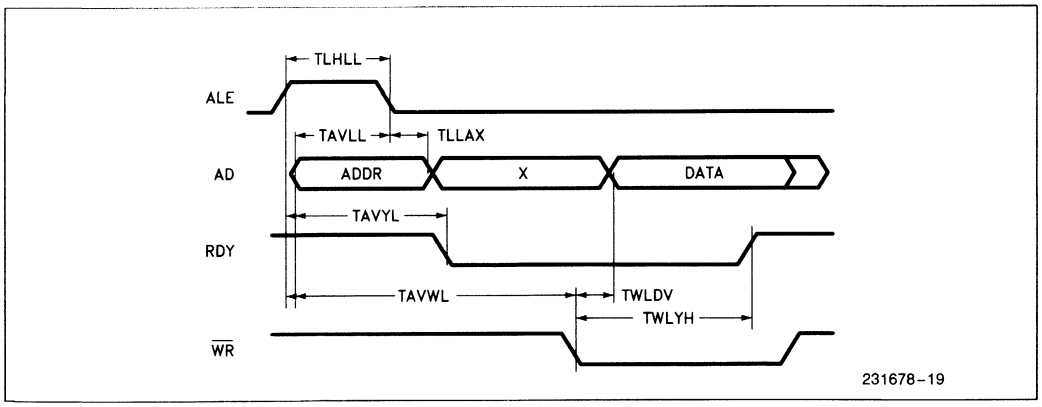


Figure 16. CPU Write Cycle Using RDY to Generate Wait-States

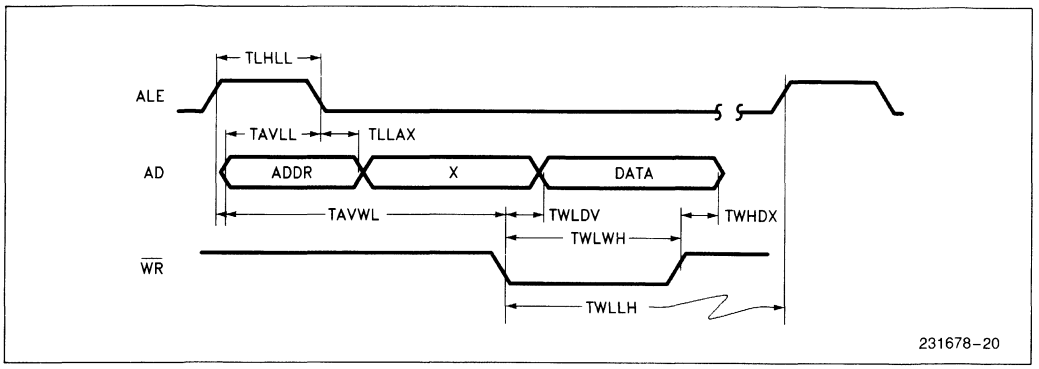


Figure 17. CPU Write Cycle Not Using RDY

If the Ready signal is not used, then $\overline{WR}$ going high latches the write data into the BIU. The CPU then continues its own program execution while the VSDD goes through its sequence of transferring the data to the MIU and generating the MIU write cycle. Timing parameters appropriate to this mode of access are shown in Figure 17. Since the CPU operating in this mode doesn't wait for the VSDD to actually commence its MIU write cycle, the parameter TWLLH is included. This parameter is measured from the leading edge of $\overline{WR}$ to the leading edge of ALE **of the next read/write cycle** to ensure that the CPU doesn't try to access the DRAM faster than the VSDD can service the requests.

Read Cycle

$\overline{RD}$ going low to an address that is within one of the VSDD's windows activates a read request to the Task Scheduler. The VSDD responds by pulling RDY low and enabling the outputs of its bus drivers.

Initially these drivers will be emitting data from the previous read cycle.

Depending on what the VSDD was doing when the read request was activated, 8 to 14 oscillator periods will elapse before the MIU read cycle begins. The MIU read cycle, Figure 10(a), takes either 4 or 5 oscillator periods, depending on the state of the Slow Access Bit (SAB) in register R0.

There is a control bit in the VSDD named PRE (Pipeline Read Enable). If PRE is 0, the output drivers switch from previous read cycle data to current read cycle data during state T5 of the MIU read cycle. RDY pulls high at the end of this time state.

For the CPU, RDY going high indicates that the new data is now available. The processor now completes its read cycle and continues program execution. Waveforms and appropriate timing parameters are shown in Figure 18.

Pipeline Read Mode

Reference is made above to the fact that activating $\overline{RD}$ enables the outputs of the VSDD's output drivers, which are initially emitting data from the **previous** read cycle. It is noted that if the PRE bit is 0, the output drivers switch to emitting current read cycle data when that data becomes available. In fact if the PRE bit is set (1), then the drivers will continue to emit data from the previous read cycle for the entire duration of the current read cycle.

The effect of PRE = 1, then, is that any CPU read cycle will always return data from the previous read cycle. This pipeline read mode is implemented specifically to facilitate interfacing to CPUs that do not use the Ready signal to insert wait states in their read/write cycles.

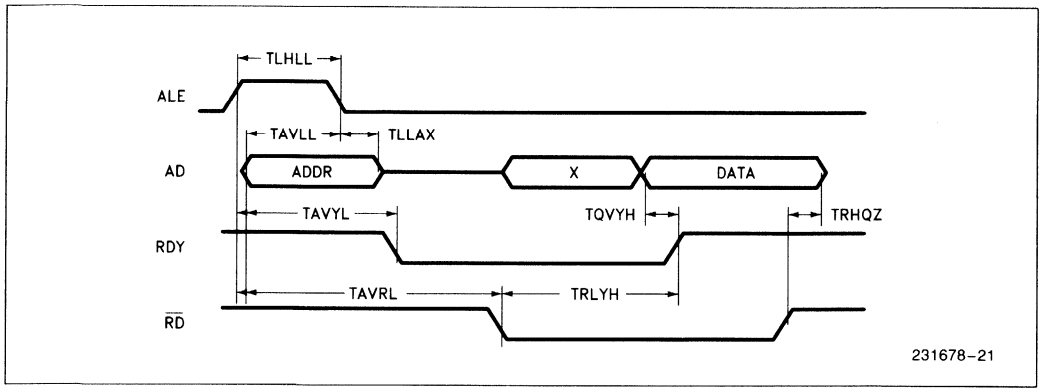


Figure 18. CPU Read Cycle (PRE = 0) Using RDY to Generate Wait-States

Wait states aren't really needed for a write cycle, since the BIU section of the VSDD latches the write data if $\overline{WR}$ goes away before the MIU section commences its write cycle. For a read cycle, however, the data is not available for the CPU until the MIU has completed its read cycle, which may or may not be within the CPU's read cycle. In this case the PRE bit ensures that the data returned to the CPU is from the previous read cycle, even if the current read cycle is long enough without wait states for the current read data to have become available. PRE = 1 prevents the output data from being updated until $\overline{RD}$ is deactivated.

Waveforms and timing parameters for the PRE = 1 Mode are shown in Figure 19. Since the CPU doesn't wait for the VSDD to actually commence its MIU read cycle, the parameter TRLLH is included. This parameter, measured from the leading edge of $\overline{RD}$ to the leading edge of ALE **of the next read/ write cycle**, is to ensure that the CPU doesn't try to access the DRAM faster than the VSDD can service the requests.

Using the Pipeline Read Mode

Any CPU that does not use the Ready signal should invoke the pipeline read mode by setting the PRE bit. Otherwise there's no way to be sure if data returned in a read cycle is for this read cycle or the previous one.

In this mode a single read access to the DRAM (as opposed to a series of accesses) in general requires two CPU read cycles. The first one is to address the desired DRAM location, but will not return data from that location. The second read cycle can be to any DRAM address, but will return the data that was addressed in the first cycle.

Less overhead is required for a series of reads: the first read cycle returns random data, but after that each read cycle returns the data that was addressed in the previous cycle.

For this purpose a "series" of reads means only that the address for the (N + 1)th read is known at the time of the Nth CPU read cycle. The individual reads in the series can be separated by any amount of time. However, in pipeline read mode, data requests can happen no quicker than 22 VSDD/MIU clock cycles.

Free Access Enable

If the Ready signal is not being used, there is no reason to dedicate a pin to its function. The 82716 contains a control bit named FAE (Free Access Enable). Setting this bit selects an alternate function for the RDY pin. The alternate function is that the pin will emit a logic high when the VSDD is "between frames," i.e., above or below the Active Vertical Zone of the screen (Figure 5). During these times the CPU can have more-or-less "free access" to the DRAM.

Of course the CPU is free to access the DRAM at any time, and these accesses will always take priority over any line construction under way by the VSDD. However, too many such accesses during the Active Vertical Zone may leave some lines with incomplete objects. This state of affairs would be flagged to the CPU by the VSDD's CTO (Construction Time Overflow) pin going high.

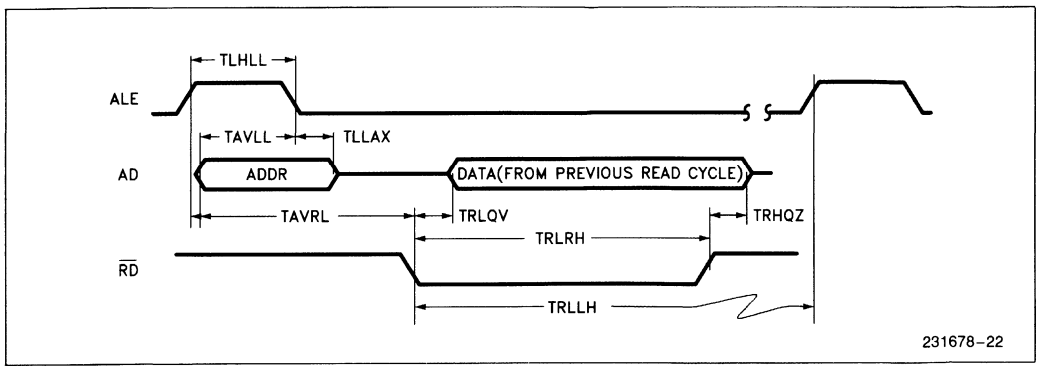


Figure 19. CPU Read Cycle (PRE = 1) Not Using RDY

Programmable Priority Accesses

CPUs that have a true Ready input can take advantage of another feature of the 82716. It is noted above that CPU accesses always take priority over VSDD accesses during line construction. Too many such accesses during the Active Vertical Zone, however, will leave some lines with incomplete objects. The 82716 can be programmed to allow only N such accesses if desired. The (N + 1)th such access (during the Active Vertical Zone) will result in RDY going low and being held there until the line under construction is completed.

This feature is enabled by setting the PCE (Priority Counter Enable) bit, and the number N is defined by loading it to a VSDD register named PAQ (Priority Access Quantity). The width of PAQ restricts N to a maximum of 15.

It should be noted that if the CPU is an 8096 or a derivative of it, the RDY line can be used, since the 8096 does have a Ready input, but the Programmable Priority Accesses feature can't be used. That is because the 8096 imposes a maximum amount of time that its Ready line can be held low. With a 70 ns clock the VSDD will meet this timing specification during normal read and write cycles, but use of the Programmable Priority Accesses feature could result in RDY being held low for several microseconds.

Programmable Sync Generator

The PSG, or Programmable Sync Generator, provides timing signals to other sections of the 82716 and to the CRT. The timings are programmable in units of the video clock period. The video clock signal can be selected by the EVC (External Video Clock) bit to be either the XTALIN clock (EVC = 0) or a separate video clock applied to the CKIO pin (EVC = 1).

Video Timing

The PSG provides basically four timing signals: for the CRT it generates the horizontal and vertical sync pulses HSYNC and VSYNC, and for the Pixel Unit it defines the active horizontal and vertical zone signals AHZ and AVZ.

Figure 20 shows the horizontal signals HSYNC and AHZ. The timing parameters that are programmable are the width and period of HSYNC, and the start and stop times of AHZ.

AHZ defines the time during which video information comes out on the R, G, and B lines. When AHZ = 0, the lines are blanked. The start and stop times of AHZ establish the left and right edges of the display. The trailing edge of AHZ flags a LINESTOP interrupt to the Task Scheduler.

The LINESTOP interrupt signals the VSDD to generate a burst of 12 refresh cycles to the DRAM and, if AVZ is active, to commence construction of the next scan line.

Figure 21 shows the vertical timing signals VSYNC and AVZ. The width and period of VSYNC are programmable, as are the start and stop times of AVZ. The start and stop times of the AVZ establish the top and bottom margins of the display.

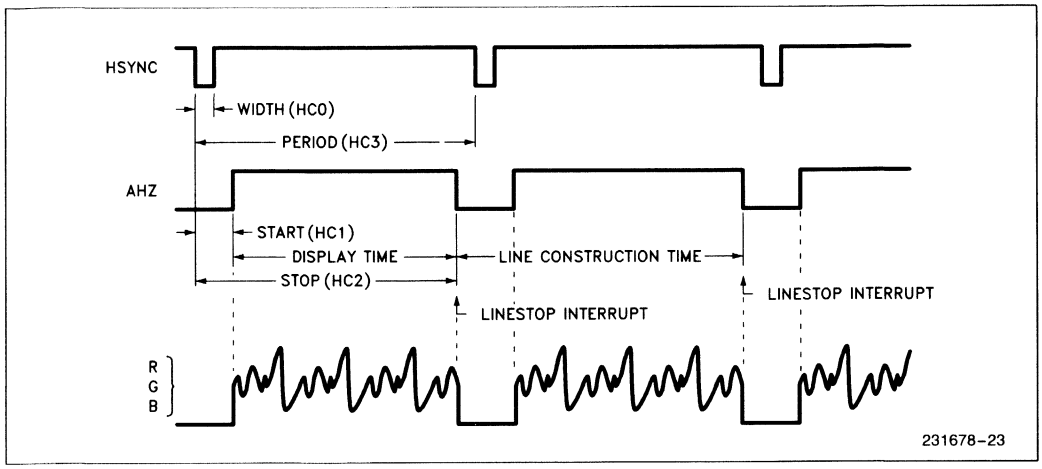


Figure 20. Horizontal Timing

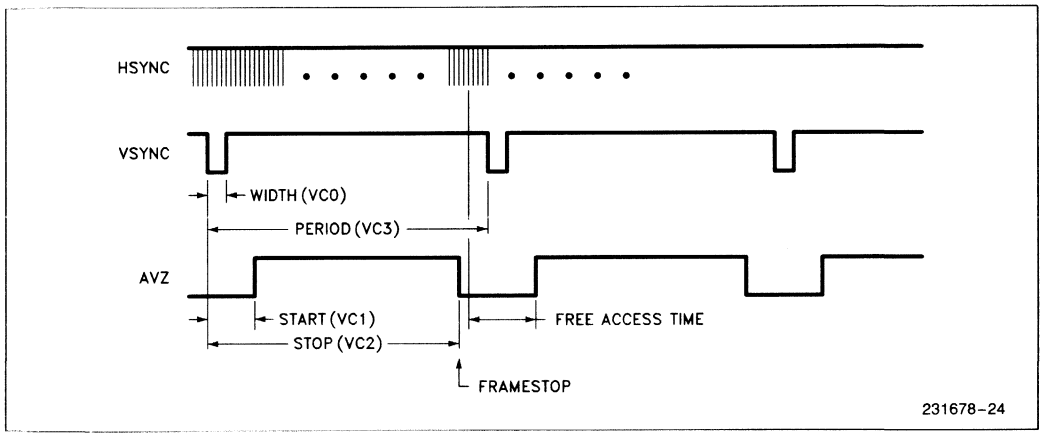


Figure 21. Vertical Timings

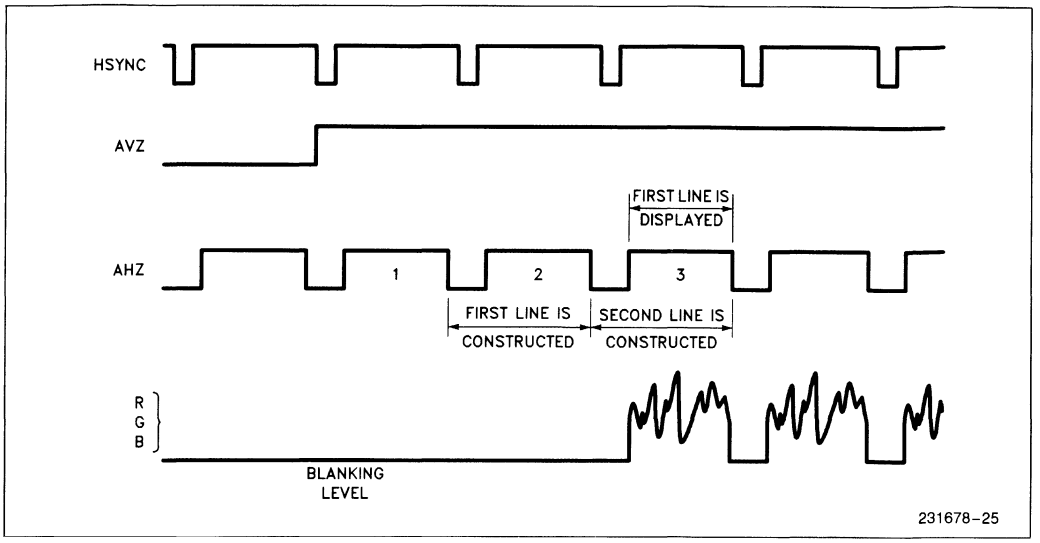


Figure 22. First Display Lines at the Beginning of the Active Vertical Zone

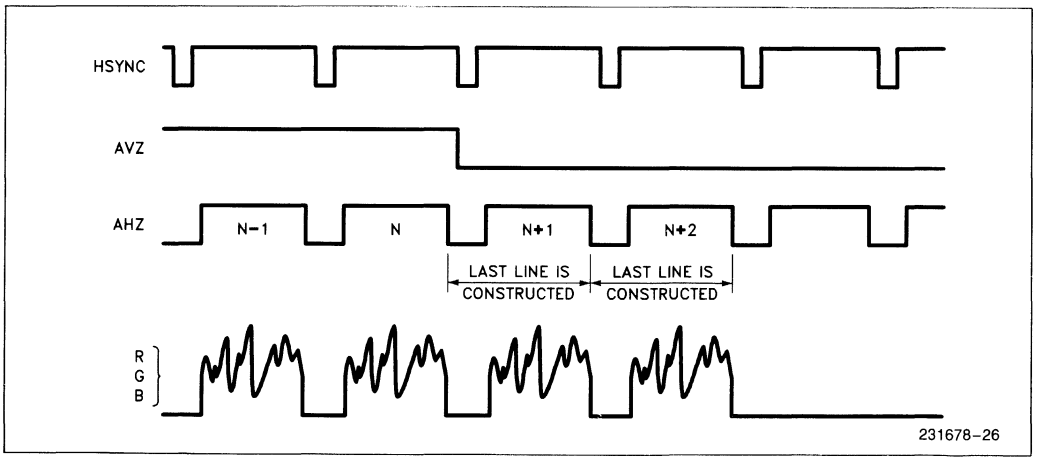


Figure 23. Last Display Lines at the End of the Active Vertical Zone

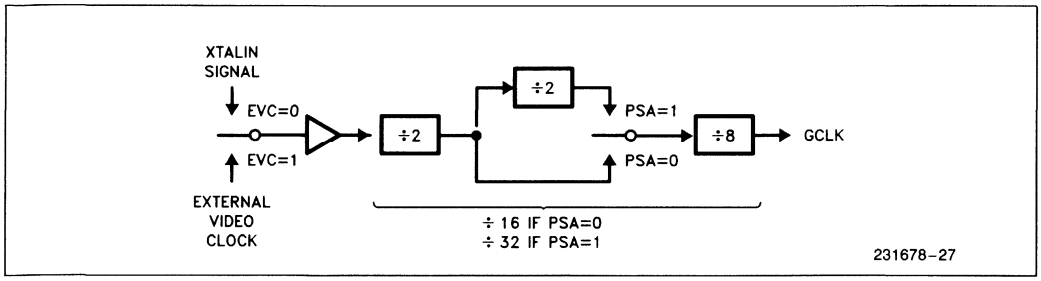


Figure 24. Derivation of GCLK for Horizontal Timings

At the beginning of the Active Vertical Zone the sync and timing signals are as shown in Figure 22. Construction of the first line begins at the end of the first AHZ following the leading edge of AVZ. Construction of this line can continue until the end of the second AHZ, at which time the Task Scheduler will execute a LINESTOP sequence and commence construction of the second line. The first line will then be displayed during AHZ number 3.

At the end of the Active Vertical Zone the sync and timing signals are as shown in Figure 23. Construction of the last line begins at the trailing edge of AHZ number N, and is displayed during AHZ number N + 2.

After the last line has been displayed, the VSDD executes a FRAMESTOP sequence, during which it reinitializes various data entry points and internal registers. During the execution of this sequence, LINESTOP interrupts are disabled. Consequently the DRAM refresh cycles are suspended during this time. These considerations were discussed in conjunction with the MIU description (DRAM Refresh).

Following the execution of the FRAMESTOP sequence, and for the rest of the time that AVZ is not active, DRAM refresh resumes but line construction does not.

Programmable Screen Constants

The timings are programmed by assigning values to the screen constants HC0 through HC3 (horizontal timings), and VC0 through VC3, (vertical timings). The screen constants and what they program are shown below:

Screen Constant	What It Programs
HC0	Width of HSYNC
HC1	AHZ Start Time
HC2	AHZ Stop Time
HC3	Horizontal Sweep Time
VC0	Width of VSYNC
VC1	AVZ Start Time
VC2	AVZ Stop Time
VC3	Vertical Sweep Time

The time base for the horizontal screen constants is the period of a signal named GCLK, which is derived from the video clock as shown in Figure 24.

The control bit PSA (PreScaler Active) selects whether the period of GCLK is 16 or 32 periods of the video clock. If PSA = 0, a GCLK period is 16 video clock periods. If PSA = 1, it is 32 video clock periods. For example, with a 70 ns video clock and PSA = 0, the basic time unit for the horizontal constants is 16 × 70 = 1120 ns.

The constants HC0 through HC3 are programmed in units of GCLK periods, offset by 1. That is, if the programmed constant is 0, the actual time programmed is 1 GCLK period. If the programmed constant is 1, the actual time is 2 GCLK periods, etc.

The width of the field reserved for these constants is 6 bits. Therefore, the maximum value any of the horizontal constants can have is 63, which means the maximum value that any horizontal timing can have is 64 GCLK periods. Thus with a 70 ns video clock and PSA = 0, the horizontal timings are programmable from 1.12 μs to 71.68 μs in steps of 1.12 μs. HC3 = 56 would give a horizontal period of 57 × 1.12 = 63.84 μs.

For internal hardware requirements, the horizontal constants should be programmed such that they satisfy the following relation:

$$HC0 < HC1 < HC3/2 < HC2 < HC3$$

The vertical screen constants VC0 through VC3 are programmed in units of horizontal lines, offset by 1; that is, if the programmed constant is 0, the actual time is 1 horizontal line duration, etc.

The width of the field reserved for these constants is 10 bits. Therefore the maximum value any of the vertical constants can have is 1023, which means the maximum value that any vertical timing can have is 1024 line times. VC3 = 261 (decimal) would give a 262-line field.

The vertical constants should be programmed to meet the following relation:

$$VC0 < VC1 < VC2 < VC3$$

Interlaced Mode

The 82716 will generate an interlaced raster if its control bit INL is set (1). In an interlaced raster, each frame consists of two fields: an even field and an odd field. The vertical sync pulse following the odd field is delayed by 1/2 line length, so that the lines of the even field fall in between the lines of the odd field on the screen. The VSDD displays the same information on both fields.

The interlaced mode is supported in order to be compatible with the broadcast standards.

To obtain, for example, a 525-line interlaced raster, one would program the Programmable Sync Generator (PSG) to give a 262-line field and set the INL bit.

Composite Sync Mode

The 82716 contains a control bit named SM (Sync Mode) that can be used to change the functionality of the HSYNC pin. If SM = 0, the HSYNC and VSYNC pins emit the HSYNC and VSYNC signals, respectively. If SM = 1, the VSYNC pin emits the VSYNC signal, and the HSYNC pin emits the logical XOR of HSYNC and VSYNC for use as a composite sync signal.

External Synchronization

A control bit named MAS (Master) allows the user to select between internal and external synchronization. If MAS = 1, the 82716 generates sync pulses for the CRT and emits them at the HSYNC and VSYNC pins, which in this mode are configured as outputs. If MAS = 0, the device configures itself to accept externally generated synchronization.

If the external sync signals are separate, HSYNC and VSYNC, they are applied to the corresponding pins of the 82716, which are both configured as inputs in this case (MAS = 0, SM = 0). If they are composite, the composite sync signal is applied to the HSYNC pin of the 82716. In this case (MAS = 0, SM = 1), the HSYNC pin is configured as an input and the VSYNC signal, as an output. It will emit the VSDD's VSYNC signal.

To use external synchronization the PSG in the 82716 must be programmed to agree with the external sync source in the width and period of HSYNC and VSYNC, in the selection of separate or composite mode for these signals, and in the selection of interlaced or non-interlaced raster. In addition, the video clock oscillator must generate a voltage-controlled frequency, so that it can close a phase-locked loop (PLL). The frequency range of this VCO needs to include the frequency for which the PSG can generate a horizontal period exactly equal to the external sync source's horizontal period.

With these requirements satisfied, external synchronization is performed on two levels: the frame level and the pixel level. At the frame level, a 1-to-0 transition in the external VSYNC resets the VSDD to the top of its own field. At the pixel level, a 1-to-0 transition in the external HSYNC causes the VSDD to sample its own internal HSYNC to determine if its video clock is fast or slow. If the internal HSYNC is at 0, the video clock is judged to be fast. If the internal HSYNC is at 1, the video clock is judged to be slow. A correcting signal is emitted at PLLCTL (PLL Control).

The correcting signal at PLLCTL is binary. If the video clock is slow, PLLCTL emits a 1. If the video clock is fast, PLLCTL emits a 0. This binary signal can be used to control the frequency of the VSDD's video clock oscillator.

For example, the frequency of a crystal oscillator in parallel resonance can normally be pulled over a range of about 0.03% by changing the load capacitance seen by the crystal. This means, with reference to Figure 8(a), showing the VSDD system oscillator in operation, by changing the values of C1 and/or C2. The pull range is about 0.3% if a ceramic resonator is used instead of a quartz crystal. In any case the PLLCTL signal can be used to switch additional capacitance into the circuit (to lower the frequency) or to remove it (to raise the frequency).

This type of control results in the video clock frequency constantly being in the process of shifting towards a value that is either above or below the target frequency, such that it hovers at the correct frequency.

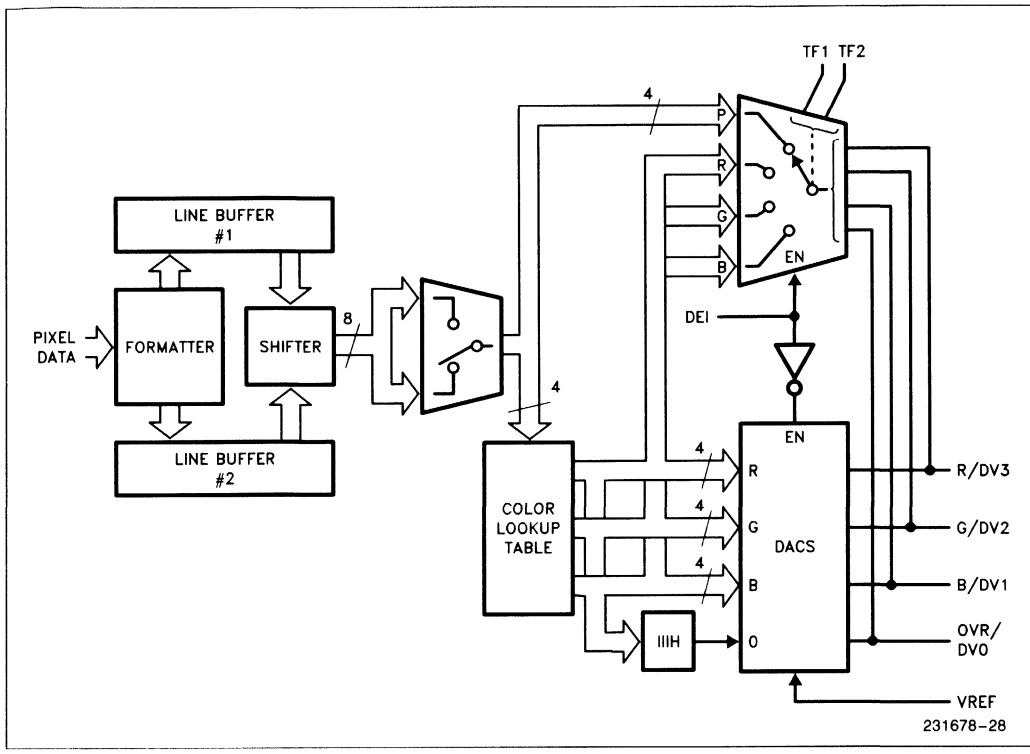


Figure 25. Pixel Unit

Pixel Unit

The Pixel Unit is a group of functional blocks on the 82716 which convert the display data stored in external memory into RGB signals. It contains basically the Formatter, two Scan Line Buffers, the Color Lookup Table, and the RGB DACs, as shown in Figure 25.

The Formatter receives raw data from external memory and performs whatever conversions and manipulations are necessary to load pixel specifications into the correct locations in the Line Buffers. This process is controlled by PLAs driven by parameters from the Video Configuration Registers and the Object Descriptor and Character Attribute fields. (Descriptors and Attributes are discussed with VSDD Data Structures.)

The Line Buffers are implemented on-chip in dynamic RAM of sufficient capacity to store data for two complete scan lines. Each buffer consists of forty 64-bit words, each word representing at 4 bits/pixel a sequence of 16 pixels. At 4 bits/pixel, then, each line buffer can hold up to 640 pixels.

The Number of Pixels in a Line

The line buffer has the capacity to hold, at the user's selection, up to 640 pixels at 4 bits/pixel or up to 320 pixels at 8 bits/pixel.

Four bits/pixel is the selection of choice if the display requires more than 320 pixels/line. This is called the High Resolution Screen mode. In this mode the 4 bits/pixel select one of 16 colors from the Color Lookup Table. This mode of operation is selected by setting the HRS (High Resolution Screen) bit to 1.

Higher **color** resolution is available using the 8 bits/pixel mode with external color tables and DACs. To enable the 8 bits/pixel mode the HRS bit is cleared. To allow the use of external color tables, the internal color table and DACs are bypassed by setting the DEI (Digitally Encoded Information) bit and two other bits that will be discussed presently. The RGB and OVR pins then emit pixel codes digitally, 4 bits at a time, 8 bits/pixel. The data can be demultiplexed using the timing signal available at the CKIO pin. (See Output Options.)

CRT displays of standard resolution are obtained by clearing both the HRS and DEI bits. With HRS = 0 each pixel uses 8 bits of the line buffer (but not necessarily 8 bits of the external memory). Clearing DEI enables the internal Color Lookup Table and DACs, which use only the lower 4 bits of each pixel specification. Lines can be up to 320 pixels long.

Thus the maximum line length is 320 pixels with HRS = 0 and 640 pixels with HRS = 1. The actual number of pixels on a line can be calculated as the duration of the Active Horizontal Zone (AHZ) multiplied by the pixel rate. The pixel rate is the video clock frequency if HRS = 1. If HRS = 0, the pixel rate is ½ that frequency. For example, if the video clock is 14.5 MHz and the duration of the AHZ is 40 μs, then there will be 580 pixels/line if HRS = 1, and 290 pixels/line if HRS = 0.

Since the Programmable Sync Generator uses the video clock for its time base, the number of pixels per line can be calculated directly from the constants that program the AHZ. The duration of the AHZ is

$$(HC2 - HC1) \times 16 \times 2^{PSA} / f_{video} \text{ (sec/line)}$$

see Figure 24

and the pixel rate is

$$f_{video} \times 2^{(HRS - 1)} \text{ (pixels/sec)}$$

The number of pixels/line is the pixel rate times the duration of AHZ:

$$\text{no. of pixels/line} = (HC2 - HC1) \times 8 \times 2^{PSA} \times 2^{HRS}$$

Thus settings of PSA = 0, HRS = 0, and HC2 − HC1 = 40 give exactly 320 pixels/line.

Screen Boundaries

Horizontal positions along the scan line are indicated by an x coordinate. This is a 10-bit signed number in 2's complement format. The left edge of the screen is the x = 0 position. If HRS = 0, each increment in x represents a displacement of one pixel to the right. If HRS = 1, each increment in x represents a two-pixel displacement. The right edge of the screen is therefore at

$$x = (HC2 - HC1) \times 8 \times 2^{PSA}$$

The Line Buffers each have the capacity to hold pixels from x = 0 to x = 319.

Pixels that would fall to the left of x = 0 are not written into the Line Buffer. In fact, if the entire object falls to the left of x = 0, the object is not processed at all.

The VSDD will also not spend time fetching pixel data that falls beyond the screen boundary, but the x coordinate of the right edge depends on the system configuration [HC1, HC2, PSA, SB3–SB9]. For simplicity, the VSDD does not compute this value. The CPU should compute it and write the upper 7 bits of its value into the "Screen Boundary" field in external memory. This will be described in Data Structures.

Note that only the upper 7 bits of this 10-bit number are specified to the VSDD. The VSDD will continue processing pixels out to x = this number with the lower 3 bits taken to be 1s. For example, if the right edge of the screen is at x = 290 = 0100100010B, the value written to the Screen Boundary field would be 0100100B, and the Pixel Unit would continue processing pixels to x = 0100100111B = 295.

Transparent Pixels

The Formatter checks each pixel for transparency before writing it into the Line Buffer. Pixels that are "transparent" are simply not written into the buffer. The buffers will retain the previous pixel data. The method of indicating transparency depends on whether the object type is bitmapped or character.

If the object is bitmapped, and its TDE (Transparency Detect Enable) bit is set (1), then transparency is indicated by the pixel code being all 0s. If TDE = 0, then the pixel data is written into the Line Buffer even if it is all 0s. Thus if TDE = 0, an all 0s code can be a valid entry to the Color Lookup Table and in that case will produce whatever color mixture is specified at that location in the table.

There are several ways of indicating pixel transparency in character objects, depending on how the characters are stored in external memory. These will be described in Data Structures.

Color Lookup Table

The on-chip Color Lookup Table contains 16 color mixtures of 12-bit definition at 4 bits per RGB color. These color mixtures are addressed by the output of whichever of the two Line Buffers is being currently displayed via the video shifter as shown in Figure 26(a). As noted, pixel codes may be 4 bits wide or 8 bits wide. If they're 8 bits wide, only the low 4 bits are used by the Color Table.

The color palette in external DRAM consists of 16 entries. Each entry is 16 bits long with the lowest 4-bits specifying the address of the entry in the CLUT and the upper 12 bits specifying the color as shown in Figure 26(b). Four bit pixel codes are used to address the CLUT. The pixel-code is matched with the lowest four bits of the CLUT entry and the pixel is given the color specified by the upper 12 bits. The color corresponding to the address 0010B is reserved for the background. The first step taken by the VSDD during line construction is to fill the Line Buffer with this code, so that unspecified pixels always come out in background color.

The Color Table is rewritten to the chip from external memory during the FRAMESTOP sequence to allow the user to make real time changes in it if desired. In fact, just by changing the Color Table Base Address in external memory, one can select an entirely different table.

On-Chip Color Lookup Table (CLUT)

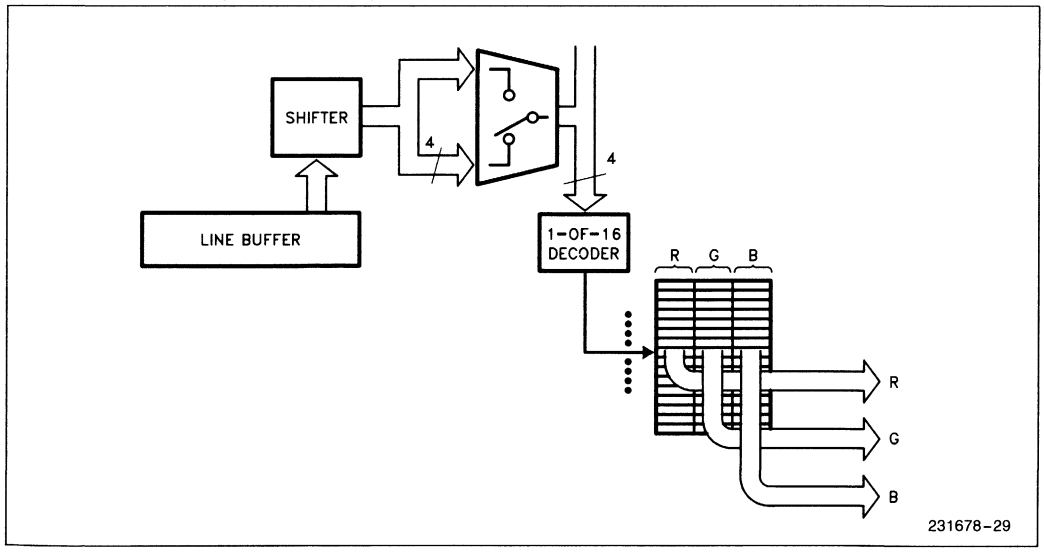


Figure 26(a). Color Lookup Table

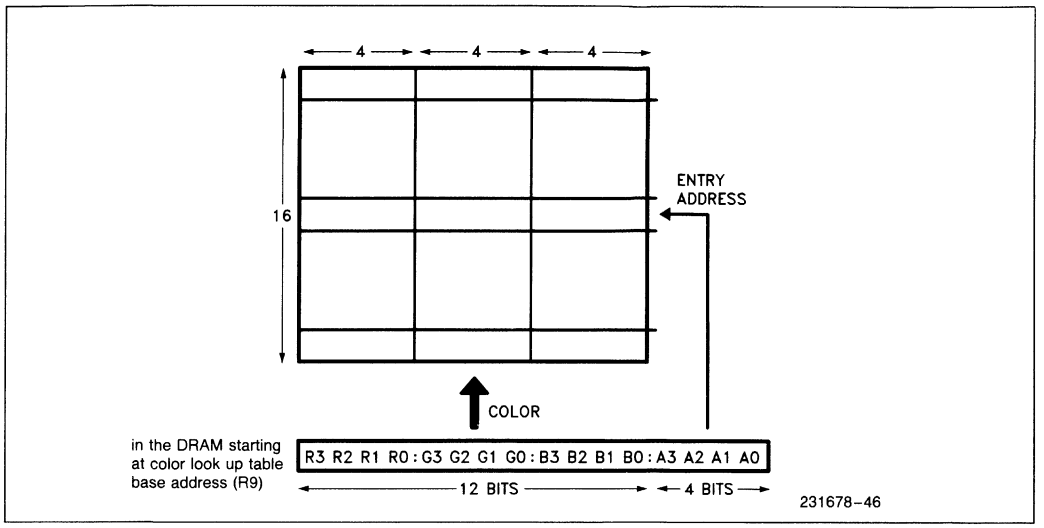


Figure 26(b). Filling the CLUT

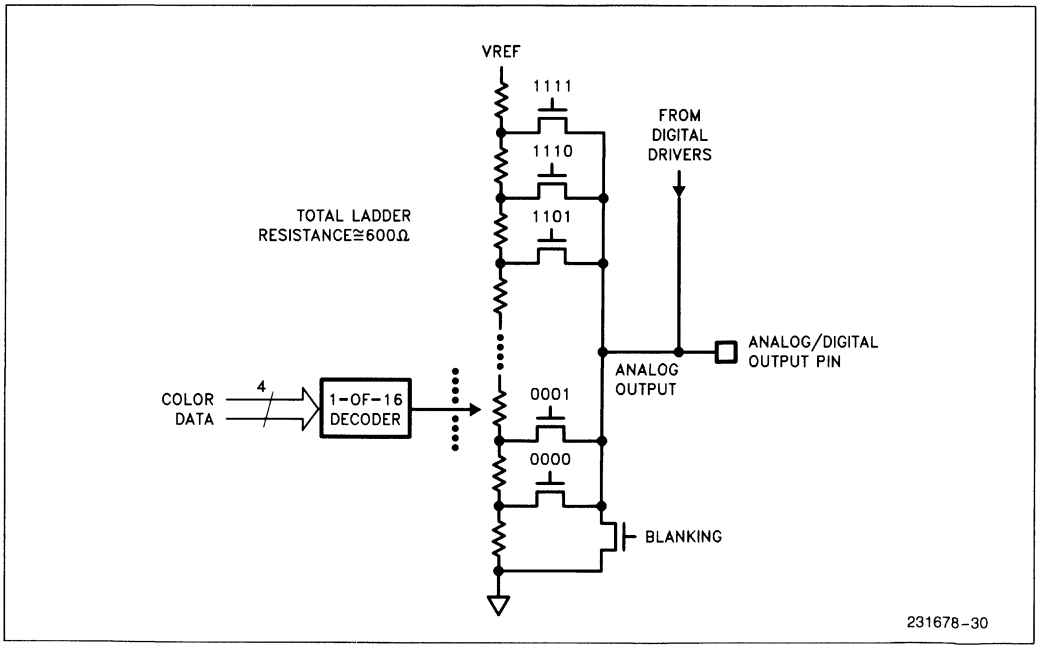


Figure 27(a). Digital-to-Analog Converter Circuit

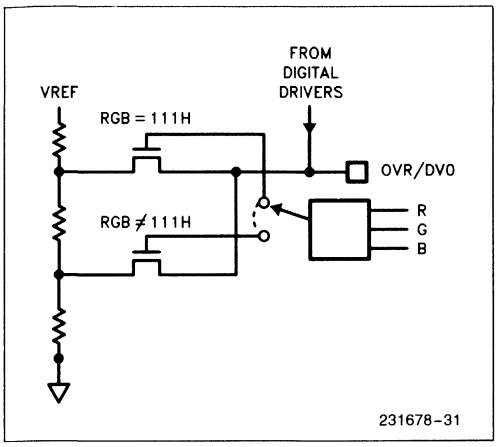


Figure 27(b). Overlay DAC

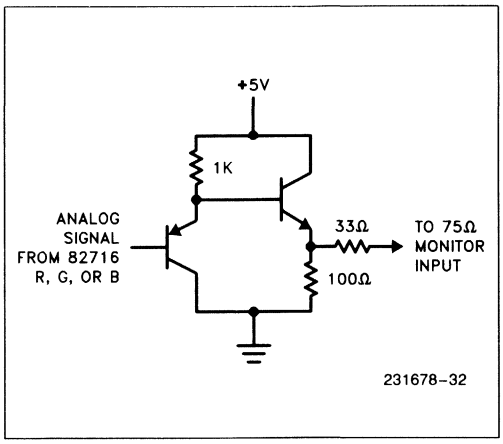


Figure 28. Buffering 82716 Analog Output to Low Input Resistance Monitor

DACs

The D/A Converters receive digital data from the Color Lookup Table and convert it to analog signals. There are four DACs: one for each of the three primary colors, plus one to emit the OVR (Overlay) signal. The DACs are implemented as diffused resistor ladders tapped at appropriate levels to provide a linear conversion function based on an externally supplied reference voltage (VREF). A schematic diagram of one of the DACs is shown in Figure 27a. As they have high output resistance, external circuitry is required to interface them to low impedance monitor inputs, as suggested in Figure 28.

The Red, Green, and Blue DACs each receive 4-bit codes from their respective fields in the Color Lookup Table. The Overlay pin can be used to overlay external video signals. The OVR DAC is controlled by the outputs of the Color Lookup Table. Whenever the color being displayed is RGB = 111H (0001 0001 0001B), the OVR pin emits the DAC output at "white" level (0FH or 1111B). Any other color will cause the OVR pin to go to "black" level (00H or 0000B). See Figure 27(b).

In a system where VSDD generated video will overlay video from an external video source, the "background" palette location 0010B would be programmed with 111H. Then, whenever the background color is displayed, user-supplied logic will switch in the external video source.

Output Options

As noted above, the DACs are enabled by clearing the DEI bit, in which case the RGBO pins emit analog signals. Setting the DEI bit bypasses the DACs, so the RGBO pins emit digital data. In this case the R, G, B, and OVR outputs become DV3, DV2, DV1 and DV0 digital outputs respectively.

There are two other bits, TF2 and TF1, which determine what type of digital data is emitted in the case where DEI = 1. If TF2 and TF1 are both set, then the color table is disabled, and the digital data that comes out are the pixel codes directly from the Line Buffers. The other three combinations of TF2 and TF1 result in the Color Table not being disabled. In these cases the digital data that comes out are the 4-bit color codes from one of the three color fields in the Lookup Table.

The output options are summarized in the table below:

DEI	TF2	TF1	Outputs	Signals
0	X	X	Analog	RGBO
1	0	0	Digital	Reds Only
1	0	1	Digital	Greens Only
1	1	0	Digital	Blues Only
1	1	1	Digital	Pixel Code

The digital single-color modes are implemented mainly for testing purposes, but they can also be used to facilitate hard copy processing. In this application the system can be programmed to emit reds during one frame, greens during the next, and blues during the third.

As has already been noted, the digital pixel code outputs facilitate high color resolution at 8 bits/pixel when external color tables and DACs are used. This mode can also be useful for hard copy processing.

The digital output modes emit data 4 bits at a time. If HRS = 1, the four bits represent a complete pixel or color code, and are valid at the falling edge of the signal at CKIO, as shown in Figure 29.

In 8 bits/pixel mode (HRS = 1), the data is available in two 4-bit nibbles. Low nibble always precedes the high nibble. VSDD provides the CKIO signal to latch low and high nibbles using off-chip edge detectors. See Figure 30.

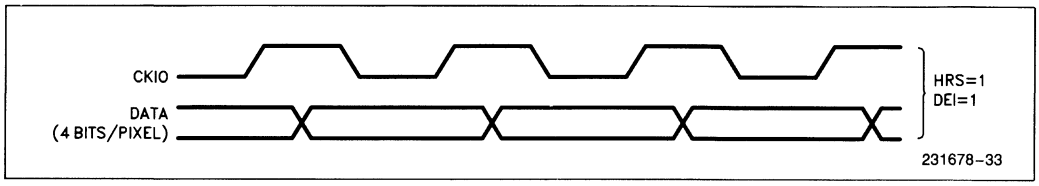


Figure 29. Digital Output Data, 4 Bits/Pixel (HRS = 1)

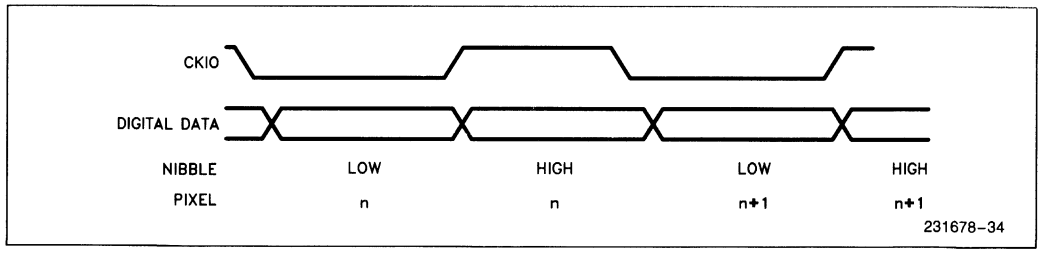


Figure 30. Digital Output Data, 8 Bits/Pixel (HRS = 0)

PART III: DATA STRUCTURES

Memory Mapping

The VSDD can support up to 512K bytes of DRAM. The DRAM is organized by the VSDD as 256K words of 16 bits for its own access. The CPU accesses this memory through the Bus Interface Unit of the VSDD using a 16-bit address and a programmable chip select, A16. The VSDD allows the CPU to access 512K bytes of DRAM via memory mapping. There are four banks of 128K bytes each in the DRAM. Even byte-addresses are in the lower half of a word and odd byte-addresses are in the upper half. The CPU also programs the two bank select bits.

The VSDD provides two windows to the CPU to map portions of the CPU's address space into portions of the VSDD's byte-address space. These are the Data Window and the Register Window.

As shown in Figure 31, information written into the Data Window maps into the VSDD's Data Segment, and information written into the Register Window maps into the VSDD's Register Segment. The Register Segment is for system configuration data, and the Data Segment is for other types of data.

Both windows are relocatable to the CPU, and their locations are programmed by the CPU. The Register Segment is not relocatable to the VSDD—it always occupies the lowest 32 bytes in DRAM. The Data Segment is relocatable to the VSDD—its base address and length are specified by the CPU.

Data Window

The VSDD provides a Data Window Base Address register of 5 bits that can be written into by the CPU to define the Data Window base address. Call these bits W16 through W12. These define the CPU's Data Window to begin at

W16	W15	W14	W13	W12	0	0	0	0	0	0	0	0	0	0	0	0
A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

in the CPU's address space. W16 is programmed to work with the chip select, A16. A16 is active low. Thus W16 must be set to 0 to select the VSDD.

Addresses within the Data Window will be mapped by the VSDD into its Data Segment, which is relocatable within the VSDD's 512K byte-address space. The Data Segment Base Address is specified by the CPU by writing 7 bits into the Data Segment Base Address register. The two highest bits in this register are the Bank Select bits BS1 and BS0. Call the other bits S16 through S12. These define the VSDD's Data Segment to begin at

BS1	BS0	S16	S15	S14	S13	S12	0	0	0	0	0	0	0	0	0	0	0	0
A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

in the VSDD's 512K byte-address space.

The VSDD also provides a means of specifying the length of the Data Window in bytes. The Data Segment that it maps into is of course the same length. The length is specified by writing 5 bits into the Data Window Length Mask (L16 through L12) as follows:

L16	L15	L14	L13	L12	Data Window Length
1	1	1	1	1	4K bytes
1	1	1	1	0	8K bytes
1	1	1	0	0	16K bytes
1	1	0	0	0	32K bytes
1	0	0	0	0	64K bytes

There are some restrictions on where the Data Window and Data Segment can be located for a given Window Length. To understand these restrictions it is necessary to understand how the VSDD recognizes that an address generated by the CPU is within the Data Window, and how it maps these addresses into its own Data Segment.

The VSDD compares CPU address bits 16 through 12 with Data Window Base Address bits W16 through W12. If all 5 pairs match, and the Length Mask is all 1s, then this address is recognized as being within the Data Window. If the Length Mask is set for 8K (L12 = 0), then a match between W12 and CPU address bit 12 is not required. If the other 4 pairs match, then this address is recognized as being within the Data Window. If the Length Mask is set for 16K, then only bits W16 through W14 need to match. The logical operation is shown in Figure 32.

If the CPU address is within the Data Window, then it must map into the VSDD's Data Segment, which begins at the Data Segment Base Address. If the Length Mask is all 1s, then the highest 7 VSDD byte-address bits are BS1, BS0, and S16 through S12. The lower 12 byte-address bits are the same as the CPU's. If the Length Mask is set for 8K (L12 = 0), then the highest 6 VSDD byte-address bits are BS1, BS0, and S16 through S13. The lower 13 byte-address bits are the same as the CPU's. If the Length Mask is set for 16K, then the lower 14 VSDD byte-address bits are the same as the CPU's. The logical operation is shown in Figure 33.

The basic restriction, therefore, is that if the Data Window length is 4K bytes, then the Data Window and Data Segment base addresses can be on any 4K boundary. If the Data Window length is 8K bytes, then the base addresses have to be on 8K boundaries. If the length is 16K bytes, the bases have to be on 16K boundaries, etc.

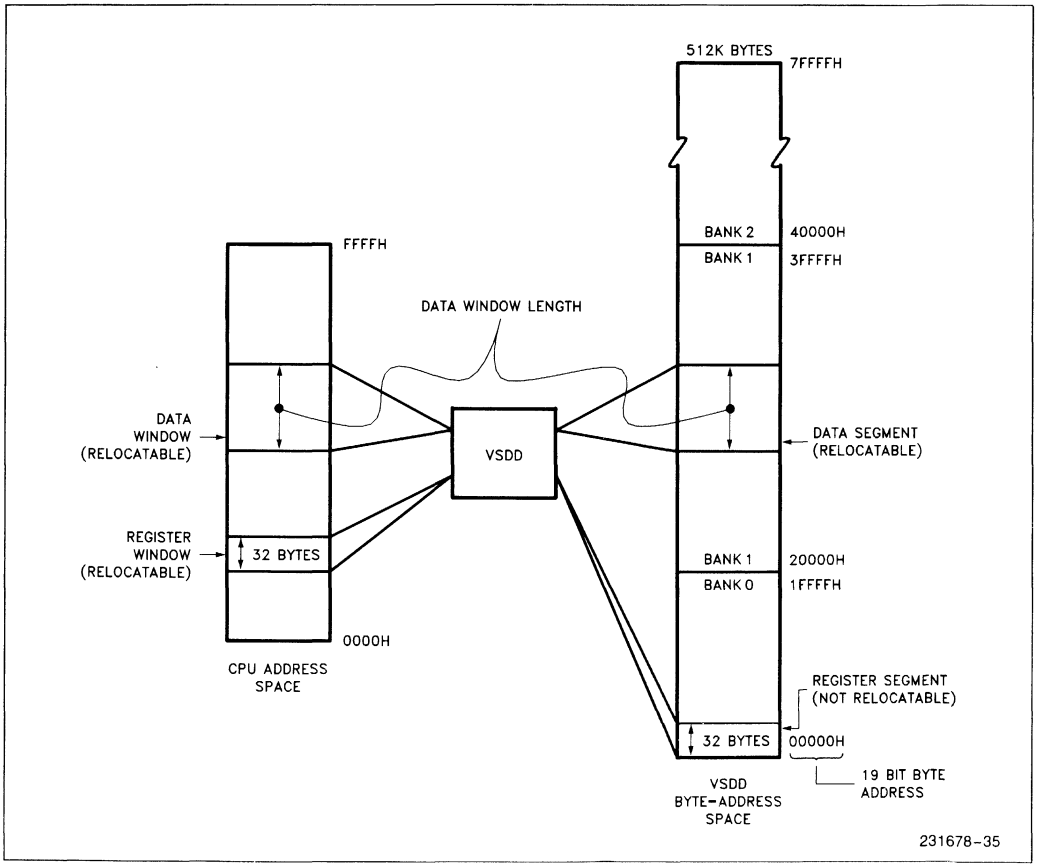


Figure 31. Memory Mapping in the VSDD

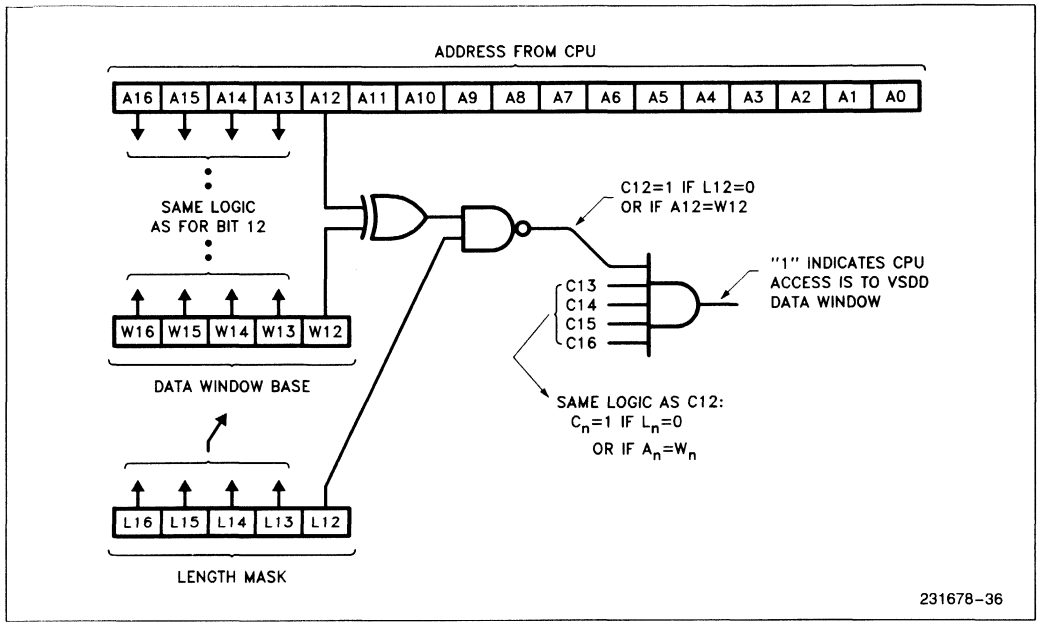


Figure 32. How the VSDD recognizes that a CPU access is to the VSDD Data Window. Thus a window length of NK bytes requires that the Window Base be on an NK boundary.

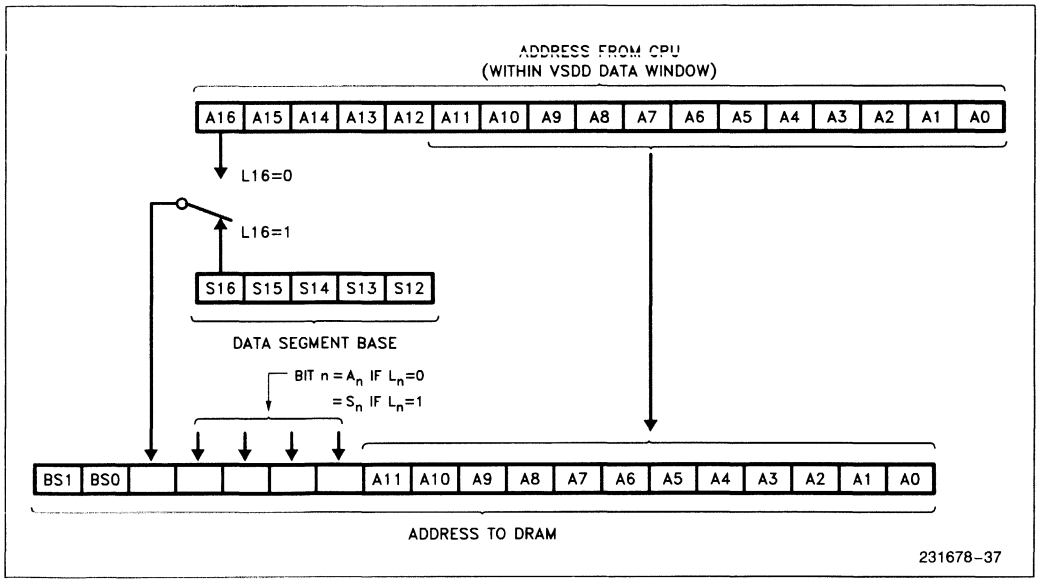


Figure 33. How the VSDD uses the Length Mask register to generate the Data Segment address (byte-address in DRAM) from the Data Segment Base and the address from the CPU.

Register Window

The Register Segment is a block of DRAM which contains 32 bytes of system configuration data. The VSDD locates this block in byte-addresses 0 to 1FH. The CPU accesses these registers through its Register Window. The VSDD provides a Register Window Base Address register of 12 bits that can be written into by the CPU to define the Register Window base address. These bits, R16 through R5, define the base address to be

R16	R15	R14	R13	R12	R11	R10	R9	R8	R7	R6	R5	0	0	0	0	0
A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

in the CPU's address space.

Addresses within the Register Window, i.e. having A16 through A5 equal bit by bit to R16 through R5, will be mapped by the VSDD into the Register Segment of the DRAM.

If the Data Window and Register Window overlap, then accesses to the overlapped area are directed to the Register Segment.

Register Segment

The lowest 16 words are reserved for registers R0 through R15. These registers contain system configuration information: control registers, base addresses, bits for switch options, etc. These registers have hardware counterparts on the VSDD chip. At the end of each Active Vertical Zone, the VSDD reads register R0 to an internal register on the chip. If bit UCF (Update Control Flag) in R0 is set, the other registers are also re-written to the chip. The registers are listed below.

			DRAM Loc.
R15	Horiz. Constant 3	Vert. Constant 3	1EH
R14	Horiz. Constant 2	Vert. Constant 2	1CH
R13	Horiz. Constant 1	Vert. Constant 1	1AH
R12	Horiz. Constant 0	Vert. Constant 0	18H
R11	Access Table Base Address Counter		16H
R10	Character Base Address		14H
R9	Color Table Base Address		12H
R8	Access Table Base Address		10H
R7	Object Descriptor Table Base Address		0EH
R6	Priority Access Quantity		0CH
R5	Data Segment Base Address		0AH
R4	Data Length Mask		08H
R3	Data Window Base Address		06H
R2	Register Window Base Address		04H
R1	Video Configuration Register 1		02H
R0	Video Configuration Register 0		00H

NOTE:
Where zeros are shown in register locations, 0 must be written to those bits in order to ensure proper operation and upward compatibility with any future versions of this chip.

Following is a description of the information contained in each of these registers.

R0: Video Configuration Register 0

Duty Cyc	Blink Rate	DS1	DS0	DOF	HRS	DEN	SAB	DEI	UCF

Blink Rate of selected object is set from 8 frames to 256 frames in multiples of 8 frames. For 50 Hz/60 Hz, this translates into blink rate increments of 160 ms/133 ms starting from 6.2 Hz/7.5 Hz (code 00000) down to 0.20 Hz/0.23 Hz (code 11111).

Duty Cycle. The duty cycle of the blink rate can be selected as below:

111	always on	
110	12.5% off	87.5% on
101	25.0% off	75.0% on
100	37.5% off	62.5% on
011	50.0% off	50.0% on
010	62.5% off	37.5% on
001	75.0% off	25.0% on
000	87.5% off	12.5% on

DS1, DS0, and **DOF** tell the 82716 how the external DRAM is configured. DS1 and DS0 give the address range of the DRAM chips. For 16K DRAMs, DOF tells if it's organized bit-wide or nibble-wide.

DS1	DS0	DOF	DRAM Configuration	Byte Capacity	ADDR Pins Used		
					Row	Col	Bank Select
0	0	0	16K × 1	32K	0–6	0–6	(none)
0	0	1	16K × 4	128K	0–7	0–5	6, 7 at $\overline{CAS}$
0	1	0	64K × 1	128K	0–7	0–7	8 at $\overline{RAS}$, $\overline{CAS}$
0	1	1	64K × 4	512K	0–7	0–7	8 at $\overline{RAS}$, $\overline{CAS}$
1	0	0	256K × 1	512K	0–8	0–8	(none)

HRS (High Resolution Screen) = 1 means that the maximum screen resolution is 640 pixels. If HRS = 0, the resolution is 320 pixels.

DEN (Display Enable) = 0 stops the 82716 from generating scan line data. This is so the system can be initialized before it starts displaying uninitialized data. It can also be used to shut the display off temporarily during times when the CPU wants to have full access to the DRAM. DEN is set (1) to enable the display.

SAB (Slow Access Bit) = 1 means that the slow DRAM can be used. If not (0), then fast DRAM can be used.

DEI (Digitally Encoded Information) = 1 means that the RGB and OVR outputs are digital. If DEI = 0, then RGB and OVR are analog.

UCF (Update Control Flag): Register 0 is read during every FRAMESTOP sequence. If UCF = 1, then the other Registers are also read and the information used to update the VSDD. If UCF = 0, only the configuration parameters in Register 0 are updated (and, of course, the Access Table Base Address, which is reinitialized during every FRAMESTOP sequence).

During system initialization, the first access to R0 must write a 0 to the UCF bit, so that the Register Segment can be correctly defined before allowing the VSDD to initialize its configuration parameters. There is more about system initialization following the section on Data Structures.

R1: Video Control Register 1

Char Height	INL	MAS	SM	TMM	TMS	0	EVC	PCE	FAE	RE	PSA	PRE

Char Height is 4 bits encoding the number of scan lines per character. The number is encoded as a simple unsigned binary integer, except that 0000 means 16. If the character height is programmed to be 10, 12, 14, or 16 lines, then characters of double height (20, 24, 28, or 32 lines) can also be defined, as will be described in the section on character objects.

INL (Interlaced) = 1 causes the device to adjust its sync signals so as to generate an interlaced raster.

MAS (Master) = 0 means the device locks onto external sync signals via PLL.

SM (Sync Mode) = 1 enables the HSYNC pin to operate in composite sync mode. That means the vertical and horizontal sync pulses both come out on the HSYNC pin.

TMM, TMS (Twin Mode Master, Twin Mode Slave) enable a pair of 82716s to operate in "twin mode". In this mode two VSDDs generate alternate lines of the same display (using separate DRAMs). For synchronization purposes, one is the master (TMM = 1, TMS = 0) and the other is the slave (TMM = 0, TMS = 1). The master displays the even lines while the slave displays the odd lines. TMM = 1, TMS = 1 is illegal.

The combination TMM = 0, TMS = 0 means twin mode operation is not in use.

EVC (External Video Clock) = 1 configures the CKIO (Clock I/O) pin as an input for an externally generated video clock signal. EVC = 0 enables the video clock to be derived from the XTALIN signal.

PCE (Priority Counter Enable) = 1 enables the VSDD to limit the number of CPU accesses to DRAM that it will allow during scan line building, until the line currently being constructed is completed. The limiting number is the value written into the low 4 bits of Register 6 ("Priority Access Quantity"). The way it limits CPU access is by using the Ready signal. Therefore the mode requires FAE = 0 (see below), and that the CPU have a true Ready input.

FAE (Free Access Enable) = 1 enables the RDY pin to operate in the Free Access signal mode instead of as a Ready signal.

RE (Read Enable) flag, when set (1) enables the CPU to read data from the display memory through 82716. If not set (0), the output buffers of the VSDD are disabled, thus preventing CPU from reading the DRAM. Upon Reset, this flag is set to zero.

PSA (PreScaler Active) defines the relationship between the video clock frequency and the Sync Generator clock frequency. If PSA = 0, then GCLK is $1/16$ the frequency of video clock. If PSA = 1, then GCLK is $1/32$ the frequency of video clock. If the video clock exceeds 16 MHz the PSA bit must be set (1).

PRE (Pipeline Read Enable) = 1 enables the pipeline read mode: CPU read cycles always return data from the previous read cycle.

R2: Register Window Base Address

Register Window Base Address: R16–R5	O	TF2	TF1	ME

R16–R5 bits specify the Register Window base address. This is the window through which the CPU accesses the Register Segment of the DRAM.

TF2, TF1 (Test Flags): If DEI = 1, then these flags determine what type of digital output is emitted. The output options are summarized in the table below:

DEI	TF2	TF1	Outputs	Signals
0	X	X	Analog	RGBO
1	0	0	Digital	Reds Only
1	0	1	Digital	Greens Only
1	1	0	Digital	Blues Only
1	1	1	Digital	Pixel Code

ME (Margin Enable) = 1 generates a margin display in background color. This is intended for applications with standard TV sets, in which line timing precision is limited and varying amounts of overscan are used. If ME = 0, the RGB signals are blanked outside the Active Horizontal Zone. If ME = 1, they're driven to background color between GCLK step 7 and the start of the AHZ, and between the end of the AHZ and GCLK step 57. If the Programmable Sync Generator is set up to generate standard TV timings (AHZ starts at GCLK step 12 and stops at GCLK step 52), then turning background color on at step 7 and off at step 57 gives 5 GCLK periods of margin at either end of the Active Horizontal Zone.

R3: Data Window Base Address

Data Window W16–W12					0	Screen Boundary SB9–SB3							0	0	0

W16–W12 bits specify the Data Window base address. This is the window through which the CPU accesses the Data Segment of the DRAM.

SB9–SB3 Screen Boundary bits specify the upper 7 bits of the 10-bit **x** coordinate of the right edge of the screen. These bits and the need to specify them are discussed in conjunction with the Pixel Unit.

R4: Data Length Mask

L16	L15	L14	L13	L12	0	0	0	0	0	0	0	0	0	0	0

L16–L12 bits are the Data Window Length Mask, which specify the length of the Data Window in bytes, as follows:

L16	L15	L14	L13	L12	Data Window Length
1	1	1	1	1	4K bytes
1	1	1	1	0	8K bytes
1	1	1	0	0	16K bytes
1	1	0	0	0	32K bytes
1	0	0	0	0	64K bytes

R5: Data Segment Base Address

S16	S15	S14	S13	S12	0	0	BS0	BS1	0	0	0	0	0	0	0

S16–S12 bits specify the Data Segment base address in the VSDD's address space.

BS1–BS0 divide the 512K byte-address space into four banks, and should be considered part of the Data Segment Base address. Note however that only bitmapped object data can reside in banks 1 through 3. All other display data must be written to bank 0.

R6: Priority Access Quantity

0	0	0	0	0	0	0	0	0	0	0	0	P A Quantity

PAQ 4 bits indicate the maximum number of CPU accesses to the DRAM that are allowed during scan line building, if PCE bit (in R1) is 1.

R7: Object Descriptor Table Base Address

Object Descriptor Table Base: A15–A6	0	0	0	0	0	0

A15–A6 bits specify the Object Descriptor Table base address. This is a word-address. In the 256K word-address space, which requires 18-bit addresses, the two highest bits in the Object Descriptor Table base address, corresponding to A16 and A17, are 0. In other words, the Object Descriptor Table must reside in bank 0.

R8: Access Table Base Address

Access Table Base Address: B15–B0

B15–B0 bits specify the Access Table base address. This is a word-address. In the 256K word-address space, which requires 18-bit addresses, the two highest bits in the Access Table Base Address, corresponding to B16 and B17, are 0. In other words, the Access Table must reside in bank 0.

R9: Color Table Base Address

Color Table Base Address: D15–D0

D15–D0 bits specify the Color Table base address. This is a word-address. In the 256K word-address space, which requires 18-bit addresses, the two highest bits of the Color Table Base Address, corresponding to D16 and D17, are 0. In other words, the Color Table must reside in bank 0.

R10: Character Generator Base Address

0	0	0	0	0	0	0	0	Char Gen 1	Char Gen 2

Char Gen 1, Char Gen 2 bits specify bits 15 through 12 of the base addresses of the two character generators. Bits 16 and 17 are necessarily 0, meaning both character generators must be in bank 0. The remaining bits of these base addresses, bits 11 through 0, are 0. When a character generator is being accessed, bits 11 through 0 are filled in with slice number and ASCII code.

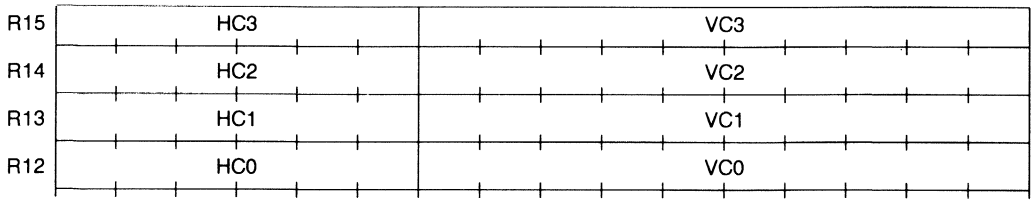
R11: Access Table Address Counter

Access Table Address Counter: C15–C0

C15–C0 bits point to the next Access Table entry to be used. C16 and C17 are of course 0, since the Access Table must reside in bank 0.

The Access Table is maintained by the VSDD, and should not be written to by the CPU. However, it can be **read** by the CPU, for the purpose of ascertaining which line is currently under construction.

R12–R15: Horizontal and Vertical Constants



HCn, VCn are screen constants described in conjunction with the Programmable Sync Generator.

Note that the 10-bit field for the vertical screen constants limits each of them to a maximum value of 1023, in effect limiting the vertical period to 1024 line-times, and imposing a practical limit to the number of lines in the raster.

The 6-bit field for the horizontal screen constants similarly imposes limitations on their values. The maximum horizontal sweep period is 64 GCLK periods.

Access Table

The Access Table contains the vertical positioning information for each object. The Table begins at the location designated by the Access Table Base Address register, R8 in the Register Bank. The Table contains one word in DRAM for each scan line in the Active Vertical Zone of the display.

The first line (at the top of the display) is associated with the word at the Table's base address. Within each word, bit number **i** is the Access Flag associated with object number **i** in the display and has the priority **i**.

Access Flag Register

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0

b0 is the access flag for object 0, b1 for object 1 etc.

The Access Flags indicate to the VSDD which objects are to be present on which lines of the display. If an Access Flag is set (1), then there is to be no change in the object's display status; that is, if the object did not appear on the previous line it will also not appear on this line. If the object's Access Flag is clear (0), the object's display status is reversed from what it was in the previous line. All objects are disabled at the end of the Active Vertical Zone.

An object is activated by putting a zero in the word corresponding to the scan line on which the object is first displayed. This turns on the object for all the following scan lines. The object is toggled off by putting a zero in the scan line following the last line of the object.

At the beginning of each frame, the VSDD writes the contents of Access Table Base Address, R8, into Access Table Counter, R11. At the end of each line this counter is read into the Access Table Entry Address Register (on the VSDD). See Figure 34. This entry address is then used to read the access flags for the line that is to be constructed. Access Table Entry Address is then incremented and written back into R11 preparing it for the next line.

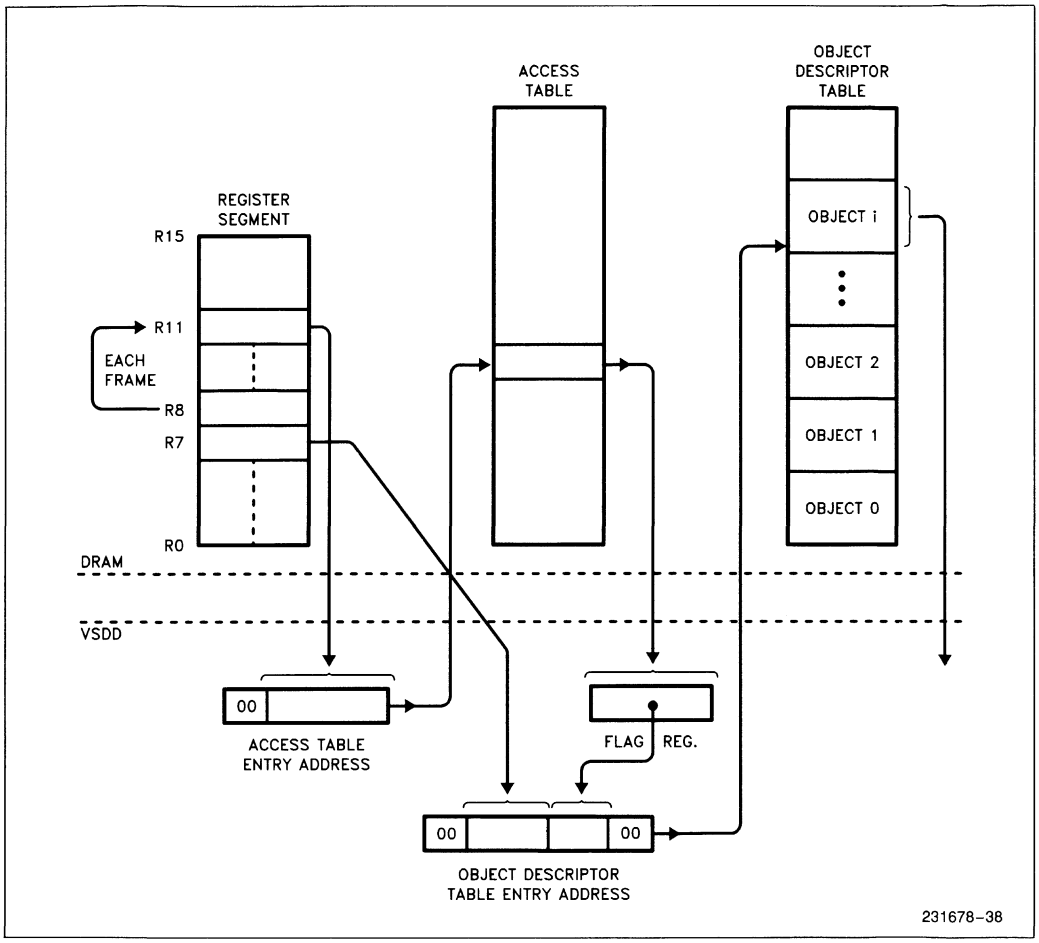


Figure 34. Addressing the Access Table and the Object Descriptor Table

Then the Access Flag Register is examined bit by bit to determine if there is a change in any object's display status. If object number **i** is to be displayed, then its Object Descriptor field is read. The base address for the Object Descriptor field for object number **i** is constructed from the Object Descriptor Table Base Address register, R7, by concatenating bits A15 through A6 from R7 with 4 bits representing the number **i** (i = 0 to 15), as shown in Figure 34. Then the Object Descriptor field base Address is

0	0	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	i3	i2	i1	i0	0	0

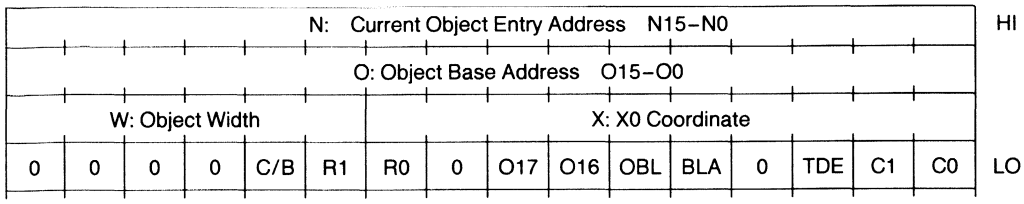
An Object Descriptor field is 4 words long, and all 4 words are read, so the last two bits in the above address are incremented to get all 4 words.

Note that the Access Table and the Object Descriptor Table base addresses are 18-bit word-addresses, the two highest bits in both addresses being 0s.

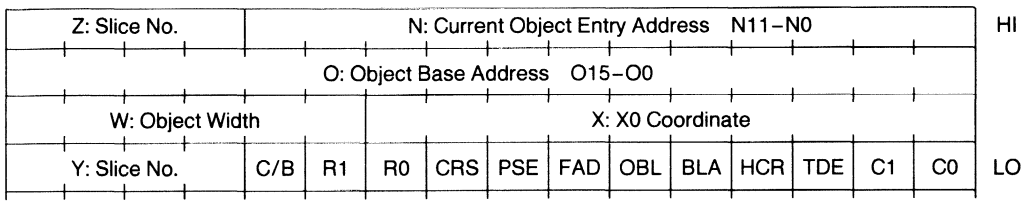
Object Descriptor Table

The Object Descriptor Table contains a 4-word Object Descriptor field for each object in the display. There are two types of objects: bitmapped and character. Their descriptor fieds are as shown below.

Bitmapped



Character



The meanings of these attributes are as follows:

Bitmapped Objects

N: **Current Object Entry Address** is the address of the pixel data for the next scan line of the object. At the beginning of each frame, the VSDD copies Object Base Address into this field. This is maintained by the VSDD and should not be altered by the CPU, except upon initialization when the object base address should be written in.

O: **Object Base Address** points to the beginning of the object's data base. Word-addresses are 18 bits long, and only 16 are specified here. The two highest bits are O17 and O16, which are specified elsewhere in the Object Descriptor Table.

W: **Width** indicates how wide the object is in 4-word units in DRAM. When object data is read, it's always read in blocks of 4 words. 000001 specifies a width of 1 "64-bit word", 111111 a width of 63 "64-bit words". What the width corresponds to in terms of distance along a scan line depends on how many pixels are stored in that number of 4-word units. Pixels in external memory may be 2, 4, or 8 bits wide, so a 4-word unit may contain 32, 16, or 8 pixels.

X: **X0 Coordinate** is a 10-bit signed number (2's complement) encoding the horizontal position of the leftmost pixel in the object. X0 = 0 is the left edge of the screen. Incrementing X0 by 1 moves the object 1 pixel to the right if HRS = 0, 2 pixels to the right if HRS = 1. X0 can be −512 to +511. Pixels to the left of x = 0 or to the right of the screen boundary are not displayed. If an object begins beyond the screen boundary value in R3, the object will not be processed.

Bit Descriptors for Bitmapped Objects

C/B: Character/Bitmap Specifier = 0 indicates the object type is bitmapped.

R1–R0: Resolution: For a bitmap object, these 2 bits specify how many bits each pixel takes up in external memory, as shown in the table below.

Bitmapped Objects (C/B = 0)

HRS	R1	R0	Bits/Pixel
0	0	0	do not use
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	do not use
1	0	1	do not use
1	1	0	2
1	1	1	4

Note that in the Line Buffers each pixel takes up either 4 bits (HRS = 1) or 8 bits (HRS = 0). Two-bit pixels from external memory are extended to four bits in the Line Buffer by prefixing them with 2 bits (C1, C0) from the Object Descriptor. If HRS = 0, four-bit pixels are extended to 8 bits in the Line Buffer by prefixing them with 0s.

O17–O16: For bitmapped objects, these are the highest two bits of the Object Base Address. The only display data that can occupy banks 1 through 3 are bitmapped objects.

OBL: Object Blinker = 1 causes the object to blink. The blink rate and duty cycle are specified in R0 in the Register Segment.

BLA: Blanker = 1 turns the object off.

TDE: Transparency Detect Enable. If TDE = 1, then pixels that are encoded as all 0s are not written into the Line Buffers. The buffers will retain the previous pixel data. Thus a low priority object will be visible through the transparent pixels of a higher priority object. When TDE = 0, then pixels that are encoded as all 0's are written into the line buffer. "0000B" is then one of the 16 color codes.

C1–C0: Default Color Specification. For bitmapped objects that are stored in external memory in the 2 bits/pixel mode, these two bits extend the pixel specification to 4 bits.

Character Objects

N: **Current Object Entry Address** is the low 12 bits of the beginning address of the current line of text. The upper 6 bits are 00 concatenated with "O15–O12". Thus a character object cannot extend across a 4K word boundary.

When the bottom slice of this line of text is completed, N is updated by the 82716 to point to the next line of text. The update is N: = N + 4 × W. Lowest 12 bits of Object Base Address are written into N at the end of each frame.

Z: **Current Slice Number** of the characters in this object. This field is concatenated with a byte ASCII code to form part of the address used in accessing the character generator. The bottom of the character is slice number 0, and the top is slice number = character height − 1 (from Register R1). As each slice is displayed, Z is decremented. When it underflows, it is reloaded with the character height −1 and the N field is updated. During the FRAMESTOP sequence, Z is initialized to Y.

O: **Object Base Address** points to the beginning of the object's data base. Word-addresses are 18 bits long, and only 16 are specified here. For character objects the two highest bits are 00.

W: **Width** indicates how wide the object is in 4-word units in DRAM. When object data is read, it's always read in blocks of 4 words. What the width corresponds to in terms of distance along a scan line depends on how many characters are stored in that number of 4-word units, and how wide they are.

X: **X0 Coordinate** is a 10-bit signed number (2's complement) encoding the horizontal position of the leftmost pixel in the object. X0 = 0 is the left edge of the screen. Incrementing X0 by 1 moves the object 1 pixel to the right if HRS = 0, 2 pixels to the right if HRS = 1. X0 can be −512 to +511. Pixels to the left of x = 0 or to the right of the screen boundary are not displayed.

Y: **Start Slice Number** is the number of the first (topmost) character slice of this object to appear in a frame. The system CPU need only modify this field to produce a scrolling effect in the display of the text. Y = 0 is the bottom of the character and Y = Character height −1 (defined in R1) is the top.

Bit Descriptors for Character Objects

C/B: Character/Bitmap Specifier = 1 indicates the object type is character.

R1–R0: Resolution: For character objects, R1 and R0 specify the width of the character, as shown in the table below.

Character Objects (C/B = 1)

HRS	R1	R0	Pixels/Char
0	0	0	6
0	0	1	8
0	1	0	12
0	1	1	16
1	0	0	16
1	0	1	6
1	1	0	8
1	1	1	12

FAD: Full Attribute Definition = 1 means character descriptions are 3 bytes long. Each character is encoded as an ASCII byte plus a 2-byte attribute word. FAD = 0 means character descriptions are 1 byte long. Each character is encoded as an ASCII byte.

HCR: High Color Resolution. If FAD = 1, then HCR = 1 means use 16-color palette for characters and their backgrounds. If FAD = 1, then HCR = 0 means use 8-color palette. If FAD = 0, then HCR is a don't-care.

CRS: Conceal/Reveal/Select. If FAD = 1, then CRS = 1 enables the MSK bit in the character's attribute word to cause the character to be concealed. If FAD = 0, then CRS selects one of two character generators. CRS = 0 selects CGBA0 as specified in the register segment. CRS = 1 selects CGBA1.

PSE: Proportional Spacing = 1 enables proportional spacing of characters.

OBL: Object Blinker = 1 to blink. The blink rate and duty cycle are specified in R0 in the Register Segment.

BLA: Blanker = 1 turns the object off.

TDE: Transparency Detect Enable. If TDE = 1, then pixels that are encoded as all 0s are not written into the Line Buffers. Thus they are transparent. If TDE = 0, then pixels that are encoded as all 0s are written into the Line Buffers (not transparent).

C1–C0: Default Color Specification. For character objects that are stored in external memory in the 1 byte/character mode, these two bits become the MSBs of the foreground/background color specifications for the characters, as shown below.

foreground color = C1 C0 0 1
background color = C1 C0 0 0

(Foreground color means the color of the character itself. Background color can be made transparent by selecting C1 = C0 = 0 and TDE = 1.)

Object Data

Objects are rectangular windows on the screen. The X0 Coordinate, specified in the X field in the object's Descriptor Table, defines the position of the left edge of the object for every scan line in which the object appears.

Object data begins at the Object Base Address specified in the "O" field (with O17, O16, for bitmap objects) of the Descriptor Table. The length of the object data file depends on the object's height and width on the screen, and how densely the data is stored. The minimum length of an object data file in DRAM is 4 words, and the length is always a multiple of 4 words.

The object width, as displayed on the screen, is given in 4-word units by the W field in the Object Descriptor. For example, if the Width field contains 001010, then the object is ten 4-word units wide. The VSDD will read in $10 \times 4 = 40$ words of object data for each scan line in which this object appears. In order to have a specific width number to refer to, we'll use this hypothetical width of 40 words in the object data discussions to follow.

Bitmap Objects

The beginning address of each block of 40 words is constructed from the N field in the object's Descriptor Table. Between frames the N field is initialized to the Object Base Address, "O". As each block of words is read, the N field is updated for the next scan line.

The width of an object in pixels depends on how many pixels are contained in a word. Pixels may be specified in 2, 4, or 8 bits/pixel, depending on bits R1 and R0 in the Object Descriptor. Therefore, one 16-bit word may contain 8, 4, or 2 pixels. Since width is given in 4-word units, the minimum width specification would be 32, 16, or 8 pixels. If the object is ten 4-word units wide, and encoded as 4 bits/pixel, then it would be $10 \times 16 = 160$ pixels wide.

For bitmap objects the N field is 16 bits wide. At the beginning of each frame the "O" field is written into the N field, so the first scan line in which the object appears has data in it from locations "O" up to but not including "O + 4W". For example, suppose "O" = 1234H, O17 O16 = 01, and W = 10. Then the Object Base Address is 11234H, and the first scan line in which the object appears has data in it from locations

1st 4-word unit: 11234H, 11235H, 11236H, 11237H
2nd 4-word unit: 11238H, 11239H, 1123AH, 1123BH
. . . . . .
10th 4-word unit: 11258H, 11259H, 1125AH, 1125BH

Now the N field is updated for the next scan line, so the second scan line in which the object appears has data in it from locations "O + 4W" to . . . "O + 2(4W). In the above example the N field would be updated to 1125CH, and the second scan line in which the object appears would have data in it from locations 1125CH up to but not including 1125CH + (4 × 10) = 11284H. 11284H is what the N field would next be updated to.

Note that the highest 2 bits in each word-address, O17 and O16, are treated like bank select bits, and all the address updating is done on only the lower 16 bits. O17 and O16 are not included in the updating process. Therefore, the data file for a single bitmap object can't be allowed to cross a 64K-word boundary, or else there will be addressing errors.

Pixel specifications for bitmap objects are simply stored sequentially in each word in the object data file. If pixels are 4-bit specifications, then each word contains 4 pixels, the lowest nibble being the first one to come out on the line, and the highest nibble being the 4th one on the line. Two-bit and 8-bit pixels are treated similarly.

Figure 35 shows how Object Data is addressed for bitmap objects.

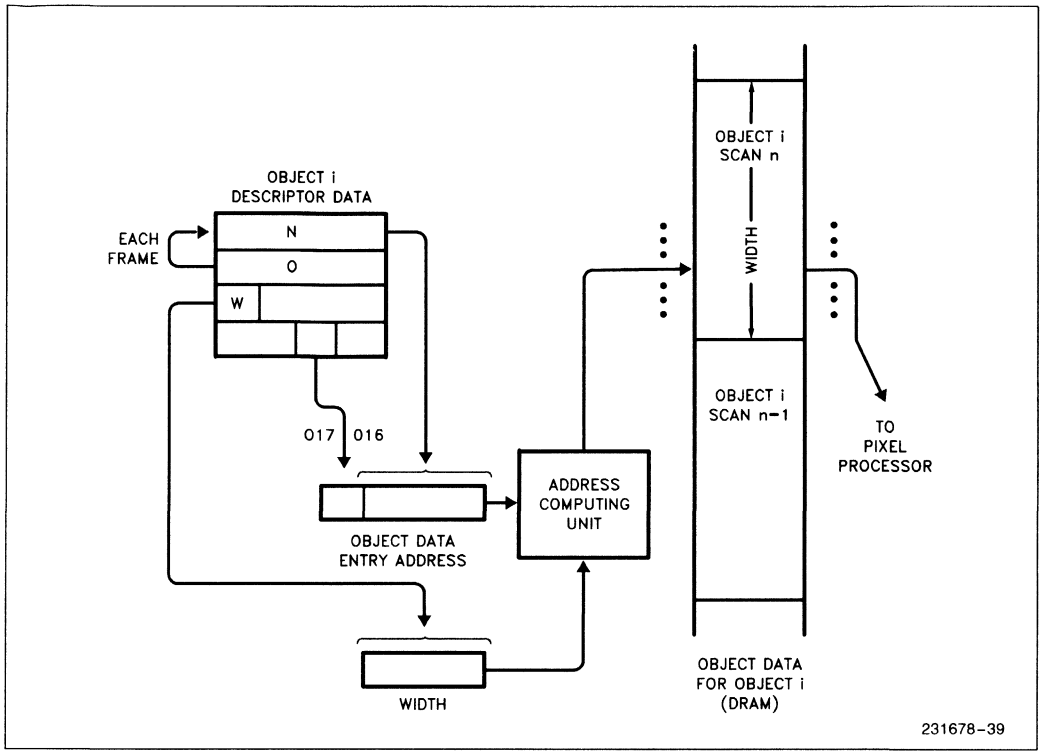


Figure 35. Addressing Bitmap Object Data

Character Objects

For character objects the N field is 12 bits wide. This field is used to construct the beginning address of a line of ASCII characters, as shown in Figure 36. The object itself may consist of many lines of characters. Each line of characters consists of individual scan lines, each scan line presenting one "slice" of the text characters. Slice numbers are kept track of in the Z field. Note that slice 0 is the bottom of the character. The top slice number is the character height. When the final slice of a line of characters has gone out on a scan line, the N field is updated to the next line of characters, which follows sequentially in DRAM. During the FRAMESTOP sequence, the lowest 12 bits of the "O" field are written into the N field.

The W field works the same way as for bitmap objects. For example, if the Width field contains 10, then the object is ten 4-word units wide. Whereas for bitmap objects a different 40-word block would be read for each scan line, for character objects the same 40-word block is read as many times as there are slices in the character height. That's because this 40-word block specifies ASCII characters, not pixel data. The pixel data is in the character generator. After the last slice of the current line of characters has been accessed, "N" field is updated to "N + 4W".

Characters are encoded either as plain ASCII (FAD = 0 in the object's descriptor field) or as ASCII with attributes (FAD = 1), as shown in Figure 36. If they're encoded just as plain ASCII, then each word holds two characters, and the unit of width is 4 × 2 = 8 characters. If they're encoded as ASCII with attributes, then

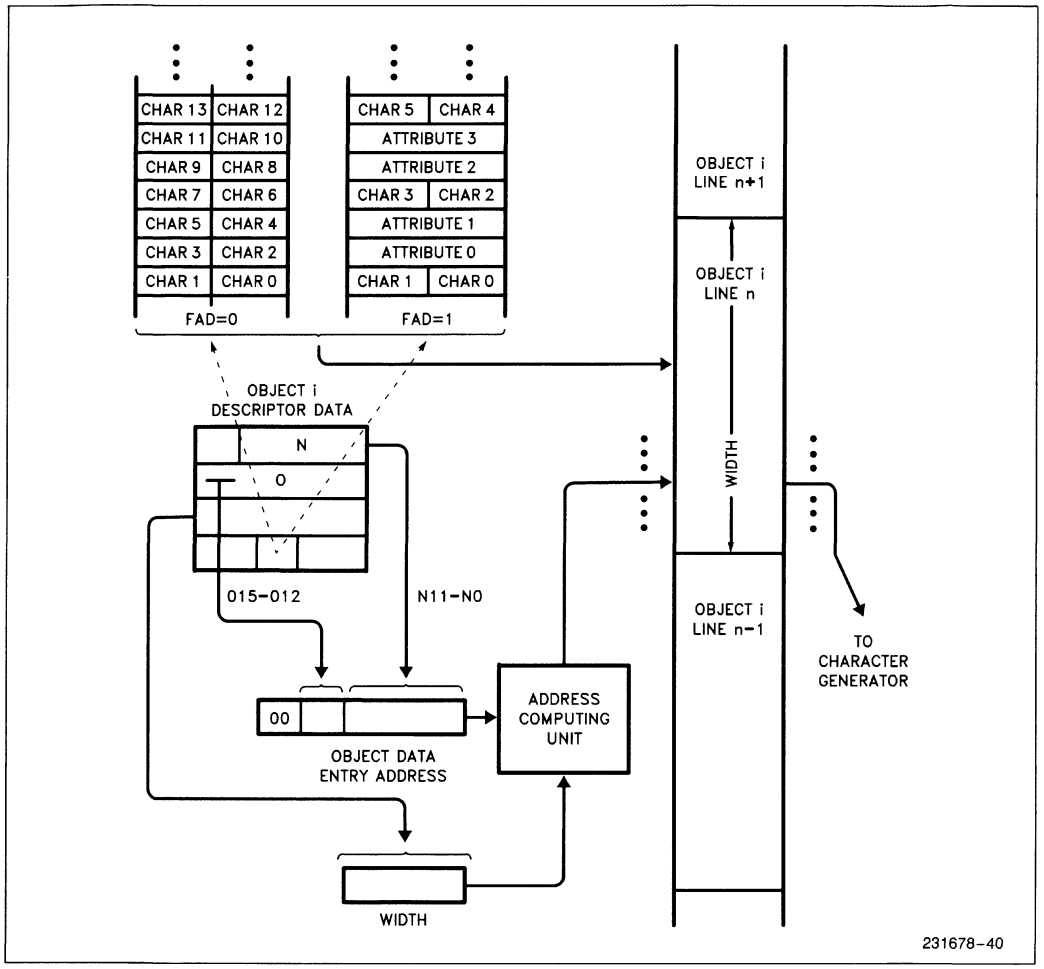
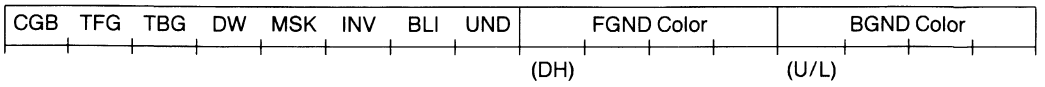


Figure 36. Addressing Character Object Data

three words hold two characters. The first of these three words contains the two ASCII-coded characters, and the next two words contain their attributes. The second word contains attributes for the character in the low byte of the first word, and the third word contains attributes for the character in the high byte. The attribute word is as follows:



where:

FGND Color: A character is a rectangular block of a certain color (background) with the character drawn inside it, in foreground color. The FGND field specifies the color table entry to be used as the Foreground Color.

BGND Color: Background Color specification.

CGB: Selects one of the two character generators. CGB = 1 selects CGBA1.

TFG: Transparent Foreground = 1 causes the foreground of the character to be transparent.

TBG: Transparent Background = 1 causes the background color of the character to be transparent. If the background is transparent, then objects behind the character are visible except where the character itself blocks them.

DW: Double Width = 1 elongates the character to twice its normal width on the screen. When this is done, the ASCII code and attributes corresponding to the next character are ignored. Effectively, then, this also causes an elongated character to take twice as much memory as a single-width character.

MSK: Mask = 1 causes the foreground color of the character to be the same as the background color (hence the character is invisible). But MSK has no effect unless the CRS bit (Conceal/Reveal/Select) bit is set (1).

INV: Inverted = 1 causes the character's background and foreground to exhcange colors.

BLI: Blinking = 1 causes the foreground to alternate between foreground and background colors. The blink rate is specified in R0.

UND: Underline = 1 causes Slice 1 of the character to be in foreground color.

DH: Double Height = 1 doubles the height of the character if the HCR (High Color Resolution) bit in this object's Descriptor Table entry is 0. DH is actually the MSB of the field reserved for FGND Color. HCR = 1 means the object is supposed to use a full 4-bit color specification, and there is in that case no Double Height attribute. HCR = 0 means the FGND Color is only a 3-bit specification, and the fourth bit is to be interpreted as DH.

U/L: Upper/Lower = 1 means this is the upper half of a double height character. U/L is actually the MSB of the field reserved for BGND Color, so it can't have this interpretation unless HCR = 0. And of course, unless DH = 1. When HCR = 0 and DH = 0, U/L must be zero.

For double height characters (HCR = 0, DH = 1), the character has to be specified twice: once for U/L = 1 and once for U/L = 0, to get both halves. And of course the two halves have to be correctly positioned, one above the other. Treat them like two separate characters.

Figure 36 shows how Object Data is addressed for character objects. Note that the Object Data entry address is formed of the 12-bit N field, concatenated with the 4 highest bits of the O field, and that the two highest bits in this address are 00. That means object data can't be allowed to cross a 4K word-address boundary, and that all character object data has to be in memory bank 0.

Character Generators

Given an ASCII code, the VSDD must decide which slice of the character to display and where to get it. Where it gets it from is one of two possible character generators.

A character generator consists of H blocks of 256 words, where H is the character height in scan lines. H, which is specified in R1 in the Register Segment, can be up to 16. Each block of 256 words contains one slice of each of the 256 characters.

The VSDD allows the simultaneous use of two independent character generators of 256 characters each. Bits 15–12 of their base addresses are specified in R10 in the Register Segment. The other bits in each base address are 0. Thus each character generator must begin on a 4K word-address boundary in memory bank 0.

If the characters are encoded in plain ASCII (FAD = 0), then the character generator is selected by the CRS bit in the Object Descriptor. If the characters are encoded with full attributes (FAD = 1), then the character generator is selected by attribute bit CGB.

Individual slices are addressed as shown in Figure 37. The slice address is formed by concatenating the four bits of the character generator base address with the slice number field, Z in the Object Descriptor, and the ASCII code, itself.

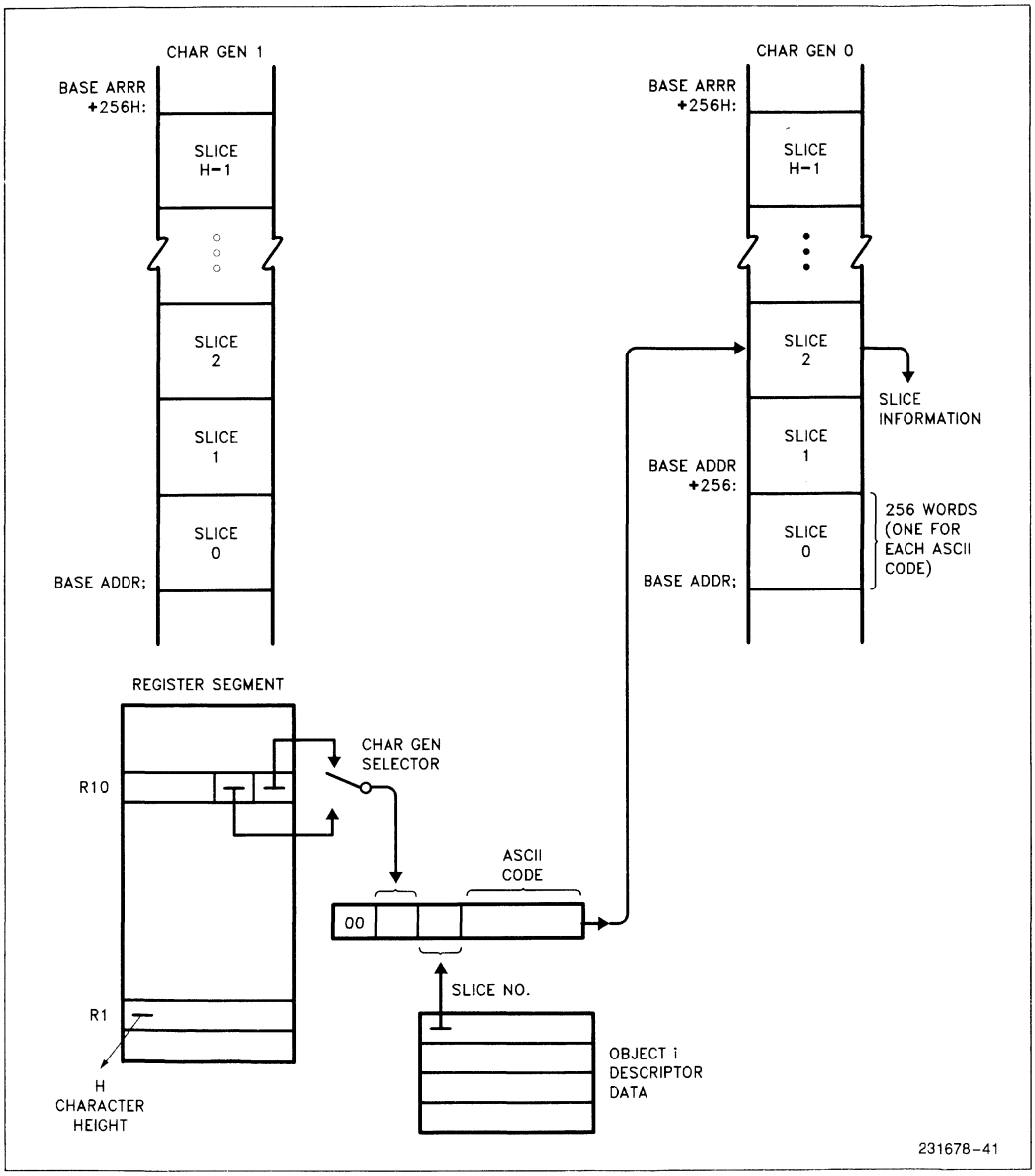


Figure 37. Addressing Character Slices

Slice number 0 is the bottom scan line for the character, and slice number H-1 is the top line. Each slice of each character occupies one word in DRAM. Within the word, the slice is encoded as a sequence of pixel bits, the leftmost pixel being the LSB in the word. If a pixel bit is 1, then the pixel is to be given foreground color. If a pixel bit is 0, the pixel is given background color. This is shown in Figure 38.

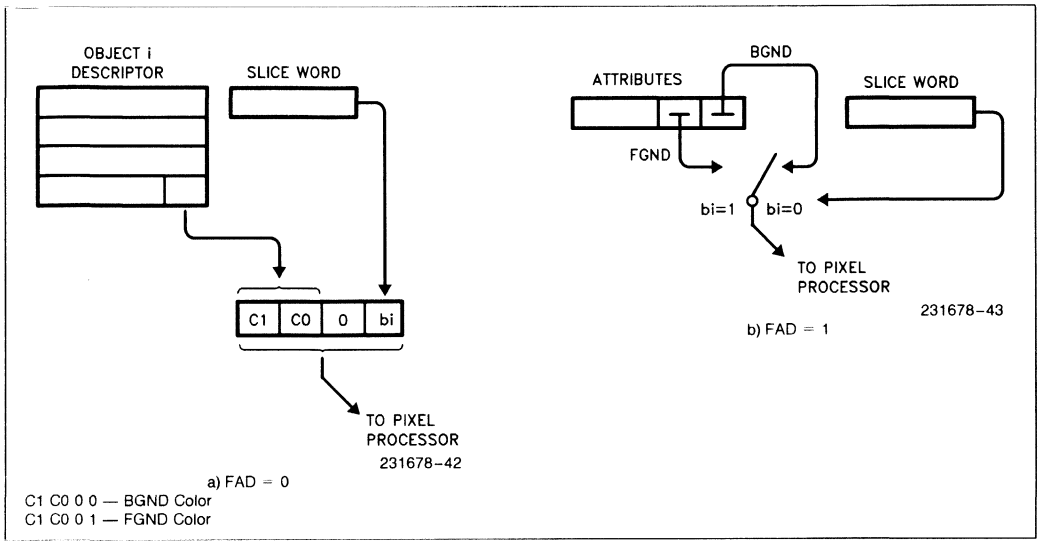


Figure 38. Character Generator Pixel Construction

Characters within an object can be either monospaced, meaning every character is the same width, or proportionally spaced, meaning the width depends on the character. The selection is made by setting or clearing the PSE bit in the Object Descriptor.

If PSE = 0, monospaced characters are selected. Monospaced characters can be 6, 8, 12, or 16 pixels in width, as selected by the R1, R0 bits in the Object Descriptor. In this case, each slice word contains 6, 8, 12, or 16 pixel bits, as shown below. Leftover bits are don't-cares, as far as the VSDD is concerned. But they should be programmed as zeroes for future compatibility.

Pixels/Char	Information in Slice Word															
6	x	x	x	x	x	x	x	x	x	x	p5	p4	p3	p2	p1	p0
8	x	x	x	x	x	x	x	x	p7	p6	p5	p4	p3	p2	p1	p0
12	x	x	x	x	p11	p10	p9	p8	p7	p6	p5	p4	p3	p2	p1	p0
16	p15	p14	p13	p12	p11	p10	p9	p8	p7	p6	p5	p4	p3	p2	p1	p0

Each bit specifies one pixel. If the bit = 1, a pixel of foreground color is specified. If the bit = 0, a pixel of background color is specified. Figure 39 shows how the pixels are stored for character "E" (ASCII Code = 45H). 12 pixel fixed character width is chosen. Character height is specified as 10 scan lines in register R1. Note that the left most pixel is stored in the LSB(p0) of the slice word. The most 4 significant bits of the slice word are don't cares. But they are programmed as zeroes. Figure 40 shows the actual output on the display.

	Char. Gen Address (Word-Address)						Pixel Storage												Hex Dump				
							P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0					
Slice #9 →	0	1	9	4	5	H	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	1	F	8	H
	0	1	8	4	5	H	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	1	F	8	H
	0	1	7	4	5	H	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	8	H
	0	1	6	4	5	H	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	8	H
	0	1	5	4	5	H	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	7	8	H
	0	1	4	4	5	H	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	7	8	H
	0	1	3	4	5	H	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	8	H
	0	1	2	4	5	H	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	8	H
	0	1	1	4	5	H	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	1	F	8	H
	0	1	0	4	5	H	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	1	F	8	H
Slice #0 ↑																							

12 Pixel Character Width
10 Scan Lines Character Height
Bank 0
Character Generator Base Address = 0 1 0 0 0 H

Figure 39. Bit Storage for Character "E"

Setting the PSE bit in the Object Descriptor specifies proportional spacing. In this case, individual characters may be 2, 4, 6, 8, 10, 12, or 14 pixels wide, depending on the character. The three highest bits in the slice word indicate the character width (the two highest bits, if the character width is 14 pixels). The pixel bits are specified as shown below. For example, if 12 pixel variable character width is selected, then in Figure 39, the most 3 significant bits in each slice word would be programmed as 101.

Width	Information in Slice Word															
2	0	0	0	x	x	x	x	x	x	x	x	x	x	x	p1	p0
4	0	0	1	x	x	x	x	x	x	x	x	x	p3	p2	p1	p0
6	0	1	0	x	x	x	x	x	x	x	p5	p4	p3	p2	p1	p0
8	0	1	1	x	x	x	x	x	p7	p6	p5	p4	p3	p2	p1	p0
10	1	0	0	x	x	x	p9	p8	p7	p6	p5	p4	p3	p2	p1	p0
12	1	0	1	x	p11	p10	p9	p8	p7	p6	p5	p4	p3	p2	p1	p0
14	1	1	p13	p12	p11	p10	p9	p8	p7	p6	p5	p4	p3	p2	p1	p0

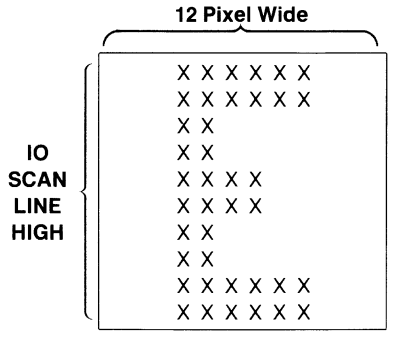


Figure 40. Character "E" on the Display

PART IV: INITIALIZATION

The RESET pin on the 82716 is active high. It is a high impedance input to a Schmitt Trigger. On power up, RESET should be held active long enough to allow the VSDD system oscillator (on XTALIN) to start, and then be held for a minimum of 20 clock periods with the oscillator running.

Reset Status

RESET puts the 82716 into a special initialization mode. When RESET is released, the device generates 12 sets of refresh cycles to the external DRAM. The user must then initialize R0 for continuous refresh cycles to commence. The first write to R0 loads the SAB bit. This data contained in the SAB bit position in R0, is then ignored until the next reset. The CPU should not try to access the DRAM for at least 10 μs after RESET has been released.

The status of the device at this time is as follows:

Control Bit	Reset Value	Effect
TMM TMS	0 0	Single Mode
DEN	0	No Display
DEI	0	Analog Mode
MAS	0	Sync Pins are in high impedance state
RE	0	CPU can Access DRAM only to Write to it
PCE	0	Priority Access Mode Disabled
FAE	0	RDY Pin Emits Ready Signal
EVC	0	Video Clock is from XTALIN Signal

DEN = 0 means no display is being generated. DEI = 0 means the device is in the analog mode, and, since DEN = 0, the R, G, B, and OVR pins are emitting retrace black.

MAS = 0 configures the sync pins as inputs. EVC = 0 would normally configure the CKIO pin as an output, but in the initialization mode CKIO is in a high impedance state.

Note that the device is at this time configured to emit a Ready signal at the RDY pin.

The device comes out of reset with the following default values for the screen constants:

HC0 = 1	HSYNC Width = 32 XTALIN Periods	2.24 μs
HC1 = 2	AHZ Start Time = 48 XTALIN Periods	3.26 μs
HC2 = 5	AHZ Stop Time = 96 XTALIN Periods	6.72 μs
HC3 = 8	HSYNC Period = 144 XTALIN periods	10.08 μs
VC0 = 1	VSYNC Width = 2 Lines	20.16 μs
VC1 = 2	AVZ Start Time = 3 Lines	30.24 μs
VC2 = 3	AVZ Stop Time = 4 Lines	40.32 μs
VC3 = 7	VSYNC period = 8 lines	80.64 μs

This is called the "fast frame" mode. Its main purpose is to give the PSG something to work with that is not contradictory (such as HC0 > HC3) until the CPU has written the correct values into DRAM. The timings listed assume a 70 ns clock (14.3 MHz) at XTALIN.

Most important to the initialization procedure are the reset values of the Register and Data Windows:

Register Window Base Address: 00400H

Data Window Base Address: (undefined)

Initialization Procedure

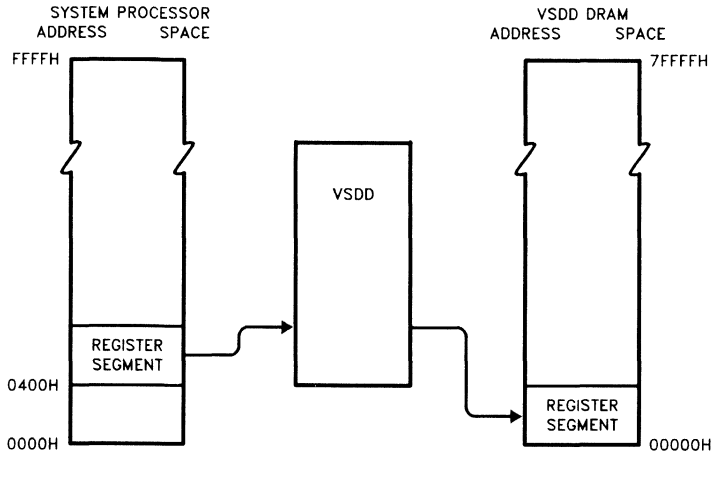
The first access must be a write cycle to Register R0 at 00400H. The written data can be 16 bits wide or 8 bits wide. If it is 8 bits wide, it should be to the low byte of the register, since these bits include the DRAM configuration bits DS1, DS0, and DOF. The VSDD needs these to be able to generate row and column addresses correctly, and will take them directly from the CPU during the first write.

The first write cycle should leave DEN and UCF at 0. UCF (Update Control Flag) should be left clear till the entire Register Segment has been initialized, lest the device "update" its on-chip control bits with random data.

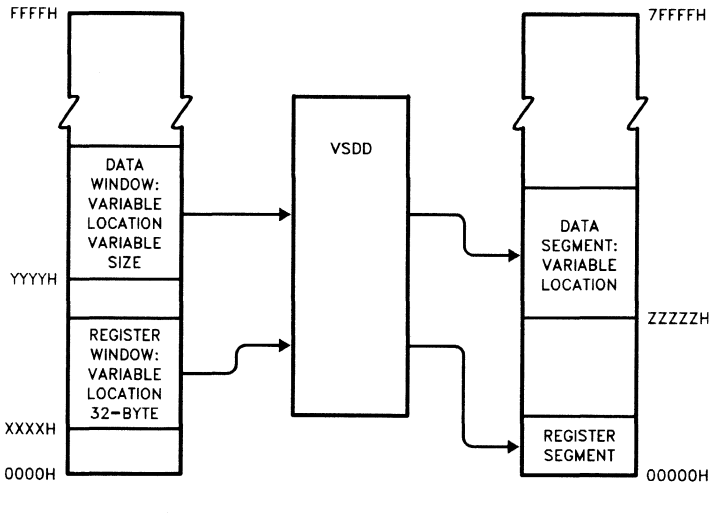
After the first write, the CPU can continue to initialize the Register Segment by writing to the Register Window addresses, 00400H through 0041FH. This information is mapped by the VSDD into its register segment from 00000H to 0001FH. Except for the control bits written into R0, the values as written do not take effect as long as UCF = 0. See Figure 41.

After the Register Segment has been completely initialized, one more write cycle is directed to R0 to set UCF to 1, while holding DEN = 0. The CPU now waits 1 frame time (1160 XTALIN cycles in the "fast frame" mode). At the end of the frame time in progress, during the FRAMESTOP sequence, the VSDD will be flagged by UCF = 1 to update its internal registers from the Register Segment in DRAM. Further access to the Register Segment by the CPU must be through the newly defined Register Window.

Now, through the newly defined Data Window, the CPU can commence initializing the display data. One would not want to set the DEN flag until the display data has been loaded.



(a) Pre-Initialization Memory Mapping



(b) Post-Initialization Memory Mapping

Figure 41. VSDD Memory Mapping

DOMESTIC SALES OFFICES

ALABAMA

Intel Corp.
5015 Bradford Drive
Suite 2
Huntsville 35805
Tel: (205) 830-4010

ARIZONA

Intel Corp.
11225 N. 28th Drive
Suite 214D
Phoenix 85029
Tel: (602) 869-4980

Intel Corp.
1161 N. El Dorado Place
Suite 301
Tucson 85715
Tel: (602) 299-6815

CALIFORNIA

Intel Corp.
21515 Vanowen Street
Suite 116
Canoga Park 91303
Tel: (818) 704-8500

Intel Corp.
2250 E. Imperial Highway
Suite 218
El Segundo 90245
Tel: (213) 640-6040

Intel Corp.
1510 Arden Way, Suite 101
Sacramento 95815
Tel: (916) 920-8096

Intel Corp.
4350 Executive Drive
Suite 105
San Diego 92121
Tel: (619) 452-5880

Intel Corp.*
400 N. Tustin Avenue
Suite 450
Santa Ana 92705
Tel: (714) 835-9642
TWX: 910-595-1114

Intel Corp.*
San Tomas 4
2700 San Tomas Expressway
Santa Clara, CA 95051
Tel: (408) 986-8086
TWX: 910-338-0255

COLORADO

Intel Corp.
3300 Mitchell Lane, Suite 210
Boulder 80301
Tel: (303) 442-8088

Intel Corp.
4445 Northpark Drive
Suite 100
Colorado Springs 80907
Tel: (303) 594-6622

Intel Corp.*
650 S. Cherry St., Suite 915
Denver 80222
Tel: (303) 321-8086
TWX: 910-931-2289

CONNECTICUT

Intel Corp.
26 Mill Plain Road
Danbury 06811
Tel: (203) 748-3130
TWX: 710-456-1199

FLORIDA

Intel Corp.
242 N. Westmonte Dr., Suite 105
Altamonte Springs 32714
Tel: (305) 869-5588

Intel Corp.
6363 N.W. 6th Way, Suite 100
Ft. Lauderdale 33309
Tel: (305) 771-0600
TWX: 510-956-9407

Intel Corp.
11300 4th Street North
Suite 170
St. Petersburg 33702
Tel: (813) 577-2413

GEORGIA

Intel Corp.
3280 Pointe Parkway
Suite 200
Norcross 30092
Tel: (404) 449-0541

ILLINOIS

Intel Corp.*
300 N. Martingale Road, Suite 400
Schaumburg 60173
Tel: (312) 310-8031

INDIANA

Intel Corp.
8777 Purdue Road
Suite 125
Indianapolis 46268
Tel: (317) 875-0623

IOWA

Intel Corp.
St. Andrews Building
1930 St. Andrews Drive N.E.
Cedar Rapids 52402
Tel: (319) 393-5510

KANSAS

Intel Corp.
8400 W. 110th Street
Suite 170
Overland Park 66210
Tel: (913) 345-2727

MARYLAND

Intel Corp.*
7321 Parkway Drive South
Suite C
Hanover 21076
Tel: (301) 796-7500
TWX: 710-862-1944

Intel Corp.
7833 Walker Drive
Greenbelt 20770
Tel: (301) 441-1020

MASSACHUSETTS

Intel Corp.*
Westford Corp. Center
3 Carlisle Road
Westford 01886
Tel: (617) 692-3222
TWX: 710-343-6333

MICHIGAN

Intel Corp.
7071 Orchard Lake Road
Suite 100
West Bloomfield 48033
Tel: (313) 851-8096

MINNESOTA

Intel Corp.
3500 W. 80th St., Suite 360
Bloomington 55431
Tel: (612) 835-6722
TWX: 910-576-2867

MISSOURI

Intel Corp.
4203 Earth City Expressway
Suite 131
Earth City 63045
Tel: (314) 291-1990

NEW JERSEY

Intel Corp.*
Parkway 109 Office Center
328 Newman Springs Road
Red Bank 07701
Tel: (201) 747-2233

Intel Corp.
280 Corporate Center
75 Livingston Avenue
First Floor
Roseland 07068
Tel: (201) 740-0111

NEW MEXICO

Intel Corp.
8500 Menaul Boulevard N.E.
Suite B 295
Albuquerque 87112
Tel: (505) 292-8086

NEW YORK

Intel Corp.
127 Main Street
Binghamton 13905
Tel: (607) 773-0337

Intel Corp.*
850 Cross Keys Office Park
Fairport 14450
Tel: (716) 425-2750
TWX: 510-253-7391

Intel Corp.*
300 Motor Parkway
Hauppauge 11787
Tel: (516) 231-3300
TWX: 510-227-6236

Intel Corp.
Suite 2B Hollowbrook Park
15 Myers Corners Road
Wappinger Falls 12590
Tel: (914) 297-6161
TWX: 510-248-0060

NORTH CAROLINA

Intel Corp.
5700 Executive Center Drive
Suite 213
Charlotte 28212
Tel: (704) 568-8966

Intel Corp.
2700 Wycliff Road
Suite 102
Raleigh 27607
Tel: (919) 781-8022

OHIO

Intel Corp.*
3401 Park Center Drive
Suite 220
Dayton 45414
Tel: (513) 890-5350
TWX: 810-450-2528

Intel Corp.*
25700 Science Park Dr., Suite 100
Beachwood 44122
Tel: (216) 464-2736
TWX: 810-427-9298

OKLAHOMA

Intel Corp.
6801 N. Broadway
Suite 115
Oklahoma City 73116
Tel: (405) 848-8086

OREGON

Intel Corp.
15254 N.W. Greenbrier Parkway, Bldg. B
Beaverton 97006
Tel: (503) 645-8051
TWX: 910-467-8741

PENNSYLVANIA

Intel Corp.
1513 Cedar Cliff Drive
Camp Hill 17011
Tel: (717) 737-5035

Intel Corp.*
455 Pennsylvania Avenue
Fort Washington 19034
Tel: (215) 641-1000
TWX: 510-661-2077

Intel Corp.*
400 Penn Center Blvd., Suite 610
Pittsburgh 15235
Tel: (412) 823-4970

PUERTO RICO

Intel Microprocessor Corp.
South Industrial Park
P.O. Box 910
Las Piedras 00671
Tel: (809) 733-8616

TEXAS

Intel Corp.
313 E. Anderson Lane
Suite 314
Austin 78752
Tel: (512) 454-3628

Intel Corp.*
12300 Ford Road
Suite 380
Dallas 75234
Tel: (214) 241-8087
TWX: 910-860-5617

Intel Corp.*
7322 S.W. Freeway
Suite 1490
Houston 77074
Tel: (713) 988-8086
TWX: 910-881-2490

UTAH

Intel Corp.
5201 Green Street
Suite 290
Murray 84123
Tel: (801) 263-8051

VIRGINIA

Intel Corp.
1603 Santa Rosa Road
Suite 109
Richmond 23288
Tel: (804) 282-5668

WASHINGTON

Intel Corp.
155-108 Avenue N.E.
Suite 386
Bellevue 98004
Tel: (206) 453-8086
TWX: 910-443-3002

Intel Corp.
408 N. Mullan Road
Suite 102
Spokane 99206
Tel: (509) 928-8086

WISCONSIN

Intel Corp.
450 N. Sunnyslope Road
Suite 130
Chancellory Park 1
Brookfield 53005
Tel: (414) 784-8087

CANADA

BRITISH COLUMBIA

Intel Semiconductor of Canada, Ltd.
301-2245 W. Broadway
Vancouver V6K 2E4
Tel: (604) 738-6522

ONTARIO

Intel Semiconductor of Canada, Ltd.
2650 Queensview Drive
Suite 250
Ottawa K2B 8H6
Tel: (613) 829-9714
TELEX: 053-4115

Intel Semiconductor of Canada, Ltd.
190 Attwell Drive
Suite 500
Rexdale M9W 6H8
Tel: (416) 675-2105
TELEX: 06983574

QUEBEC

Intel Semiconductor of Canada, Ltd.
620 St. Jean Boulevard
Pointe Claire H9R 3K3
Tel: (514) 694-9130
TWX: 514-694-9134

*Field Application Location

DOMESTIC DISTRIBUTORS

ALABAMA

Arrow Electronics, Inc.
1015 Henderson Road
Huntsville 35805
Tel: (205) 837-6955

†Hamilton/Avnet Electronics
4940 Research Drive
Huntsville 35805
Tel: (205) 837-7210
TWX: 810 726-2162

Pioneer/Technologies Group Inc
4825 University Square
Huntsville 35805
Tel: (205) 837-9300
TWX: 810-726-2197

ARIZONA

†Hamilton/Avnet Electronics
505 S. Madison Drive
Tempe 85281
Tel: (602) 231-5100
TWX: 910-950-0077

Kierulff Electronics, Inc.
4134 E. Wood Street
Phoenix 85040
Tel: (602) 437-0750
TWX: 910-951-1550

Wyle Distribution Group
17855 N. Black Canyon Highway
Phoenix 85023
Tel: (602) 866-2888

CALIFORNIA

Arrow Electronics, Inc.
19748 Dearborn Street
Chatsworth 91311
Tel: (818) 701-7500
TWX: 910-493-2086

Arrow Electronics, Inc.
1502 Crocker Avenue
Hayward 94544
Tel: (408) 487-4600

Arrow Electronics, Inc.
9511 Ridgehaven Court
San Diego 92123
Tel: (619) 565-4800
TLX: 888064

†Arrow Electronics, Inc.
521 Weddell Drive
Sunnyvale 94086
Tel: (408) 745-6600
TWX: 910-339-9371

Arrow Electronics, Inc.
2961 Dow Avenue
Tustin 92680
Tel: (714) 838-5422
TWX: 910-595-2860

†Avnet Electronics
350 McCormick Avenue
Costa Mesa 92626
Tel: (714) 754-6051
TWX: 910-595-1928

Hamilton/Avnet Electronics
1175 Bordeaux Drive
Sunnyvale 94086
Tel: (408) 743-3300
TWX: 910-339-9332

†Hamilton/Avnet Electronics
4545 Viewridge Avenue
San Diego 92123
Tel: (619) 571-7500
TWX: 910-595-2638

†Hamilton/Avnet Electronics
20501 Plummer Street
Chatsworth 91311
Tel: (818) 700-6271
TWX: 910-494-2207

†Hamilton/Avnet Electronics
4103 Northgate Boulevard
Sacramento 95834
Tel: (916) 920-3150

Hamilton/Avnet Electronics
3002 G Street
Ontario 91311
Tel: (714) 989-9411

Hamilton/Avnet Electronics
19515 So. Vermont Avenue
Torrance 90502
Tel: (213) 615-3909
TWX: 910-349-6263

Hamilton Electro Sales
9650 De Soto Avenue
Chatsworth 91311
Tel: (818) 700-6500

†Hamilton Electro Sales
10950 W. Washington Boulevard
Culver City 90230
Tel: (213) 558-2458
TWX: 910-340-6364

CALIFORNIA (Cont'd)

Hamilton Electro Sales
1361 B West 190th Street
Gardena 90248
Tel: (213) 558-2131

†Hamilton Electro Sales
3170 Pullman Street
Costa Mesa 92626
Tel: (714) 641-4150
TWX: 910-595-2638

Kierulff Electronics, Inc.
10824 Hope Street
Cypress 90430
Tel: (714) 220-6300

Kierulff Electronics, Inc.
1180 Murphy Avenue
San Jose 95131
Tel: (408) 971-2600
TWX: 910-379-6430

Kierulff Electronics, Inc.
14101 Franklin Avenue
Tustin 92680
Tel: (714) 731-5711
TWX: 910-595-2599

†Kierulff Electronics, Inc.
5650 Jillson Street
Commerce 90040
Tel: (213) 725-0325
TWX: 910-580-3666

Wyle Distribution Group
26560 Agoura Street
Calabasas 91302
Tel: (818) 880-9000
TWX: 818-372-0232

†Wyle Distribution Group
124 Maryland Street
El Segundo 90245
Tel: (213) 322-8100
TWX: 910-348-7140 or 7111

†Wyle Distribution Group
17872 Cowan Avenue
Irvine 92714
Tel: (714) 843-9953
TWX: 910-595-1572

†Wyle Distribution Group
11151 Sun Center Drive
Rancho Cordova 95670
Tel: (916) 638-5282

†Wyle Distribution Group
9525 Chesapeake Drive
San Diego 92123
Tel: (619) 565-9171
TWX: 910-335-1590

†Wyle Distribution Group
3000 Bowers Avenue
Santa Clara 95051
Tel: (408) 727-2500
TWX: 910-338-0296

Wyle Military
18910 Teller Avenue
Irvine 92750
Tel: (714) 851-9958
TWX: 310-371-9127

Wyle Systems
7382 Lampson Avenue
Garden Grove 92641
Tel: (714) 851-9953
TWX: 910-595-2642

COLORADO

Arrow Electronics, Inc.
1390 S. Potomac Street
Suite 136
Aurora 80012
Tel: (303) 696-1111

†Hamilton/Avnet Electronics
8765 E. Orchard Road
Suite 708
Englewood 80111
Tel: (303) 740-1017
TWX: 910-935-0787

†Wyle Distribution Group
451 E. 124th Avenue
Thornton 80241
Tel: (303) 457-9953
TWX: 910-936-0770

CONNECTICUT

†Arrow Electronics, Inc.
12 Beaumont Road
Wallingford 06492
Tel: (203) 265-7741
TWX: 710-476-0162

†Hamilton/Avnet Electronics
Commerce Industrial Park
Commerce Drive
Danbury 06810
Tel: (203) 797-2800
TWX: 710-456-9974

CONNECTICUT (Cont'd)

†Pioneer Northeast Electronics
112 Main Street
Norwalk 06851
Tel: (203) 853-1515
TWX: 710-468-3373

FLORIDA

†Arrow Electronics, Inc.
350 Fairway Drive
Deerfield Beach 33441
Tel: (305) 429-8200
TWX: 510-955-9456

†Arrow Electronics, Inc.
1001 N.W. 62nd Street
Suite 108
Ft. Lauderdale 33309
Tel: (305) 776-7790
TWX: 510-955-9456

†Arrow Electronics, Inc.
50 Woodlake Drive W., Bldg. B
Palm Bay 32905
Tel: (305) 725-1480
TWX: 510-959-6337

†Hamilton/Avnet Electronics
6801 N.W. 15th Way
Ft. Lauderdale 33309
Tel: (305) 971-2900
TWX: 510-956-3097

†Hamilton/Avnet Electronics
3197 Tech Drive North
St. Petersburg 33702
Tel: (813) 576-3930
TWX: 810-863-0374

Hamilton/Avnet Electronics
6947 University Boulevard
Winterpark 32792
Tel: (305) 628-3888
TWX: 810-853-0322

†Pioneer Electronics
337 N. Lake Boulevard
Suite 1000
Alta Monte Springs 32701
Tel: (305) 834-9090
TWX: 810-853-0284

†Pioneer Electronics
674 S. Military Trail
Deerfield Beach 33442
Tel: (305) 428-8877
TWX: 510-955-9653

GEORGIA

†Arrow Electronics, Inc.
3155 Northwoods Parkway, Suite A
Norcross 30071
Tel: (404) 449-8252
TWX: 810-766-0439

Hamilton/Avnet Electronics
5825 D. Peachtree Corners
Norcross 30092
Tel: (404) 447-7500
TWX: 810-766-0432

Pioneer Electronics
3100 F. Northwoods Place
Norcross 30071
Tel: (404) 448-1711
TWX: 810-766-4515

ILLINOIS

†Arrow Electronics, Inc.
2000 E. Alonquin Street
Schaumberg 60195
Tel: (312) 397-3440
TWX: 910-291-3544

†Hamilton/Avnet Electronics
1130 Thorndale Avenue
Bensenville 60106
Tel: (312) 860-7780
TWX: 910-227-0060

MTI Systems Sales
1100 West Thorndale
Itasca 60143
Tel: (312) 773-2300

†Pioneer Electronics
1551 Carmen Drive
Elk Grove Village 60007
Tel: (312) 437-9680
TWX: 910-222-1834

INDIANA

†Arrow Electronics, Inc.
2495 Directors Row, Suite H
Indianapolis 46241
Tel: (317) 243-9353
TWX: 810-341-3119

Hamilton/Avnet Electronics
485 Gradle Drive
Carmel 46032
Tel: (317) 844-9333
TWX: 810-260-3966

INDIANA (Cont'd)

†Pioneer Electronics
6408 Castleplace Drive
Indianapolis 46250
Tel: (317) 849-7300
TWX: 810-260-1794

KANSAS

†Hamilton/Avnet Electronics
9219 Quivera Road
Overland Park 66215
Tel: (913) 888-8900
TWX: 910-743-0005

KENTUCKY

Hamilton/Avnet Electronics
1051 D. Newton Park
Lexington 40511
Tel: (606) 259-1475

MARYLAND

Arrow Electronics, Inc.
8300 Gulford Road #H
Rivers Center
Columbia 21046
Tel: (301) 995-0003
TWX: 710-236-9005

†Hamilton/Avnet Electronics
6822 Oak Hall Lane
Columbia 21045
Tel: (301) 995-3500
TWX: 710-862-1861

†Mesa Technology Corporation
9720 Patuxentwood Dr.
Columbia 21046
Tel: (301) 290-8150
Twx: 710-828-9702

†Pioneer Electronics
9100 Gaither Road
Gaithersburg 20877
Tel: (301) 948-0710
TWX: 710-828-0545

MASSACHUSETTS

†Arrow Electronics, Inc.
1 Arrow Drive
Woburn 01801
Tel: (617) 933-8130
TWX: 710-393-6770

†Hamilton/Avnet Electronics
10D Centennial Drive
Peabody 01960
Tel: (617) 532-3701
TWX: 710-393-0382

MTI Systems Sales
13 Fortune Drive
Billerica 01821

Pioneer Northeast Electronics
44 Hartwell Avenue
Lexington 02173
Tel: (617) 863-1200
TWX: 710-326-6617

MICHIGAN

Arrow Electronics, Inc.
755 Phoenix Drive
Ann Arbor 48104
Tel: (313) 971-8220
TWX: 810-223-6020

†Hamilton/Avnet Electronics
32487 Schoolcraft Road
Livonia 48150
Tel: (313) 522-4700
TWX: 810-242-8775

Hamilton/Avnet Electronics
2215 29th Street S.E.
Space A5
Grand Rapids 49508
Tel: (616) 243-8805
TWX: 810-273-6921

†Pioneer Electronics
13485 Stamford
Livonia 48150
Tel: (313) 525-1800
TWX: 810-242-3271

MINNESOTA

†Arrow Electronics, Inc.
5230 W. 73rd Street
Edina 55435
Tel: (612) 830-1800
TWX: 910-576-3125

Hamilton/Avnet Electronics
12400 White Water Drive
Minnetonka 55343
Tel: (612) 932-0600
TWX: (910) 576-2720

†Pioneer Electronics
10203 Bren Road East
Minnetonka 55343
Tel: (612) 935-5444
TWX: 910-576-2738

†Microcomputer System Technical Demonstrator Centers