TMS320C80 Frame Buffer

Application Report

SPRA156
February 1997

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain applications using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

Contents

List of Figures

List of Tables

TMS320C80 Frame Buffer

ABSTRACT

The TMS320C80 digital signal processor (DSP) provides direct support for two independent frame memories through on-chip controllers. This application report presents a 4M-byte video random-access memory (VRAM) based frame buffer interface to the ’C80 DSP.

The report discusses the hardware interface for the frame buffer card palette. Also, included are a VRAM overview, information on frame-timer interrupts, and appendices covering materials, schematics, and ABEL™ files.

1 Introduction

High-end graphics and imaging applications often rely on a sophisticated frame-buffering mechanism, wherein large amounts of data are displayed quickly and efficiently, without imposing significant constraints on the system’s ability to process the data. The TMS320C80 is well suited for such applications. The ’C80 provides direct support for two independent frame memories through on-chip frame controllers. Additionally, the ’C80’s external bus interface directly provides the address multiplexing, bus-width selection, wait-state support, configurable page size, and refresh control required for multiple banks of dynamic memory typically used in frame buffers.

This application report discusses a 4M-byte VRAM (video random-access memory) based frame buffer interface to the TMS320C80 DSP. A 4M-byte frame buffer is large enough to support double buffering of most of the larger resolutions and also addresses the issue of serial output multiplexing. The VRAMs’ serial ports feed directly to a color palette, the TVP3020 RAMDAC. Control of the TVP3020 is maintained by the ’C80. This report discusses the hardware interface for the frame buffer and palette.

ABEL is a trademark of DATA I/O.

2 Video Timing

It is important to understand the many timing relationships that must exist from one part of the system to the next in order to construct a complete frame buffer and display system. The three most important clocks in video — the pixel clock, the shift clock, and the video, or frame, clock — are discussed in the following paragraphs.

2.1 Pixel Clock

Pixel clock frequency refers to the conversion frequency. This is the frequency at which the digital-to-analog converters (DACs) of the palette must be able to convert the digital data stored in the frame buffer to analog red, green, and blue (RGB) outputs to drive the display device. The pixel clock rate is influenced by the amount of horizontal and vertical blanking required to conform to the monitor specifications.

2.2 Shift Clock

The serial shift clock (SCLK) is required to control the transfer of digital pixel information from the VRAM frame buffer to the RAMDAC pixel port. SCLK is normally a divide-down of the pixel clock frequency, where the divisor value is determined by the ratio of pixel port width to pixel size. For example, if 8-bit pixels are stored in the frame buffer that interfaces with a 64-bit serial bus, then eight pixels can be transferred in each SCLK cycle. SCLK is, therefore, pixel clock divided by 8. Shift clock is slightly different from the other clocks in that it is not continuous. SCLK does not function during periods of blanking, since no pixel data is to be displayed then. The TVP3020 used in this design provides this feature.

2.3 Frame Clock

The video, or frame, clock is used to produce the horizontal timing signals required to drive the display device. The frame clock also indirectly produces the vertical timing signals. For many processors, this requires an external frame timing chip to generate the sync and blanking signals from the frame clock. The ’C80 has two on-chip frame timers that can produce all the required signals. These signals are derived from the frame clock (FCLK) input. The duration of blanking, sync, and a general-purpose area signal are completely programmable, allowing for many standards to be supported. Additionally, each frame timer can be user-programmed to interrupt the master processor during a frame. ’C80 frame timers support both interlaced and non-interlaced modes.

3 Video Controller

The TMS320C80 provides two identical on-chip frame timers. Each frame timer has its own frame clock and operates asynchronously to the rest of the ’C80. Each timer can be programmed to generate timing pulses on five video signals that can be used to control a capture or display device. Figure 1 shows the functional block diagram of the TMS320C80 video controller.

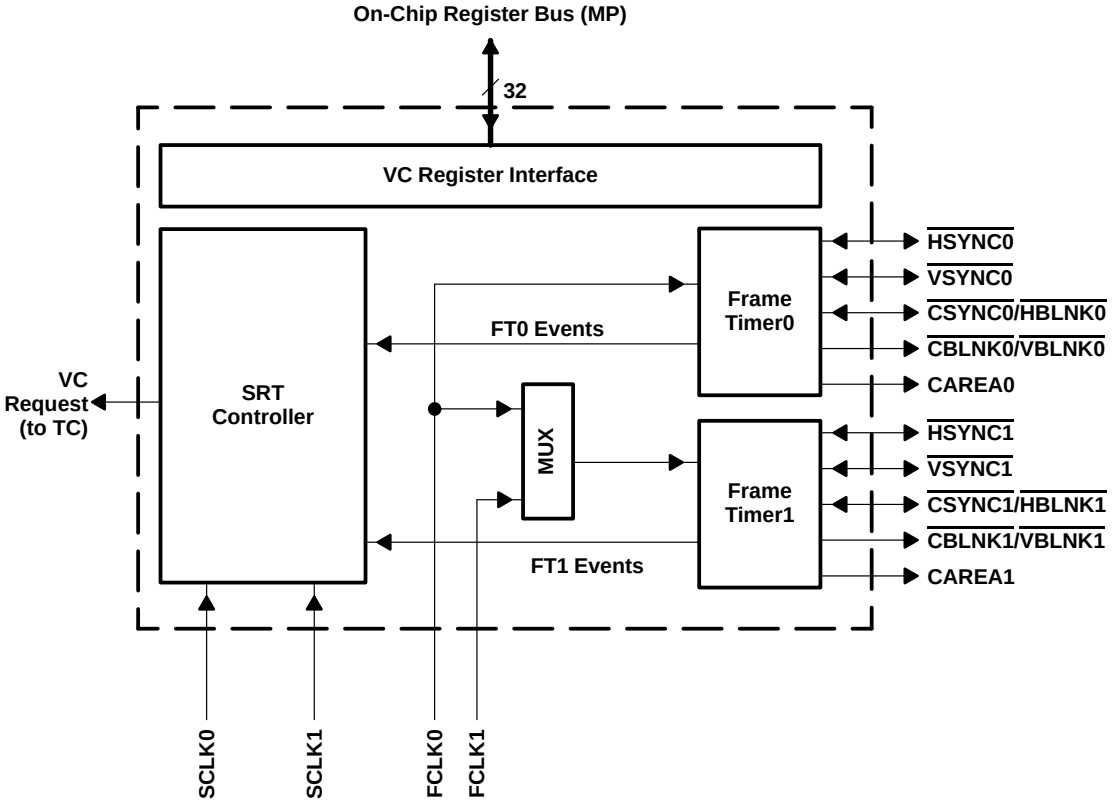


Figure 1. TMS320C80 Video Controller

The five video signals are described below:

$\overline{\text{HSYNC}}$	I / O / Hi-Z	Horizontal sync
$\overline{\text{VSYNC}}$	I / O / Hi-Z	Vertical sync
$\overline{\text{CSYNC}}$ / $\overline{\text{HBLNK}}$	I / O / Hi-Z	Composite sync / horizontal blank (user-selectable)
$\overline{\text{CBLNK}}$ / $\overline{\text{VBLNK}}$	O	Composite blank / vertical blank (user-selectable)
CAREA	O	Composite area (general purpose)

NOTE: I = input
O = output
Hi-Z = high impedance

3.1 Frame Timer Registers

All horizontal timing, and horizontal timing components of composite signals, are programmed in terms of an integral number of frame clock periods. Vertical timing parameters are programmed in terms of an integral number of lines (half-lines for interlaced). In this report, only the non-interlaced mode is discussed.

The duration of sync, blanking, and area signals is completely user-programmable on the ’C80. Control is maintained though on-chip memory-mapped registers. In this example, only frame timer 1 is used. Frame timer 1 is chosen because it allows more system flexibility. If a capture device is also to be controlled by the ’C80 in the system, it is desirable to use frame timer 1 for the display. This allows the display system to be clocked from the input if desired (frame timer 1 can be clocked from FCLK0, but the contrary is not true).

Figure 2 and Figure 3 show the frame timer registers. As shown, horizontal timing registers are located at even halfword addresses; their vertical counterparts are located at odd halfword addresses.

	ADDRESS
SETVCT0	0x01820206
VFTINT0	0x0182020A
VESYNC0	0x0182020E
VEBLNK0	0x01820212
VSAREA0	0x01820216
VEAREA0	0x0182021A
VSBLNK0	0x0182021E
VTOTAL0	0x01820222
VCOUNT0	0x0182023E

	ADDRESS
FTCTL0	0x01820200
SETHCT0	0x01820204
HESERR0	0x01820208
HESYNC0	0x0182020C
HEBLNK0	0x01820210
HSAREA0	0x01820214
HEAREA0	0x01820218
HSBLNK0	0x0182021C
HTOTAL0	0x01820220
HALINE0	0x01820224
HBLINE0	0x01820228
HCOUNT0	0x0182023C

Figure 2. Frame Timer 0 Register Map

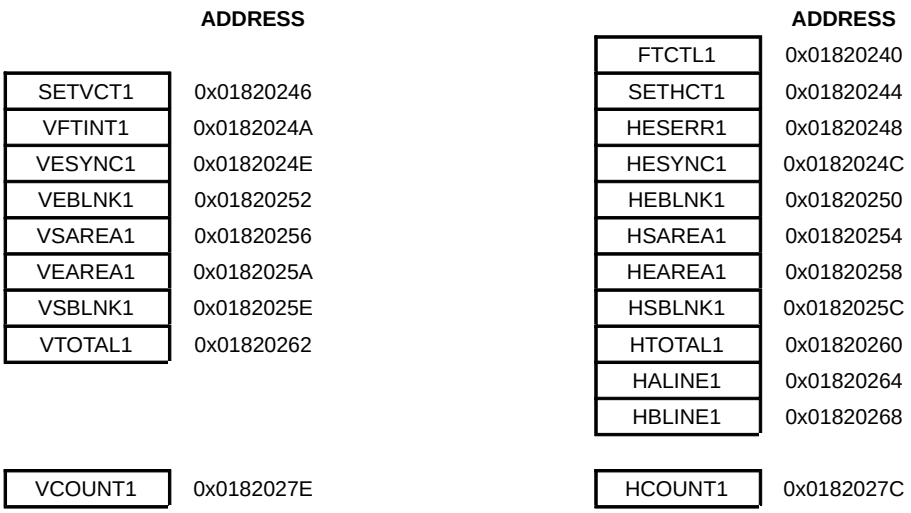


Figure 3. Frame Timer 1 Register Map

Table 1 summarizes the programming of the frame timer registers for noninterlaced mode. For a more detailed description of each of the frame timer registers, please refer to the *TMS320C80 Video Controller User's Guide*, (literature number SPRU111A). Note that the programming of the registers is the same for both composite and non-composite modes.

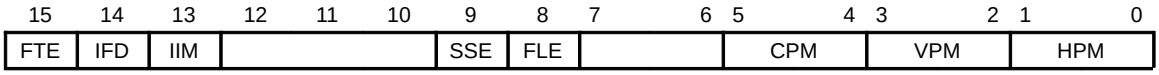
Table 1. Video-Timing Registers for Noninterlaced Video

REGISTER	IS PROGRAMMED TO ...
HTOTAL	(The number of FCLKs per line) – 1
HESYNC	(The number of FCLKs in horizontal sync) – 1
HESERR	(The number of FCLKs in horizontal serration) – 1
HEBLNK	(The number of FCLKs from the start of horizontal sync to the end of horizontal blank) – 1
HSAREA	(The number of FCLKs from the start of horizontal sync to the start of horizontal area) – 1
HEAREA	(The number of FCLKs from the start of horizontal sync to the end of horizontal area) – 1
HSBLNK	(The number of FCLKs from the start of horizontal sync to the start of horizontal blank) – 1
VTOTAL	(The number of lines per frame) – 1
VESYNC	(Twice the number of lines in vertical sync) – 1
VEBLNK	(The number of lines from the start of vertical sync to the end of vertical blank) – 1
VSAREA	(The number of lines from the start of vertical sync to the start of vertical area) – 1
VEAREA	(The number of lines from the start of vertical sync to the end of vertical area) – 1
VSBLNK	(The number of lines from the start of vertical sync to the start of vertical blank) – 1
VFTINT	(The number of lines from the start of vertical sync to the interrupt point) – 1

NOTES: 1. HESERR needs to be programmed only for composite mode; it is unused when separate horizontal and vertical syncs are used.
2. VFTINT is discussed in more detail in Section 14.

3.1.1 FTCTL Register

Control of the frame timer is maintained through the FTCTL register. Bits in this register enable / disable the frame timer, select between interlaced and non-interlaced modes, and control the direction of the frame timer pins (CAREA is always an output). See Figure 4 for FTCTLx register description. For a complete description of the FTCTL register, please refer to the *TMS320C80 Video Controller User's Guide*, (literature number SPRU111A).



Legend:

FTE –	Frame timer enable			VPM –	$\overline{\text{VSYNC}}$ pin mode		
IFD –	Interlaced frame disable			00 –	Hi-Z	10 –	Output
IIM –	Interlace interrupt mode			01 –	Input	11 –	Reserved
SSE –	Set synchronization enable			HPM –	$\overline{\text{HSYNC}}$ pin mode		
FLE –	Frame lock enable			00 –	Hi-Z	10 –	Output
CPM –	CSYNC / HBLNK pin mode			01 –	Input	11 –	Reserved
00 –	CSYNC Hi-Z	10 –	CSYNC output				
01 –	CSYNC input	11 –	HBLNK output				

Figure 4. FTCTLx Register

3.2 Horizontal and Vertical Timing Relationship

In this application report, a non-interlaced display is considered. The following figure illustrates the relationship between the horizontal and vertical timing signals, and the frame timer registers that control the signal transitions for non-interlaced video.

- The horizontal sync and blanking signals span a single horizontal scan within the frame and are repeated for each line.
- The vertical sync and blanking signals span one complete frame.

The relationship between the sync and blanking pulses shown in Figure 5 defines what are commonly called the *porches* in video timing. The video porches are shown in Figure 6.

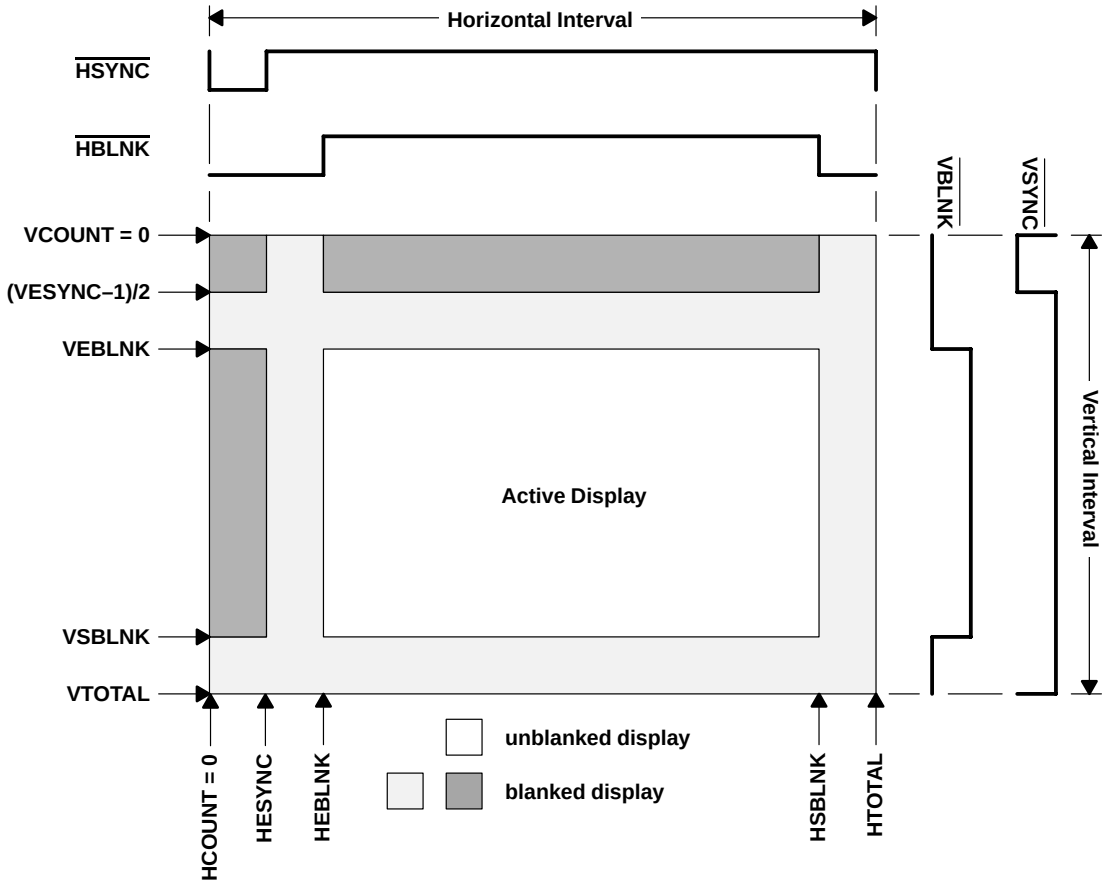


Figure 5. Horizontal and Vertical Timing Relationship

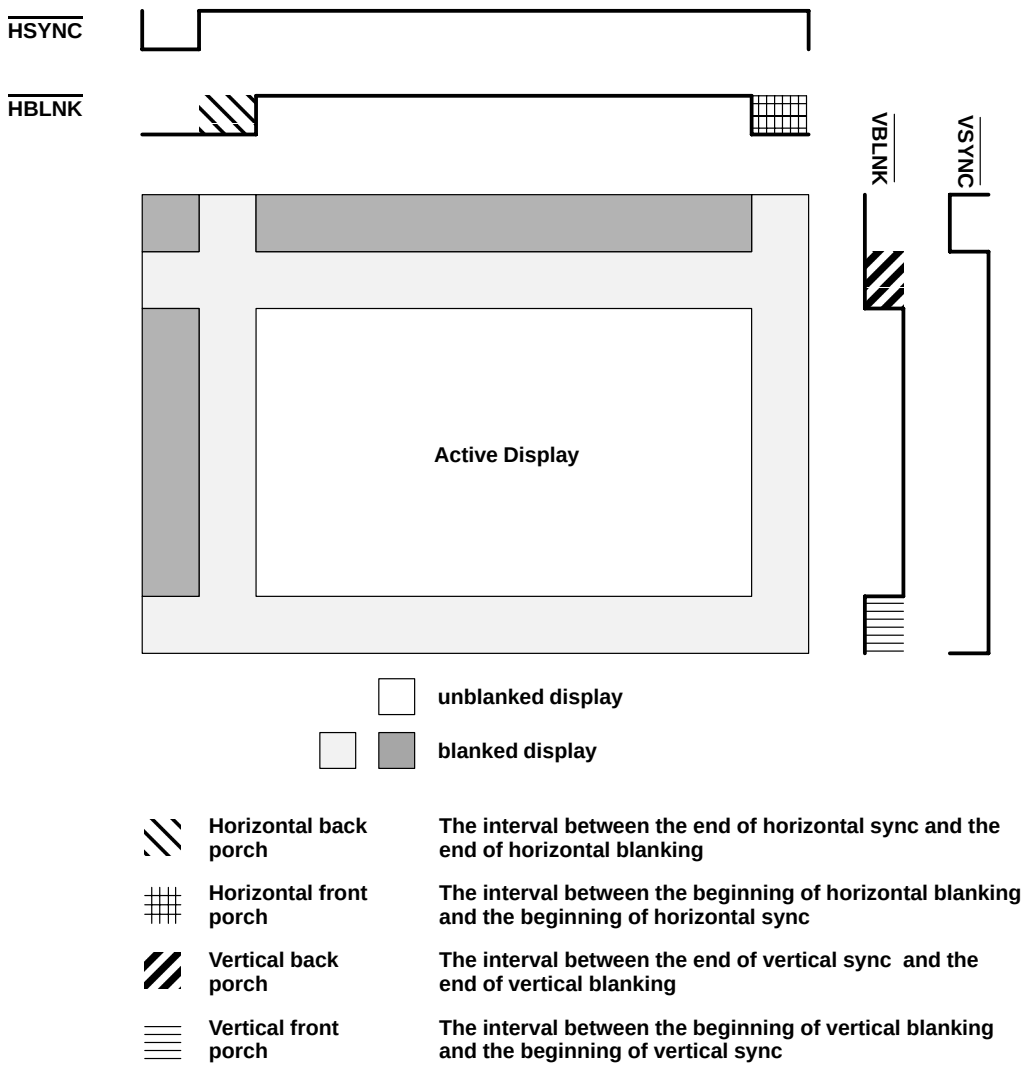


Figure 6. Video Porches

Note that in Figure 5 it was assumed that the ’C80 directly drove the sync and blanking signals to the display device. These signals can be driven from the RAMDAC instead. Skews through the RAMDAC affect the position of the sync and blanking transitions in an absolute sense; however, as the skew is constant, it does not affect the programming of most of the timing registers.

4 VRAM Overview

Architecturally, VRAM is very similar to DRAM. In fact, VRAM is actually the same dynamic memory array, with several key additions that optimize it for frame buffer designs. The two key features are the support for block write cycles, which is a special write cycle that allows the user to efficiently process pixel data, and the addition of a serial access port.

In this report, the frame buffer is made up of TMS55161 (-60) VRAMs (x8). The TMS55161 (-60) VRAMs are organized as 512 rows × 512 columns × 16 I / Os. The serial port is organized as 256 × 16 I / Os and is connected to the pixel port of the TVP3020. Data is written into the memory array from the ’C80 by way of the parallel interface. Figure 7 is a simplified block diagram showing a portion of a VRAM. Notice that when used with TMS55161s, this structure is replicated four times, providing sixteen bit planes in one device.

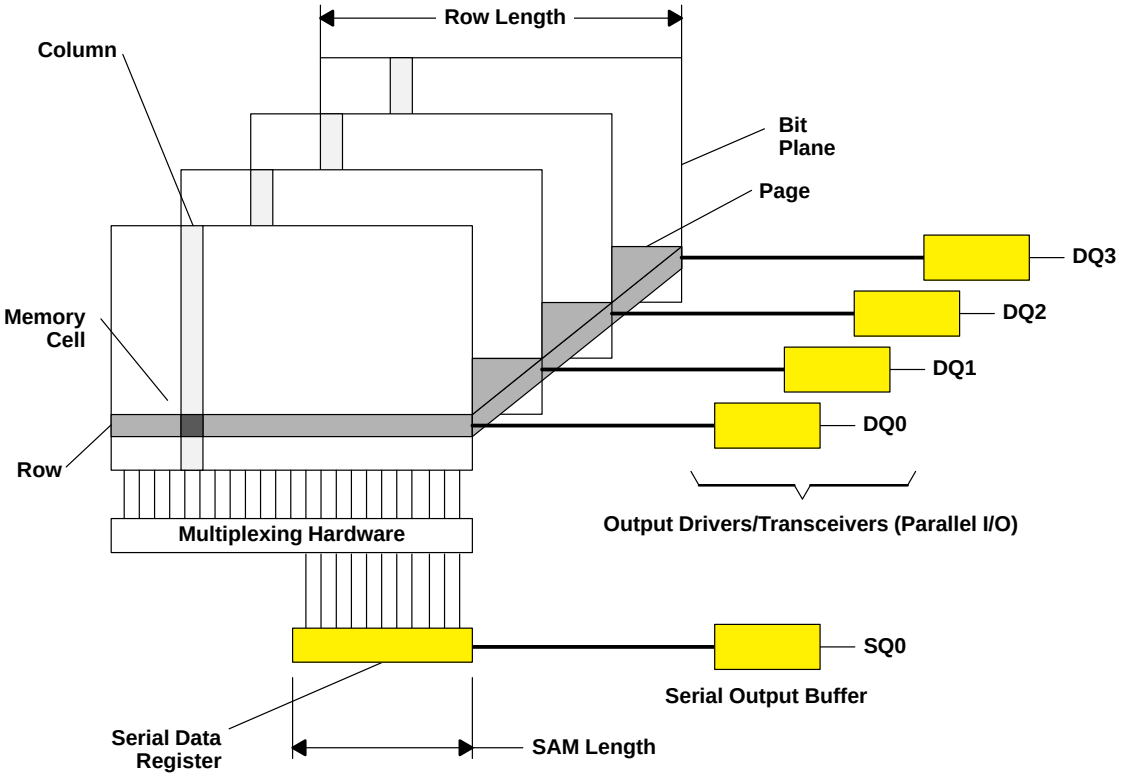


Figure 7. VRAM Architecture

Each TMS55161 features a 256-bit-long register, referred to as the serial-access memory (SAM). Each bit plane in the VRAM (16 total), has its own dedicated SAM. Data stored into the memory array is transferred into the SAM in a memory-to-register transfer. The data is then shifted out serially to the RAMDAC on the SQ outputs (SQ15−SQ0), at the rate of the shift clock (SCLK) input to the VRAMs.

4.1 Memory-to-Register Transfers

Each TMS55161 supports two modes of memory-to-register transfers: full-register transfers and split-register transfers. In both modes, the basic operation is the same: data from the memory array is copied into the SAM. The TMS55161 allows either the entire SAM, or half of the SAM to be written in each memory-to-register transfer. The split- (half-) register transfer is widely used in the following two cases:

- In systems where there is not enough data in the register to complete one horizontal line, and therefore new data must be transferred in the middle of an active scan line.
- When the resolution does not match the amount of information stored in a row of the memory array, and therefore the information needs to be packed as tightly as possible.

In the split-register transfer operation, the serial register is divided into two halves. While one half is being read out of the SAM port, the other half can be loaded from the memory array. This requires precise synchronization to maintain a smooth display. If the register-to-memory transfers are not scheduled at the proper time, the display can be corrupted. With many processors, this requires a complex array of counters, comparators, and logic to maintain synchronization. The TMS320C80 video controller also has two separate SRT (serial-register transfer) controllers to complement the frame timers. The SRT controllers are identical, and each can be used to control a frame memory for either capture or display. The scheduling of both full- and split-register transfers, as well as the generation of the proper bus cycles, is provided by the TMS320C80 video and transfer controllers. The SRT controllers are described in Section 13.

The SAM register of each TMS55161 is 256 bits in length, or one-half the length of a VRAM row; therefore, full transfers move either bits 0−255 or 256−511 of a VRAM row into the SAM. A split-register transfer transfers bits 0−127, 128−255, 256−383, or 384−512 into bits 1−127 or 128−255 of the SAM. This is illustrated in Figure 8.

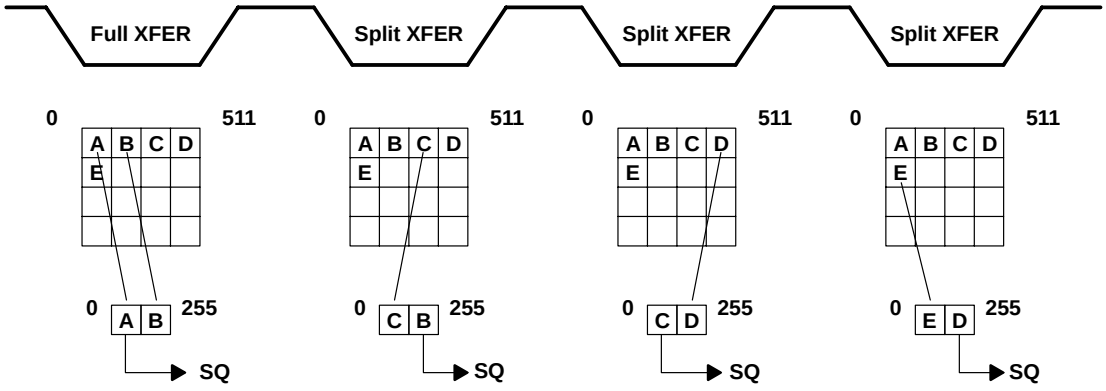


Figure 8. Split-Register Read Transfer Operation

The SQ arrow shown in Figure 8 indicates the *active* SAM half, the part of the SAM that is being serially shifted out to the RAMDAC. Split transfers, which occur during active display, are always done to the inactive SAM half. This prevents corruption of the displayed video. The ’C80’s SRT controller handles the scheduling of all full and split transfers for the system.

4.2 Block Write

A second feature that makes VRAM unique is support for special write cycles known as block-writes. Block-write is a high-level function that can greatly enhance applications such as rectangle-fill, rapid text, and clear screen operations. Many types of block-writes are available to match the many sizes of VRAM available. In general, a block-write is expressed in three dimensions.

column locations per color register $\times$ length of color register $\times$ number of color registers

The TMS320C80 supports both 8x and 4x block-write cycles; this feature allows up to 64 bits of data to be written simultaneously to each device in the memory bank. This is accomplished by transferring a color value from an internal (to the VRAM) register to the memory, in what is called a load-color-register (LCR) cycle. The ’C80 performs an LCR cycle at the start of a block-write cycle. The color register value is defined in the packet-transfer parameter table which sets up the block-write. For more information on block-write packet transfers, please refer to the *TMS320C80 (MVP) Transfer Controller User’s Guide* (literature number SPRU105A). TMS55161s described in this application report implement $4 \times 4 \times 4$ block-write cycles.

In the case of 4x block-writes, the ’C80 writes 32 bytes in a single access. This requires some bit manipulation by the transfer controller. Normally, the ’C80 bus size (BS[1:0] pins) indicates the bus width of the memory bank being addressed. However, since the ’C80 supports only 64-bit-wide block-write cycles, the BS inputs are used to define the block-write mode for these cycles. The status code 001001 is output on STATUS[5:0] to indicate that the block-write is occurring, and must be decoded by external hardware to indicate the block-write mode. The codes listed in Table 2 are defined for the BS inputs during block-write.

Table 2. BS[1:0] Clock-Write Codes

BS[1:0]		BLOCK-WRITE MODE
0	0	Simulated
0	1	Reserved
1	0	4x
1	1	8x

The “blocks” in block-writes (4x) cross four columns of a row of the VRAM array, therefore, there are 128 blocks per row (512 / 4). During the 4x block-write cycles, only the seven most significant bits (MSBs) of the column address are latched (with $\overline{\text{CAS}}$) to decode one of the 128 blocks. Each bit of the source data can force up to four consecutive columns to be written at a time. Note that the data written to the memory array is actually data from the color register, not the VRAM data inputs. An example of block-write is shown in Figure 9. It should be noted, however, that Figure 9 cannot be interpreted as an actual block-write cycle, nor does it accurately describe the VRAM hardware that implements block writes. It is included to demonstrate the relationship between data bits during block-write cycles and the value stored in the color register of the VRAM, and the writes that they produce during block-write cycles.

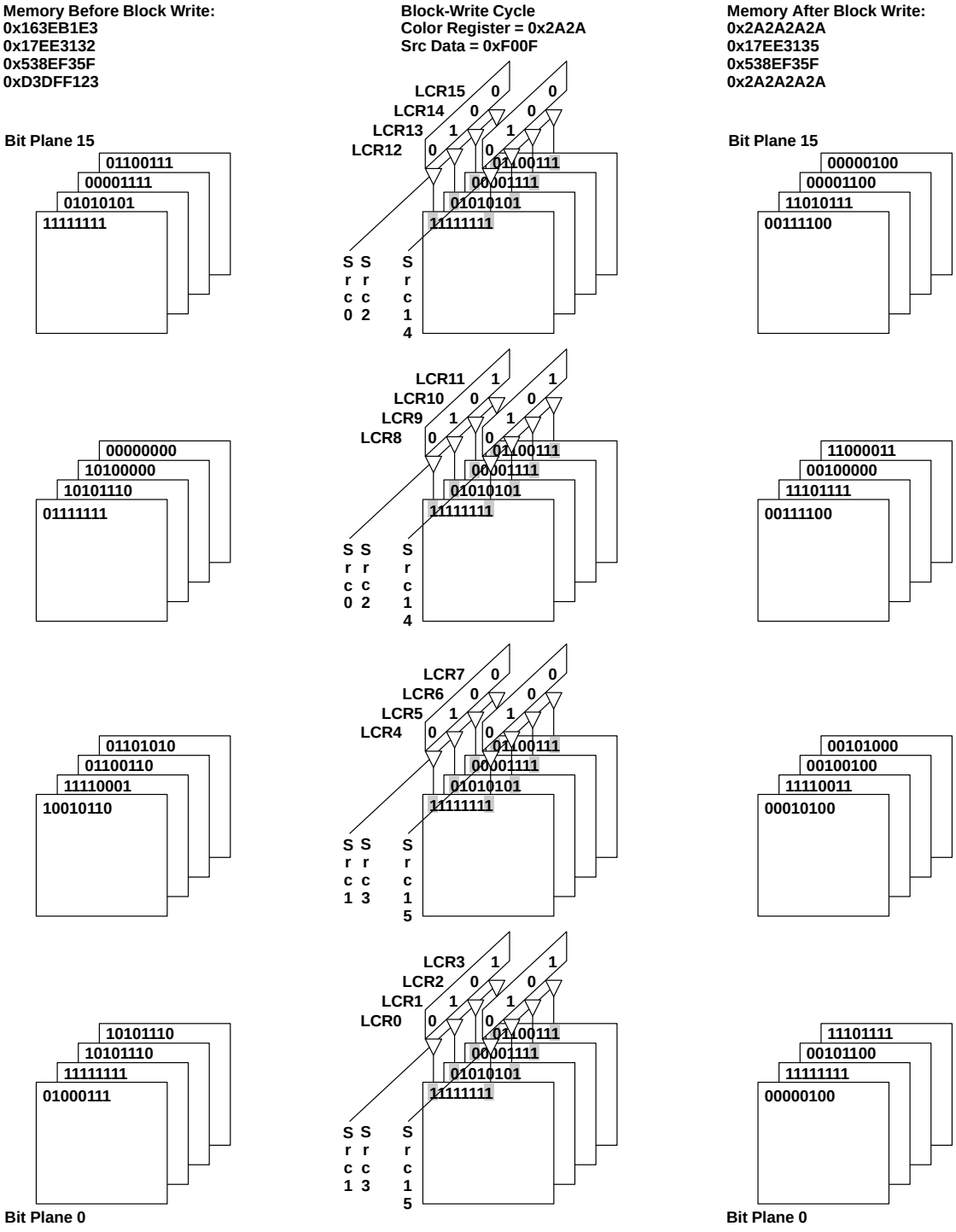


Figure 9. Example of Block-Write

Support for block-write is not mandatory. Block-write is most often used for high-end graphics applications where speed is critical. Block-write is an inherently little-endian function. In little-endian systems, byte 0 corresponds to $\overline{\text{CAS0}}$, which is controlled by bit 0 of the source data. In big-endian systems, byte 0 corresponds to $\overline{\text{CAS7}}$, but still must be controlled by data bit 0. Therefore, data lines connected to the VRAM bank must be bit-reversed, on a by-device-width basis. For the 16-bit-wide VRAMS used here, a complete reversal of 16 bits (63–48, 47–32, 31–16, and 15–0) is required in order to support block-writes. Additionally, the pixel port connection to the VRAMs' SQ outputs needs to be swapped accordingly. A second implementation is used in this application report, which involves only one bus swap. While both methods are acceptable, this single swap can be more desirable as it often simplifies board routing and decreases the number of layers required to interface to the VRAM. The single swap also requires that the $\overline{\text{CAS}}$ lines be swapped as well. Schematics for this implementation are shown in Appendix B.

Normal reads and writes are not affected. The bus is bit-reversed for both reads and writes, and therefore, functions as if the data line connection is direct. In this design, which is big-endian, the data lines are bit-reversed and block-write (4x) is supported.

5 SRT Controller Register Programming

As mentioned in Section 4.1, the TMS320C80 has two on-chip SRT controllers that are responsible for scheduling the full- and split-register transfers. Requests are passed onto the transfer controller, which, in turn, actually performs the memory-to-register (read) transfer. The SRT controllers are clocked by the system shift clock (SCLK); the same SCLK that controls the serial port of the VRAMs. Like the frame timers, the SRT controllers are programmed through on-chip memory-mapped registers. The SRT controller registers are shown in Figure 10**.**

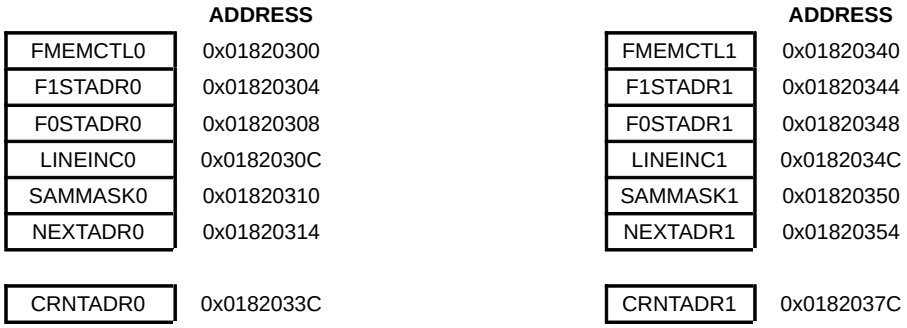
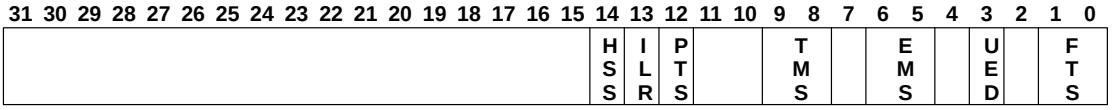


Figure 10. SRT Controller Register Map

5.1 FMEMCTL Register

Similar to the FTCTL (frame timer control) register, control over each SRT controller is facilitated through the FMEMCTL register (see Figure 11). Bits in this register control half-length SAM select, interlaced line-repeat option, and frame-timer selection among others. Also, *events* are selected with this register.



HSS – Half-SAM select
ILR – Interlace line repeat
PTS – Packet-transfer select

TMS – Transfer-mode select
00 – Display 10 – Capture
01 – Reserved 11 – Merge capture
EMS – Event-mode select
00 – SOF, line, SAM 10 – SOF, line
01 – SOF, EOF, SAM 11 – none

UED – Unblanked event disable
FTS – Frame timer sequencer
00 – ft0/disabled 10 – ft1/disabled
01 – ft0/enabled 11 – ft1/enabled

Figure 11. FMEMCTLx Register

There are four events that determine when SRT events are to be scheduled

- Start of field End of vertical blanking (once per frame)
- End of field Start of vertical blanking (once per frame)
- Line Start of horizontal blanking (once per line)
- SAM overflow Current address increment > previous tap point + SAM length

SAM overflow events are distinguished from the other event types in that they are generated from within the SRT controller itself. The point at which a SAM overflow is generated is dependent on the bus width to the VRAM port and the size of the SAM in each VRAM. These two pieces of information are stored in the SAMMASK register of the ’C80.

There are two rules that should be observed when programming the SAMMASK register.

- There are n contiguous 1’s in SAMMASK, where n = $\log_2$(split SAM length)
- The least significant 1 of SAMMASK should be aligned to correspond with the bit that increments in CRNTADR with every SCLK. Since a 64-bit bus is considered here, this is bit 3 (bits 0–2 are ignored, since the $\overline{\text{CAS}}$ lines serve as byte strobes to the VRAM).

Programming for SAMMASK is shown in Section 13.

5.2 Address Tracking

The SRT controller tracks the address of the current pixel. The address increments on every SCLK; the increment position corresponds to the least significant bit in the SAMMASK register (that is, if the least significant (LS) bit of SAMMASK is bit 3, the current address increments by 8 on each SCLK). The address is tracked in the CRNTADR register. The next address register (NEXTADR) tracks the address of the first pixel on the next line, when line events are enabled. Line and field events automatically update this register. Line events force NEXTADR to be updated by adding the line increment (LINEINC) to NEXTADR. It should be noted that the SRT address is taken from NEXTADR *before* it is updated. This pre-updated address is also stored into CRNTADR. The value in the LINEINC register represents the number of bytes between two vertically adjacent pixels in a frame.

Field events cause one or more SRTs to be performed at the address stored in the field-start-address (F0STADR) register. This register is copied into CRNTADR, and the value of F0STADR plus LINEINC is stored into NEXTADR. F1STADR is used in interlaced mode to identify the start of the other field. End-of-field events look like line events, and are used only for capture systems.

6 TVP3020 Overview

The TVP3020 video interface palette provides extensive flexibility through several key features. Among these are support for both big- and little-endian pixel formats, variable pixel bus size, and support for 24-bit true-color modes. The TVP3020 has an internal frequency doubler that provides convenient and cost-effective clock sources. The TVP3020 generates the serial shift clock, video clock, and reference clock required by the system. Additionally, the RCLK / SCLK / LCLK pixel-latching mechanism of the TVP3020 allows for very flexible control of VRAM timing.

The TVP3020 has three 8-bit digital-to-analog converters (DACs) which are sufficient to drive the RGB inputs of most monitors directly. Horizontal and vertical sync signals are passed through the device. As an option, these sync signals can be inverted. Three 256-by-8-color lookup tables (RAMs) are also provided, and are completely user-programmable. Data out of the RAMs is multiplexed with optional cursor data and overscan-boundary data (if oversacan is used) before being passed to the DACs. All color RAMs are dual-ported to support extremely fast operation. The TVP3020 comes in three speed grades: 135 MHz, 175 MHz, and 200 MHz; the speed grades are indicative of the maximum pixel clock frequency. A block diagram of the TVP3020 is shown in Figure 12.

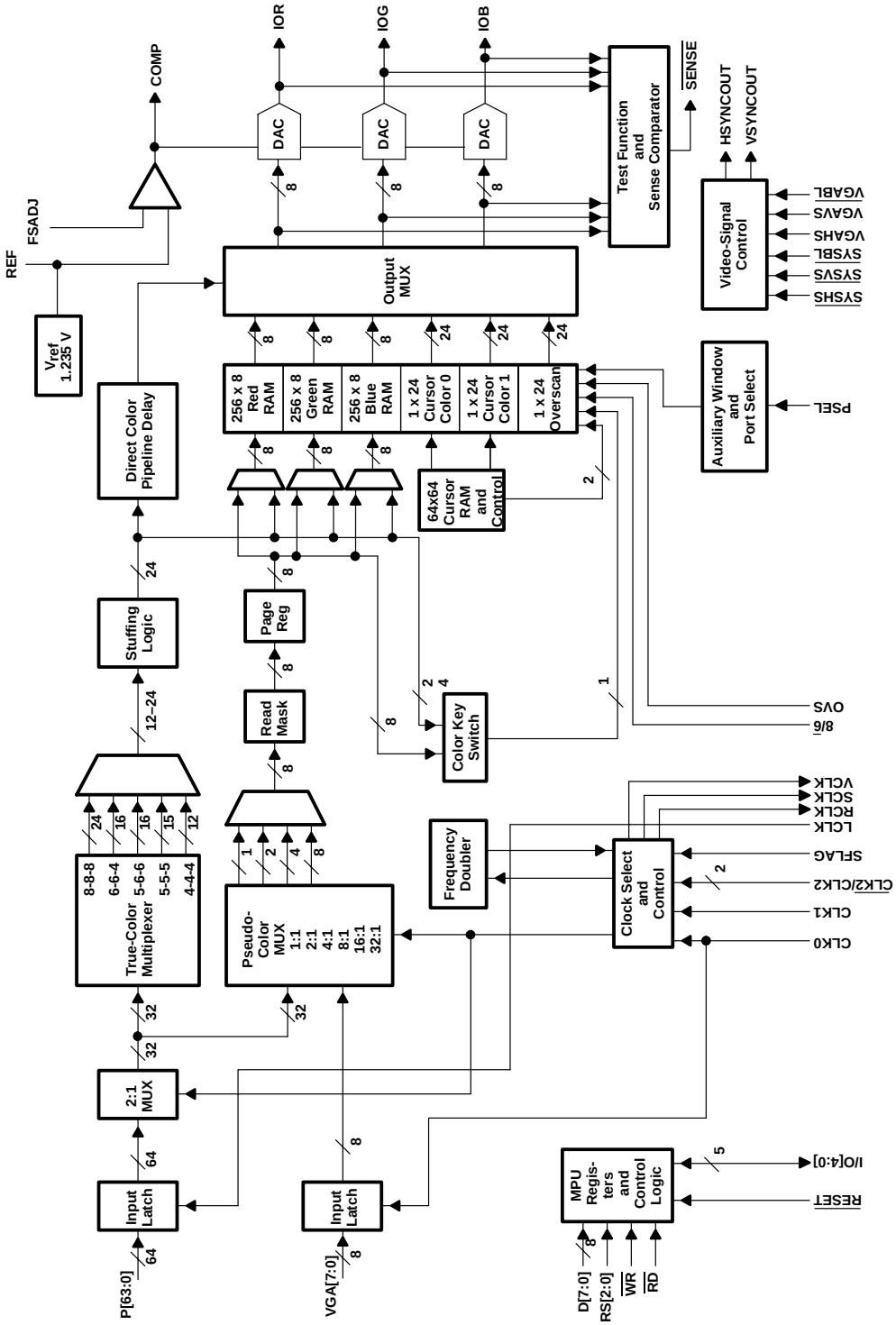


Figure 12. TVP3020 Block Diagram

6.1 TVP3020 Clocking

The TVP3020 is responsible for providing all of the clocks in the frame buffer system. All clock outputs are generated from an input reference clock. The TVP3020 allows selection between one of two TTL-compatible, or one ECL-compatible reference clock. These can be selected to drive the pixel clock directly, or can be doubled internally to produce faster pixel clock rates.

The TVP3020 provides three clock outputs: RCLK (reference clock), SCLK (shift clock), and VCLK (video or frame clock). RCLK is continuous, and is not disabled during blanking. For most applications, it is tied back into the palette as the LCLK (latch clock) input. Pixel data is latched into the device on the rising edge of LCLK. Data is requested out of the VRAMs by the rising edge of SCLK; therefore, SCLK is the same as RCLK, but it is disabled during blanking.

Because multiple pixels can be transferred on a single cycle, both the RCLK / LCLK and SCLK frequencies are a function of the desired multiplexing ratio. VCLK is used as the reference clock for generating the $\overline{\text{SYSBL}}$ (system blank, composite), $\overline{\text{SYSHS}}$ (horizontal sync), and $\overline{\text{VSYNC}}$ (vertical sync) input signals. Since these signals are provided by the ’C80, the VCLK output is tied to the ’C80’s FCLK input. Like RLCK / LCLK and SCLK, the VCLK output can be programmed as a divide-down version of the reference input.

7 System Overview

The frame buffer system described in this application report has the following architecture.

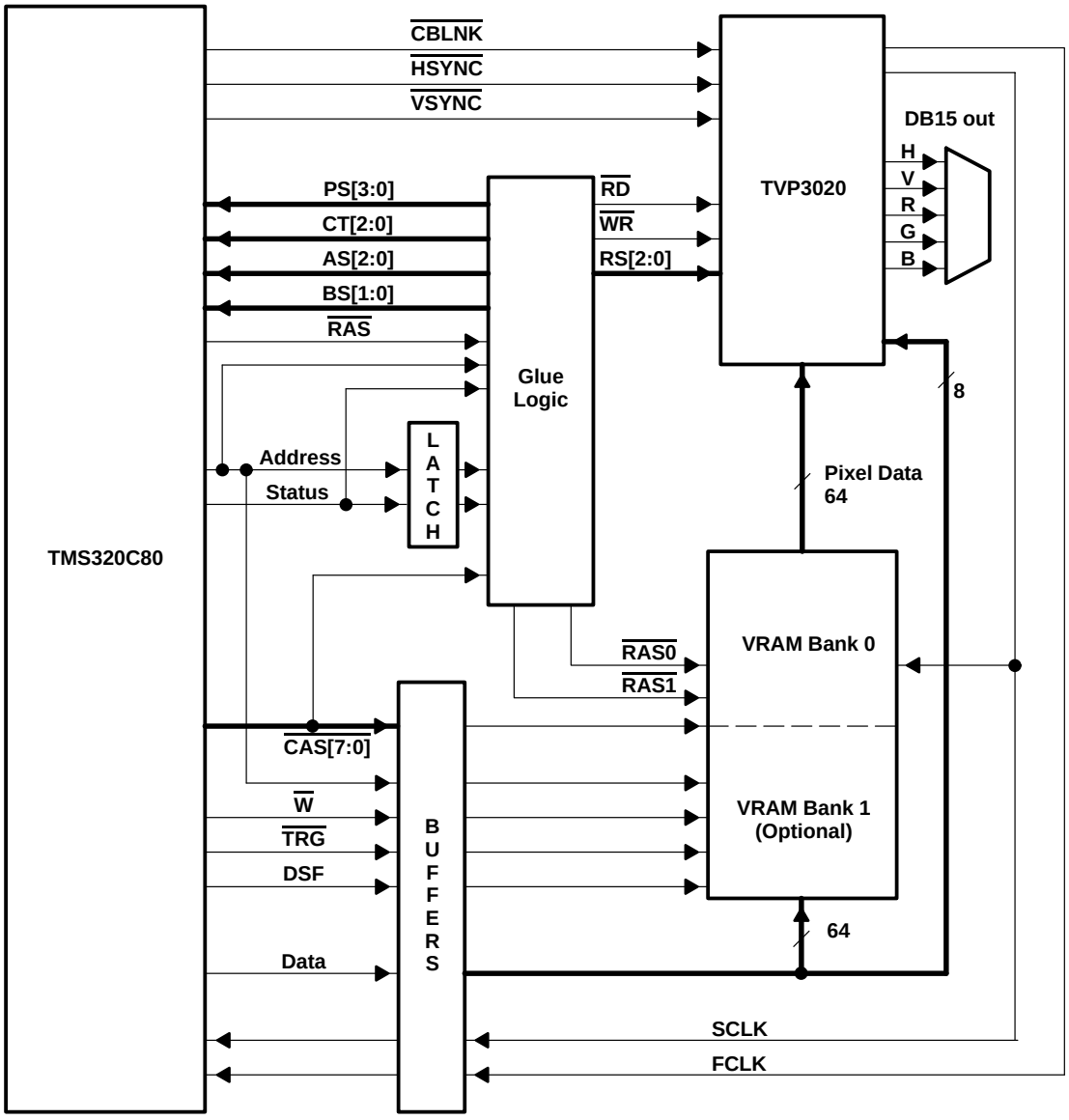


Figure 13. System Block Diagram

7.1 Video Signals

The ’C80 accepts the FCLK and SCLK inputs provided by the TVP3020. These signals are 5-V signals and, therefore, they must be buffered. A 244-type LVT buffer is used to buffer these signals. The clock inputs serve to drive one of the ’C80’s frame timers and one of the SRT controllers. In this example, frame timer 1 and SRT controller 1 are used. In turn, the ’C80 output drives $\overline{\text{HSYNC}}$, $\overline{\text{VSYNC}}$, and $\overline{\text{CBLNK}}$ to the $\overline{\text{SYSHS}}$, $\overline{\text{SYSVS}}$, and $\overline{\text{SYSBL}}$ inputs of the RAMDAC. In this example, the sync signals to the display device (DB15 connector) are driven by the palette, along with the RGB data.

7.2 VRAM Connection

The VRAM serial ports are connected to the pixel input port of the TVP3020. As discussed in Section 4.2, the pixel port does *not* need to be bit-reversed (the reversal is handled on the TMS320C80 side). SCLK from the TVP3020 controls the transfer of data from the VRAM banks.

The ’C80’s address and data buses interface with the VRAM banks, as is normally done by other banks of dynamic memory, with exception to the swapped data lines (for little-endian designs, the swap is not necessary). The data connections are made to the parallel I / O (DQ) side of the VRAMs. In this design, LVT16245 drivers are used for the data-bus connections. The VRAM banks are interfaced at 2 cycles / column for a 40-MHz design; at 50 MHz, 3 cycles / column must be used since the cycle time is violated at 2 cycles / column. Address, $\overline{\text{CAS}}$, $\overline{\text{W}}$, DSF, and the $\overline{\text{TRG}}$ signals are also buffered to the VRAM devices. This is not necessary, since the 3.3-V drive level meets the V_{IH} specification of the TMS55161 VRAMs. However, since these lines are typically heavily loaded, the addition of transceivers is advantageous.

7.3 Address Lines

The TMS55161s are arranged as 256K $\times$ 16-bit memories. As indicated on page 7–13 of the *TMS320C80 (MVP) Transfer Controller User’s Guide* (literature number SPRU105A), a 256K $\times$ n device, AS[2:0] = 010 should be used. Since the VRAM banks are addressed as 64-bit-wide memory (BS[1:0] = 11), the three least significant bits of the column address are ignored ($\overline{\text{CAS}}$ lines select bytes). Therefore, bit 3 is the least significant bit that should address the VRAM bank. For AS[2:0] = 010, bit 3 corresponds to C80A12. C80A12, is therefore, connected to A0 of the TMS55161s. Since nine bits are required for each VRAM, address lines C80A12–C80A20 are sufficient to address the entire memory array. Bank selection for the optional bank is controlled using C80A21 (see Section 7.5).

Address lines 2–0 (buffered C80A[2:0]) are also required to interface with the TVP3020. These signals are connected to the RS[2:0] inputs of the TVP3020, which decode the internal register file of the palette.

7.4 Data Lines

The TMS55161s are 5-V parts, and therefore, the data lines must be buffered to interface with the ’C80. This design supports 4x block-writes and operates in big-endian mode. This creates the need to bit-reverse the entire 64-bit data bus between the ’C80 and the VRAM bank(s). This swap is performed on the VRAM side of the data transceivers (SN74LVT16245s). While a swap on either side is acceptable, this implementation allows the transceivers to be used by other memories and peripherals without performing a dual-swap.

7.5 Glue Logic

There is a minimal amount of glue logic associated with this design. The logic in this report is implemented in PAL / GAL devices, though the equations presented here are valid for ASIC and FPGA designs as well. External logic is used to generate the cycle-configuration inputs to the ’C80, the serial-output-enables of the VRAM banks (2), separate $\overline{\text{RAS}}$ signals for the two banks, and the control inputs ($\overline{\text{RD}}$ and $\overline{\text{WR}}$) for the palette. Two banks of VRAM (4M bytes total) are considered here. Many resolutions and frame rates do not require this much frame buffer space. The inclusion of the second bank of VRAM is done here to illustrate the necessary pixel-muxing and serial output-enable-decoding that is required for a system that uses the larger frame buffer.

7.5.1 $\overline{\text{RAS}}$ Generation

In order to address the two banks of VRAM separately, a $\overline{\text{RAS}}$ signal must be generated for each bank. The signals are generated by conditioning the ’C80’s $\overline{\text{RAS}}$ signal on a valid address decode, status decode (masking out SDRAM MRS, SDRAM DCAB, and refresh cycles), and bank-select, which is determined by a latched address bit (latched with the $\overline{\text{RL}}$ output of the ’C80 at row time). Refreshes are separated into two areas (accounts for memory expansion), 0 and 1, of the refresh map; which is decoded with C80A16 and C80A17 of the refresh pseudo-address. The VRAM banks are decoded at 0xA0000000. An expression for the $\overline{\text{VRAS}}$ signals might look like the following:

```
!_VRAS0   = (!_RAS & VRAM_AV & !LC80A21 & !SDRAM_cyc & !refresh).
          #(!_RAS & refresh & !LC80A17 & !LC80A16);

!_VRAS1   = (!_RAS & VRAM_AV & LC80A21 & !SDRAM_cyc & !refresh).
          #(!_RAS & refresh & !LC80A17 & LC80A16);
```

where

```
S5 = STATUS5; externally latched by RL

S4 = STATUS4; externally latched by RL

S3 = STATUS3; externally latched by RL

S2 = STATUS2; externally latched by RL

S1 = STATUS1; externally latched by RL

S0 = STATUS0; externally latched by RL

VRAM_AV      = !LC80A31 & LC80A30 & !LC80A29 & LC80A28;

SDRAM_cyc    = !S5 & !S4 &!S3 &!S2 & S1 & S0
               #!S5 & !S4 & S3 & S2 & !S1 & !S0;

refresh      = !S5 & !S4 &!S3 &!S2 & S1 & !S0;
```

7.5.2 Serial Output Enable

In order to safely connect the serial outputs of both VRAM banks to the TVP pixel port, control logic for the VRAM serial-output-enables must be generated. This is a normal process, as the ’C80 performs the SRT cycles for both banks. While there are many solutions, one of the simplest is shown below. Again, bit A21 is used as the bank select.

```
_VSE1  = (SRT & LC80A21 & !_RAS)
        #(_VSE1 & !SRT)
        #(_VSE1 & _RAS);

_VES0  = !_VSE1;

SRT as indicated by row time status

_VSE1  serial output enable for bank 1

_VSE1  serial output enable for bank 0
```

This simple implementation guarantees that both serial outputs cannot be enabled at the same time.

7.5.3 Cycle-Configuration Inputs

The cycle-configuration inputs are made up of the AS[2:0] (address shift), BS[1:0] (bus size and block-write mode), CT[2:0] (cycle timing), and PS[3:0] (page size) signals. These signals are read in at row time, and dictate how the ’C80’s transfer controller should continue with the rest of the cycle. Additionally, the ’C80 also reads in the $\overline{\text{UTIME}}$ input, which selects the user-timed modified cycle modes. These cycles are used to interface with the TVP3020.

Column timing is dependent on the system operating speed. For 50-MHz designs, 3 cycles / column must be specified (CT[2:0] = 111). At 40-MHz CLKOUT speed, 2 cycles / column (CT[2:0] = 110) can be used. Equations are presented in this application report for both a 40- and a 50-MHz system. Page size can be calculated as row length X bus width (512 X 64), which is equal to 4K bytes (PS[3:0] = 1010). Note that this page size exists for each bank *individually*, since both banks cannot be accessed simultaneously. Address shift (AS[2:0]) is set to 010 as shown in Section 7.3, and bus size (BS[1:0]) is 11 (64 bits).

The glue logic must also respond with cycle configuration inputs when addressing the TVP3020 palette. The palette interface, which is SRAM-like in nature, requires the user to modify the timing cycles. To enable these cycles, $\overline{\text{UTIME}}$ must be sampled low at row time during the r2 state. The $\overline{\text{RAS}}$ signal is significantly modified for the user-timed accesses. It does not transition until column time, and only remains low for 1 CLKOUT cycle. External logic must use these transitions to generate the read and write signals to the RAMDAC. The TVP3020 specifies a minimum $\overline{\text{RD}}$ or $\overline{\text{WR}}$ pulse width of 50 ns. This requires a 3-cycle / column user-timed cycle with one wait state inserted by pulling the READY signal low. The cycle diagram for the palette read and writes is shown in Figure 17.

Bus size for the palette is 8 bits (BS[1:0] = 00), and page-mode cycles are disabled (PS[3:0] = 1000). Since the palette is a non-paged device, AS[2:0] = 000 for palette reads and writes.

Expressions for the cycle configuration inputs might look like those shown in the following example. In this application, VRAM banks are decoded at 0xA0000000. The optional bank 1 is contiguous to bank 0. The palette is decoded at 0x80000000. Note that in the following equations, S5–S0 refer to STATUS5–STATUS0 signals from the ’C80. These signals should be taken from the ’C80 directly *without* latching in order to meet access time. Refresh cycles are decoded using the ’C80’s pseudo-address output on C80A[31:16]. VRAM banks 1 and 0 are decoded to refresh banks 1 (C80A17 = 0, C80A16 = 1) and 0 (C80A17 = 0, C80A16 = 0), respectively.

Example 1.

Equations

```
AS2 =   SYSTEM_SPECIFIC_REQUIRING_AS2=1;

AS1 =   (VRAM_AV) & !(SDRAM_cyc # refresh)
        #(SYSTEM_SPECIFIC_REQUIRING_AS1=1);

AS0 =   (SYSTEM_SPECIFIC_REQUIRING_AS0=1);

BS1 =   (VRAM_AV) & !(SDRAM_cyc)
        #(VRAM_AV) & block_wrt
        #(SYSTEM_SPECIFIC_REQUIRING_BS1=1);

BS0 =   (VRAM_AV) & !(SDRAM_cyc # blk_wrt)
        #(SYSTEM_SPECIFIC_REQUIRING_BS0=1);

CT2 =   (VRAM_AV) & !(SDRAM_cyc # refresh)
        # VRAM_refresh
        #(TVP_AV) & !(SDRAM_cyc # refresh)
        #(SYSTEM_SPECIFIC_REQUIRING_CT2=1);

CT1 =   (VRAM_AV) & !(SDRAM_cyc # refresh)
        # VRAM_refresh
        #(TVP_AV) & !(SDRAM_cyc # refresh)
        #(SYSTEM_SPECIFIC_REQUIRING_CT1=1);

CT0 =   (VRAM_AV) & !(SDRAM_cyc # refresh)
        # VRAM_refresh
        #(TVP_AV) & !(SDRAM_cyc # refresh)
        #(SYSTEM_SPECIFIC_REQUIRING_CT0=1);

PS3 =   (VRAM_AV) & !(SDRAM_cyc # refresh)
        #(TVP_AV) & !(SDRAM_cyc # refresh)
        #(SYSTEM_SPECIFIC_REQUIRING_PS3=1);

PS2 =   (SYSTEM_SPECIFIC_REQUIRING_PS2=1);

PS1     (VRAM_AV) & !(SDRAM_cyc # refresh)
        #(SYSTEM_SPECIFIC_REQUIRING_PS1=1));

PS0 =   (SYSTEM_SPECIFIC_REQUIRING_PS0=1);

!UTIME = (TVP_AV) & !(SDRAM_cyc)
        #(!RESETIN);
```

Constants and alias names:

```
S5  =  STATUS5;

S4  =  STATUS4;

S3  =  STATUS3;

S2  =  STATUS2;

S1  =  STATUS1;

S0  =  STATUS0;
```

CYCLE CODE	VALUE (VRAM)
AS[2:0]	010 (512K X n)
BS[1:0]	11 (64-bit bus)
CT[2:0]	111 or 110
PS[3:0]	1010 (4K)

CYCLE CODE	VALUE (TVP3020)
AS[2:0]	000 (static)
BS[1:0]	00 (8-bit bus)
CT[2:0]	111 (3 cycles / column)
PS[3:0]	1000 (bus size)

```
VRAM_AV         =   C80A31 & !C80A30
                    & C80A29 & !C80A28;

TVP_AV          =   C80A31 & !C80A30 &
                    !C80A29 & !C80A28;

block_wrt       =   !S5 & !S4 & S3 & !S2 &
                    !S1 & S0;

SDRAM_cyc       =   !S5 & !S4 &!S3 &!S2 &
                    S1 & S0 #!S5 & !S4 &
                    S3 & S2 & !S1 & !S0;

refresh         =   !S5 & !S4 &!S3 &!S2 &
                    S1 & !S0;

VRAM_refresh    =   !S5 & !S4 &!S3 &!S2 &
                    S1 & !S0 &((!C80A17 &
                    C80A16) #(!C80A17 &
                    !C80A16));
```

NOTES:
SYSTEM_SPECIFIC_REQUIRING_xxx conditions should not set the cycle configuration inputs for the VRAM cycles.

Signal SDRAM_cyc does not need to be generated if SDRAM is not present in the system.

RESETIN is the system reset signal (active low).

Equations have not been simplified.

8 Timing Analysis

The VRAM interface must meet a considerable number of timing parameters. Among these are access times from both $\overline{RAS}$ and $\overline{CAS}$, address setup and hold times, data setup and hold times, and propagation delays from input to input for read, write, and refresh cycles. The parameters of interest are presented in the following timing diagrams for page-mode reads, page-mode writes, and refresh cycles, just as a standard DRAM interface. Additionally, the special VRAM cycles of block-write and load color register (LCR) must be examined. Table 4 through Table 10 contain equations and values for each of the calculated parameters. Data transceiver, $\overline{W}$, DSF, $\overline{TRG}$, $\overline{CAS}$, and address delays are considered. Also, since $\overline{RAS}$ signals for the VRAM bank(s) are generated in external logic, a maximum logic delay is considered. The following parameters are used in the timing calculations.

Table 3. Component Delays

DESCRIPTION	MNEUMONIC	PARAMETER
Maximum logic delay	$t_{PROPPALMAX}$	7.5 ns
Maximum transceiver delay	$t_{Pxx245MAX}$†	4.1 ns
Minimum transceiver delay	$t_{Pxx245MIN}$	1.0 ns
Maximum buffer delay	$t_{Pxx244MAX}$	4.1 ns
Minimum buffer delay	$t_{Pxx244MIN}$	1.0 ns

† x replaced by L and H where applicable. For example, t_{PHL} indicates high-to-low transition; t_{PLH} indicates low-to-high transition.

In addition to the VRAM interface timing, the timing for the ’C80 cycle configuration inputs should also be evaluated. As these signals are generated in external logic and feed directly back to the ’C80, the analysis is simple:

$$t_{a(MIDV-CFGV)} = t_{PROPPAL} = 7.5 \text{ ns}$$

The ’C80 only requires that $t_{a(MIDV-CFGV)}$ be less than 20 ns ($3t_H$–10) at 50 MHz.

Table 4. VRAM Timing Parameters – Access Times (2 cycles / column at 40 MHz)

NO.	PARAMETER	FORMULA (WITHOUT $\overline{CAS}$ BUFFERS)	’C80 SPECIFICATION (ns)	40 MHz RESULT (ns)
26	Access time, address	$t_{AA} + t_{Pxx244MAX} + t_{Pxx245MAX}$	$t_{a(OUTV-DV)} = 4t_H - 9 = 41$	38.2
24	Access time, $\overline{CASL}$	$t_{CAC} + t_{Pxx245MAX}$	$t_{a(CASL-DV)} = 3t_H - 12 = 25.5$	21.1
25	Access time, $\overline{CASH}$	$t_{CPA} + t_{Pxx245MAX}$	$t_{a(OUTV-DV)} = 4t_H - 9 = 41$	39.1
27	Access time, $\overline{RASL}$	$t_{RAC} + t_{PROPPALMAX} + t_{Pxx245MAX}$	$t_{a(OUTV-DV)} = 8t_H - 8 = 92$	71.6

Table 5. VRAM Timing Parameters – Setup and Hold Times (2 cycles / column; 40 MHz)

NO.	PARAMETER	FORMULA (WITHOUT $\overline{CAS}$ BUFFERS)	VRAM SPECIFICATION (ns)	40 MHz RESULT (ns)
15	Cycle time, random	$(t_{h(OUTV-OUTV)} = 14t_H - 5.5) - t_{PROPPALMAX} + t_{PROPPALMIN}$	$t_{RC} = 110$	162
14	Cycle time, page mode	$(t_{h(OUTV-OUTV)} = 4t_H - 6.5)$	$t_{PC} = 35$	43.5
2	Pulse duration, $\overline{RAS}$ high	$(t_{w(OUTV)} = 6t_H - 5.5) - t_{PROPPALMAX} + t_{PROPPALMIN}$	$t_{RP} = 60$	67.5
1	Pulse duration, $\overline{RAS}$ low	$(t_{w(OUTV)} = 8t_H - 5.5) - t_{PROPPALMAX} + t_{PROPPALMIN}$	$t_{RASP} = 60$	87
5	Pulse duration, $\overline{CAS}$ low	$(t_{w(CASL)} = 3t_H - 11)$	$t_{CAS} = 15$	26.5
6	Pulse duration, $\overline{CAS}$ high	$(t_{w(CASH)} = t_H - 2)$	$t_{CPN} = 10$	10.5
32	Pulse duration, $\overline{W}$	$(t_{w(OUTV)} = 8t_H - 5.5) - t_{PHL244MAX} + t_{PLH244MIN}$	$t_{WP} = 10$	91.4
11	Setup time, address (column)	$(t_{h(OUTV-CASL)} = t_H - 4.5) - t_{Pxx244MAX}$	$t_{ASC} = 0$	1.4
13	Hold time, address (column)	$(t_{h(CASL-OUTV)} = 3t_H - 11) + t_{Pxx244MIN}$	$t_{CAH} = 10$	27.5
8	Setup time, address (row)	$(t_{su(OUTV-OUTV)} = 6t_H - 5.5) - t_{Pxx244MAX} + t_{PROPPALMIN}$	$t_{ASR} = 0$	65.4
9	Hold time, address (row)	$(t_{h(OUTV-OUTV)} = 4t_H - 6.5) - t_{PROPPALMAX} + t_{Pxx244MIN}$	$t_{RAH} = 10$	37
34	Setup time, data to $\overline{CAS}$	$(t_{h(OUTV-CASL)} = t_H - 5) - t_{Pxx245MAX}$	$t_{DSC} = 0$	0.9
35	Hold time, data from $\overline{CAS}$	$(t_{h(CASL-OUTV)} = 3t_H - 11) + t_{Pxx245MIN}$	$t_{DH} = 15$	27.5
31	Setup time, $\overline{W}$ to $\overline{CASH}$	$(t_{h(OUTV-OUTV)} = 7t_H - 6.5) - t_{PHL244MAX}$	$t_{CWL} = 15$	76.9
33	Setup time, $\overline{W}$ to $\overline{RASH}$	$(t_{h(OUTV-OUTV)} = 7t_H - 5.5) - t_{PROPPALMAX} + t_{PLH244MIN}$	$t_{RWL} = 15$	75.5
21	Setup time, $\overline{W}$ (row)	$(t_{h(OUTV-OUTV)} = 5t_H - 5.5) + t_{PROPPALMIN} - t_{PLH244MAX}$	$t_{WSR} = 0$	52.9
18	Setup time, DSF to $\overline{RAS}$	$(t_{h(OUTV-OUTV)} = t_H - 5.5) + t_{PROPPALMIN} - t_{PLH244MAX}$	$t_{FSR} = 0$	2.9
17	Hold time, DSF from $\overline{RAS}$	$(t_{h(OUTV-OUTV)} = 9t_H - 5.5) - t_{PROPPALMAX} + t_{PLH244MIN}$	$t_{FHR} = 30$	100.5
17	Hold time, DSF from $\overline{RAS}$ (bw)	$(t_{h(OUTV-OUTV)} = 3t_H - 5.5) - t_{PROPPALMAX} + t_{PLH244MIN}$	$t_{RFH} = 10$	25.5
19	Setup time, $\overline{TRG}$ to $\overline{RAS}$	$(t_{h(OUTV-OUTV)} = 5t_H - 5.5) + t_{PROPPALMIN} - t_{PLH244MAX}$	$t_{THS} = 0$	52.9
20	Hold time, $\overline{TRG}$ from $\overline{RAS}$	$(t_{h(OUTV-OUTV)} = 3t_H - 5.5) - t_{PROPPALMAX} + t_{PLH244MIN}$	$t_{THH} = 10$	25.5
28	Setup time, DSF to $\overline{CAS}$	$(t_{h(OUTV-OUTV)} = 4t_H - 6.5) - t_{PHL244MAX}$	$t_{FSC} = 0$	39.4
29	Hold time, DSF from $\overline{CAS}$	$(t_{h(OUTV-OUTV)} = 4t_H - 5.5) + t_{PHL244MIN}$	$t_{CFH} = 10$	45.5
30	Hold time, $\overline{W}$ (row)	$(t_{h(OUTV-OUTV)} = 3t_H - 5.5) - t_{PROPPALMAX} + t_{PHL244MIN}$	$t_{WHR} = 10$	25.5

Table 5. VRAM Timing Parameters – Setup and Hold Times (2 cycles / column; 40 MHz) (Continued)

NO.	PARAMETER	FORMULA (WITHOUT $\overline{CAS}$ BUFFERS)	VRAM SPECIFICATION (ns)	40 MHz RESULT (ns)
23	Hold time, read from $\overline{RAS}$	$(t_{h(OUTV-OUTV)} = 17t_H - 5.5) - t_{PROPPALMAX} + t_{PHL244MIN}$	$t_{RRH} = 0$	200.5
22	Setup time, read to $\overline{CAS}$	$(t_{h(OUTV-CASL)} = 10t_H - 3.5) - t_{PLH244MAX}$	$t_{RCS} = 0$	117.4

Table 6. VRAM Timing Parameters – Delay Times (2 cycles / column at 40 MHz)

NO.	PARAMETER	FORMULA (WITHOUT $\overline{CAS}$ BUFFERS)	VRAM SPECIFICATION (ns)	40 MHz RESULT (ns)
38	$\overline{RASL}$ to $\overline{CASH}$ (refresh)	$(t_{h(OUTV-OUTV)} = 8t_H - 6.5) - t_{PROPPALMAX}$	$t_{CHR} = 10$	86
3	$\overline{CASH}$ to $\overline{RASL}$	$(t_{h(OUTV-OUTV)} = 6t_H - 5.5) + t_{PROPPALMIN}$	$t_{CRP} = 0$	69.5
10	$\overline{RASL}$ to $\overline{CASH}$	$(t_{h(OUTV-OUTV)} = 8t_H - 6.5) - t_{PROPPALMAX}$	$t_{CSH} = 60$	86
37	$\overline{CASL}$ to $\overline{RASL}$ (refresh)	$(t_{h(OUTV-OUTV)} = 2t_H - 5.5) + t_{PROPPALMIN}$	$t_{CSR} = 10$	19.5
12	Column address to $\overline{CASH}$	$(t_{h(OUTV-OUTV)} = 4t_H - 6.5) - t_{Pxx244MAX}$	$t_{CAL} = 30$	39.4
4	$\overline{RASL}$ to $\overline{CASL}$	$(t_{h(OUTV-CASL)} = 5t_H - 3.5) - t_{PROPPALMAX}$	$t_{RCD} = 20$	51.5
36	$\overline{RASH}$ to $\overline{CASL}$ (refresh)	$(t_{h(OUTV-CASL)} = 4t_H - 3.5) - t_{PROPPALMAX}$	$t_{RPC} = 0$	39
7	$\overline{CASL}$ to $\overline{RASH}$	$(t_{h(CASL-OUTV)} = 3tH - 11) + t_{PROPPALMIN}$	$t_{RSH} = 17$	26.5

Table 7. VRAM Timing Parameters – Access Times (3 cycles / column at 50 MHz)

NO.	PARAMETER	FORMULA	'C80 SPECIFICATION (ns)	50 MHz RESULT (ns)
26	Access time, address	$t_{AA} + t_{Pxx244MAX} + t_{Pxx245MAX}$	$t_{a(OUTV-DV)} = 6tH - 7 = 53$	38.2
24	Access time, $\overline{CASL}$	$t_{CAC} + t_{Pxx245MAX}$	$t_{a(CASL-DV)} = 4tH - 7 = 33$	21.1
25	Access time, $\overline{CASH}$	$t_{CPA} + t_{Pxx245MAX}$	$t_{a(OUTV-DV)} = 6t_H - 7 = 53$	39.1
27	Access time, $\overline{RASL}$	$t_{RAC} + t_{PROPPALMAX} + t_{Pxx245MAX}$	$t_{a(OUTV-DV)} = 10t_H - 6.5 = 93.5$	71.6

Table 8. VRAM Timing Parameters – Setup and Hold Times (3 cycles / column at 50 MHz)

NO.	PARAMETER	FORMULA	VRAM SPECIFICATION (ns)	50 MHz RESULT (ns)
15	Hold time, random	$(t_{h(OUTV-OUTV)} = 18t_H - 5) - t_{PROPPALMAX} + t_{PROPPALMIN}$	$t_{RC} = 110$	168.5
14	Hold time, page mode	$(t_{h(OUTV-OUTV)} = 6t_H - 5.5) - t_{Pxx244MAX} + t_{Pxx244MIN}$	$t_{PC} = 35$	51.4
2	Pulse duration, $\overline{RAS}$ high	$(t_{w(OUTV)} = 8t_H - 5) - t_{PROPPALMAX} + t_{PROPPALMIN}$	$t_{RP} = 60$	67.5

Table 8. VRAM Timing Parameters – Setup and Hold Times (3 cycles / column at 50 MHz) (Continued)

NO.	PARAMETER	FORMULA	VRAM SPECIFICATION (ns)	50 MHz RESULT (ns)
1	$\overline{RAS}$ low	$(t_{w(OUTV)} = 10t_H - 5) - t_{PROPPALMAX} + t_{PROPPALMIN}$	$t_{RASP} = 60$	87.5
5	$\overline{CAS}$ low	$(t_{w(CASL)} = 4t_H - 5.5) - t_{Pxx244MAX} + t_{Pxx244MIN}$	$t_{CAS} = 15$	31.4
6	$\overline{CAS}$ high	$(t_{w(CASH)} = 2t_H - 2) - t_{Pxx244MAX} + t_{Pxx244MIN}$	$t_{CPN} = 10$	14.9
32	Pulse duration, $\overline{W}$	$(t_{w(OUTV)} = 8t_H - 5) - t_{PHL244MAX} + t_{PLH244MIN}$	$t_{WP} = 10$	71.9
11	Setup time, address (column)	$(t_{h(OUTV-OUTV)} = 2t_H - 5.5) - t_{Pxx244MAX} + t_{Pxx244MIN}$	$t_{ASC} = 0$	11.4
13	Hold time, address (column)	$(t_{h(OUTV-OUTV)} = 4t_h - 5.5) - t_{Pxx244MAX} + t_{Pxx244MIN}$	$t_{CAH} = 10$	31.4
8	Setup time, address (row)	$(t_{su(OUTV-OUTV)} = 8t_H - 5) - t_{Pxx244MAX} + t_{PROPPALMIN}$	$t_{ASR} = 0$	70.9
9	Hold time, address (row)	$(t_{h(OUTV-OUTV)} = 4t_H - 5.5) - t_{PROPPALMAX} + t_{Pxx244MIN}$	$t_{RAH} = 10$	36
34	Setup time, data to $\overline{CAS}$	$(t_{h(OUTV-OUTV)} = 2t_H - 5.5) + t_{Pxx244MIN} - t_{Pxx245MAX}$	$t_{DSC} = 0$	11.4
35	Hold time, data from $\overline{CAS}$	$(t_{h(OUTV-OUTV)} = 4t_H - 6.5) - t_{Pxx244MAX} + t_{Pxx245MIN}$	$t_{DH} = 15$	30.4
31	Setup time, $\overline{W}$ to $\overline{CASH}$	$(t_{h(OUTV-OUTV)} = 7t_H - 5.5) - t_{PHL244MAX} + t_{PLH244MIN}$	$t_{CWL} = 15$	61.4
33	Setup time, $\overline{W}$ to $\overline{RASH}$	$(t_{h(OUTV-OUTV)} = 7t_H - 5) - t_{PROPPALMAX} + t_{PLH244MIN}$	$t_{RWL} = 15$	58.5
21	Setup time, $\overline{W}$ (row)	$(t_{h(OUTV-OUTV)} = 7t_H - 5) + t_{PROPPALMIN} - t_{PLH244MAX}$	$t_{WSR} = 0$	60.9
18	Setup time, DSF to $\overline{RAS}$	$(t_{h(OUTV-OUTV)} = 3t_H - 5) + t_{PROPPALMIN} - t_{PLH244MAX}$	$t_{FSR} = 0$	20.9
17	Hold time, DSF from $\overline{RAS}$	$(t_{h(OUTV-OUTV)} = 9t_H - 5) - t_{PROPPALMAX} + t_{PLH244MIN}$	$t_{FHR} = 30$	78.5
17	Hold time, DSF from $\overline{RAS}$ (bw)	$(t_{h(OUTV-OUTV)} = 3t_H - 5) - t_{PROPPALMAX} + t_{PLH244MIN}$	$t_{RFR} = 10$	18.5
19	Setup time, $\overline{TRG}$ to $\overline{RAS}$	$(t_{h(OUTV-OUTV)} = 7t_H - 5) + t_{PROPPALMIN} - t_{PLH244MAX}$	$t_{THS} = 0$	60.9
20	Hold time, $\overline{TRG}$ from $\overline{RAS}$	$(t_{h(OUTV-OUTV)} = 3t_H - 5) - t_{PROPPALMAX} + t_{PLH244MIN}$	$t_{THH} = 10$	18.5
28	Setup time, DSF to $\overline{CAS}$	$(t_{h(OUTV-OUTV)} = 3t_H - 5.5) - t_{PHL244MAX} + t_{PLH244MIN}$	$t_{FSC} = 0$	21.4
29	Hold time, DSF from $\overline{CAS}$	$(t_{h(OUTV-OUTV)} = 5t_H - 5) + t_{PHL244MIN} - t_{PHL244MAX}$	$t_{CFH} = 10$	41.9

Table 8. VRAM Timing Parameters – Setup and Hold Times (3 cycles / column at 50 MHz) (Continued)

NO.	PARAMETER	FORMULA	VRAM SPECIFICATION (ns)	50 MHz RESULT (ns)
30t	Hold time, $\overline{W}$ hold (row)	$(t_{h(OUTV-OUTV)} = 3t_H - 5) - t_{PROPPALMAX} + t_{PHL244MIN}$	$t_{WHR} = 10$	18.5
23	Hold time, read from $\overline{RAS}$	$(t_{h(OUTV-OUTV)} = 21t_H - 5) - t_{PROPPALMAX} + t_{PHL244MIN}$	$t_{RRH} = 0$	198.5
22	Setup time, read to $\overline{CAS}$	$(t_{h(OUTV-OUTV)} = 13t_H - 5.5) - t_{PHL244MAX} + t_{PLH244MIN}$	$t_{RCS} = 0$	121.4

Table 9. VRAM Timing Parameters – Delay Times (3 cycles / column at 50 MHz)

NO.	PARAMETER	FORMULA	VRAM SPECIFICATION (ns)	50 MHz RESULT (ns)
38	$\overline{RASL}$ to $\overline{CASH}$ (refresh)	$(t_{h(OUTV-OUTV)} = 10t_H - 5.5) - t_{PROPPALMAX} + t_{PLH244MIN}$	$t_{CHR} = 10$	88
3	$\overline{CASH}$ to $\overline{RASL}$	$(t_{h(OUTV-OUTV)} = 8t_H - 5) + t_{PROPPALMIN} - t_{PLH244MAX}$	$t_{CRP} = 0$	70.9
10	$\overline{RASL}$ to $\overline{CASH}$	$(t_{h(OUTV-OUTV)} = 10t_H - 5.5) - t_{PROPPALMAX} + t_{PLH244MIN}$	$t_{CSH} = 60$	88
37	$\overline{CASL}$ to $\overline{RASL}$ (refresh)	$(t_{h(OUTV-OUTV)} = 4t_H - 5) + t_{PROPPALMIN} - t_{PLH244MAX}$	$t_{CSR} = 10$	30.9
12	Column address to $\overline{CASH}$	$(t_{h(OUTV-OUTV)} = 6t_H - 5.5) - t_{Pxx244MAX} + t_{PLH244MIN}$	$t_{CAL} = 30$	51.4
4	$\overline{RASL}$ to $\overline{CASL}$	$(t_{h(OUTV-OUTV)} = 6t_H - 5.5) - t_{PROPPALMAX} + t_{PHL244MIN}$	$t_{RCD} = 20$	48
36	$\overline{RASH}$ to $\overline{CASL}$ (refresh)	$(t_{h(OUTV-OUTV)} = 4t_H - 5.5) - t_{PROPPALMAX} + t_{PHL244MIN}$	$t_{RPC} = 0$	28
7	$\overline{CASL}$ to $\overline{RASH}$	$(t_{h(OUTV-OUTV)} = 4t_H - 5) + t_{PROPPALMIN} - t_{PLH244MAX}$	$t_{RSH} = 17$	30.9

Table 10. TVP3020 Interface Timing Parameters (50 MHz)

NO.	PARAMETER	FORMULA	SPECIFICATION	50 MHz RESULT (ns)
39	Hold time, address from $\overline{W}$	$(t_{h(OUTV-OUTV)} = 12t_H - 5) - t_{Pxx244MAX} + t_{PHL244MIN}$	10	111.9
40	Setup time, address to $\overline{W}$	$(t_{h(OUTV-OUTV)} = 8t_H - 5.5) - t_{PHL244MAX} + t_{Pxx244MIN}$	10	71.4
41	Pulse duration, $\overline{W}$	$(t_{w(OUTV)} = 7t_H) - t_{PROPPALMAX}$	50	62.5
42	Setup time, data to $\overline{W}$	$(t_{h(OUTV-OUTV)} = 7t_H - 5) - t_{Pxx245MAX} + t_{PLH244MIN}$	10	60.9
43	Setup time, data to read	$(8t_h - (t_{en1} = 40) - t_{Pxx245MAX} + t_{PLH244MIN}$	6.1	36.9

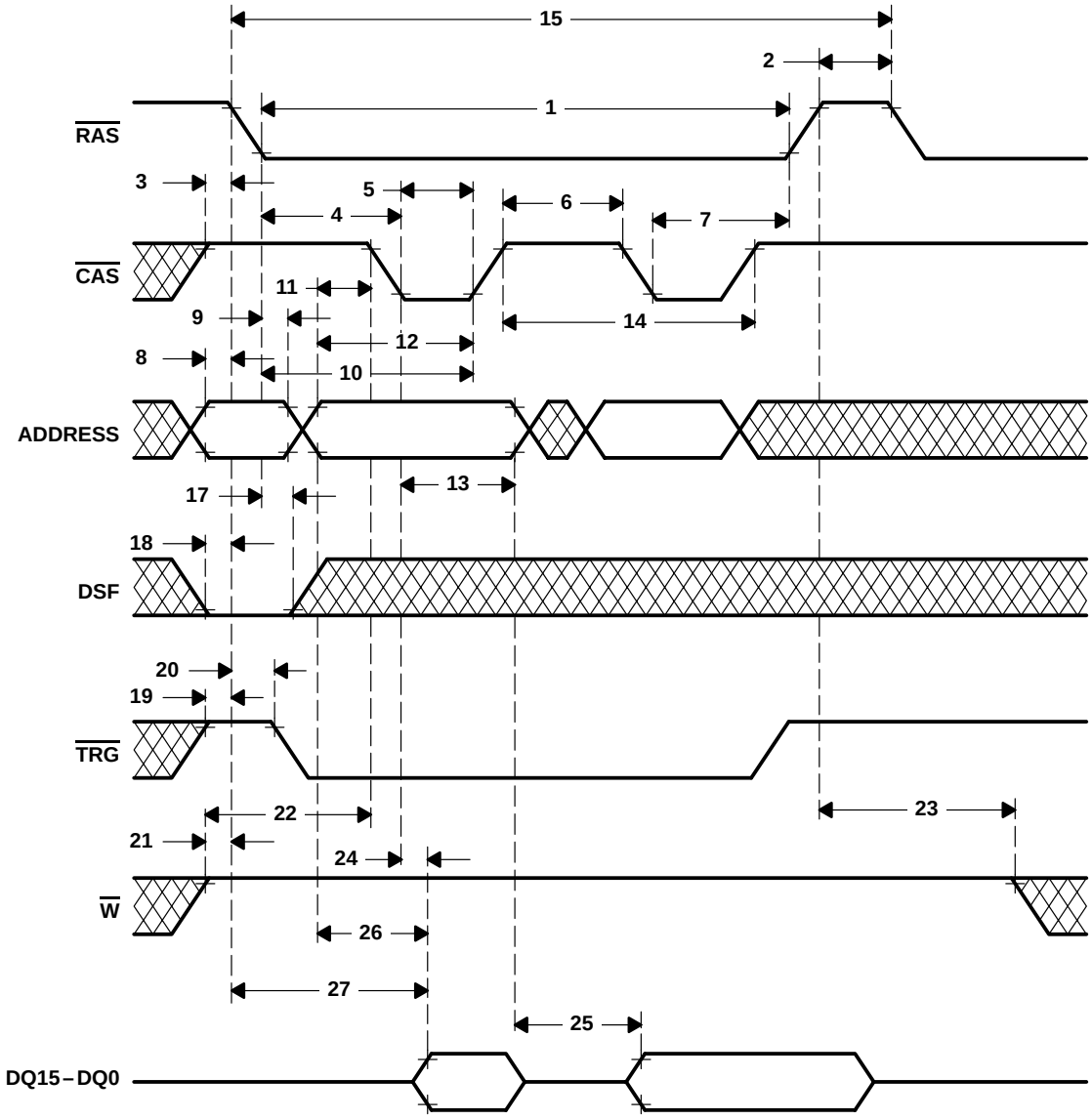
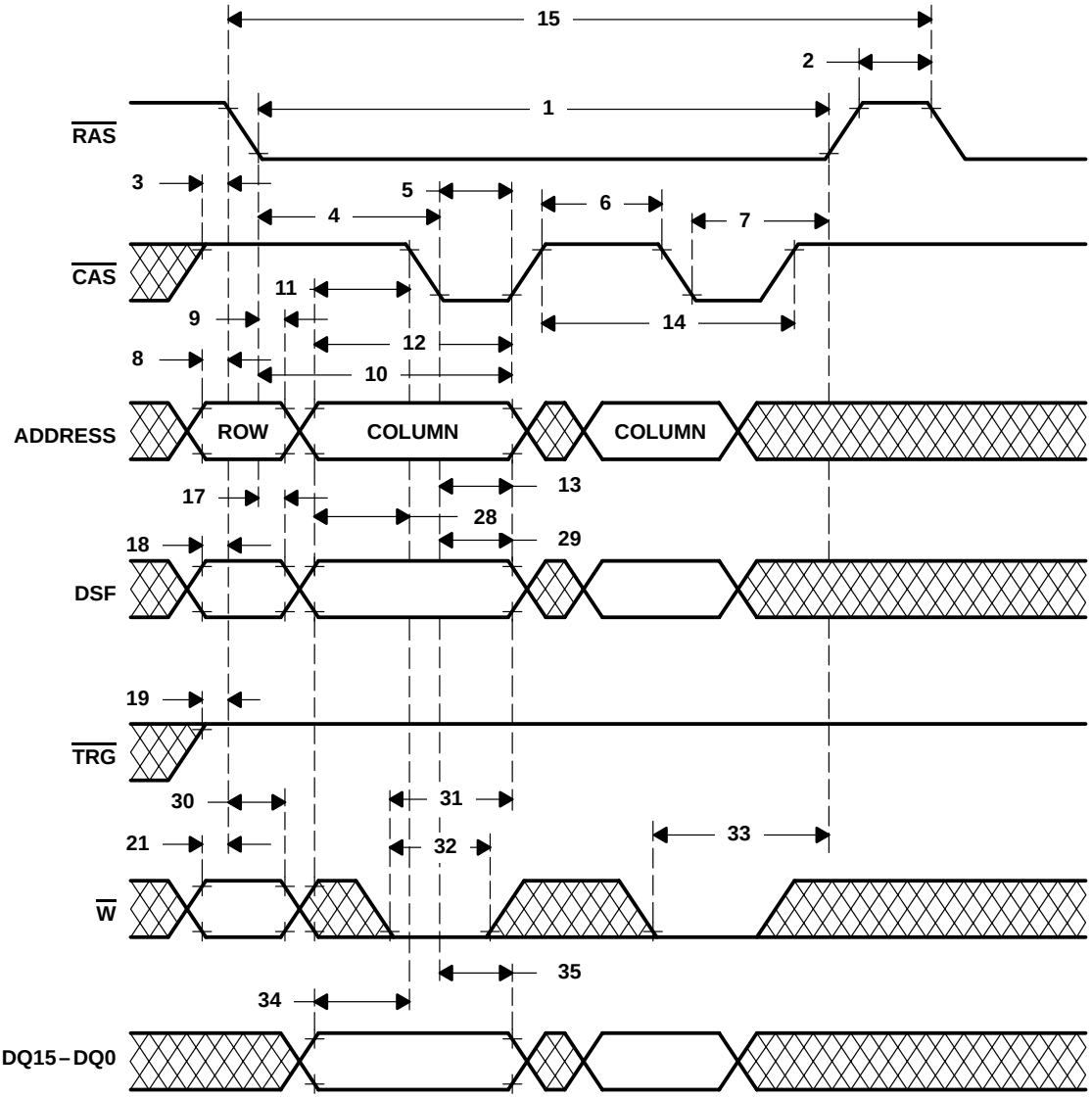


Figure 14. VRAM Read Cycle (Page Mode)



NOTE: Parameters apply to block-write and read SRTs also.

Figure 15. VRAM Write Cycle (Page Mode)

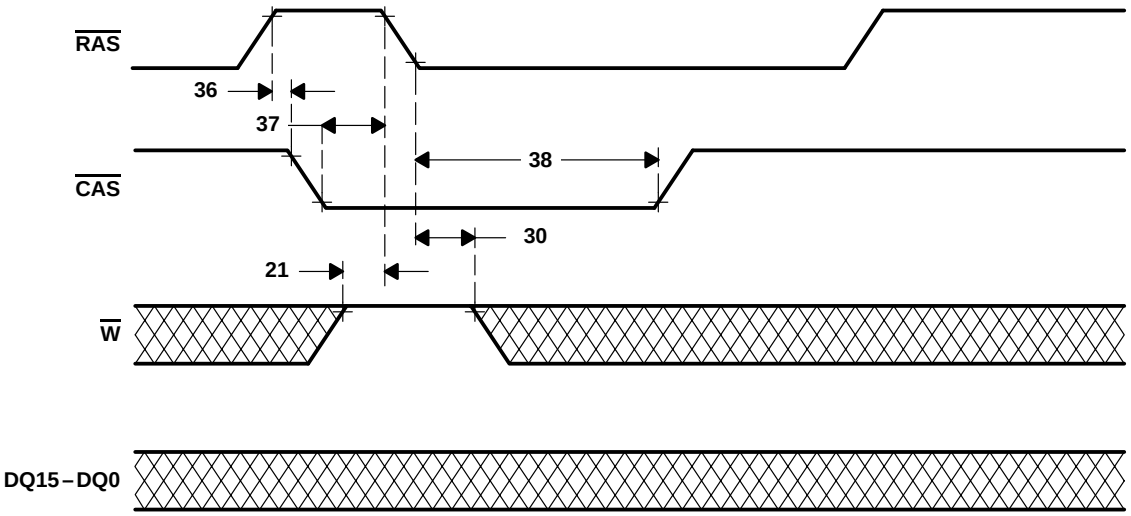


Figure 16. VRAM Refresh Cycle

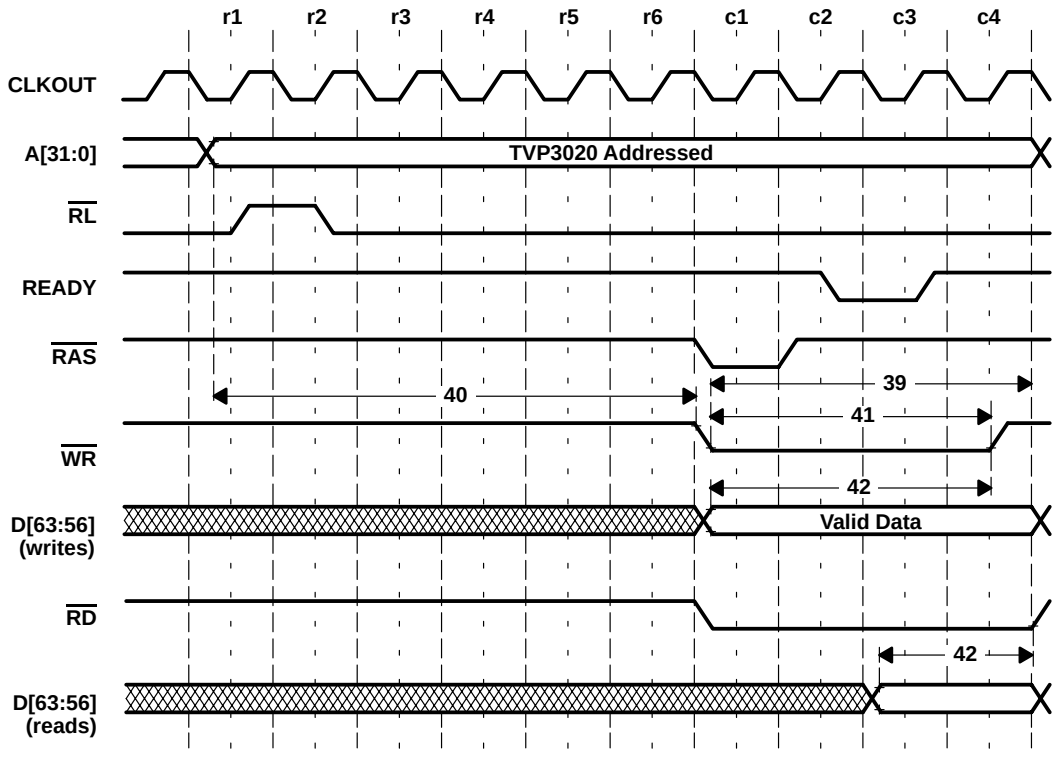


Figure 17. TVP3020 Interface Timing

9 Pixel Port Timing

In addition to the ’C80 interface timing, attention should be paid to the latching of data from the VRAM serial outputs into the RAMDAC pixel port (P[63:0]).

The TVP3020 latches pixel data on the P[63:0] bus on the rising edge of latch clock (LCLK). In this application, the LCLK input is tied directly to the RCLK output. An internal delay on the LCLK input guarantees proper operation. When the $\overline{\text{SYSBL}}$ signal goes high (beginning of active line), the first SCLK pulse is generated, which causes the VRAM serial outputs to push out the first group of data. On the next SCLK positive-going edge, the first group of data is latched into the RAMDAC, and the VRAM clocks out the second group. This action continues throughout the active line, and consequently, the last SCLK pulse occurs after the $\overline{\text{SYSBL}}$ signal has gone low to force the VRAM to output the final data group. Since this “pipeline” exists at both the beginning and end of a line, the net effect is not noticeable by the ’C80 or in the display.

A timing parameter of particular importance is the VRAM serial output access time from the SCLK input, as this determines the maximum SCLK frequency. The TMS55161s used in this application report specify $t_{a(SQ)}$ at 15 ns, which limits the maximum SCLK frequency to 67 MHz. Generally, the SQ outputs of the VRAM are tied directly to the P port of the RAMDAC. As this path can be a very high-speed one, care should be taken to make it as short as possible.

In some systems, the access time from SE also requires a strong consideration. However, as this signal is generated from logic tied to the ’C80 and not the SCLK input, it is not an issue in this design. Note that the switch from one bank to another (SE toggle) is expected to be performed during vertical blanking, presumably during a switching of frame buffers. Switching between the two VRAM banks in general should be avoided during display, as it can produce undesirable results.

10 PCB Layout Considerations

As the TVP3020 is a mixed-signal (analog and digital) part, special considerations must be made during the PCB layout stage. It is recommended that at least four layers be provided for the TVP interface: one for 5-V power, one for ground, and two signal layers. The layout should be optimized for low noise on the power and ground planes by shielding digital inputs and good decoupling. Additionally, the VRAM banks should be as close as possible to the TVP pixel port.

10.1 Power Planes

Split power planes are recommended for the TVP3020. One of these planes is for the TVP's analog circuitry, the other is for the digital interface. The split in the two planes should be made underneath the TVP3020, as shown in Figure 18. Connection between the two planes is made with a Ferrite Bead (Fair-Rite 2743001111) — this bead should be located as close as possible to where the power supply connects to the board.

The ground plane for both the analog and digital planes is the same. The use of separate ground planes is not recommended and should be actively discouraged.

10.2 Supply Decoupling

For optimal performance, a 0.1-μF and 0.01-μF capacitor pair should be used to decouple each of the groups of power terminals to ground. These capacitors should be placed as close as possible to the device, as shown in Figure 18.

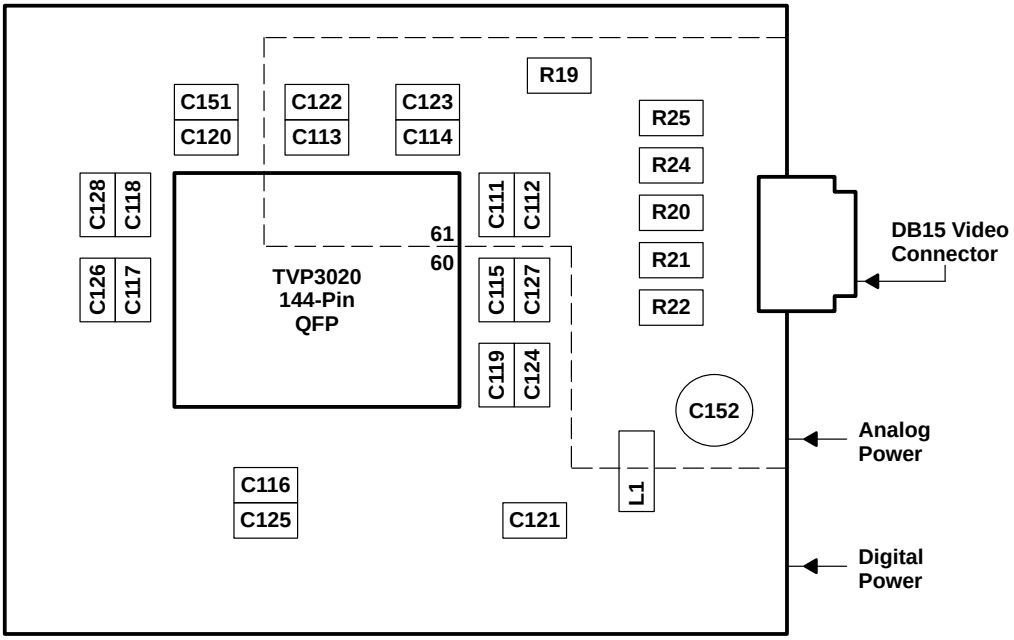


Figure 18. Component Placement for Split Power Plane

11 Clock Considerations

This section discusses several considerations regarding the frame timer and video clocks as they relate to the display that the system is meant to control. In this application report, a 1024 × 768 frame buffer is implemented, using 16-bit pixels. The equations presented in the following sections can be easily modified for other resolutions.

11.1 Screen Resolution

Screen resolution is a function of several factors, including the monitor timing, VRAM size and speed, and pixel size. The maximum possible screen resolution is determined by the pixel clock frequency necessary to display it, whether or not the hardware supports it, and the size of the VRAM being used. To determine if the resolution desired is supported by the VRAM, one must first determine the pixel size. This is often a function of the pallette and/or RAMDAC that interfaces with the monitor. To determine the required frame memory size, multiply the pixel size times the number of pixels per screen . For example, two megabytes of VRAM memory would support the following screen resolutions:

- 1024 by 512 pixels at 32 bits/pixel
- 1024 by 1024 pixels at 16 bits/pixel
- 1280 by 1024 pixels at 8 bits/pixel

In this application report, four megabytes of VRAM are used, so many more resolutions are possible. For the purposes of this example, a 1024 x 768 display with 16-bit pixels is considered.

11.2 Monitor Specifications

The monitor in this example has the following specifications. This information can typically be found in the user's guide supplied with the monitor.

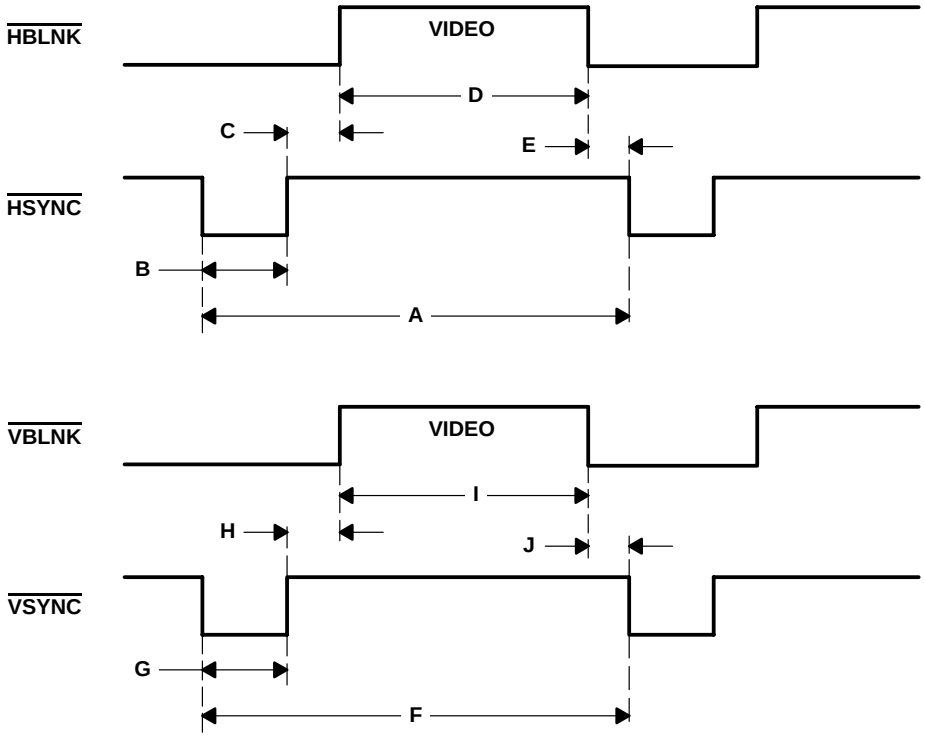


Figure 19. Noninterlaced Monitor Timing (Separate SYNC)

Table 11. Monitor Timings (Typical)†

LINE RATE	48.5 kHz		FRAME RATE	60 Hz	
A (µs)	20.625	LD – line duration	F (ms)	16.665	FD – frame duration
B (µs)	1.0	HS – horizontal sync	G (µs)	83	VS – vertical sync
C (µs)	2.875	HBP – horizontal back porch	H (µs)	660	VBP – vertical back porch
D (µs)	16.0	HAT – horizontal active time	I (µs)	15.84	VAT – vertical active time
E (µs)	0.75	HFP – horizontal front porch	J (µs)	82	VFP – vertical front porch

† Resolution: 1024 pixels by 768 lines

11.3 Pixel Clock Selection

The timing specifications of the monitor limit the available pixel clock frequencies. The pixel clock rate is influenced by the amount of horizontal and vertical blanking needed to conform to the monitor specifications. Use the following horizontal timing information to determine the appropriate FCLK frequency:

horizontal active time (HAT) = LD – HS – HPB – HFP = 16.000 µs

At 1024 pixels per line, this corresponds to 15.625 ns/pixel. This is known as the dot clock rate, or the rate at which pixels must be output from the RAMDAC.

In this design, the VRAM frame buffers are 64 bits wide; therefore, four 16-bit pixels can be output on each SCLK cycle and, consequently, SCLK should be dot clock/4. To support the required pixel rate, dot clock needs to be 64 MHz (1/15.625 ns); therefore, SLCK is 16 MHz. The maximum FCLK frequency is determined by the speed grade of the 'C80 that is used. While many combinations of FCLK frequency and register setting provide a suitable result, a few guidelines ensure that an optimal solution is achieved. The first point is that it should be noted that the FCLK/SCLK ratio determines the overall granularity with which a display can be controlled. For example, if FCLK is set to SCLK/4, then the display can only be of certain widths equal to four times the number of pixels shifted out per SCLK cycle. The second point is that the FCLK should not be greater than SCLK as this can cause undesirable effects with some RAMDACs (for example, $\overline{\text{SYSBL}}$ switching in the middle of an SCLK cycle, rather than at the end). For this application, an FCLK frequency of 16 MHz was chosen. Previously, it was mentioned that the VCLK output of the TVP3020 is intended to be used to generate the $\overline{\text{SYSBL}}$ input. As this is provided from the 'C80 ($\overline{\text{CBLNK1}}$ output), VCLK should be connected to FCLK1; while SCLK goes to both the 'C80's SCLK1 input and the VRAM SCLK inputs.

The TVP3020 drives both of the clock signals to the ’C80. Both signals are derived from the reference (dot) clock input of the TVP, which can be up to 140 MHz. In this example, a 64-MHz TTL-type oscillator is used to drive the CLK0 input of the TVP3020, and serves as the dot clock directly. The divide-down ratios for SCLK and FCLK (VCLK) are programmed in the TVP’s output-clock-selection register (indirect address 0x1B). A complete set of register settings is provided in Table 12.

Table 12. TVP3020 Register Settings

TVP REGISTER	VALUE
MUX control register 1	0x45 (see Note 3)
MUX control register 2	0x04 (see Note 3)
Input clock selection register	0x00 (CLK0 input)
Output clock selection register	0x52 (VCLK = SCLK = CLK0 / 4)
General control register	0x48 (see Note 4)
Auxiliary control register	0x08 (seld clock select)

NOTES: 3. Mux control registers 1 and 2 select the pixel format and multiplexing ratios for the TVP3020. The above settings are valid for 16-bit true color pixels in the 5-6-5 format. For a complete set of register settings, please refer to the *TVP3020 Video Interface Palette Data Manual* (literature number SLAS080A)

4. This register setting selects the big-endian pixel format and enables overlay. While not required (or used in this design), the overlay feature is useful for a variety of applications. The ’C80’s CAREA output is a general-purpose output that provides the overscan input to the TVP. Please refer to the *TVP3020 Video Interface Palette Data Manual* (literature number SLAS080A) for more information on overscan.

12 Frame Timer Register Programming

Table 13 summarizes frame timer regiser configurations for noninterlaced video. The horizontal timing registers are programmed in terms of an integral number of FCLK cycles. VCOUNT increments only after HCOUNT = HTOTAL for noninterlaced video. Therefore, with one exception, all of the vertical timing registers are programmed in terms of an integral number of horizontal lines. For purposes associated with composite interlaced video, the VESYNC register detects the end of vertical sync at half the value it is programmed to, and therefore represents twice the number of horizontal lines in vertical sync. Since all of the timing registers begin at 0, their contents represent the number of appropriate events minus one, as described in Table 13.

Table 13. Video Timing Registers (Noninterlaced)

REGISTER	IS PROGRAMMED TO ...
HESYNC	(number of FCLKS in horizontal sync) – 1
HEBLNK	(number of FCLKS from start of horizontal sync to end of horizontal blanking) – 1
HSBLNK	(number of FCLKS from start of horizontal sync to start of horizontal blanking) – 1
HTOTAL	(number of FCLKS in horizontal line) – 1
HESERR†	(number of FCLKS in horizontal serration) – 1
VESYNC	(twice the number of lines in vertical sync) – 1
VEBLNK	(number of lines from start of vertical sync to end of vertical blanking) – 1
VSBLNK	(number of lines from start of vertical sync to start of vertical blanking) – 1
VTOTAL	(number of lines in the frame) – 1

† HESERR can be left unprogrammed for noncomposite video.

12.1 Procedure

Use the steps in Table 14 and refer to Figure 20 to configure the frame timer registers for a 1024 × 768 display. Use the monitor timings described in Table 15. Remember that FCLK is 16 MHz (T = 62.5 ns).

Table 14. Frame Timer Register Programming†

STEP	CALCULATE	FORMULA	RESULT	REGISTER VALUE
1	Number of FCLKs per line	$\frac{LD}{T_{FCLK}}$	330	HTOTAL = Result – 1 = 0x149
2	Number of FCLKs in horizontal sync	$\frac{HS}{T_{FCLK}}$	16	HESYNC = Result – 1 = 0x000F
3	Number of FCLKs in horizontal back porch	$\frac{HBP}{T_{FCLK}}$	46	N/A
4	Number of FCLKs from start of horizontal sync to end of horizontal blank	Add results from step 2 and step 3	62	HEBLNK = Result – 1 = 0x003D
5	Number of FCLKs in horizontal front porch	$\frac{HFP}{T_{FCLK}}$	12	N/A
6	Number of FCLKs from start of horizontal sync to start of horizontal blank	Subtract the result of step 5 from step 1	318	HSBLNK = Result – 1 = 0x013D
7	Number of lines per frame	$\frac{FD}{LD}$	808	VTOTAL = Result – 1 = 0x0327
8	Number of lines in vertical sync	$\frac{VS}{LD}$	4	VESYNC = 2 × Result – 1 = 0x0007
9	Number of lines in vertical back porch	$\frac{VBP}{LD}$	32	N/A
10	Number of lines from start of vertical sync to end of vertical blank	Add results from step 8 and step 9	36	VEBLNK = Result – 1 = 0x0023
11	Number of lines in vertical front porch	$\frac{VFP}{LD}$	4	N/A
12	Number of lines from start of vertical sync to start of vertical blank	Subtract the result of step 11 from step 7	804	VSBLNK = Result – 1 = 0x0323

† All results rounded to the nearest whole number.

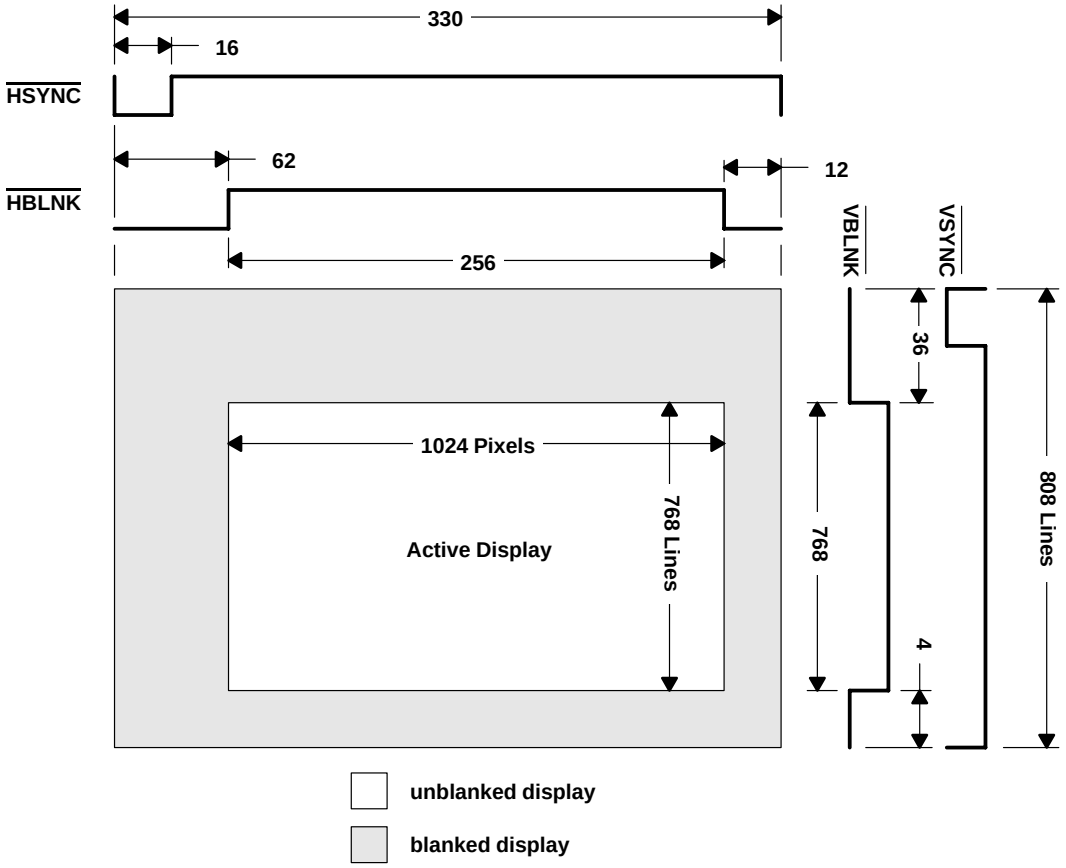


Figure 20. 1024 × 768 Display (Noninterlaced)

Table 15. Frame Timer Registers

FT REGISTER	VALUE
HESYNC1	0x000F
HEBLNK1	0x003D
HSBLNK1	0x013D
HTOTAL1	0x0149
VESYNC1	0x0007
VEBLNK1	0x0023
VSBLNK1	0x0323
VTOTAL1	0x0327

13 SRT Controller Registers

Programming the SRT controller registers requires information about the VRAMs that are used. To program these registers, calculate the memory address of the first pixel displayed and the difference in the memory addresses between two vertically adjacent pixels; 1024 $\times$ 2 bytes = 2048 (in this case). This should be placed in the LINEINC register. The address of the first pixel displayed is most often the base address of the VRAM, though the ’C80 places no restrictions on this. This address should be programmed into F0STADR, F1STADR, NEXTADR, and CRNTADR. NEXTADR and CRNTADR are automatically updated by the VC during display, but need to be programmed during initialization.

The TMS55161 VRAMs have a row length of 512 bits with a SAM length of 256 bits. The split SAM length is therefore 128 bits. The ’C80’s SRT controller is responsible for scheduling SRT events to the TC, which in turn performs either full- or split-register transfers. These transfers move data from the VRAM memory array to the SAM inside the VRAM, which, is then, transferred out serially to the TVP3020 at the SCLK rate. To prevent corruption of the display, the register transfers must be scheduled and performed such that the inactive half of the SAM is updated (see Figure 8).

In order to ensure that memory-to-register transfers occur at proper times, the SRT controller must know the size of the VRAM being used. This information is entered in two registers: FMEMCTL and SAMMASK. The SAMMASK register is programmed with a number of contiguous 1’s (n), where 2^n is the split SAM length. In this example, n is 7 since 2^7 = 128. The placement of these seven contiguous 1s in the SAMMASK register depends on what address pins are wired to the VRAM bank. In this application, each VRAM bank is 64 bits wide, and therefore, the least significant *logical* bit of address is A3 (this is the bit that toggles from one column address to an adjacent one). The least significant bit of the contiguous 1s in SAMMASK should correspond to the least significant bit of the address bus wired to the VRAM. Therefore, SAMMASK1 should be programmed to 0x000003F8 (that is, seven contiguous 1s, least significant 1 in bit 3).

The FMEMCTL0 register controls how and when data is transferred to and from the VRAM. In this example, FMEMCTL1 is programmed to 0x00004043. This enables clocking from frame timer 1 and sets the HSS (half SAM select) bit, since the SAM is half the length of a VRAM row. SRT events (see Table 16) are scheduled at the start of the field and when the SAM overflows (end of field events are for capture only). Line events are not used in this application. For systems that require line events, they can be enabled through the FMEMCTLx register. In those systems, the HALINE or HBLINE register must also be programmed to schedule the line events, presumably during blanking to prevent corruption of the display (for example, HALINE/HBLINE = HSBLNK + 1).

Table 16. SRT Controller Register Values

SRT CONTROLLER REGISTER	VALUE
FMEMCTL1	0x00004043
SAMMASK1	0x000003F8
LINEINC1	0x00000800 (1024 2-byte pixels)
CRNTADR1	0xA0000000 (base of VRAM)
NEXTADR1	0xA0000000 (base of VRAM)
F0STADR1	0xA0000000 (base of VRAM)
F1STADR1	0xA0000000 (base of VRAM)

14 A Note on Frame-Timer Interrupts

Another key feature of the ’C80’s on-chip frame timer mechanism is the ability to interrupt the MP to maintain synchronization between the application code and the display. Frame-timer interrupts are controlled by way of a memory-mapped register in the VC, VFTINTx. VFTINT, like the rest of the vertical timing registers, is programmed in terms of horizontal lines (halflines for interlaced modes).

Setting up a frame-timer interrupt involves three steps:

- Programming VFTINTx
- Setting up the interrupt vector
- Enabling the interrupt

VFTINTx should be programmed to the line number after which you want the interrupt to occur. For many applications, this corresponds to the last line of active display, as this identifies the point at which it is all right to begin writing over the entire frame buffer for the next frame. However, the ’C80 places no restrictions on the location of the interrupt. The interrupt is generated when the condition HCOUNT=HTOTAL (end of a line) on the line in which VCOUNT = VFTINT.

The frame-timer-interrupt vectors are located at the following addresses, in the MP’s parameter RAM:

Frame Timer 0	0x010101A0
Frame Timer 1	0x010101A4

The interrupt vector should be loaded with the address of the first instruction to be executed when the interrupt occurs.

The final step is to enable the interrupt by setting the appropriate bits in the MP’s INTEN register. As a good practice, you should clear the INTPEN register before doing this to avoid unwanted interrupts. The enable bits for the frame-timer interrupts are bit 8 (frame timer 0) and bit 9 (frame timer 1). When an interrupt occurs, the correcponding bit in the MP’s INTPEN register is set. The interrupt-service routine should clear this bit by writing a 1 to it in the INTPEN register before exiting.

Frame-timer interrupts are often used for double-buffering applications, is which a frame is being drawn in one part of the memory while the previous frame is being displayed. The frame-timer interrupt is used to reprogram the F0STADR0, F0STADR1, CRNTADR, and NEXTADR registers to display the next frame. The ISR also tracks the current frame so that the application maintains syncronization and does not write to the active frame. It should be noted that in this application, the VFTINT register should be programmed to a value between VSBLNK and VTOTAL, so that the writing of the aforementioned registers does not occur during the active display.

Appendix A Bill of Materials

QUANTITY	TYPE	DESIGNATORS
62	0.1 µF	C111 C112 C113 C114 C115 C116 C117 C118 C119 C120 C121 C55 C56 C57 C58 C59 C60 C61 C62 C63 C64 C65 C66 C67 C68 C69 C70 C71 C72 C73 C74 C75 C76 C77 C78 C79 C80 C81 C82 C83 C84 C85 C86 C87 C88 C89 C90 C91 C92 C93 C94 C95 C96 C97 C98 C99 C100 C101 C102 C103 C104 C105
8	0.01 µF	C122 C123 C124 C125 C126 C127 C128 C151
3	4.7 µF	C106 C107 C108
6	10 kΩ	R1 R2 R3 R4 R7 R18
1	523 Ω	R19
1	Ferrite Bead	L1
5	75 Ω	R20 R21 R22 R23 R24
4	22 Ω	RP5 RP6 RP7 RP8
2	22 µF	C109 C110
1	33 µF	C152
2	74LVT16244	U47 U48
4	74LVT16245	U26 U27 U28 U34
1	74LVT16373	U45
1	14-pin header	U14
1	HD15 connector (female)	U46
2	Crystal oscillators	U2 U12
1	Crystal oscillator (3.3 V)	U1
1	Pushbutton switch	S1
1	TL7705A	U49
8	TMS55161-60	U15 U16 U18 U19 U20 U22 U24 U25
1	TMS320C80GF-50	U13
1	TVP3020	J1
4	PAL22LV10	U35 U36 U37 U43

Appendix B Schematics

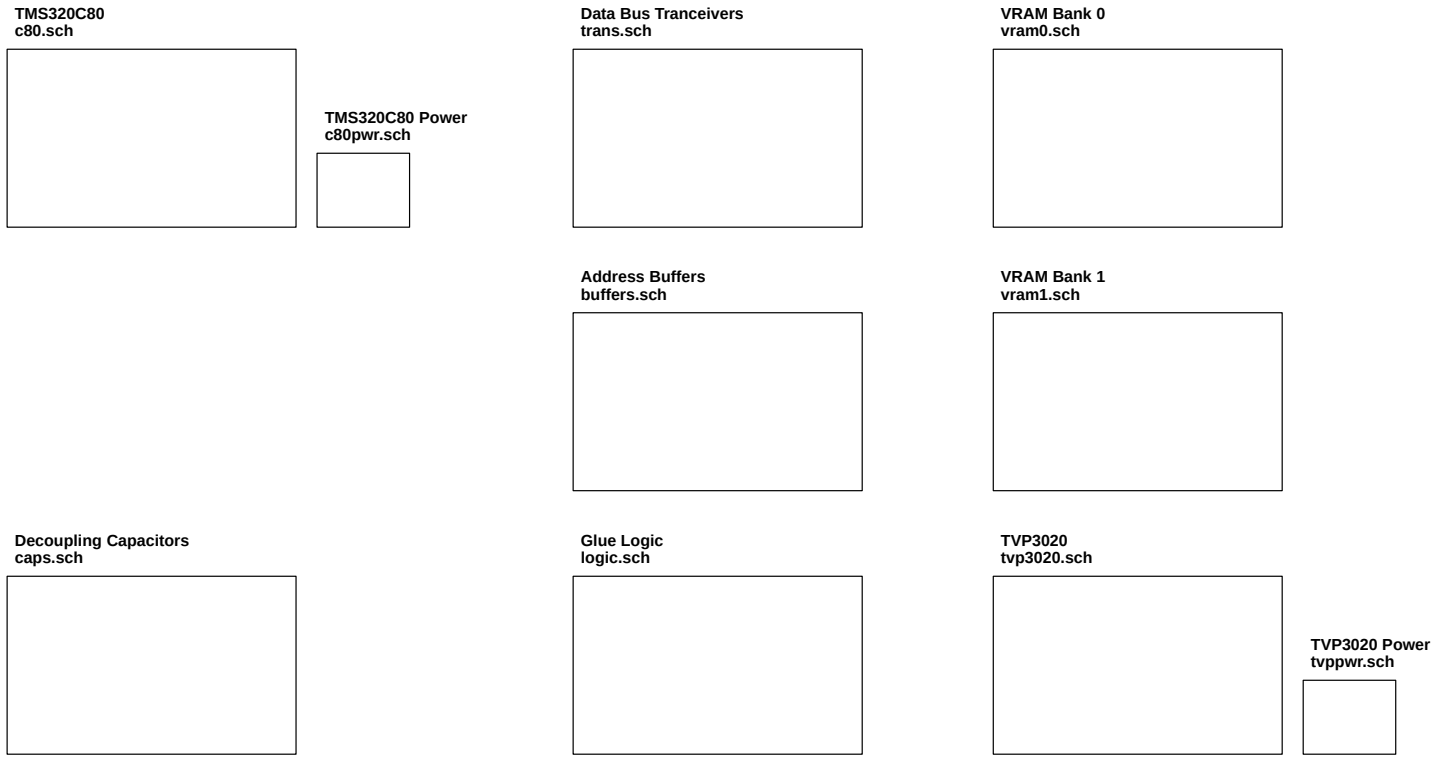


Figure B–1. Schematics

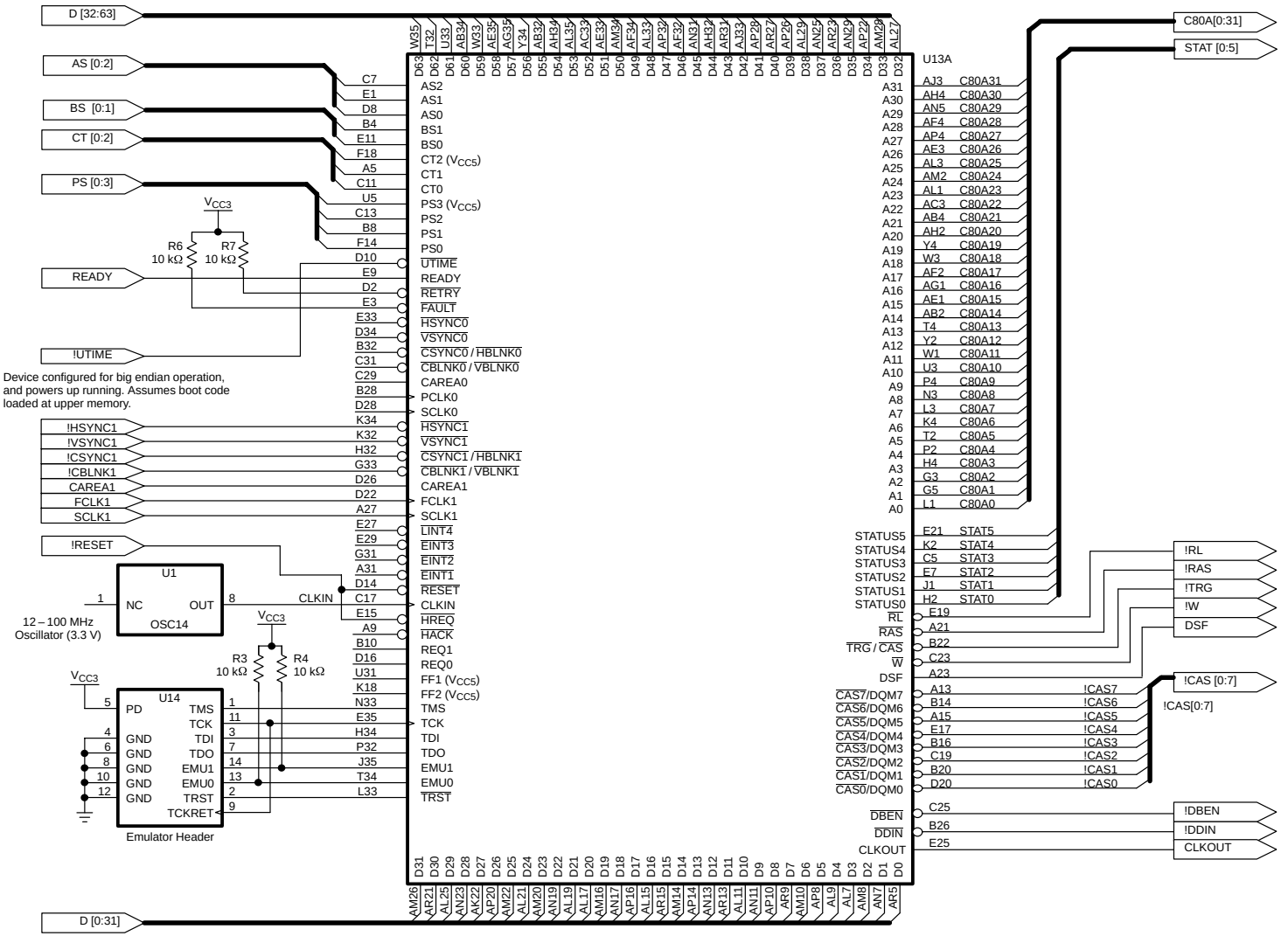


Figure B–2. TMS320C80-GF

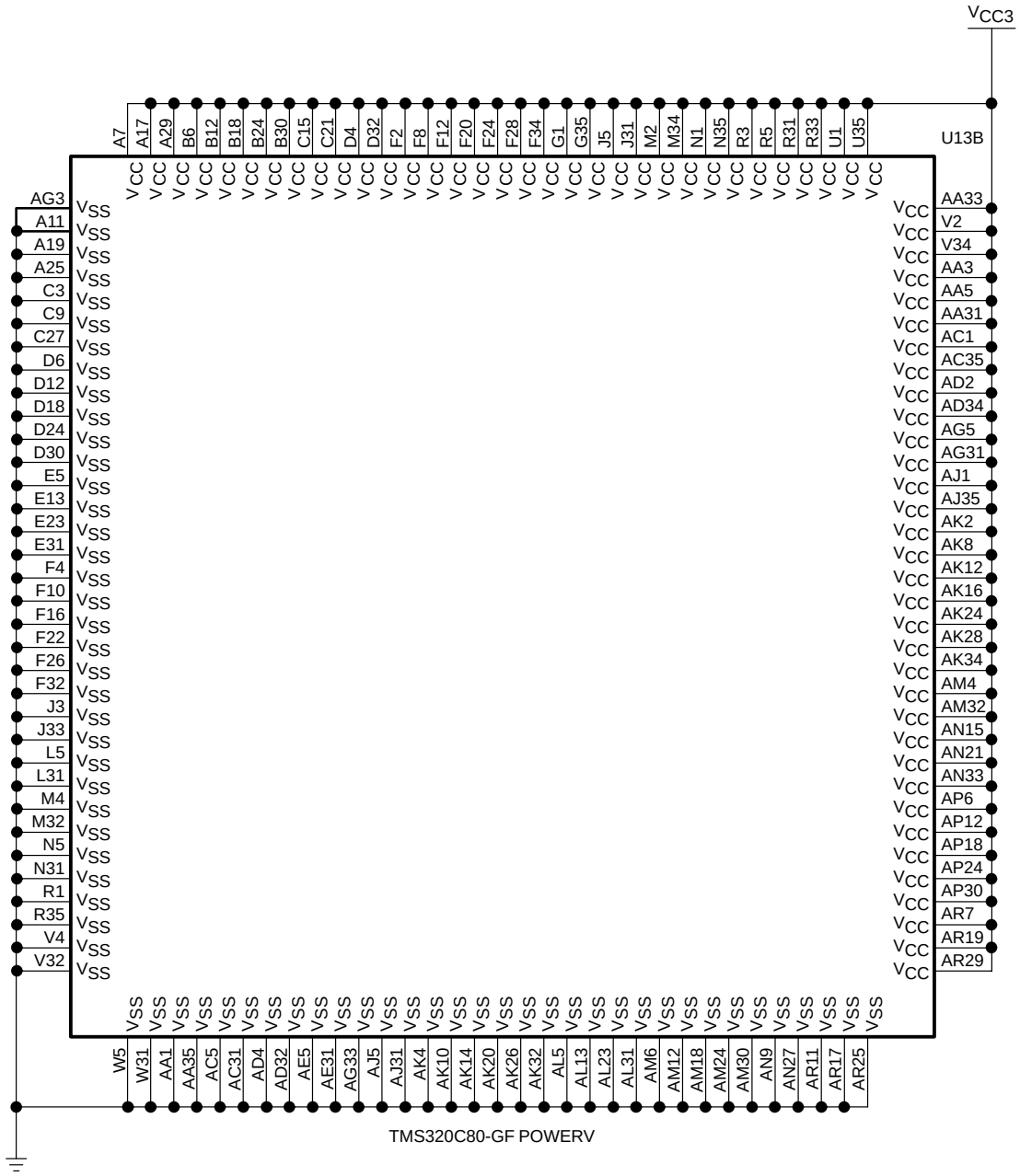


Figure B–3. TMS320C80 Power and Ground Connections

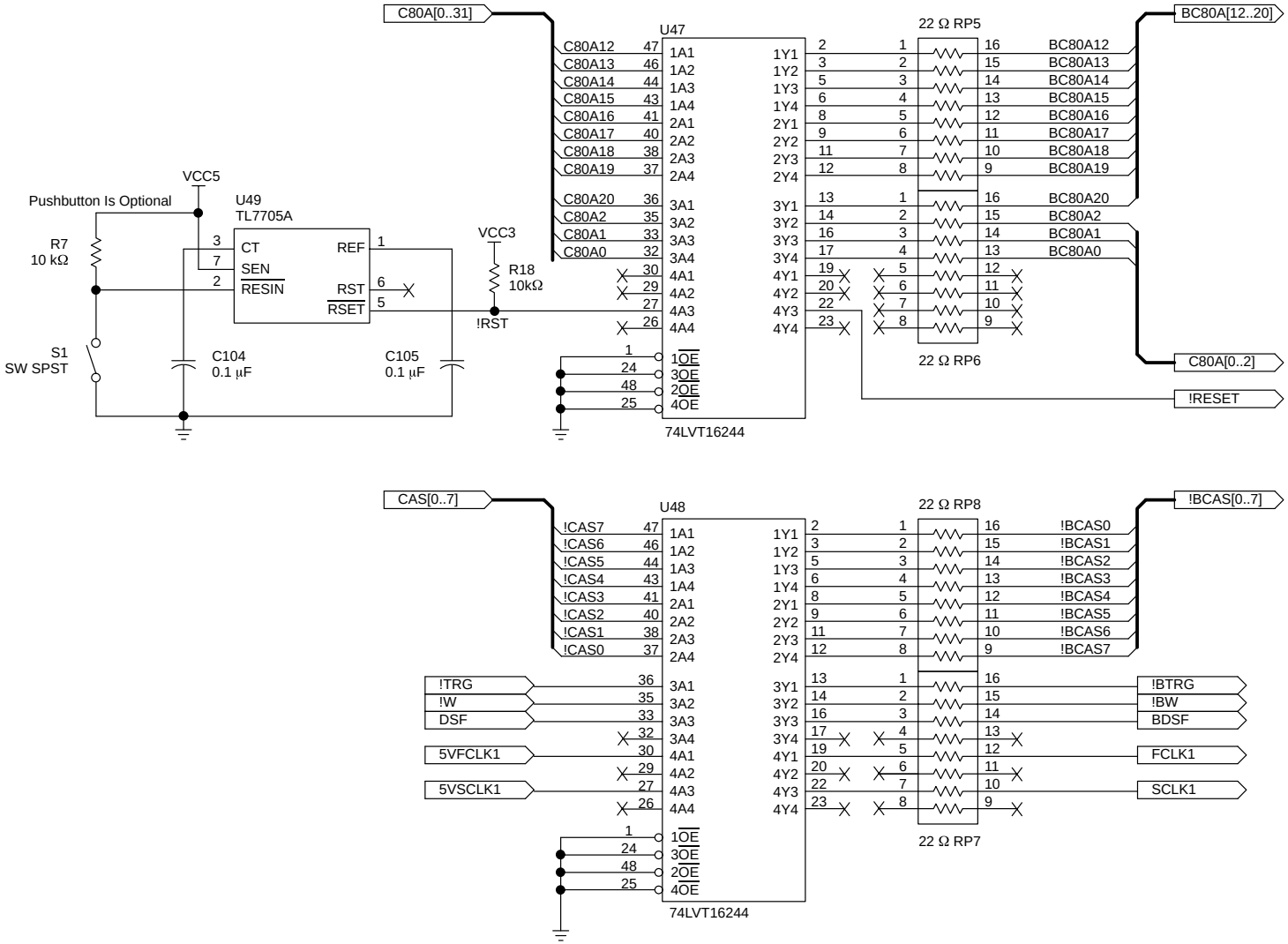


Figure B–4. Address Buffers

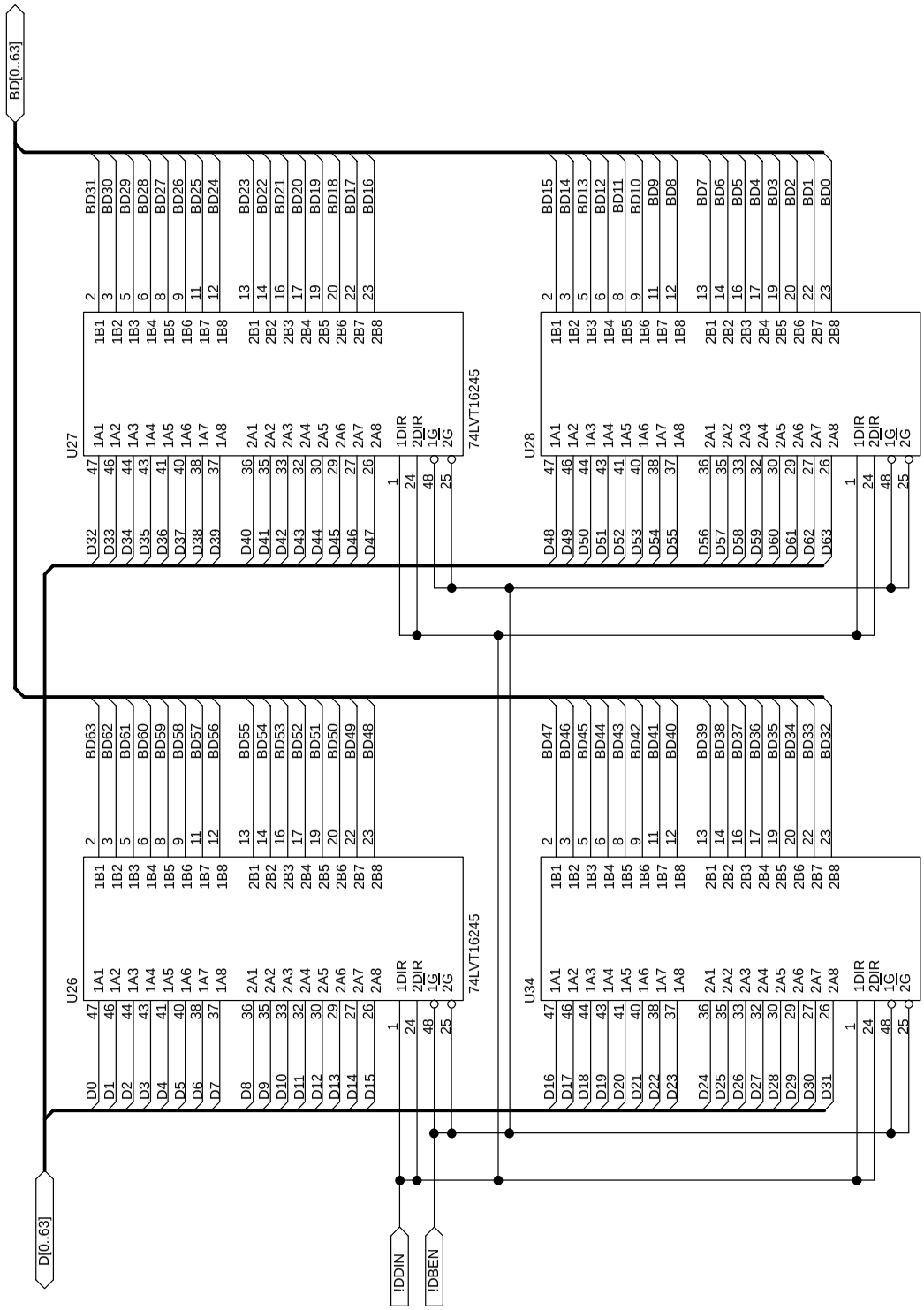


Figure B–5. Data Transceivers (Big-Endian Configuration)

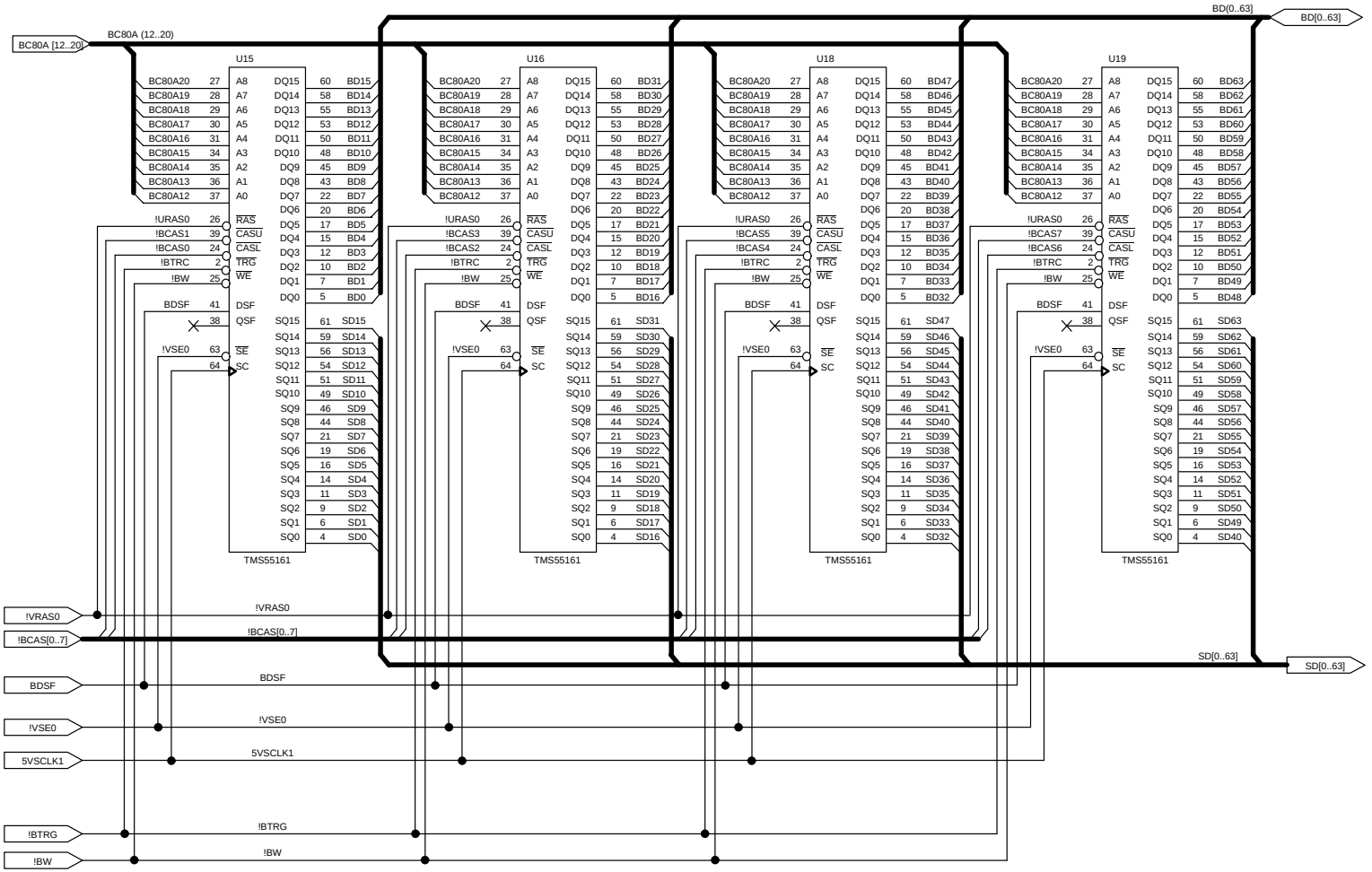


Figure B–6. VRAM Bank 0

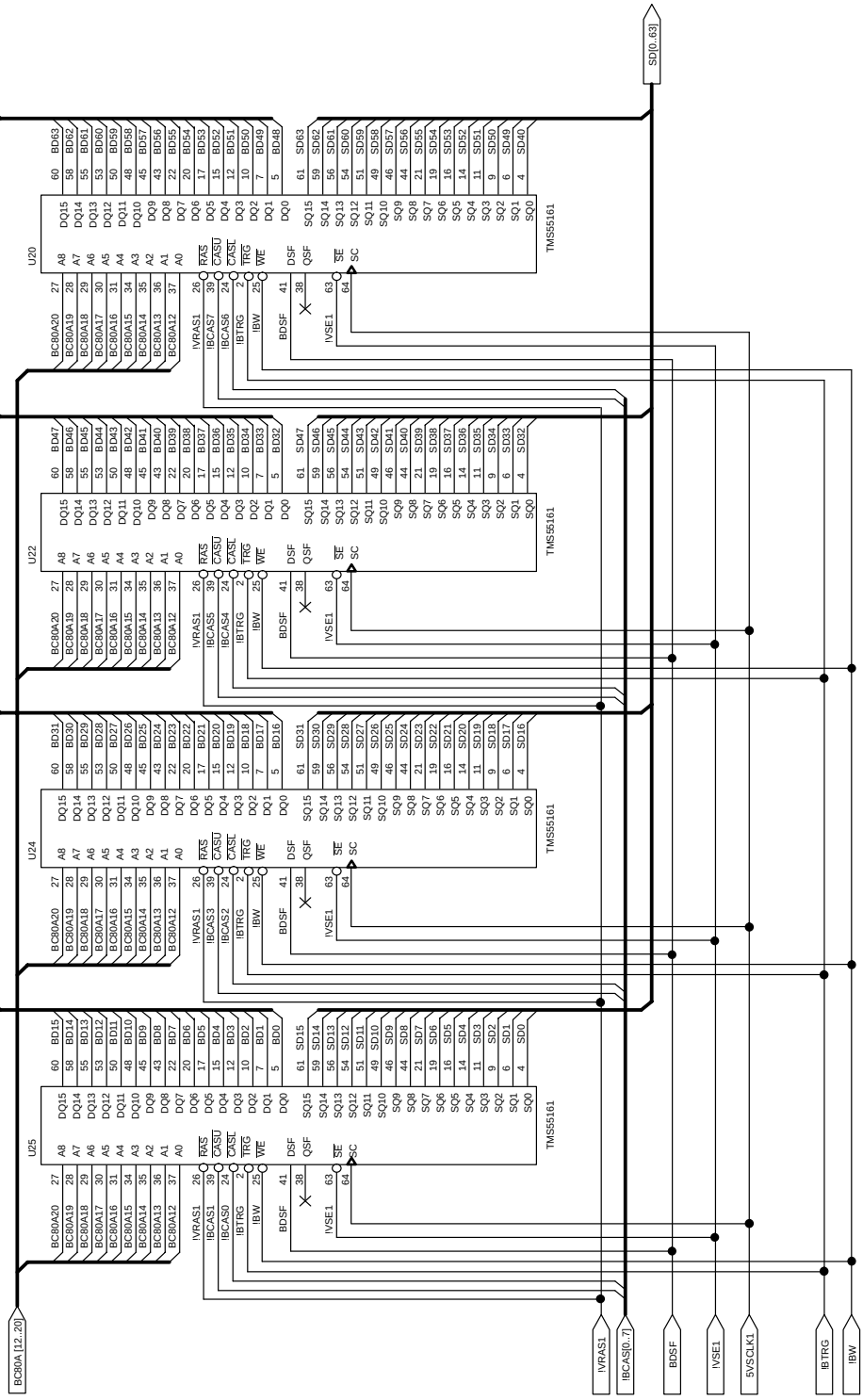


Figure B–7. VRAM Bank1

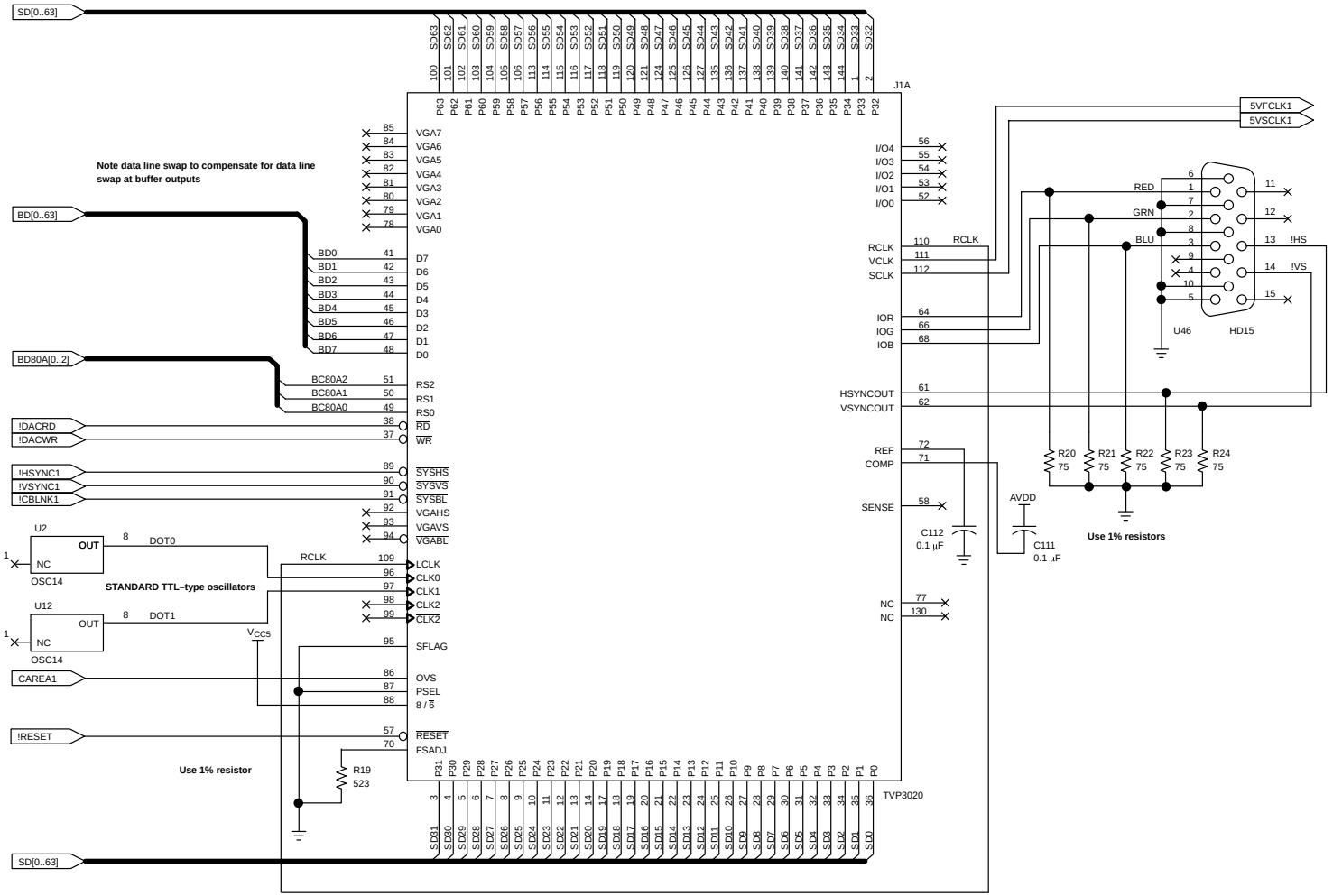
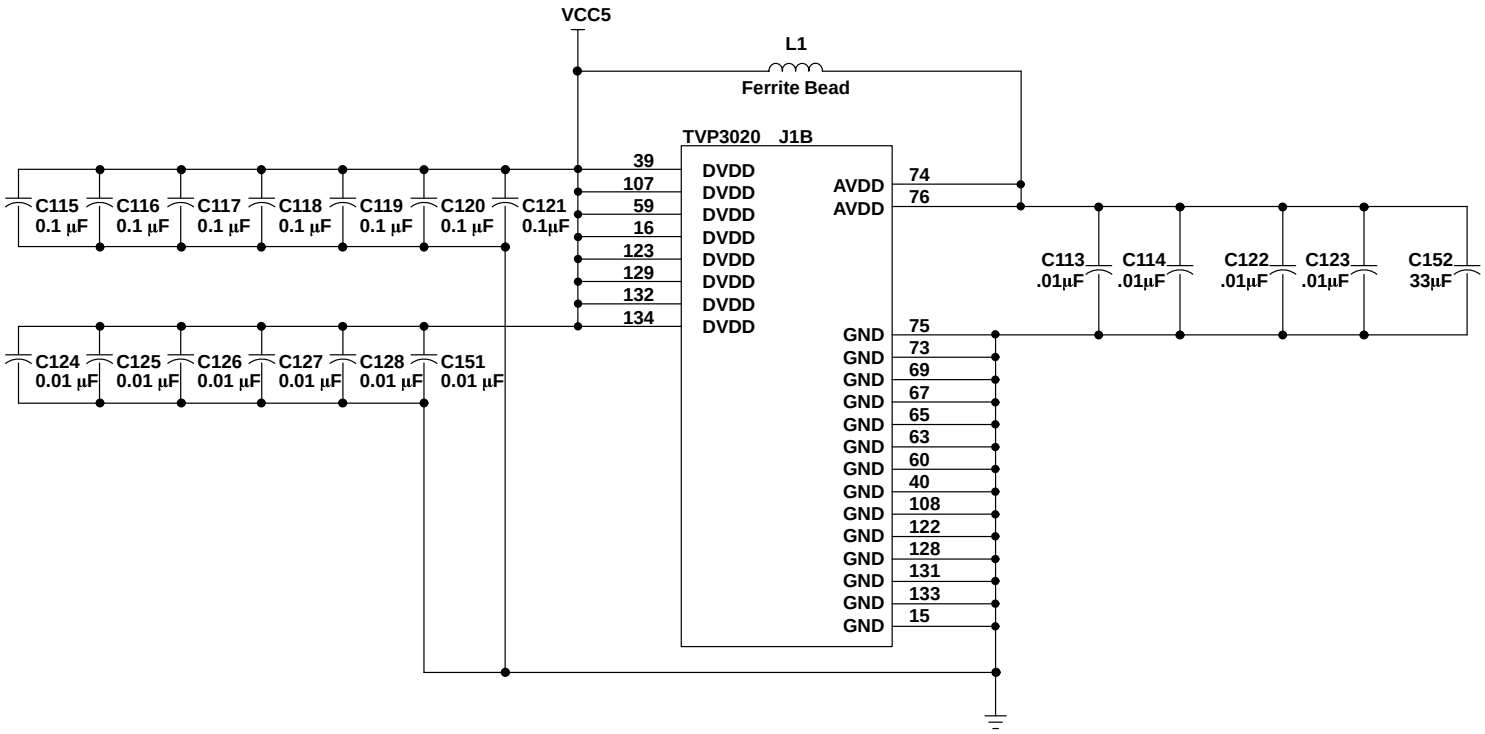


Figure B–8. TVP3020 Pallette



Note - For actual layout, please refer to AP Note text.

Figure B–9. TVP3020 Power and Ground Connections

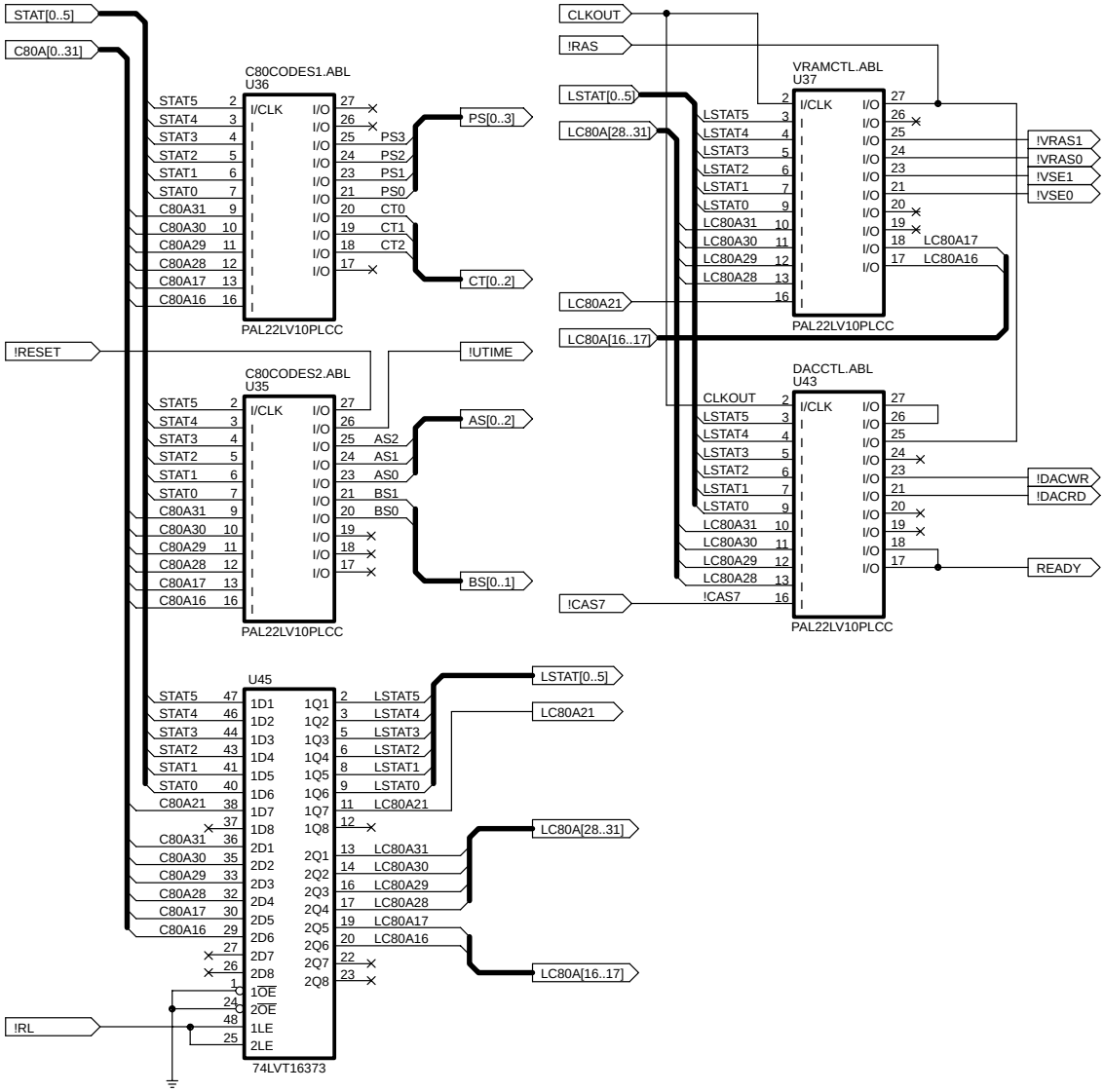


Figure B–10. Logic

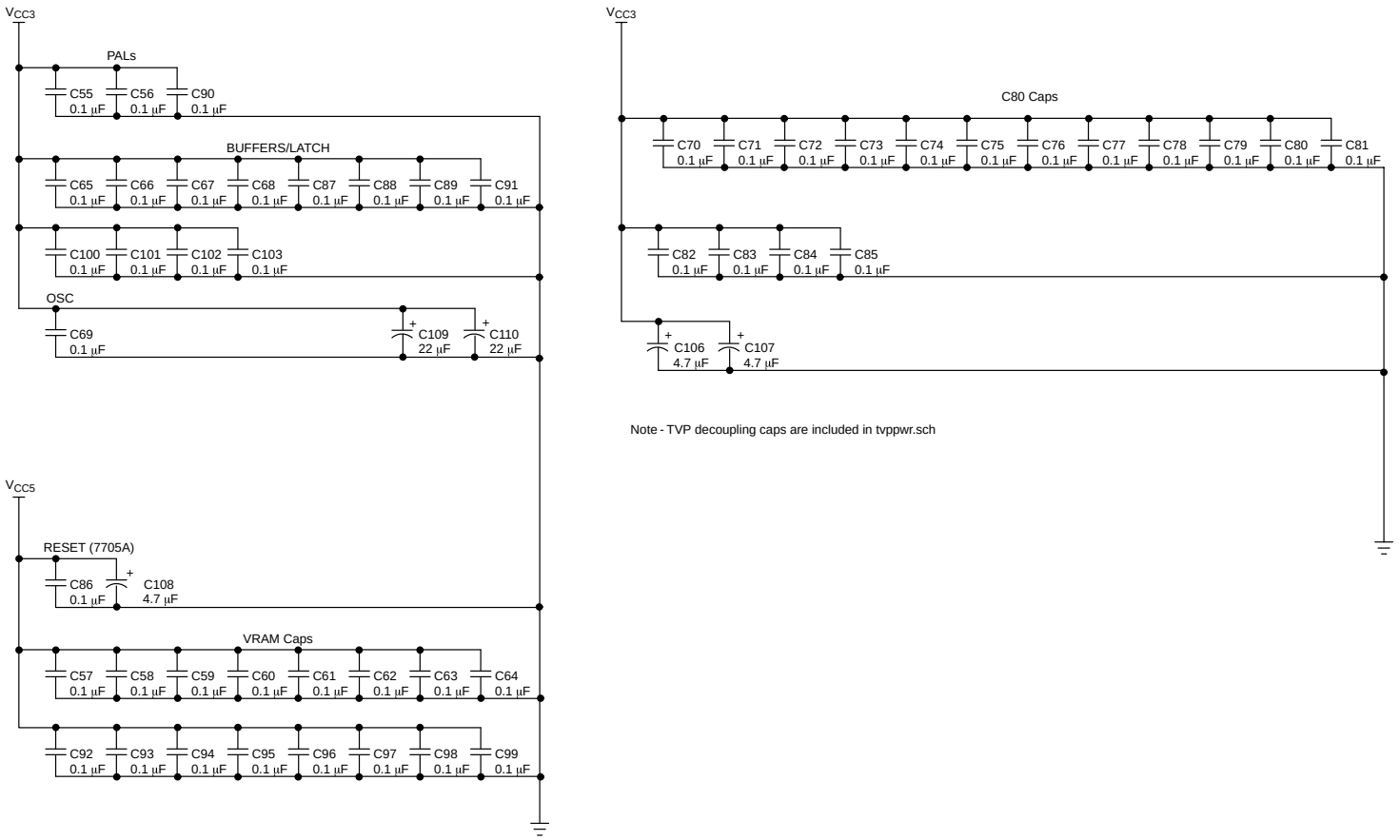


Figure B–11. TMS320C80-Decoupling Caps

Appendix C ABEL™ Files

```
module C80codes1
title'
DWG NAME  LOGIC.SCH
PAL #     U36
COMPANY   TEXAS INSTRUMENTS INCORPORATED
ENGINEER  C80 APPLICATIONS
DATE      02_28_96'
   xx_001 device 'P22V10C';   " PAL22LV10–7 PLCC
   STAT5        Pin 2;       "C80 STATUS[5]
   STAT4        Pin 3;       "C80 STATUS[4]
   STAT3        Pin 4;       "C80 STATUS[3]
   STAT2        Pin 5;       "C80 STATUS[2]
   STAT1        Pin 6;       "C80 STATUS[1]
   STAT0        Pin 7;       "C80 STATUS[0]
   A31          Pin 9;       "C80 address line 31
   A30          Pin 10;      "C80 address line 30
   A29          Pin 11;      "C80 address line 29
   A28          Pin 12;      "C80 address line 28
   A17          Pin 13;      "C80 address line 17
   A16          Pin 16;      "C80 address line 16
   vss          Pin 14;      "Ground
   vcc          Pin 28;      "Power
   "NC          Pin 27;
   "NC          Pin 26;
   PS3          Pin 25;      "C80 PS[3] input (page size)
   PS2          Pin 24;      "C80 PS[2] input (page size)
   PS1          Pin 23;      "C80 PS[1] input (page size)
   PS0          Pin 21;      "C80 PS[0] input (page size)
   CT0          Pin 20;      "C80 CT[0] input (cycle timing)
   CT1          Pin 19;      "C80 CT[1] input (cycle timing)
   CT2          Pin 18;      "C80 CT[2] input (cycle timing)
   "NC          Pin 17;
```

ABEL is a trademark of DATA I/O.

```
"constants and alias names
   ADDR          = [A31..A28] ;
   REFADD        = [A17..A16] ;
   STAT          = [STAT5..STAT0] ;
   refr1         = 0 ; "VRAM bank refresh pseudo–address
   refr2         = 1 ; "VRAM bank refresh pseudo–address
   MRS           = 12 ;
   DCAB          = 3 ;
   REFRESH       = 2 ;
   READ          = 0 ;
   WRITE         = 1 ;
   SDRAM_cyc     =(!STAT5 & !STAT4 & !STAT3 & !STAT2 & STAT1 & STAT0)
                 #(!STAT5 & !STAT4 & STAT3 & STAT2 & !STAT1 & !STAT0);
   VRAM_refresh =(!STAT5 & !STAT4 & !STAT3 & !STAT2 & STAT1 &
                 !STAT0) &(( !A17 & A16) # (!A17 & !A16));
   REFH          = (!STAT5 & !STAT4 & !STAT3 & !STAT2 & STAT1 & !STAT0);
   VRAM_AV       = A31 & !A30 & A29 & !A28 ;


   TVP_AV        = A31 & !A30 & !A29 & !A28 ;


equations
   PS3    = (VRAM_AV) & !(SDRAM_cyc # REFH)
          #(TVP_AV) & !(SDRAM_cyc # REFH);
          "#(SYSTEM_SPECIFIC_REQUIRING_PS3=1)
   PS2    = 0 ;
          "#(SYSTEM_SPECIFIC_REQUIRING_PS2=1)
   PS1    = (VRAM_AV) & !(SDRAM_cyc # REFH);
          "#(SYSTEM_SPECIFIC_REQUIRING_PS1=1);
   PS0    = 0 ;
          "#(SYSTEM_SPECIFIC_REQUIRING_PS0=1)


   CT2    = (VRAM_AV) & !(SDRAM_cyc # REFH)
          #VRAM_refresh
          #(TVP_AV) & !(SDRAM_cyc # REFH);
          "#(SYSTEM_SPECIFIC_REQUIRING_CT2=1)
   CT1    = (VRAM_AV) & !(SDRAM_cyc # REFH)
          #VRAM_refresh
          #(TVP_AV) & !(SDRAM_cyc # REFH);
          "#(SYSTEM_SPECIFIC_REQUIRING_CT1=1)
```

```
   CT0    = (VRAM_AV) & !(SDRAM_cyc # REFH)
          #(VRAM_refresh)
          #(TVP_AV) & !(SDRAM_cyc # REFH);
          "#(SYSTEM_SPECIFIC_REQUIRING_CT0=1)
test_vectors"PS3
([ ADDR  , STAT  ] -> [PS3])
 [  0    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  1    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  2    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  3    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  4    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  5    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  6    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  7    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  8    , .X.   ] -> [ 1 ] ;" TVP3020 addressed
 [  9    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  10   ,  0    ] -> [ 1 ] ;" VRAM addressed, reads
 [  10   ,  1    ] -> [ 1 ] ;" VRAM addressed, writes
 [  10   ,REFRESH] -> [ 0 ] ;" VRAM addressed, refresh cycle
 [  10   ,  MRS  ] -> [ 0 ] ;" VRAM addressed, MRS cycle
 [  10   ,  DCAB ] -> [ 0 ] ;" VRAM addressed, DCAB cycle
 [  11   , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  12   , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  13   , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  14   , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  15   , .X.   ] -> [ 0 ] ;" VRAM not addressed

test_vectors"PS2
([ ADDR  , STAT  ] -> [PS2])
 [  0    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  1    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  2    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  3    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  4    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  5    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  6    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  7    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  8    , .X.   ] -> [ 0 ] ;" TVP3020 addressed
```

```
 [  9    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  10   ,  0    ] -> [ 0 ] ;" VRAM addressed, reads
 [  10   ,  1    ] -> [ 0 ] ;" VRAM addressed, writes
 [  10   ,REFRESH] -> [ 0 ] ;" VRAM addressed, refresh cycle
 [  10   ,  MRS  ] -> [ 0 ] ;" VRAM addressed, MRS cycle
 [  10   ,  DCAB ] -> [ 0 ] ;" VRAM addressed, DCAB cycle
 [  11   , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  12   , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  13   , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  14   , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  15   , .X.   ] -> [ 0 ] ;" VRAM not addressed

test_vectors"PS1
([ ADDR  , STAT  ] -> [PS1])
 [  0    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  1    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  2    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  3    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  4    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  5    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  6    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  7    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  8    , .X.   ] -> [ 0 ] ;" TVP3020 addressed
 [  9    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  10   ,  0    ] -> [ 1 ] ;" VRAM addressed, reads
 [  10   ,  1    ] -> [ 1 ] ;" VRAM addressed, writes
 [  10   ,REFRESH] -> [ 0 ] ;" VRAM addressed, refresh cycle
 [  10   ,  MRS  ] -> [ 0 ] ;" VRAM addressed, MRS cycle
 [  10   ,  DCAB ] -> [ 0 ] ;" VRAM addressed, DCAB cycle
 [  11   , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  12   , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  13   , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  14   , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  15   , .X.   ] -> [ 0 ] ;" VRAM not addressed

test_vectors"PS0
([ ADDR  , STAT  ] -> [PS0])
 [  0    , .X.   ] -> [ 0 ] ;" VRAM not addressed
```

```
 [  1    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  2    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  3    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  4    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  5    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  6    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  7    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  8    , .X.   ] -> [ 0 ] ;" TVP3020 addressed
 [  9    , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  10   ,  0    ] -> [ 0 ] ;" VRAM addressed, reads
 [  10   ,  1    ] -> [ 0 ] ;" VRAM addressed, writes
 [  10   ,REFRESH] -> [ 0 ] ;" VRAM addressed, refresh cycle
 [  10   ,  MRS  ] -> [ 0 ] ;" VRAM addressed, MRS cycle
 [  10   ,  DCAB ] -> [ 0 ] ;" VRAM addressed, DCAB cycle
 [  11   , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  12   , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  13   , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  14   , .X.   ] -> [ 0 ] ;" VRAM not addressed
 [  15   , .X.   ] -> [ 0 ] ;" VRAM not addressed

test_vectors"CT2
([ ADDR , STAT, REFADD ] -> [ CT2 ])
 [  0   , .X. ,  .X.   ] -> [  0  ] ;" VRAM not addressed
 [  1   , .X. ,  .X.   ] -> [  0  ] ;" VRAM not addressed
 [  2   , .X. ,  .X.   ] -> [  0  ] ;" VRAM not addressed
 [  3   , .X. ,  .X.   ] -> [  0  ] ;" VRAM not addressed
 [  4   , .X. ,  .X.   ] -> [  0  ] ;" VRAM not addressed
 [  5   , .X. ,  .X.   ] -> [  0  ] ;" VRAM not addressed
 [  6   , .X. ,  .X.   ] -> [  0  ] ;" VRAM not addressed
 [  7   , .X. ,  .X.   ] -> [  0  ] ;" VRAM not addressed
 [  8   , READ,  .X.   ] -> [  1  ] ;" TVP3020 addressed
 [  8   ,WRITE,  .X.   ] -> [  1  ] ;" TVP3020 addressed
 [  9   , .X. ,  .X.   ] -> [  0  ] ;" VRAM not addressed
 [  10  , READ,  .X.   ] -> [  1  ] ;" VRAM addressed
 [  10  ,WRITE,  .X.   ] -> [  1  ] ;" VRAM addressed
 [  11  , .X. ,  .X.   ] -> [  0  ] ;" VRAM not addressed
 [  12  , .X. ,  .X.   ] -> [  0  ] ;" VRAM not addressed
 [  13  , .X. ,  .X.   ] -> [  0  ] ;" VRAM not addressed
```

```
 [  14  , .X. ,  .X.   ] -> [  0  ] ;" VRAM not addressed
 [  15  , .X. ,  .X.   ] -> [  0  ] ;" VRAM not addressed
 [  .X. ,REFRESH,refr1 ] -> [  1  ] ;" VRAM refresh
 [  .X. ,REFRESH,refr2 ] -> [  1  ] ;" VRAM refresh
 [  .X. ,REFRESH,  2   ] -> [  0  ] ;" refresh- not VRAM
 [  .X. ,REFRESH,  3   ] -> [  0  ] ;" refresh- not VRAM
 [  .X. , MRS ,  .X.   ] -> [  0  ] ;" MRS
 [  .X. , DCAB,  .X.   ] -> [  0  ] ;" DCAB

test_vectors"CT1
([ ADDR , STAT, REFADD ] -> [ CT1 ])
 [  0   , .X. ,  .X.   ] -> [  0  ] ;" VRAM not addressed
 [  1   , .X. ,  .X.   ] -> [  0  ] ;" VRAM not addressed
 [  2   , .X. ,  .X.   ] -> [  0  ] ;" VRAM not addressed
 [  3   , .X. ,  .X.   ] -> [  0  ] ;" VRAM not addressed
 [  4   , .X. ,  .X.   ] -> [  0  ] ;" VRAM not addressed
 [  5   , .X. ,  .X.   ] -> [  0  ] ;" VRAM not addressed
 [  6   , .X. ,  .X.   ] -> [  0  ] ;" VRAM not addressed
 [  7   , .X. ,  .X.   ] -> [  0  ] ;" VRAM not addressed
 [  8   , READ,  .X.   ] -> [  1  ] ;" TVP3020 addressed
 [  8   ,WRITE,  .X.   ] -> [  1  ] ;" TVP3020 addressed
 [  9   , .X. ,  .X.   ] -> [  0  ] ;" VRAM not addressed
 [  10  , READ,  .X.   ] -> [  1  ] ;" VRAM addressed
 [  10  ,WRITE,  .X.   ] -> [  1  ] ;" VRAM addressed
 [  11  , .X. ,  .X.   ] -> [  0  ] ;" VRAM not addressed
 [  12  , .X. ,  .X.   ] -> [  0  ] ;" VRAM not addressed
 [  13  , .X. ,  .X.   ] -> [  0  ] ;" VRAM not addressed
 [  14  , .X. ,  .X.   ] -> [  0  ] ;" VRAM not addressed
 [  15  , .X. ,  .X.   ] -> [  0  ] ;" VRAM not addressed
 [  .X. ,REFRESH,refr1 ] -> [  1  ] ;" VRAM refresh
 [  .X. ,REFRESH,refr2 ] -> [  1  ] ;" VRAM refresh
 [  .X. ,REFRESH,  2   ] -> [  0  ] ;" refresh- not VRAM
 [  .X. ,REFRESH,  3   ] -> [  0  ] ;" refresh- not VRAM
 [  .X. , MRS ,  .X.   ] -> [  0  ] ;" MRS
 [  .X. , DCAB,  .X.   ] -> [  0  ] ;" DCAB
```

```
test_vectors”CT0
([ ADDR , STAT, REFADD ] –> [ CT0 ])
 [  0   , .X. ,  .X.   ] –> [  0  ] ;” VRAM not addressed
 [  1   , .X. ,  .X.   ] –> [  0  ] ;” VRAM not addressed
 [  2   , .X. ,  .X.   ] –> [  0  ] ;” VRAM not addressed
 [  3   , .X. ,  .X.   ] –> [  0  ] ;” VRAM not addressed
 [  4   , .X. ,  .X.   ] –> [  0  ] ;” VRAM not addressed
 [  5   , .X. ,  .X.   ] –> [  0  ] ;” VRAM not addressed
 [  6   , .X. ,  .X.   ] –> [  0  ] ;” VRAM not addressed
 [  7   , .X. ,  .X.   ] –> [  0  ] ;” VRAM not addressed
 [  8   , READ,  .X.   ] –> [  1  ] ;” TVP3020 addressed
 [  8   ,WRITE,  .X.   ] –> [  1  ] ;” TVP3020 addressed
 [  9   , .X. ,  .X.   ] –> [  0  ] ;” VRAM not addressed
 [  10  , READ,  .X.   ] –> [  1  ] ;” VRAM addressed
 [  10  ,WRITE,  .X.   ] –> [  1  ] ;” VRAM addressed
 [  11  , .X. ,  .X.   ] –> [  0  ] ;” VRAM not addressed
 [  12  , .X. ,  .X.   ] –> [  0  ] ;” VRAM not addressed
 [  13  , .X. ,  .X.   ] –> [  0  ] ;” VRAM not addressed
 [  14  , .X. ,  .X.   ] –> [  0  ] ;” VRAM not addressed
 [  15  , .X. ,  .X.   ] –> [  0  ] ;” VRAM not addressed
 [  .X. ,REFRESH,refr1 ] –> [  1  ] ;” VRAM refresh
 [  .X. ,REFRESH,refr2 ] –> [  1  ] ;” VRAM refresh
 [  .X. ,REFRESH,  2   ] –> [  0  ] ;” refresh– not VRAM
 [  .X. ,REFRESH,  3   ] –> [  0  ] ;” refresh– not VRAM
 [  .X. , MRS ,  .X.   ] –> [  0  ] ;” MRS
 [  .X. , DCAB,  .X.   ] –> [  0  ] ;” DCAB
end C80codes1
```

```
module C80codes2
title'
DWG NAME  LOGIC.SCH
PAL #     U35
COMPANY   TEXAS INSTRUMENTS INCORPORATED
ENGINEER  C80 APPLICATIONS
DATE      02_28_96'

   xx_001    device 'P22V10C';   " PAL22LV10–7C PLCC
   STAT5     Pin 2;              "C80 STATUS[5]
   STAT4     Pin 3;              "C80 STATUS[4]
   STAT3     Pin 4;              "C80 STATUS[3]
   STAT2     Pin 5;              "C80 STATUS[2]
   STAT1     Pin 6;              "C80 STATUS[1]
   STAT0     Pin 7;              "C80 STATUS[0]
   A31       Pin 9;              "C80 address line 31
   A30       Pin 10;             "C80 address line 30
   A29       Pin 11;             "C80 address line 29
   A28       Pin 12;             "C80 address line 28
   A17       Pin 13;             "C80 address line 17
   A16       Pin 16;             "C80 address line 16
   vss       Pin 14;             "Ground
   vcc       Pin 28;             "Power
   _RESET    Pin 27;             "C80 _RESET – used as an input here
   _UTIME    Pin 26;             "C80 /UTIME input (user timed cycles)
   AS2       Pin 25;             "C80 AS[2] input (address shift)
   AS1       Pin 24;             "C80 AS[1] input (address shift)
   AS0       Pin 23;             "C80 AS[0] input (address shift)
   BS1       Pin 21;             "C80 BS[1] input (bus size)
   BS0       Pin 20;             "C80 BS[0] input (bus size)
   "NC       Pin 19;
   "NC       Pin 18;
   "NC       Pin 17;
"constants and alias names
   ADDR      = [A31..A28] ;
   REFADD    = [A17..A16] ;
   STAT      = [STAT5..STAT0];
   refr1     = 0 ;               "VRAM bank refresh pseudo–address
```

```
   refr2     = 1 ; ”VRAM bank refresh pseudo–address
   MRS       = 12 ;
   READ      = 0 ;
   WRITE     = 1 ;
   blk_wr    = 9 ;
   VRAM_AV   = A31 & !A30 & A29 & !A28 ;
   TVP_AV    = A31 & !A30 & A29 & !A28 ;
   SDRAM_cyc =(!STAT5 & !STAT4 & !STAT3 & !STAT2 & STAT1 & STAT0)
             #(!STAT5 & !STAT4 & STAT3 & STAT2 & !STAT1 & !STAT0);

equations
   AS2       = 0 ;
             ”#(SYSTEM_SPECIFIC_REQUIRING_AS2=1)
   AS1       = (VRAM_AV) & !(SDRAM_cyc) ;
             ”#(SYSTEM_SPECIFIC_REQUIRING_AS1=1)
   AS0       = 0 ;
             ”#(SYSTEM_SPECIFIC_REQUIRING_AS0=1)
   BS1       = (VRAM_AV) & !(SDRAM_cyc) ;
             ”#(SYSTEM_SPECIFIC_REQUIRING_BS1=1)
   BS0       = (VRAM_AV) & !(SDRAM_cyc # blk_wrt);
             ”#(SYSTEM_SPECIFIC_REQUIRING_BS0=1)
   !_UTIME   = (TVP_AV) & !(SDRAM_cyc)
             #(!_RESET);
             ”#(SYSTEM_SPECIFIC_REQUIRING_UTIME=0)
   blk_wrt   =(!STAT5 & !STAT4 & STAT3 & !STAT2 & !STAT1 & STAT0);

test_vectors”AS2
([ ADDR  , STAT  ] –> [AS2])
 [  0    , .X.   ] –> [ 0 ] ;  ” VRAM not addressed
 [  1    , .X.   ] –> [ 0 ] ;  ” VRAM not addressed
 [  2    , .X.   ] –> [ 0 ] ;  ” VRAM not addressed
 [  3    , .X.   ] –> [ 0 ] ;  ” VRAM not addressed
 [  4    , .X.   ] –> [ 0 ] ;  ” VRAM not addressed
 [  5    , .X.   ] –> [ 0 ] ;  ” VRAM not addressed
 [  6    , .X.   ] –> [ 0 ] ;  ” VRAM not addressed
 [  7    , .X.   ] –> [ 0 ] ;  ” VRAM not addressed
 [  8    , .X.   ] –> [ 0 ] ;  ” TVP3020 addressed
 [  8    , MRS   ] –> [ 0 ] ;  ” TVP3020 addressed– MRS
```

```
 [  9    , .X.   ] -> [ 0 ] ;  " VRAM not addressed
 [  10   , .X.   ] -> [ 0 ] ;  " VRAM addressed
 [  10   , MRS   ] -> [ 0 ] ;  " VRAM addressed - MRS
 [  11   , .X.   ] -> [ 0 ] ;  " VRAM not addressed
 [  12   , .X.   ] -> [ 0 ] ;  " VRAM not addressed
 [  13   , .X.   ] -> [ 0 ] ;  " VRAM not addressed
 [  14   , .X.   ] -> [ 0 ] ;  " VRAM not addressed
 [  15   , .X.   ] -> [ 0 ] ;  " VRAM not addressed

test_vectors"AS1
([ ADDR  , STAT  ] -> [AS1])
 [  0    , .X.   ] -> [ 0 ] ;  " VRAM not addressed
 [  1    , .X.   ] -> [ 0 ] ;  " VRAM not addressed
 [  2    , .X.   ] -> [ 0 ] ;  " VRAM not addressed
 [  3    , .X.   ] -> [ 0 ] ;  " VRAM not addressed
 [  4    , .X.   ] -> [ 0 ] ;  " VRAM not addressed
 [  5    , .X.   ] -> [ 0 ] ;  " VRAM not addressed
 [  6    , .X.   ] -> [ 0 ] ;  " VRAM not addressed
 [  7    , .X.   ] -> [ 0 ] ;  " VRAM not addressed
 [  8    , .X.   ] -> [ 0 ] ;  " TVP3020 addressed
 [  8    , MRS   ] -> [ 0 ] ;  " TVP3020 addressed- MRS
 [  9    , .X.   ] -> [ 0 ] ;  " VRAM not addressed
 [  10   , .X.   ] -> [ 1 ] ;  " VRAM addressed
 [  10   , MRS   ] -> [ 0 ] ;  " VRAM addressed - MRS
 [  11   , .X.   ] -> [ 0 ] ;  " VRAM not addressed
 [  12   , .X.   ] -> [ 0 ] ;  " VRAM not addressed
 [  13   , .X.   ] -> [ 0 ] ;  " VRAM not addressed
 [  14   , .X.   ] -> [ 0 ] ;  " VRAM not addressed
 [  15   , .X.   ] -> [ 0 ] ;  " VRAM not addressed

test_vectors"AS0
([ ADDR  , STAT  ] -> [AS0])
 [  0    , .X.   ] -> [ 0 ] ;  " VRAM not addressed
 [  1    , .X.   ] -> [ 0 ] ;  " VRAM not addressed
 [  2    , .X.   ] -> [ 0 ] ;  " VRAM not addressed
 [  3    , .X.   ] -> [ 0 ] ;  " VRAM not addressed
 [  4    , .X.   ] -> [ 0 ] ;  " VRAM not addressed
 [  5    , .X.   ] -> [ 0 ] ;  " VRAM not addressed
```

```
 [  6    , .X.   ] –> [ 0 ] ;  ” VRAM not addressed
 [  7    , .X.   ] –> [ 0 ] ;  ” VRAM not addressed
 [  8    , .X.   ] –> [ 0 ] ;  ” TVP3020 addressed
 [  8    , MRS   ] –> [ 0 ] ;  ” TVP3020 addressed– MRS
 [  9    , .X.   ] –> [ 0 ] ;  ” VRAM not addressed
 [  10   , .X.   ] –> [ 0 ] ;  ” VRAM addressed
 [  10   , MRS   ] –> [ 0 ] ;  ” VRAM addressed – MRS
 [  11   , .X.   ] –> [ 0 ] ;  ” VRAM not addressed
 [  12   , .X.   ] –> [ 0 ] ;  ” VRAM not addressed
 [  13   , .X.   ] –> [ 0 ] ;  ” VRAM not addressed
 [  14   , .X.   ] –> [ 0 ] ;  ” VRAM not addressed
 [  15   , .X.   ] –> [ 0 ] ;  ” VRAM not addressed

test_vectors”BS1
([ ADDR  , STAT ] –> [BS1])
 [  0    , .X.  ] –> [ 0 ] ;   ” VRAM not addressed
 [  1    , .X.  ] –> [ 0 ] ;   ” VRAM not addressed
 [  2    , .X.  ] –> [ 0 ] ;   ” VRAM not addressed
 [  3    , .X.  ] –> [ 0 ] ;   ” VRAM not addressed
 [  4    , .X.  ] –> [ 0 ] ;   ” VRAM not addressed
 [  5    , .X.  ] –> [ 0 ] ;   ” VRAM not addressed
 [  6    , .X.  ] –> [ 0 ] ;   ” VRAM not addressed
 [  7    , .X.  ] –> [ 0 ] ;   ” VRAM not addressed
 [  8    , .X.  ] –> [ 0 ] ;   ” TVP3020 addressed
 [  9    , .X.  ] –> [ 0 ] ;   ” VRAM not addressed
 [  10   ,WRITE ] –> [ 1 ] ;   ” VRAM addressed (writes)
 [  10   ,READ  ] –> [ 1 ] ;   ” VRAM addressed (reads)
 [  10   ,blk_wr] –> [ 1 ] ;   ” VRAM addressed (block write)
 [  11   , .X.  ] –> [ 0 ] ;   ” VRAM not addressed
 [  12   , .X.  ] –> [ 0 ] ;   ” VRAM not addressed
 [  13   , .X.  ] –> [ 0 ] ;   ” VRAM not addressed
 [  14   , .X.  ] –> [ 0 ] ;   ” VRAM not addressed
 [  15   , .X.  ] –> [ 0 ] ;   ” VRAM not addressed
 [  .X.  , MRS  ] –> [ 0 ] ;   ” MRS (change as req’d)

test_vectors”BS0
([ ADDR  , STAT ] –> [BS0])
 [  0    , .X.  ] –> [ 0 ] ;   ” VRAM not addressed
```

```
 [  1    , .X.  ] -> [ 0 ] ;  " VRAM not addressed
 [  2    , .X.  ] -> [ 0 ] ;  " VRAM not addressed
 [  3    , .X.  ] -> [ 0 ] ;  " VRAM not addressed
 [  4    , .X.  ] -> [ 0 ] ;  " VRAM not addressed
 [  5    , .X.  ] -> [ 0 ] ;  " VRAM not addressed
 [  6    , .X.  ] -> [ 0 ] ;  " VRAM not addressed
 [  7    , .X.  ] -> [ 0 ] ;  " VRAM not addressed
 [  8    , .X.  ] -> [ 0 ] ;  " TVP3020 addressed
 [  9    , .X.  ] -> [ 0 ] ;  " VRAM not addressed
 [  10   ,WRITE ] -> [ 1 ] ;  " VRAM addressed (writes)
 [  10   ,READ  ] -> [ 1 ] ;  " VRAM addressed (reads)
 [  10   ,blk_wr] -> [ 0 ] ;  " VRAM addressed (block write)
 [  11   , .X.  ] -> [ 0 ] ;  " VRAM not addressed
 [  12   , .X.  ] -> [ 0 ] ;  " VRAM not addressed
 [  13   , .X.  ] -> [ 0 ] ;  " VRAM not addressed
 [  14   , .X.  ] -> [ 0 ] ;  " VRAM not addressed
 [  15   , .X.  ] -> [ 0 ] ;  " VRAM not addressed
 [  .X.  , MRS  ] -> [ 0 ] ;  " MRS (change as req'd)
end C80codes2
```

```
module vramctl
title'
DWG NAME  LOGIC.SCH
PAL #     U37
COMPANY   TEXAS INSTRUMENTS INCORPORATED
ENGINEER  C80 APPLICATIONS
DATE      02_28_96'
   xx_001    device 'P22V10C'; " PAL22LV10-7 PLCC
   CLKOUT    Pin 2;  "C80 CLKOUT
   LSTAT5    Pin 3;  "C80 STATUS[5] --EXTERNALLY LATCHED BY _RL
   LSTAT4    Pin 4;  "C80 STATUS[4] --EXTERNALLY LATCHED BY _RL
   LSTAT3    Pin 5;  "C80 STATUS[3] --EXTERNALLY LATCHED BY _RL
   LSTAT2    Pin 6;  "C80 STATUS[2] --EXTERNALLY LATCHED BY _RL
   LSTAT1    Pin 7;  "C80 STATUS[1] --EXTERNALLY LATCHED BY _RL
   LSTAT0    Pin 9;  "C80 STATUS[0] --EXTERNALLY LATCHED BY _RL
   LA31      Pin 10;  "C80 address line 31 --EXTERNALLY LATCHED BY _RL
   LA30      Pin 11; "C80 address line 30 --EXTERNALLY LATCHED BY _RL
   LA29      Pin 12; "C80 address line 29 --EXTERNALLY LATCHED BY _RL
   LA28      Pin 13; "C80 address line 28 --EXTERNALLY LATCHED BY _RL
   LA21      Pin 16; "C80 address line 21 --EXTERNALLY LATCHED BY _RL
   vss       Pin 14; "Ground
   vcc       Pin 28; "Power
   _RAS      Pin 27; "C80 _RAS
   VSTATE    Pin 26;
   _VRAS1    Pin 25; "VRAM bank 1 _RAS
   _VRAS0    Pin 24; "VRAM bank 0 _RAS
   _VSE1     Pin 23; "VRAM Bank1 serial output enable
   _VSE0     Pin 21; "VRAM Bank0 serial output enable
   "NC       Pin 20;
   "NC       Pin 19;
   LA17      Pin 18; "C80 address line 17 --EXTERNALLY LATCHED BY _RL
   LA16      Pin 17; "C80 address line 16 --EXTERNALLY LATCHED BY _RL
"constants and alias names
   ADDR      = [LA31..LA28] ;
   REFADD    = [LA17..LA16] ;
   STAT      = [LSTAT5..LSTAT0] ;
   MRS    = 12 ;
```

```
    DCAB      = 3 ;
    READ      = 0 ;
    WRITE     = 1 ;
    RFR       = 2 ;
    SDRAM_cyc =(!LSTAT5 & !LSTAT4 & !LSTAT3 & !LSTAT2 & LSTAT1 & LSTAT0)
              #(!LSTAT5 & !LSTAT4 & LSTAT3 & LSTAT2 & !LSTAT1 & !LSTAT0);
    REFH      = (!LSTAT5 & !LSTAT4 & !LSTAT3 & !LSTAT2 & LSTAT1 & !LSTAT0);

    SRT       = (!LSTAT5 & LSTAT4 & !LSTAT3 & LSTAT2  & !LSTAT0);

    VRAM_AV   = LA31 & !LA30 & LA29 & !LA28 ;
equations
    !_VRAS1   = (!_RAS & VRAM_AV & LA21) & !(SDRAM_cyc # REFH)
               #(REFH & !_RAS & !LA17 & LA16);
    !_VRAS0   = (!_RAS & VRAM_AV & !LA21) & !(SDRAM_cyc # REFH)
               #(REFH & !_RAS & !LA17 & !LA16);
    !_VSE1    = (SRT & LA21 & !_RAS)
               #(!_VSE1 & !SRT)
               #(!_VSE1 & _RAS);
    !_VSE0    = !_VSE1 ;
```

```
test_vectors”_VRAS1, _VRAS0
([ ADDR  , STAT, _RAS, REFADD, LA21  ] -> [_VRAS1, _VRAS0])
 [  .X.  , RFR ,   1 ,  .X.  , .X.   ] -> [   1  ,    1  ];”Refresh- RAS high
 [  .X.  , RFR ,   0 ,   0   , .X.   ] -> [   1  ,    0  ];”Refresh- bank 0
 [  .X.  , RFR ,   0 ,   1   , .X.   ] -> [   0  ,    1  ];”Refresh- bank 1
 [  .X.  , RFR ,   0 ,   2   , .X.   ] -> [   1  ,    1  ];”Refresh- not VRAM
 [  .X.  , RFR ,   0 ,   3   , .X.   ] -> [   1  ,    1  ];”Refresh- not VRAM
 [  .X.  , RFR ,   0 ,   3   , .X.   ] -> [   1  ,    1  ];”Refresh- not VRAM
 [   0   ,READ ,   0 ,  .X.  , .X.   ] -> [   1  ,    1  ];”access- not VRAM
 [   8   ,READ ,   0 ,  .X.  , .X.   ] -> [   1  ,    1  ];”access- not VRAM
 [   9   ,READ ,   0 ,  .X.  , .X.   ] -> [   1  ,    1  ];”access- not VRAM
 [   10  ,READ ,   0 ,  .X.  ,  0    ] -> [   1  ,    0  ];”access- bank 0
 [   10  ,WRITE,   0 ,  .X.  ,  0    ] -> [   1  ,    0  ];”access- bank 0
 [   10  ,READ ,   0 ,  .X.  ,  1    ] -> [   0  ,    1  ];”access- bank 1
 [   10  ,WRITE,   0 ,  .X.  ,  1    ] -> [   0  ,    1  ];”access- bank 1
 [   10  , MRS ,   0 ,  .X.  , .X.   ] -> [   1  ,    1  ];”MRS cycle
 [   10  , DCAB,   0 ,  .X.  , .X.   ] -> [   1  ,    1  ];”DCAB cycle
end vramctl
```

```
module dacctl
title'
DWG NAME     LOGIC.SCH
PAL #        U43
COMPANY      TEXAS INSTRUMENTS INCORPORATED
ENGINEER     C80 APPLICATIONS
DATE         02_28_96'

   xx_001    device 'P22V10C'; " PAL22LV10-7 PLCC
   CLKOUT    Pin 2;  "C80 CLKOUT
   LSTAT5    Pin 3;  "C80 STATUS[5] --EXTERNALLY LATCHED BY _RL
   LSTAT4    Pin 4;  "C80 STATUS[4] --EXTERNALLY LATCHED BY _RL
   LSTAT3    Pin 5;  "C80 STATUS[3] --EXTERNALLY LATCHED BY _RL
   LSTAT2    Pin 6;  "C80 STATUS[2] --EXTERNALLY LATCHED BY _RL
   LSTAT1    Pin 7;  "C80 STATUS[1] --EXTERNALLY LATCHED BY _RL
   LSTAT0    Pin 9;  "C80 STATUS[0] --EXTERNALLY LATCHED BY _RL
   LA31      Pin 10; "C80 address line 31 --EXTERNALLY LATCHED BY _RL
   LA30      Pin 11; "C80 address line 30 --EXTERNALLY LATCHED BY _RL
   LA29      Pin 12; "C80 address line 29 --EXTERNALLY LATCHED BY _RL
   LA28      Pin 13; "C80 address line 28 --EXTERNALLY LATCHED BY _RL
   _CAS7     Pin 16; "C80 _CAS line
   vss       Pin 14; "Ground
   vcc       Pin 28; "Power
   _DRAS     Pin 27; "C80 _RAS delayed 1/2 clk cycle
   _DDRAS    Pin 26; "_DRAS with ext delay- compensates for nonoverlap on /W
   _RAS      Pin 25; "C80 _RAS
   "NC       Pin 24;
   _DACWR    Pin 23; "TVP /WR pulse
   _DACRD    Pin 21; "TVP /RD pulse
   "NC       Pin 20;
   _DRAS3    Pin 19; "C80 _RAS delay 2 1/2 cycles
   _DRAS2    Pin 18; "C80 _RAS delay 1 1/2 cycles
   _DDRAS2   Pin 17; "_DDRAS2 with ext- compensates for nonoverlap on /W
"constants and alias names
   ADDR      = [LA31..LA28] ;
   STAT      = [LSTAT5..LSTAT0] ;

   MRS       = 12 ;
```

```
    DCAB      = 3 ;
    READ      = 0 ;
    WRT       = 1 ;
    RFR       = 2 ;
    SDRAM_cyc =(!LSTAT5 & !LSTAT4 & !LSTAT3 & !LSTAT2 & LSTAT1 & LSTAT0)
              #(!LSTAT5 & !LSTAT4 & LSTAT3 & LSTAT2 & !LSTAT1 & !LSTAT0);
    REFH      = (!LSTAT5 & !LSTAT4 & !LSTAT3 & !LSTAT2 & LSTAT1 & !LSTAT0);
    TVP_AV    = LA31 & !LA30 & !LA29 & !LA28 ;
equations
    !_DACRD   = (TVP_AV)
              & (!LSTAT5 & !LSTAT4 & !LSTAT3 & !LSTAT2 & !LSTAT1 & !LSTAT0)
              & !_CAS7 ;
    !_DACWR   = (TVP_AV)
              & (!LSTAT5 & !LSTAT4 & !LSTAT3 & !LSTAT2 & !LSTAT1 & LSTAT0)
              & (!_RAS # !_DDRAS # !_DDRAS2 # !_DRAS3);
    !_DRAS    := (!_RAS) & (TVP_AV)
              & (!LSTAT5 & !LSTAT4 & !LSTAT3 & !LSTAT2 & !LSTAT1);
    _DRAS2    := _DRAS ;
    _DRAS3    := _DDRAS2 ;
```

```
test_vectors"_DACWR,
_DACRD([CLKOUT,ADDR,STAT,_RAS,_DDRAS,_DRAS2,_DDRAS2,_DRAS3,_CAS7] ->
[_DACRD,_DACWR])"READY
 [  0   ,  0 ,   0,  0 ,  1   ,   1  ,   1   ,   1  ,  1  ] -> [  1   ,   1  ];"  1
 [  1   ,  0 ,   0,  0 ,  1   ,   1  ,   1   ,   1  ,  1  ] -> [  1   ,   1  ];"  1
 [  0   ,  0 ,   0,  0 ,  1   ,   1  ,   1   ,   1  ,  1  ] -> [  1   ,   1  ];"  1
 [  1   ,  0 ,   0,  0 ,  1   ,   1  ,   1   ,   1  ,  1  ] -> [  1   ,   1  ];"  1
 [  0   ,  8 , WRT,  1 ,  1   ,   1  ,   1   ,   1  ,  1  ] -> [  1   ,   1  ];"  1
 [  1   ,  8 , WRT,  1 ,  1   ,   1  ,   1   ,   1  ,  1  ] -> [  1   ,   1  ];"  1
 [  0   ,  8 , WRT,  0 ,  1   ,   1  ,   1   ,   1  ,  0  ] -> [  1   ,   0  ];"  1
 [  1   ,  8 , WRT,  0 ,  0   ,   1  ,   1   ,   1  ,  0  ] -> [  1   ,   0  ];"  1
 [  0   ,  8 , WRT,  1 ,  0   ,   1  ,   1   ,   1  ,  0  ] -> [  1   ,   0  ];"  1
 [  1   ,  8 , WRT,  1 ,  1   ,   0  ,   0   ,   1  ,  0  ] -> [  1   ,   0  ];"  0
 [  0   ,  8 , WRT,  1 ,  1   ,   0  ,   0   ,   1  ,  0  ] -> [  1   ,   0  ];"  0
 [  1   ,  8 , WRT,  1 ,  1   ,   1  ,   1   ,   0  ,  0  ] -> [  1   ,   0  ];"  1
 [  0   ,  8 , WRT,  1 ,  1   ,   1  ,   1   ,   0  ,  0  ] -> [  1   ,   0  ];"  1
 [  1   ,  8 , WRT,  1 ,  1   ,   1  ,   1   ,   1  ,  0  ] -> [  1   ,   1  ];"  1
 [  0   ,  8 , WRT,  1 ,  1   ,   1  ,   1   ,   1  ,  1  ] -> [  1   ,   1  ];"  1
 [  1   ,  8 , WRT,  1 ,  1   ,   1  ,   1   ,   1  ,  1  ] -> [  1   ,   1  ];"  1
 [  0   ,  8 , .X.,  1 ,  1   ,   1  ,   1   ,   1  ,  1  ] -> [  1   ,   1  ];"  1
 [  1   ,  8 , .X.,  1 ,  1   ,   1  ,   1   ,   1  ,  1  ] -> [  1   ,   1  ];"  1
 [  0   ,  8 , .X.,  1 ,  1   ,   1  ,   1   ,   1  ,  1  ] -> [  1   ,   1  ];"  1
 [  1   ,  8 , .X.,  1 ,  1   ,   1  ,   1   ,   1  ,  1  ] -> [  1   ,   1  ];"  1
 [  0   ,  8 ,READ,  1 ,  1   ,   1  ,   1   ,   1  ,  1  ] -> [  1   ,   1  ];"  1
 [  1   ,  8 ,READ,  1 ,  1   ,   1  ,   1   ,   1  ,  1  ] -> [  1   ,   1  ];"  1
 [  0   ,  8 ,READ,  1 ,  1   ,   1  ,   1   ,   1  ,  1  ] -> [  1   ,   1  ];"  1
 [  1   ,  8 ,READ,  1 ,  1   ,   1  ,   1   ,   1  ,  1  ] -> [  1   ,   1  ];"  1
 [  0   ,  8 ,READ,  0 ,  1   ,   1  ,   1   ,   1  ,  0  ] -> [  0   ,   1  ];"  1
 [  1   ,  8 ,READ,  0 ,  0   ,   1  ,   1   ,   1  ,  0  ] -> [  0   ,   1  ];"  1
 [  0   ,  8 ,READ,  1 ,  0   ,   1  ,   1   ,   1  ,  0  ] -> [  0   ,   1  ];"  1
 [  1   ,  8 ,READ,  1 ,  1   ,   0  ,   0   ,   1  ,  0  ] -> [  0   ,   1  ];"  0
 [  0   ,  8 ,READ,  1 ,  1   ,   0  ,   0   ,   1  ,  0  ] -> [  0   ,   1  ];"  0
 [  1   ,  8 ,READ,  1 ,  1   ,   1  ,   1   ,   0  ,  0  ] -> [  0   ,   1  ];"  1
 [  0   ,  8 ,READ,  1 ,  1   ,   1  ,   1   ,   0  ,  0  ] -> [  0   ,   1  ];"  1
 [  1   ,  8 ,READ,  1 ,  1   ,   1  ,   1   ,   1  ,  0  ] -> [  0   ,   1  ];"  1
 [  0   ,  8 , .X. , 1 ,  1   ,   1  ,   1   ,   1  ,  1  ] -> [  1   ,   1  ];"  1
 [  1   ,  8 , .X. , 1 ,  1   ,   1  ,   1   ,   1  ,  1  ] -> [  1   ,   1  ];"  1
 [  0   ,  8 , .X. , 1 ,  1   ,   1  ,   1   ,   1  ,  1  ] -> [  1   ,   1  ];"  1
 [  1   ,  9 , .X. , 1 ,  1   ,   1  ,   1   ,   1  ,  1  ] -> [  1   ,   1  ];"  1
```

```
 [  0   ,  3 , .X. , 1 ,  1   ,   1  ,   1   ,   1  ,  1  ] -> [  1   ,   1  ];"  1
 [  1   ,  6 , .X. , 1 ,  1   ,   1  ,   1   ,   1  ,  1  ] -> [  1   ,   1  ];"  1
end dacctl
```