Part Number 800-1187-01
Revision 50 of 28 September 1984

Engineering Manual

for the

Sun-2/120 Video Board

Sun Microsystems, Inc.,
2550 Garcia Avenue
Mountain View
California 94043
(415) 960-3300

Revision History

Revision	Date	Comments
50	28 September 1984	First release of this Theory of Operation Manual.

Contents

Contents

Figures

Tables

Preface

Purpose and Audience

The purpose of this manual is to enable Sun customers and licensors of the Sun Workstation design to understand how the video board works. Sun customers should be able to use this manual to isolate hardware failures on the video board. Licensors of the Sun Workstation design should use this manual to aid them in modifying the video board design.

Organization

This manual provides a video board overview and Theory of Operation. It includes relevant timing information, a video memory map, and video board PALs and PROMs.

Chapter 1 — *Overview* — provides a summary of the video board features and capabilities.

Chapter 2 — *Functional Description* — describes the three functional blocks on the video board.

Chapter 3 — *Operation* — explains in detail the video memory controller and video refresh cycles.

Chapter 4 — *P2-Bus Interface* — describes the P2 bus interface logic.

Chapter 5 — *Video Controller* — covers the video monitor timing and attributes.

Chapter 6 — *Video Clock and Shifter* — explains the video clock and video data.

Appendix A — *Video Board Schematics*

Appendix B — *PALs and PROMs* — lists the video board PALs and PROMs.

At the end of this manual, we have supplied a reader comment form. Please use the comment form to list errors and omissions. Your responses help a great deal in our efforts to keep our documentation up to date.

Notations Used In This Manual

When possible, the schematics were drawn to standard drafting conventions. Signal flow is shown from left to right, and top to bottom. Connected sections of the design are logically grouped together, as much as the available space allows.

Conventions used for hardware signal names in this manual are:

- Both active-high and active-low signals are used. A signal name that is followed by a minus sign (−) indicates that the signal is active LOW (<0.4V). For example, the Column Address Strobe, M.CAS0−, is such a low-active signal.

- A signal that is *not* followed by a minus sign is understood to be a HIGH active signal (>2.0V). An example of such a signal would be the parity error signal, PARERR.
- For signals with multiple meanings or synonyms, the signal names are listed as separated by a slash (/). An example of this would be the read-write signal, P.R/W–. A high signal is understood to be an assertion of a read, while a low signal is an assertion of a write.
- Bus signals are indicated by a common prefix followed by a number. For example, a 16-bit data bus might be labelled D0, D1, D2, and so on until D15.
- A group of signals that is part of a signal vector is denoted by a common prefix separated from its suffix by a period. For example, all P1 signals start with the prefix "P1.", and P1 bus address signals are P1.A00, P1.A01, etc.
- Connector signals are distinguished by a suffix of "[]" with an optional string enclosed inside the square brackets identifying the connector name.

Components

Components in the schematics are identified by ***component name*** (this is also referred to as the "body name" in the wirelist). Components are named according to their generic or industry standard names. The way the components are drawn reflects their circuit function rather than the manufacturer's definition.

Each component carries a ***location label*** identifying its component type and approximate location in the schematics. Location labels consist of a letter followed by three digits. For instance, U300 is a DIP positioned on page three of the schematics.

The letter stands for the type of component, and is one of the following:

Letter	**Component Type**
C	Standard Capacitor
D	Diode
K	Electrolytic Capacitor
L	Inductor
X	Decoupling Capacitor
J	Jumper or Connector
R	Resistor
S	Single-in-Line Component
U	Dual-in-Line Component
P..L..	Programmable Logic Array

Programmable logic components, such as PALs and PROMs, are described in a high-level functional language from which they are translated automatically into the bit patterns for programming.

Programmable logic elements are identified by name. The source code for the programmable logic elements is included in Appendix B of this manual.

Chapter 1

Overview

This manual describes the Sun-2/120™ video board. The video board is a display subsystem for the Sun-2 product family. It has a serial interface with the monitor, a keyboard and a mouse, and a parallel interface with the CPU board. The video board provides a control register to allow advanced features such as copy mode and interrupts.

The main features of the Sun-2/120 video board are:

- black-and-white display
- 1152-by-900 pixel resolution video memory
- flicker-free, non-interlaced display; refresh at 67 Hz
- dual-ported video memory between video and processor
- two serial interfaces for keyboard and mouse

The video subsystem consists of the following components:

- video memory (128 kilobytes)
- video memory controller
- data multiplexor
- P2-bus interface
- video sync controller
- video shifter

Chapter 2

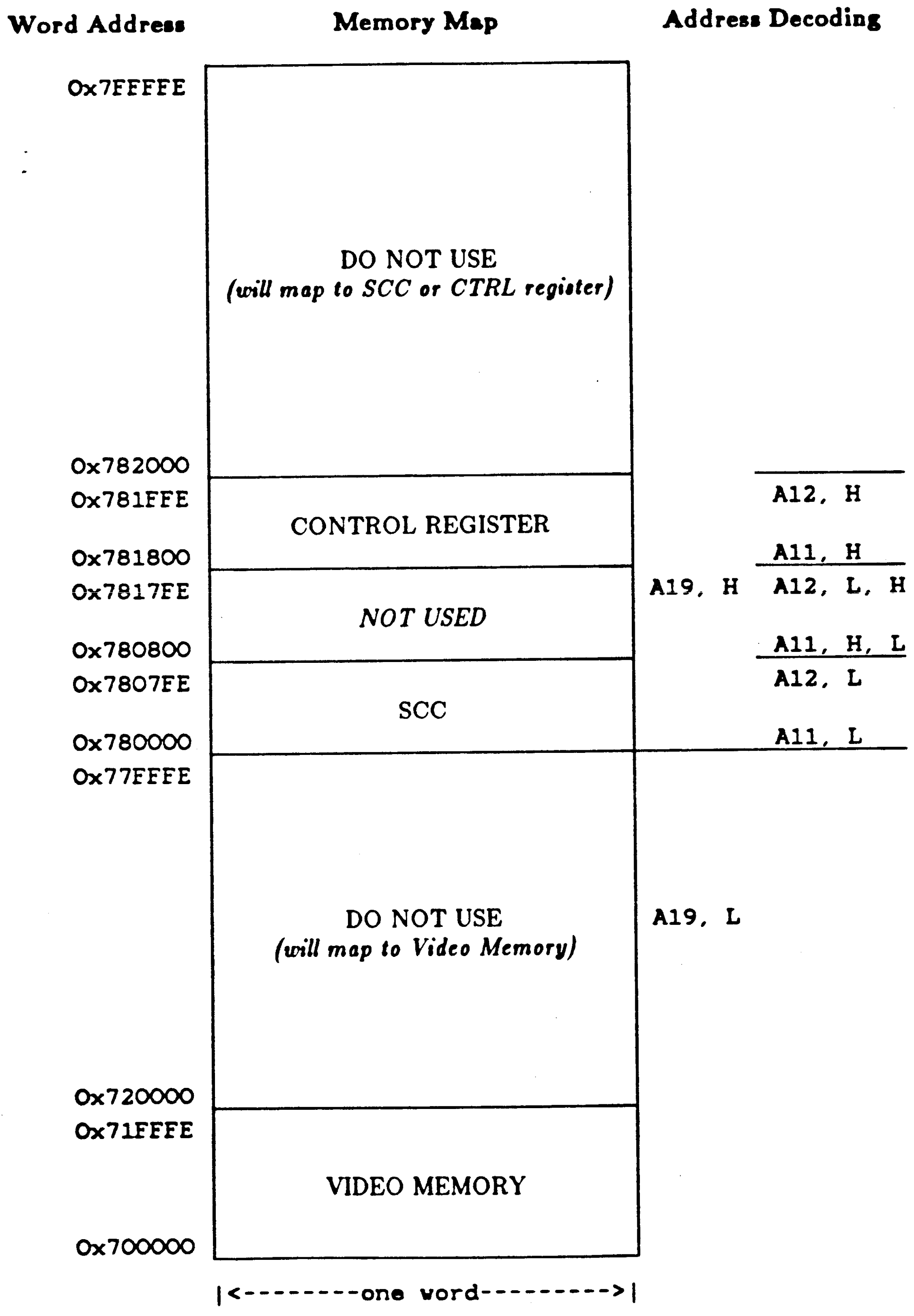
Functional Description

The three functional blocks on the video board that are accessible by the CPU are the control register, the serial controller (keyboard and mouse), and video memory. This section describes the functions of these blocks and shows their locations in memory. All addresses shown are relative to the eighth megabyte of memory the video board occupies.

2.1. Memory Map

The video board occupies the eighth megabyte of address space. The base register of the video board is fixed at 0x700000. The address space occupied by the video board is shown in the video memory map in the following table.

Figure 2-1: Video Memory Map



2.2. Control Register

The video control register is used to enable or disable various functions, set the copy mode base address, and monitor the vertical interrupt status. The processor can read the entire word and write the high or low byte individually. The starting address for the control register within the eighth megabyte for the video board is x81800 (hex) as shown in Figure 2-1. The control register is formatted as shown in the following table:

Table 2-1: Video Control Register

High Byte								Low Byte							
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
VIDEN	VCOPYEN	VINTEN	VVINT	Configuration Jumpers				0	CMBASE 17 - 22						0
read or write	read or write	read or write	read only	read only				read only	read or write						read only

VIDEN — is the video display enable bit. The processor can write to this bit and read it back to check that it was set. If the bit is a one, the video display is on; if it is a zero, the display is completely black.

VCOPYEN — is the copy-enable bit that enables copy mode. Copy mode enables read-modify-write cycles to a main memory shadow buffer to also write to the video board memory.

CMBASE — is the copy mode base address. Bits 01-06 correspond to the bus address lines 17-22. The video board monitors these address lines and if address bits 17-22 on the bus match the address in the control register (bits 01-06) the video board recognizes communication with the processor. If the copy enable bit (bit 14) is set, the video board copies the data into the video memory. If bit 14 is reset to zero, the video board ignores the matching address.

VINTEN — is the vertical interrupt enable bit. The video board generates both the horizontal and vertical synchronization pulses. Approximately every seventieth of a second, the vertical scanning finishes at the bottom of the picture and starts back up at the top again. During the vertical interval, nothing is displayed on the screen, and an interrupt can be sent back via the P1 to the processor. If an interrupt is desired, bit 13 must be set to a one. If no interrupt is desired, set this bit to a zero which disables the flip-flop. If bit 13 is set, the vertical interrupt status bit, bit 12, gets set on the next vertical interval, and the video board issues an interrupt.

After a vertical interrupt occurs, bit 12 stays set until the processor resets it. When bit 13 is toggled low, it clears the interrupt; when bit 13 is then toggled high, another interrupt is enabled for the next vertical interval. This sequence results in an interrupt about every seventieth of a second (66.66 Hz).

VVINT — is the vertical interrupt status bit which is read only. If bit 13 is enabled, the video board sends out an interrupt on the bus. The processor then has to determine where the interrupt came from. The processor reads from the control register bit 12, and if bit 12 is set, the processor knows the interrupt came from the video board.

Configuration Jumpers — these bits read back the jumper J1600, used for configuration control.

Unused bits — bits 00 and 07 can be written to by the processor, but only read back as a zero. It is therefore unnecessary to mask bits 00 and 07 for numerical computations.

2.3. Serial Communications Controller

A Zilog 8530A serial communications controller (SCC) is used to communicate with the keyboard (channel A) and the mouse (channel B). This is a byte-oriented device so data is transferred on the high byte only (data bits 08:15). On a word read, the low byte contains the lower byte of the control register so that correct parity is generated for the lower byte.

The relative starting address for the SCC is 0x780000 (hex) as shown in Figure 2-1. The SCC takes up four words in the video board megabyte, starting at hex addresses 780000, 2, 4, and 6.

Since the 8530A has an asynchronous bus interface, there is a recovery period between accesses. The recovery time for the video board 8530A is 1.4 μsec.

Addressing the device is as follows:

ADDRESS:	FUNCTION:	
780000	Channel B command data	(mouse)
780002	Channel B data	(mouse)
780004	Channel A command data	(keyboard)
780006	Channel A data	(keyboard)

Refer to the 1983 Zilog 8530A data sheet for programming details. The Serial Communications Controller is formatted as shown in the table below.

Figure 2-2: Serial Communications Controller

High Byte								Low Byte							
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SCC read or write data								write = do not care read = lower byte of control register							

2.4. Video Memory

The video memory is addressed with megabyte address 0x700000 to 0x71FFFE as shown in Figure 2-1. Accesses to the video memory can be in units of bytes or words. The processor can read words only but can write words or bytes. In other words, all read operations are in units of words, but write operations can either be to the high byte or to the low byte, or to the entire word.

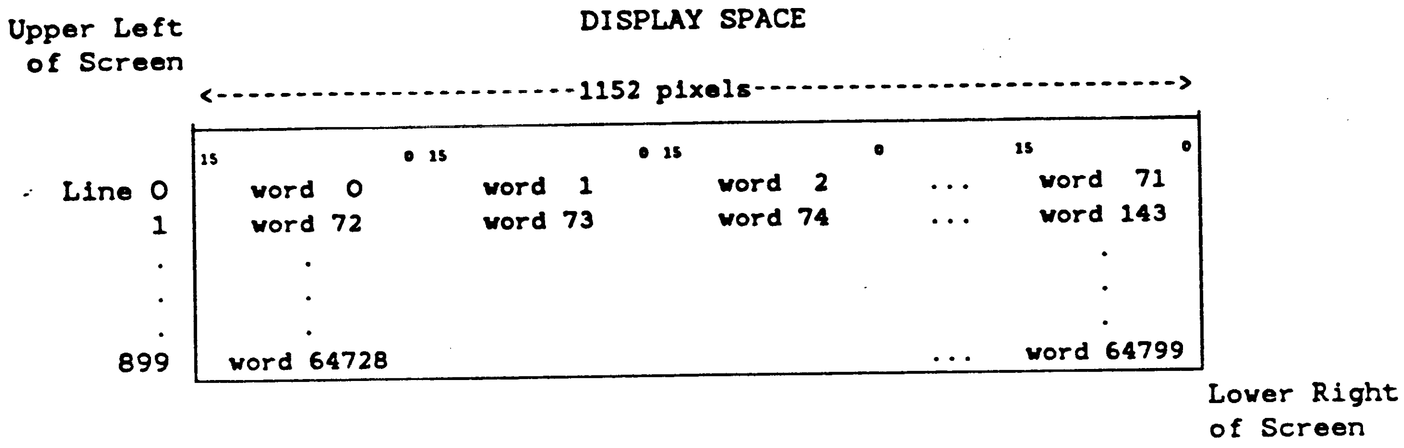
Figure 2-3: Video Memory

High Byte								Low Byte							
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Video memory read or write data															

The Sun-2/120 video board is programmable for two display formats. One is 1152 pixels by 900 lines. The mapping from linear memory space to display space is illustrated in Figure 2-4 on the next page. The other display format is 1024 pixels by 1024 lines as shown in Figure 2-5. Note that the word numbers on the memory display and memory array are base 10, and the addresses are base 16.

Switching between the two formats is achieved by two alternate sets of PROMs for the horizontal and vertical state machines.

Figure 2-4: Memory to Video Display Mapping Diagram (**1152 pixels by 900 lines**)



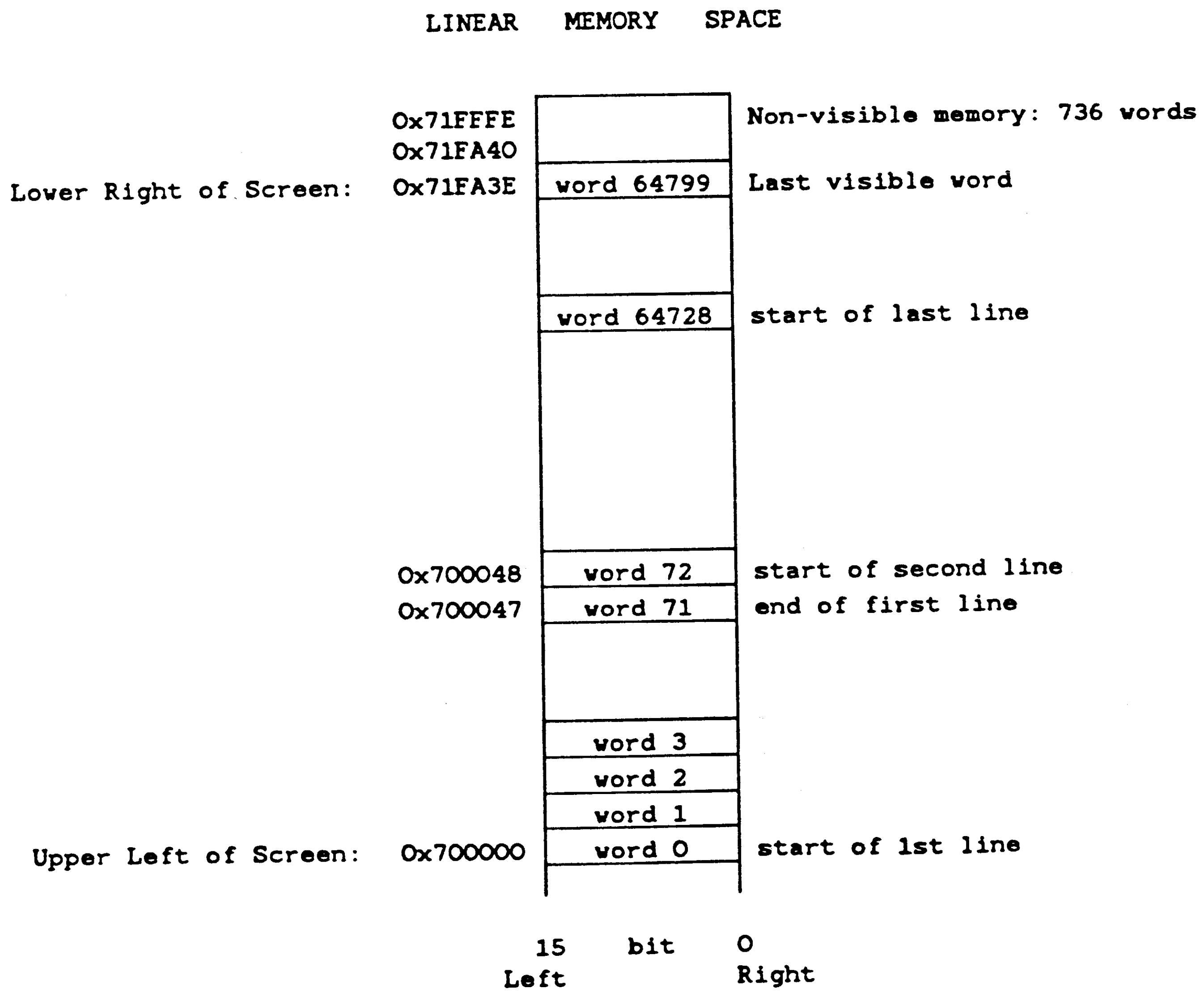
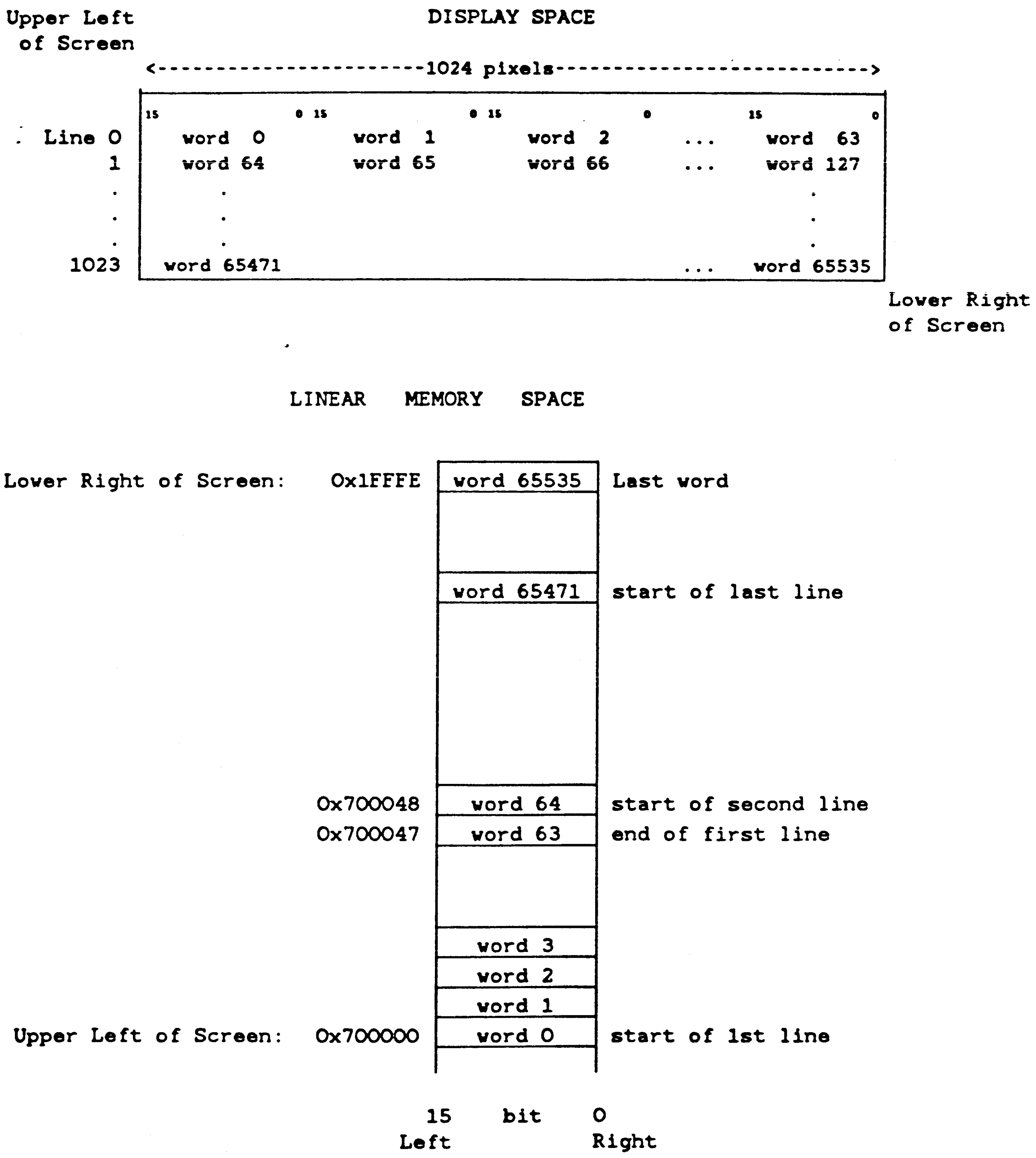


Figure 2-5: Memory to Video Display Mapping Diagram (**1024 pixels by 1024 lines**)



Chapter 3

Operation

The video memory controller state machine generates the timing for the video memory and other basic timing strobes for the video board. It consists of PROMs [P5X8:U1640,U1642] and latches [74F374:U1641,U1643]. The state machine is clocked with [V.C.40].

The memory controller has a total of 16 states, enumerated 0 through 15, that are continuously executed in sequence. Each state has a duration of 40 nsec, making the 16 state cycle repeat every 640 nsec.

The memory controller can execute four basic types of cycles:

1) Idle Cycle,

2) Video refresh cycle,

3) Processor update cycle, and

4) UART/Register cycle.

Idle Cycle — executed between state 0 through 7 if no request is pending [V.SREQ=0]. During an idle cycle, memory control signals are not asserted.

Video Refresh Cycle — executed during every memory controller cycle between state 8 and 15. During a video refresh cycle, signals [V.VRA–] and [V.VCA–] enable the video row and column address registers [74F374:U1632] and [74F374:U1633], respectively. These registers output the value contained in counters [74LS393:U1630] and [74LS590:U1631]. Video memory data is read out 64 bits in parallel and is latched at the end of state 15 in the video data register [74LS374:U1720-U1727] with the trailing edge of [V.VCA–].

In addition to executing the video refresh cycle, the current memory controller state is decoded in decoder [74F138:U1728] to enable consecutive bytes from the video data register onto the video output bus [V.O0-V.O7] via control lines [V.OE0–..V.OE7–]. Starting with [V.OE0–] in state 0, one byte from the video data register is enabled every two states. The data on the video output bus is then loaded into the video shifters [74F194:U1805,U1806].

Processor Update Cycle — executed between states 0 and 7 if synchronous request is asserted [V.SREQ=1] and if the register/video memory bit is clear [V.BS19=0]. During a processor cycle, signals [V.PRA–] and [V.PCA–] enable the processor row and column address from the processor address latches [F374:U1634] and [F374:U1635], respectively, in time for [V.RAS–] and [V.CAS–], the row and column address strobe.

Read Cycle — is executed if no external write strobes [V.WEL,V.WEU] are pending in the request latch [F374:U1615]. The memory word addressed by bank selects [V.BS1] and [V.BS2] is enabled via the RAS decoder PAL [P16R4:U1616]. The read data passes from the video RAM chips onto the internal data bus [V.B00..15] via buffers [74LS245:U1730..U1737] and is latched in the data output register [ALS374:U1602,U1603] at the rising edge of signal [V.XACK–]. At the same time, parity is computed on the internal data bus by parity checkers [74F280:U1608,U1609] and latched into the parity register [74ALS374:U1604].

Write Cycle — is executed if an external write strobe [V.WEL,V.WEU] is pending in the request latch [F374:U1615]. Write cycles are similar to read cycles except that the flow of data reverses. Write data is output enabled from the data input register [ALS374:U1600,U1601], passes through buffers [74LS245:U1730..U1737], and is then written into the RAM chips selected by active RAS strobe [V.RAS0–..3–] and write enable strobes [V.WU,V.WL]. The RAM Write Enable signal [V.WE–] is asserted starting at state 3 for early write-cycle timing.

UART/Register Cycle — almost identical to a processor update cycle. The differences are that the UART or the control register is accessed instead of the video memory and that the termination of the cycle is extended from state 7 to state 14 to allow for the longer access time of the UART. A UART/register cycle is executed if synchronous request is set [V.SREQ=1] and the register/video memory bit is set [V.BS19=1]. With [V.BS19=1], PAL [P16R4:U1616] does not assert any one of [V.RAS0–..3–] which also disables the data multiplexors [74LS245:U1730..U1737].

Figure 3-1: Timing Diagram for an Idle Cycle Followed by a Video Refresh Cycle

```
V.SREQ=0
---------------------------------------------------------------------------
State        0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15
---------------------------------------------------------------------------
V.OE         0       1       2       3       4       5       6       7

V.C.40       --__--__--__--__--__--__--__--__--__--__--__--__--__--__--__--__

V.RAS-       -------------------------------------------____________________

V.CAS-       ---------------------------------------------------____________

V.VRA-       ------------------------------------___________---------------

V.VCA-       -----------------------------------------------________________

V.PRA-       ----____________-----------------------------------------------

V.PCA-       ----------------________________-------------------------------

V.G-         ---------------------------------------------------____________

V.EN-        ---------------------------------------------------------------

V.HCLK       ------_________________________________________________________

V.ENREQ      ----___________________________________________________________
```

Figure 3-2: Timing Diagram for a Processor Update Cycle Followed by a Video Refresh Cycle

```
V.SREQ=1
------------------------------------------------------------------------------
State         0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15
------------------------------------------------------------------------------
V.OE          0       1       2       3       4       5       6       7

V.C.40        --__--__--__--__--__--__--__--__--__--__--__--__--__--__--__--__

V.RAS-        ------------____________________------------____________________

V.CAS-        --------------------____________________------------____________

V.VRA-        ------------------------------------____________----------------

V.VCA-        ------------------------------------------------________________

V.PRA-        ----____________------------------------------------------------

V.PCA-        ----------------________________--------------------------------

V.C-          --------------------________________----------------____________

V.EN-         ------------____________________________------------------------

V.ACK-        ________________________________________----____________________

V.HCLK        --------________________________________________________________

V.ENREQ       ----____________________________________________________________
```

Chapter 4

P2-Bus Interface

Major components of the P2-bus interface logic are address decoding, request generation, and interrupt logic.

4.1. P2-Bus Address Decoding

The video board responds to three types of accesses: direct reads, direct writes, and copy writes.

For direct reads, the video board is addressed if the three most significant P2 address bits [P2.A20..A22] are all ones and if P2 read/write [P2.R/W–] indicates a read cycle. In that case, decoder [F138:U1621] produces signal [V.RSEL–] which generates a video request via PAL [P16L8:U1618].

For direct writes, decoder [F138:U1622] generates a video write select [V.WSEL–] if the three most significant P2 address bits [P2.A20..A22] are all ones, P2 read/write [P2.R/W–] indicates a write cycle, and P2.RAS– and P2.CAS– are asserted.

Copy writes occur if the copy comparator [LS2521:U1623] matches P2 address bits [P2.A17..P2.A22] with video base address bits [V.BASE1..6] and if copy mode is set V.COPY=1 in the control register. If all of these conditions are true then comparator [LS2521:U109] generates [V.CSEL–] which generates a video request via PAL [P16L8:U1618].

4.2. P2-Bus Read/Write Cycles

The video board implements buffered write cycles and unbuffered reads. Reads follow the traditional conventions of memory systems. When the processor reads from the video board, the video board performs the desired access and returns the data read to the processor. Since the memory on the video board is dual-ported and asynchronous to the processor, the processor will have to wait until the read data is available. This is implemented by the video board asserting the [P2.WAIT–] signal until the read data is ready.

Write cycles, on the other hand, are buffered. The video board provides a set of registers that store all information related to a write cycle, effectively implementing a one-deep FIFO. This means that on a write cycle the processor does needs not to wait until the dual-ported video memory is available. Instead, the write cycle is automatically completed with the data stored in the registers. A second write, however, can only be initiated when the first write cycle has been completed. This is done by asserting the [P2.WAIT–] signal if a write cycle to the video board is attempted while a previous request is still in progress.

An interesting case occurs if a write cycle is immediately followed by a read cycle. In this case, the write cycle is still in progress while the new read cycle is pending. The design of the request logic assures that a read cycle is only begun after a write cycle has been completed.

This read/write cycle handshaking is implemented in PAL [P16L8:U1618]. Video request is set on the leading edge of [P2.CAS−] if direct read [V.RSEL−=1] is valid, if the direct write conditions apply [P2.A20..22=1,P2.R/W−=0], or if copy write [V.CSEL−=0] is valid.

[P2.WAIT−] is asserted on read cycles via three-state driver [74F244:U1816] as soon as [V.RSEL−] is valid. It is de-asserted on read cycles when [V.RWAIT−] is cleared. On write cycles, [P2.WAIT−] is asserted by PAL [P16L8:U1618] while a write request is in progress.

The leading edge of the request signal [V.REQ−] clocks the demultiplexed processor address into the processor address register [F374:U1634,U1635]. It also clocks the low-order address bits [P2.A01,P2.A02] and the write enable bits [P2.WEL,P2.WEU] into register [F374:U1615].

[V.REQ−] is sampled with signal [V.ENREQ] into flip-flop [74F74:U1624-0]. The sampled signal is reclocked on the next clock edge of [V.C.40] into flip-flop [74F74:U1624-1] and becomes signal [V.SREQ]. This signal controls the memory state machine to perform either a CPU cycle [V.SREQ=1] or an idle cycle [V.SREQ=0].

```
--------------------------------------------------------------------
READ CYCLE

V.RSEL-    -------_________________________----------------------------

V.RDACK-   ------------------------________----------------------------

V.REQ-     -------_________________------------------------------------

V.XACK-    ------------------------____--------------------------------

P2.WAIT-   -------_________________------------------------------------

--------------------------------------------------------------------
WRITE CYCLE

V.WSEL-    -------____-------------------------------------------------

V.RDACK-   ------------------------------------------------------------

V.REQ-     -------_________________------------------------------------

V.XACK-    ------------------------____--------------------------------

P2.WAIT-   -----------_____________------------------------------------

--------------------------------------------------------------------
WRITE CYCLE FOLLOWED BY WRITE CYCLE

V.WSEL-    -------____--------_____________----------------------------

V.RDACK-   ------------------------------------------------------------

V.REQ-     -------_________________----______________------------------

V.XACK-    ------------------------____--------------____--------------

P2.WAIT-   -----------_____________--------__________------------------

--------------------------------------------------------------------
WRITE CYCLE FOLLOWED BY READ CYCLE

V.WSEL-    -------____-------------------------------------------------

V.RSEL-    -------____--------_____________________________------------

V.RDACK-   ------------------------------------------______------------

V.REQ-     -------_________________----______________------------------

V.XACK-    ------------------------____--------------____--------------

P2.WAIT-   -----------_______________________________------------------
```

4.3. Interrupt Logic

The video interrupt flip-flop [74F74:U1803-1] is set at the leading edge of [V.VBLANK] as long as interrupt enable [V.INTEN] is enabled. Via open collector driver [U1906:7407], it drives the Multibus interrupt level selected by jumper block J1903. UART interrupt [V.SINT] is set if the SCC is programmed to generate interrupts and the corresponding condition has occurred. Open collector driver [U1906-1:7407] drives the UART interrupt to the Multibus interrupt line selected by jumper block [J.16:J1904].

Chapter 5

Video Controller

The video controller generates the timing for the video monitor. The video controller latch [74F374:U1812] latches the outputs of the horizontal and vertical decoding PROM on the rising edge of [V.HCLK]. The following description applies to the "standard Sun-2/120 video monitor". This video monitor has the following attributes:

Table 5-1: Video Monitor Attributes (**1152 pixels by 900 lines**)

Visible Display	1152 pixels by 900 lines		
Video Clock	10	nsec	100 MHz
Horizontal Cycle	16.00	μsec	62.5 kHz
Vertical Cycle	14992	μsec	66.70 Hz
Horizontal Retrace	4.48	μsec	
Vertical Retrace	592	μsec	

The alternate Sun-2/120 video display has the following attributes:

Table 5-2: Video Monitor Attributes (**1024 pixels by 1024 lines**)

Visible Display	1024 pixels by 1024 lines (no non-visible memory, see Figure 2-5)		
Video Clock	10	nsec	100 MHz
Horizontal Cycle	16.00	μsec	62.5 kHz
Vertical Cycle	16976	μsec	58.91 Hz
Horizontal Retrace	4.48	μsec	
Vertical Retrace	592	μsec	

5.1. Horizontal State Machine

Horizontal counter [74LS393:U1810] is advanced every 640 nsec with the falling edge of clock [V.HCLK]. Horizontal counter is reset with [V.HRESET] generated by video controller latch.

Horizontal decode PROM [P9X4:U1811] decodes horizontal counter inputs [V.H0] through [V.H6], plus vertical blank [VBLANK] from the vertical state machine. Horizontal decode PROM outputs are [V.HRESET, V.HSYNC, and V.DISPEN].

For the 1152-by-900 resolution, the horizontal state machine timing diagram is as follows:

Figure 5-1: Horizontal State Machine Timing Diagram

```
Signal   State

STATE+1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5

HCLK    -_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_

DISPEN  ------------------------------------________________

HSYNC   ____________________________________--------________

HRESET  --__________________________________________________
```

5.2. Vertical State Machine

Vertical counter [74LS393:U1813,U1814] is advanced on falling edge of horizontal sync [V.HSYNC]. Vertical counter is reset with [V.VRESET] from video controller latch.

Vertical decode PROM [P9X4:U1815] decodes vertical counter states [V.VSTATE1..7], the AND of [V.VSTATE8..9], and [V.VSTATE10]. Vertical decode PROM outputs are [V.VSYNC, V.RESET, and V.VBLANK.RESET–]. The vertical decode PROM function is defined in PROM A1815 in Appendix B.

For the 1152-by-900 resolution, the vertical state machine timing diagram is as follows:

Figure 5-2: Vertical State Machine Timing Diagram

```
Signal   State

STATE+1 0 0 0 0 0 0 0 0 ... 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 ... 9 9 9 9
        0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 ... 3 3 3 3
        0 1 2 3 4 5 6 7 ... 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 ... 5 6 7 8

VCLK    -_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_

VBLANK  ____________________--------------------------------------------

VSYNC   ________________________--------------------____________________

VRESET  ______________________________________________________________--
```

Chapter 6

Video Clock and Shifter

The 100-MHz video clock [V.C.10], generated by crystal oscillator [K1114A:U1800], is divided into a 50-MHz clock by flip-flop [74F112:U1801-0] and into a 25-MHz clock by flip-flop [74F112:U1801-1]..

The video data [V.O0..7] is loaded into two 50-MHz shift registers, [74F194:U1805,U8106], one shifting the odd and one the even bits, respectively. A pair of odd and even bits [V.VID0,V.VID1] together with a 10-nanosecond clock [V.C.10–] and a 20-nanosecond clock [V.C.20] are converted from TTL to ECL levels by converter [10H124:U1807] and drive the 100-MHz shift register [10H141]. Since both true and inverted data is loaded into the shifter, differential output levels [VIDEO–,VIDEO+] are available on its outputs. The differential outputs are terminated with 390-ohm resistors [R:R1800,R1801] to -5V and are intended to drive differential ECL terminated with an impedence of 100 ohms.

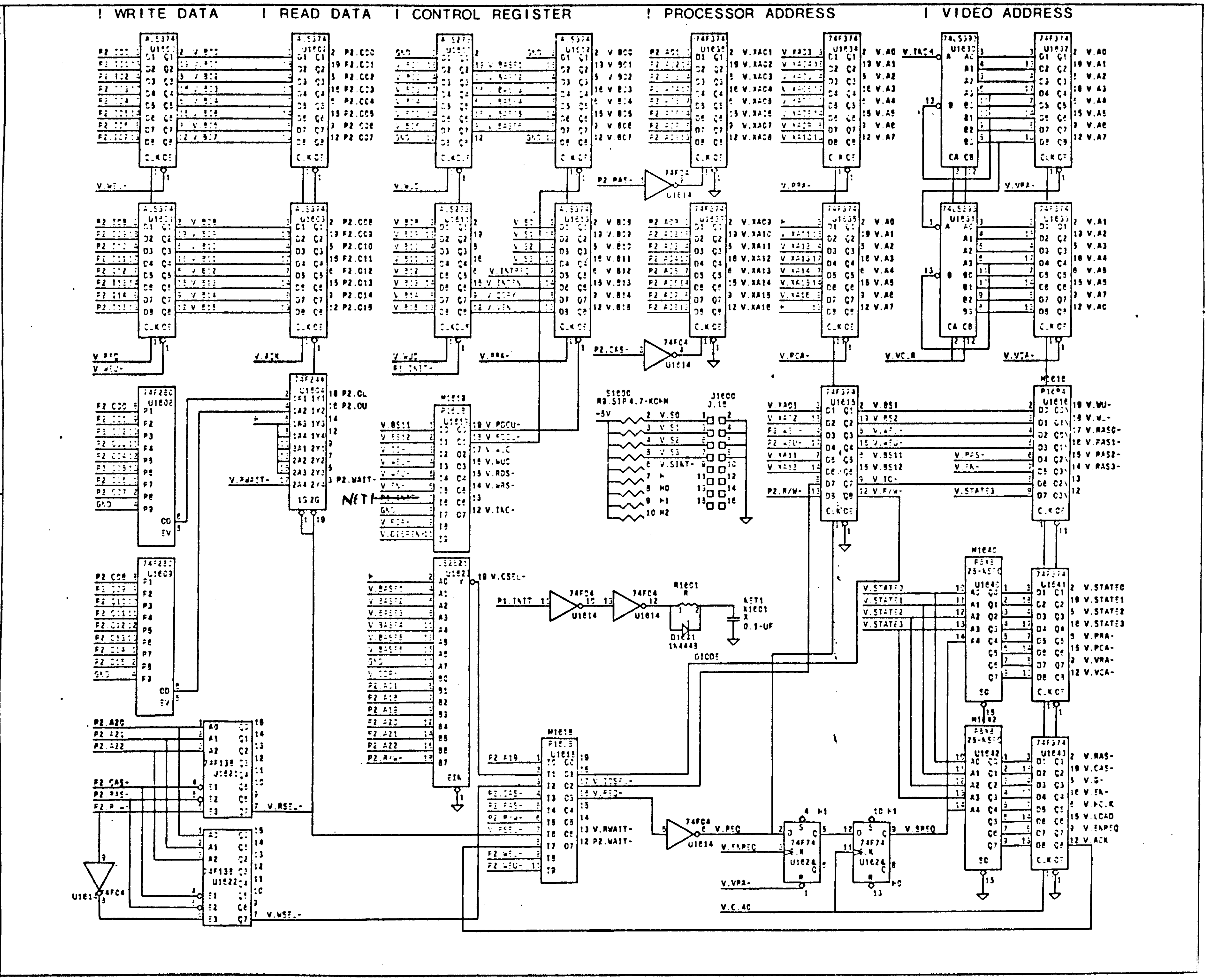
The timing is illustrated in the following figure:

Figure 6-1: Video Clock Timing Diagram

```
V.C.10      --__--__--__--__--__--__--__--__--__--__--__--__--__--__--__--__
V.C.20      ----____----____----____----____----____----____----____----____
V.C.40      --------________--------________--------________--------________
V.STATE0    ________________----------------________________----------------
V.DISPEN    --------------------------------________________________________
V.LOAD      ________________----------------________________________________
V.LDEN      ________________--------________________________________________
V.LD        ________________________--------________________________________
V.ELD       ----____----____----____----____----____----____----____----____
V.ECLK      __--__--__--__--__--__--__--__--__--__--__--__--__--__--__--__--
```

Appendix A

Video Board Schematics



TITLE PROPRIETARY SMI, SUN-2, VIDEO.

COMPUTER SCIENCE DEPARTMENT, STANFORD UNIVERSITY, STANFORD, CALIFORNIA 94305

FILE M16[1SMI]

DATE 14-Nov-84 15:29

PAGE 16 OF 19

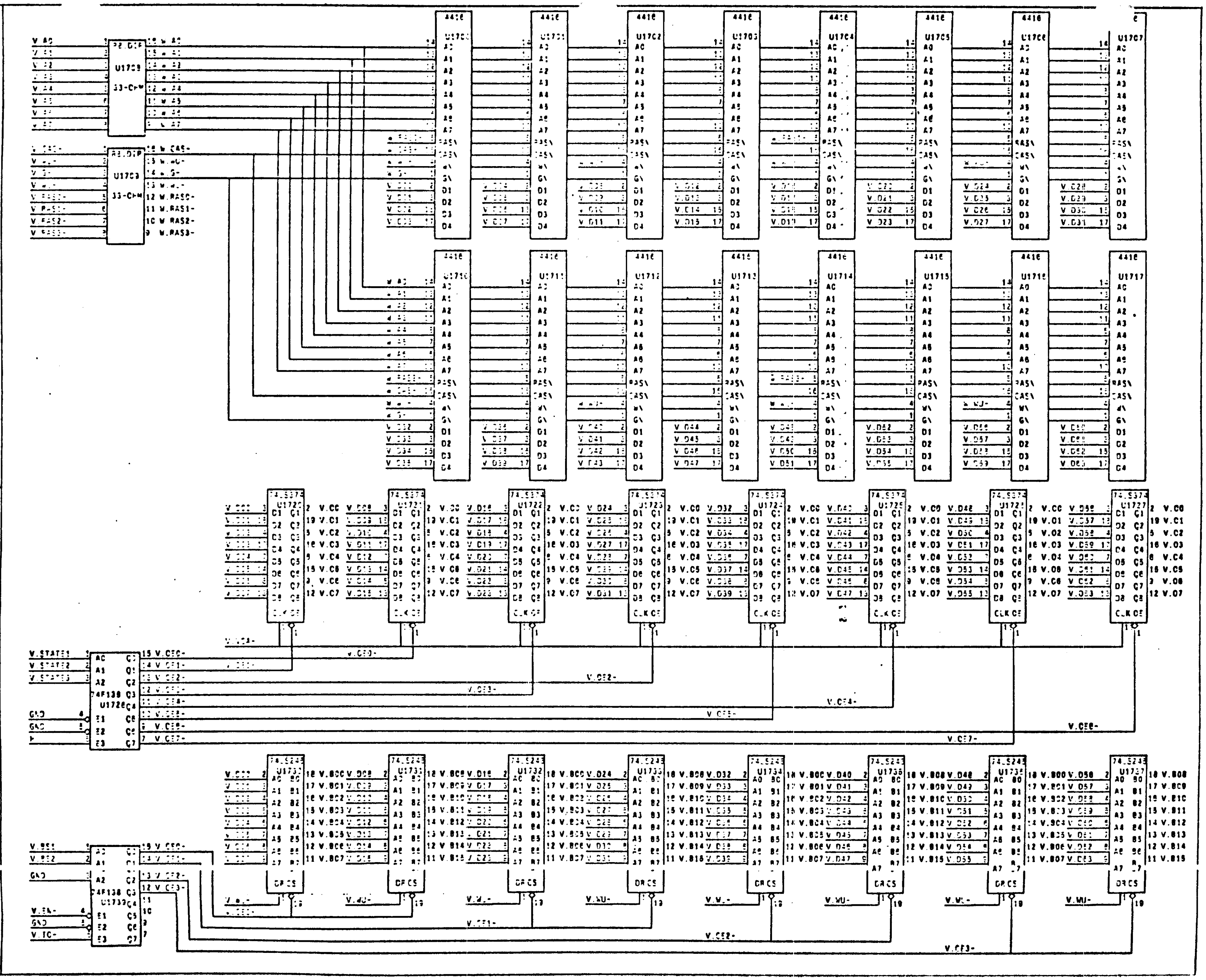
TITLE
PROPRIETARY SMI, SUN-2, FRAME BUFFER,

COMPUTER SCIENCE DEPARTMENT, STANFORD UNIVERSITY, STANFORD, CALIFORNIA 94305

FILE
M17[1SMI]

DATE
3-Oct-84 11:28

PAGE 17 OF 19



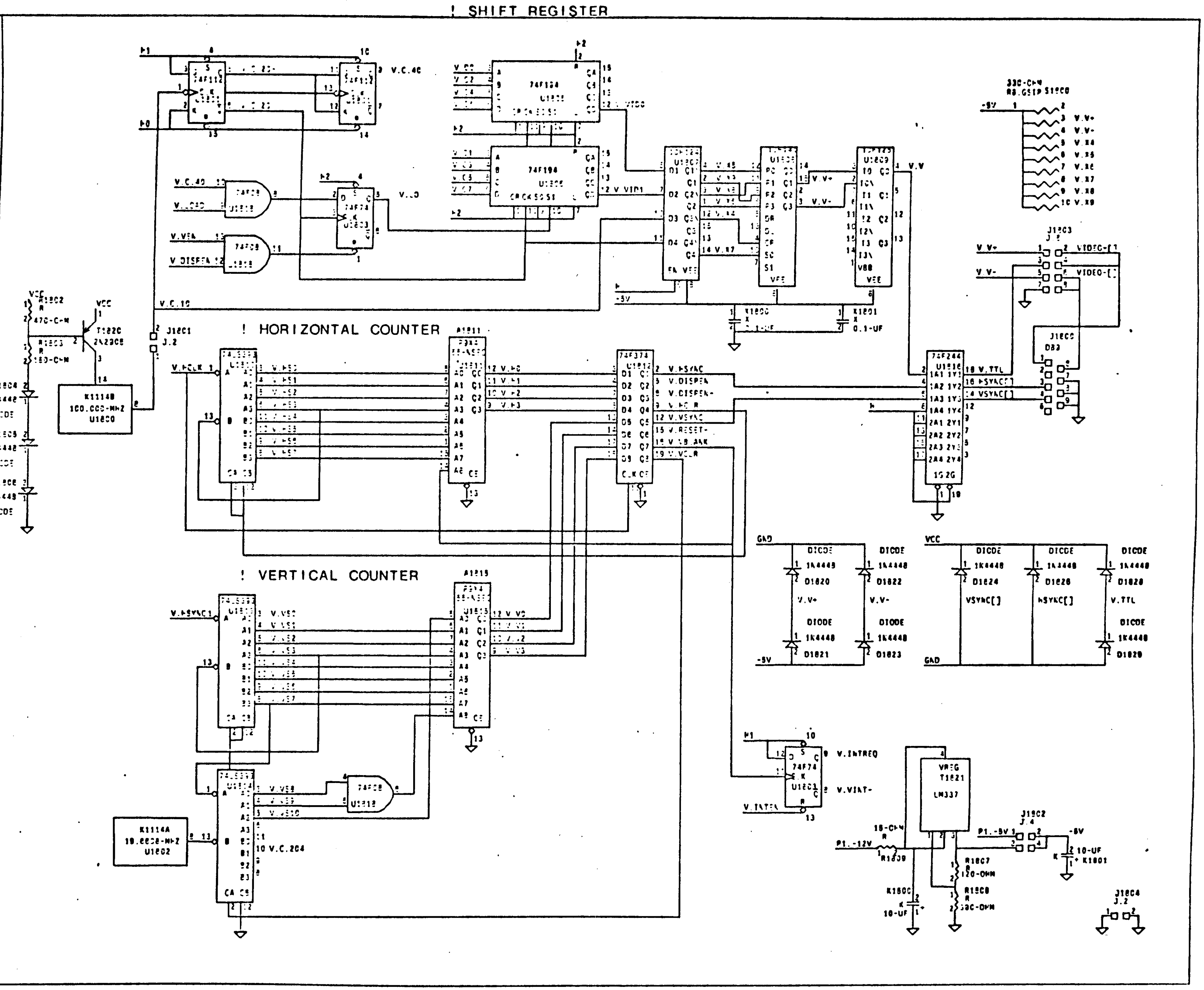
TITLE PROPRIETARY SMI, SUN-2, VIDEO.

COMPUTER SCIENCE DEPARTMENT, STANFORD UNIVERSITY, STANFORD, CALIFORNIA 94305

FILE M18[1SMI]

DATE 14-Nov-84 15:32

PAGE 18 OF 19

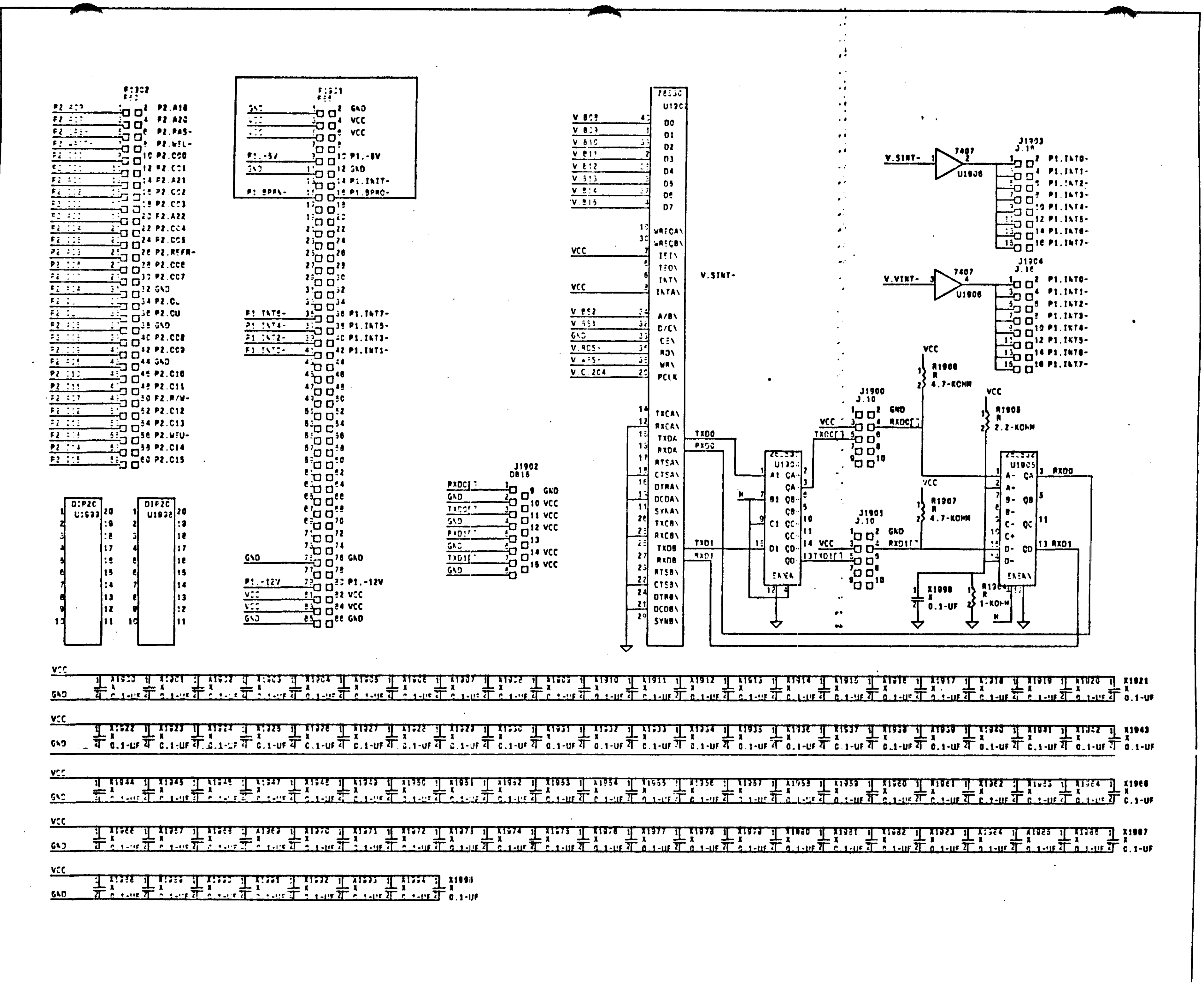


TITLE
PROPRIETARY SMI, SUN-2, VIDEO,
COMPUTER SCIENCE DEPARTMENT, STANFORD UNIVERSITY, STANFORD, CALIFORNIA 94305

FILE
M19[1SMI]

DATE
14-Nov-84 15:33

PAGE 19 OF 19



Appendix B

Video Board PALs and PROMs

```
paltype pal16r4
palname M1616
palid 1.0 84/05/15

PALBEGIN

% Inputs

2  INPUT V.BS1
3  INPUT V.BS2
4  INPUT V.LDS-
5  INPUT V.UDS-
6  INPUT V.RAS-
7  INPUT V.EN-
8  INPUT V.IO-
9  INPUT V.STATE3

1  CLOCK CLK
10 GND
11 OUTPUT-ENABLE OE
20 VCC

% Outputs
19 OUTPUT V.WU-
18 OUTPUT V.WL-
17 OUTPUT V.RAS0-
16 OUTPUT V.RAS1-
15 OUTPUT V.RAS2-
14 OUTPUT V.RAS3-

EQUATIONS

: VIDEO_CYCLE   V.STATE3 ;                      % Video Cycle
: CPU_CYCLE    / V.STATE3 & / V.IO    ;                  % CPU Memory Cycle

% Equations to generate all video RAM RAS strobes

ASSERT V.RAS0-                                         % RAS bank 0
OR        V.RAS & / V.BS2 & / V.BS1 & CPU_CYCLE        % CPU cycle to this bank
OR        V.RAS & VIDEO_CYCLE                          % Video cycle runs all banks

ASSERT V.RAS1-                                         % RAS bank 1
OR        V.RAS & / V.BS2 &   V.BS1 & CPU_CYCLE
OR        V.RAS & VIDEO_CYCLE

ASSERT V.RAS2-                                         % RAS bank 2
OR        V.RAS &   V.BS2 & / V.BS1 & CPU_CYCLE
OR        V.RAS & VIDEO_CYCLE
```

```
ASSERT V.RAS3-                                         % RAS bank 3
OR          V.RAS &   V.BS2 &   V.BS1 & CPU_CYCLE
OR          V.RAS & VIDEO_CYCLE

% Simple AND gates we use because they're handy

ASSERT V.WU-                                           % Write to upper memory byte
ENABLE ALWAYS
OR          V.EN & V.UDS

ASSERT V.WL-                                           % Write to lower memory byte
ENABLE ALWAYS
OR          V.EN & V.LDS

PALEND
```

```
paltype pal1618
palname M1618
palid 1.0 84/05/15

PALBEGIN

% Inputs

1  INPUT P2.A19
2  INPUT V.CSEL-
3  INPUT V.WSEL-
4  INPUT P2.CAS-
5  INPUT P2.RAS-
6  INPUT P2.READ
7  INPUT V.RSEL-
8  INPUT V.XACK
9  INPUT P2.WEU-
11 INPUT P2.WEL-

10 GND
20 VCC

% Outputs

18 INPUT  V.READ
17 OUTPUT V.IOSEL-
16 OUTPUT V.REQ-
15 OUTPUT V.WWAIT-
14 OUTPUT V.RDACK-
13 OUTPUT V.RWAIT-
12 OUTPUT P2.WAIT-

EQUATIONS

ASSERT V.IOSEL-                              % IO select
ENABLE ALWAYS
OR        V.RSEL & P2.A19 & P2.RAS & P2.CAS   & / V.XACK & / V.RDACK
OR        V.WSEL & P2.A19 & P2.RAS & P2.CAS & P2.WEU & / V.XACK
OR        V.WSEL & P2.A19 & P2.RAS & P2.CAS & P2.WEL & / V.XACK

ASSERT V.REQ-                                % Transfer request
ENABLE ALWAYS
OR        V.RSEL & P2.A19- & P2.RAS & P2.CAS    & / V.XACK & / V.RDACK
OR        V.WSEL & P2.A19- & P2.RAS & P2.CAS & P2.WEU & / V.XACK
OR        V.WSEL & P2.A19- & P2.RAS & P2.CAS & P2.WEL & / V.XACK
OR        V.CSEL & P2.RAS & P2.CAS & P2.WEU & / V.XACK
OR        V.CSEL & P2.RAS & P2.CAS & P2.WEL & / V.XACK
OR        V.IOSEL                              & / V.XACK
OR        V.REQ                                & / V.XACK      % Hold til ack

ASSERT V.RDACK-                              % Used to hold deassertion of V.WAIT
ENABLE ALWAYS
OR        V.XACK & V.READ                    % Set at end of read req
OR        V.RDACK & P2.CAS           % Hold till read cycle is gone

ASSERT V.RWAIT-                              % Read Wait
ENABLE ALWAYS
OR        P2.READ & P2.RAS & / V.RDACK       % Set on RD
OR        V.RWAIT & P2.RAS & / V.RDACK       % Hold until V.RDACK
```

```
ASSERT V.WWAIT-                              % Write Wait
ENABLE ALWAYS
OR        / P2.RAS & / P2.READ & V.REQ       % Set at end of write
OR        V.WWAIT & V.REQ & / V.XACK % Hold till V.XACK

ASSERT P2.WAIT-                              % Write Wait to P2-Bus
ENABLE    V.WWAIT
OR        V.WWAIT

PALEND
```

```
paltype pal1618
palname M1619
palid 1.0 84/05/15

PALBEGIN

% modified 9-29-84 : add reset to UART / V.RDS- and V.WRS- both low when
%                          P1.INIT low

% Inputs

1  INPUT V.BS11
2  INPUT V.BS12
3  INPUT V.IO-
4  INPUT V.WEL-
5  INPUT V.WEU-
6  INPUT V.EN-
7  INPUT P1.INIT-
8  INPUT PIN8
9  INPUT V.PCA-
11 INPUT V.DISPEN-

10 GND
20 VCC

% Outputs

19 OUTPUT V.RDCU-
18 OUTPUT V.RDCL-
17 OUTPUT V.WRCL-
16 OUTPUT V.WRCU-
15 OUTPUT V.RDS-
14 OUTPUT V.WRS-
13 OUTPUT PIN13
12 OUTPUT V.INC-

EQUATIONS

ASSERT V.INC-
ENABLE ALWAYS
OR        V.PCA & V.DISPEN

ASSERT   V.RDCU-
ENABLE ALWAYS
OR        V.EN & V.IO & V.BS11 & V.BS12 & / V.WEL & / V.WEU

ASSERT   V.RDCL-
ENABLE ALWAYS
OR        V.EN & V.IO & V.BS11 & V.BS12 & / V.WEL & / V.WEU
OR        V.EN & V.IO & / V.BS11 & / V.BS12 & / V.WEL & / V.WEU
          % enable lower byte during UART accesses
          % to generate valid parity on lower byte

ASSERT   V.WRCL-
ENABLE ALWAYS
OR        V.EN & V.IO & V.BS11 & V.BS12 & V.WEL

ASSERT   V.WRCU-
ENABLE ALWAYS
OR        V.EN & V.IO & V.BS11 & V.BS12 & V.WEU
```

```
ASSERT   V.RDS-
ENABLE ALWAYS
OR        V.EN & V.IO & / V.BS11 & / V.BS12 & / V.WEL & / V.WEU
OR       P1.INIT

ASSERT   V.WRS-
ENABLE ALWAYS
OR        V.EN & V.IO & / V.BS11 & / V.BS12 & V.WEU
OR       P1.INIT

PALEND
```

```
static char* sccs_id = "1.8 84/06/02";

/* This information proprietary to Sun Microsystems Inc */

/* ================================================================
 * Prom Name: M1640 and M1642
 * Prom Type: 32 x 8.
 * Speed: 25 nsec.
 * Purpose: Video Memory State Machines
 * Timing:
 *   The video state machines perform an optional read or write access
 *   to the frame buffer memory followed by a video update read cycle.
 *   The basic memory cycle consists of 16 states; the state machine
 *   is clocked every 40 nsec and, hence, repeats every 640 nsec.
 *
 */

#include "/usr/local/pl/prom.c"
#define range(low,x,high)  ((low<=x)&&(x<=high))

/* Define inputs to 32 x 8 proms */

#define  state0   (a0)
#define  state1   (a1)
#define  state2   (a2)
#define  state3   (a3)
#define  xreq     (a4)

/* Define outputs */

#define nstate nnstate()
nnstate()
{  int state,xstate;
   state = (cvb(state0)*d0+cvb(state1)*d1+cvb(state2)*d2+cvb(state3)*d3);
   xstate = ((state + 1) % 16);
   return(xstate);
}

#define ras       ((xreq && range(2,nstate,6)) || range(10,nstate,14))
#define cas       ((xreq && range(5,nstate,8)) || \
                  (range(13,nstate,15) || (nstate == 0)) )
#define g         ((xreq && range(5,nstate,8)) || range(13,nstate,15))
#define we        ((xreq && range(3,nstate,9)))

#define pra       range( 1,nstate, 3)
#define pca       range( 4,nstate, 7)
#define vra       range( 9,nstate,11)
#define vca       range(12,nstate,15)

#define ack       (xreq && (nstate==10))
#define  hclk     range(0,nstate,1)
#define load      ((nstate % 2) == 1)
#define enreq     (nstate == 0)

main()
{
prom32x8;

prombegin
prom(0,d0, (nstate&d0))
prom(0,d1, (nstate&d1))
prom(0,d2, (nstate&d2))
prom(0,d3, (nstate&d3))
prom(0,d4,!pra)
prom(0,d5,!pca)
prom(0,d6,!vra)
prom(0,d7,!vca)

prom(1,d0,!ras)
prom(1,d1,!cas)
prom(1,d2,!g)
prom(1,d3,!we)
prom(1,d4, hclk)
prom(1,d5, load)
prom(1,d6, enreq)
prom(1,d7, ack)
promend;

writeprom("M1640",0);
writeprom("M1642",1);

}
```

```
static char* sccs_id = "1.12 84/06/20";

#include "/usr/local/pl/prom.c"
#define range(low,x,high) ((low<=x)&&(x<=high))

/* This information proprietary to Sun Microsystems Inc */

/* ==================================================================
 * Date :  March 9, 1984
 * Prom Name: A1811
 * Prom Type: 512 x 4.
 * Speed: 55 nsec.
 * Purpose: Video horizontal state machine.
 * Timing:
 *
 * 1 Horizontal state = 64 pixel; 1 pixel = 10 nsec.
 *
 *              Range     Length   Length  Time
 *              [State]   [State]  [Pixel] [usec]
 * -----------------------------------------------------------------
 * *** 1152 x 900 Display ***
 * cycle       00..24      25      1600    16.00           HFreq = 62.5 KHz
 * visible     00..17      18      1152    11.52
 * invisble    18..24      7        448     4.48
 * frontporch    ..        0          0        0
 * hsync       18..19      2        128     1.28
 * backporch   20..24      5        320     3.20
 * *** 1024 x 1024 Display ***
 * cycle       00..24      25      1600    16.00           HFreq = 62.5 KHz
 * visible     00..15      18      1152    11.52
 * invisble    16..24      7        448     4.48
 * frontporch  16..16      1         64     0.64
 * hsync       17..18      2        128     1.28
 * backporch   19..24      6        384     3.84
 * ------------------------------------------------------------
 */

/* Define Inputs */

int res_1152x900;

#define h0      (a0)
#define h1      (a1)
#define h2      (a2)
#define h3      (a3)
#define h4      (a4)
#define h5      (a5)
#define h6      (a6)
#define h7      (a7)
#define vblank  (a8)

#define state  (cvb(h0)*d0 + cvb(h1)*d1 + cvb(h2)*d2 + cvb(h3)*d3 + \
                cvb(h4)*d4 + cvb(h5)*d5 + cvb(h6)*d6 + cvb(h7)*d7 )

#define nstate  ((state + 1) % 25)

dispen()
{  int dispen;
   if (res_1152x900) {
      dispen =  (!vblank && range(0,nstate,17));
```

```
    } else {
        dispen =  (!vblank && range(0,nstate,15));
    }
    return(dispen);
}

hsync()
{  int hsync;
    if (res_1152x900) {
        hsync =  range(18,nstate,19);
    } else {
        hsync =  range(17,nstate,18);
    }
    return(hsync);
}

#define hreset  (nstate == 0)
#define vclock  (nstate == 21)

main()
{
prom512x4;

res_1152x900 = 1;
prombegin
prom(0,d0, hsync())
prom(0,d1, dispen())
prom(0,d2,!dispen())
prom(0,d3, hreset)
promend;

res_1152x900 = 0;
prombegin
prom(1,d0, hsync())
prom(1,d1, dispen())
prom(1,d2,!dispen())
prom(1,d3, hreset)
promend;

writeprom("A1811",0);
writeprom("A1811_1024",1);
}
```

```
static char* sccs_id = "1.10 84/06/20";

#include "/usr/local/pl/prom.c"
#define range(low,x,high) ((low<=x)&&(x<=high))

/* This information proprietary to Sun Microsystems Inc */

/* ======================================================================
 * Date :  March 9, 1984
 * Prom Name: A1815
 * Prom Type: 512 x 4.
 * Speed: 55 nsec.
 * Purpose: Video vertical state machine
 *
 * 1 states = 1 line = 16.00 usec  (62.50 KHz)
 *
 *             Range        Length  Time
 *             [Lines]      [Lines] [usec]
 * -------------------------------------------------------------
 * *** 1152x900 Display ***
 * cycle       000..936     937     14992     66.70 Hz
 * visible     000..899     900     14400
 * invisible   900..936     37        592
 * frontporch     ..         0         0
 * vsync       900..909     10        160
 * backporch   910..936     27        432
 * *** 1024x1024 Display ***
 * cycle       000..1060   1061     16976     58.91 Hz
 * visible     000..1023   1024     16384
 * invisible  1024..1060     37       592
 * frontporch     ..         0         0
 * vsync      1024..1033     10       160
 * backporch  1034..1060     27           432
 * -------------------------------------------------------------
 */

int res_1152x900;

/* Define Inputs to prom */

#define v0      0
#define v1      (a1)
#define v2      (a2)
#define v3      (a3)
#define v4      (a4)
#define v5      (a5)
#define v6      (a6)
#define v7      (a7)
#define v8      (a8)
#define v9      (a8)
#define v10       (a0)


#define line (cvb(v0)*d0 + cvb(v1)*d1 + cvb(v2)*d2 + cvb(v3)*d3 + \
              cvb(v4)*d4 + cvb(v5)*d5 + cvb(v6)*d6 + cvb(v7)*d7 + \
              cvb(v8)*d8 + cvb(v9)*d9 + cvb(v10)*d10)

vsync()
{  int vsync;
   if (res_1152x900) {
      vsync = range(900,line,909);
   } else {
      vsync = range(1024,line,1033);
   }
   return(vsync);
}

vblank()
{  int vblank;
   if (res_1152x900) {
      vblank = (line >= 900);
   } else {
      vblank = (line >= 1024);
   }
   return(vblank);
}

vreset()
{  int vreset;
   if (res_1152x900) {
      vreset = (line >= 936);
   } else {
      vreset = (line >= 1060);
   }
   return(vreset);
}


main()
{
prom512x4;

res_1152x900 = 1;
prombegin
prom(0,d0, vsync())
prom(0,d1,!vreset())
prom(0,d2, vblank())
prom(0,d3, vreset())
promend;

res_1152x900 = 0;
prombegin
prom(1,d0, vsync())
prom(1,d1,!vreset())
prom(1,d2, vblank())
prom(1,d3, vreset())
promend;

writeprom("A1815",0);
writeprom("A1815_1024",1);
}
```

READER COMMENT SHEET

Dear Customer,
We who work here at Sun Microsystems wish to provide the best possible documentation for our products. To this end, we solicit your comments on this manual. We would appreciate your telling us about errors in the content of the manual, and about any material which you feel should be there but isn't.

Typographical Errors:

Please list typographical errors by page number and actual text of the error.

Technical Errors:

Please list errors of fact by page number and actual text of the error.

Content:

Please list errors of fact by page number and actual text of the error.