Engineering Manual Sun-2 Model 50

Company Confidential

Sun Microsystems Inc.

Part Number: 800-1146-01

Revision: 01 of 12 October 1984

This manual describes the two board that make up a Model 50, the 2050 base board and the 2051 expansion board.

The 2050 Board is a single board workstation computer. It provides on a single, triple height eurocard all components of a high-performance engineering/scientific workstation: processor, memory, virtual memory management, display subsystem, networking, serial I/O, system bus interface, and various system utilities. The processor is based on a 10 Mhz 68010 CPU, extended with the Sun-2 multiprocess virtual memory management. The 2050 board contains one to four megabytes of memory with zero wait state access. Main memory is equipped with byte parity error detection.

The 2051 Board is an optional memory expansion board that provides an additional one to four megabytes of main memory. It also provides a slot for an optional input/output expansion board, such as a floating point processor.

Table of Contents

List of Figures

1. Data Sheet

1.1. Features

1.1.1. Processor

- 32-bit VLSI CPU
- 10 MHz operation with no wait states to main memory
- 1M Bytes (64K) or 1/2/3/4M Bytes (256K) of main memory
- 1M Bytes (64K) or 1/2/3/4M Bytes (256K) of expansion memory
- multiprocess, demand paging virtual memory management
- 16M bytes virtual address space per process
- optional DES encryption processor

1.1.2. Display

- dual-ported 128K Bytes video memory
- 1152 by 900 pixel display resolution
- 67 Hz non-interlaced video refresh

1.1.3. I/O

- integral Ethernet interface transfers directly into memory
- two programmable serial I/O ports with full modem control
- two additional serial interfaces for keyboard and mouse

1.1.4. Other Features

- VME System Bus Interface
- DVMA (direct virtual memory access) from VME Bus
- five programmable 16-bit timers
- 32K to 128K Bytes EPROM
- extensive self-diagnostic capabilities
- triple-height Eurocard form factor

1.2. Introduction

The Sun-2050 Board is a high-performance implementation of the Sun-2 architecture on a single 400mm by 366.67mm Eurocard. The board includes the CPU, virtual memory management, optional processor enhancements, one to four megabytes of main memory with parity error detection, a high-resolution display subsystem, integral Ethernet and RS-423 interfaces, and a dual-ported interface to the VME-bus.

The processor is based on the Motorola 68010 32-bit VLSI CPU, extended with the Sun-2 virtual memory management unit (MMU). The processor executes from main memory at 10 MHz without wait states. The MMU was specifically optimized to support the demand paging requirements of the the 4.2 BSD version of the Unix (TM) operating system. It provides multiple, simultaneous process contexts with up to 16 megabyte virtual memory space each. In addition, the MMU provides separate address spaces for the system and for the user.

The Sun 2050 board contains 1M Bytes (64K RAM) to 4M Bytes (256K RAM) of main memory. With the Sun 2051 memory expansion board, another 1M Bytes (64K) to 4M Bytes (256K) of main memory can be added. 64K and 256K RAMs can be intermixed between the 2050 board and the 2051 board, and overall memory can be expanded in 1M Bytes increments. Memory is equipped with byte parity error detection.

Integral to the Sun-2050 Board is a high-resolution bitmap display subsystem featuring a 1152 by 900 pixel display area and non-interlaced, 67 Hz refresh. The display is refreshed out of a dedicated, dual-ported 128K Bytes video memory, which is logically part of main memory.

The Sun-2 Single Board workstation includes an integral Ethernet interface. This interface uses a VLSI Ethernet controller that features high-performance frame handling and extensive diagnostic capabilities. Ethernet packets are directly transferred in and out of main memory through the use of direct virtual memory access (DVMA).

For serial I/O, two highly programmable serial communication channels are provided featuring software programmable baud rates from 75 Baud to 19.2 KBaud and supporting asynchronous, synchronous, or bit-stuffing protocols. Two additional ports are provided for keyboard and mouse interfaces.

The Sun-2050 Board includes a bidirectional interface to the VME Bus with master and slave capabilities. The board provides 24-bit address and 16-bit data transfer capabilities in both directions. It also implements system controller functions such as arbitration, interrupt handling, reset, and power monitoring.

Other features of the board include an optional DES encryption processor, programmable timers, and an identification PROM providing software-readable serial number and Ethernet address.

The board also includes extensive facilities for software and hardware diagnostics. Among them are a bus-error register, a diagnostic display for displaying error messages, a watchdog timer for automatic restart, and powerup self-tests.

1.3. Sun-2 Architecture Overview

The 2050 Board implements a Sun-2 architecture machine. The complete specification of the architecture is contained in the Sun-2 Architecture Manual. The following is a brief overview of the architecture and its implementation on the 2050 Board.

The Sun-2 architecture is divided into three spaces: the CPU space, MMU space, and Device space.

The CPU space comprises the central processing unit (the "CPU") together with coprocessors, such as the floating point coprocessor, and DVMA masters, such as the Ethernet interface.

The MMU space is the core of the Sun-2 architecture. It includes the Sun-2 memory management unit (the "MMU") as well as all other Sun-2 architecture extensions to the CPU, such as the bus error register, the system enable register, the diagnostic register, and the ID-PROM. The ID-PROM contains a unique serial number and configuration data for a particular implementation of the architecture.

The Device space of the Sun-2 architecture defines what devices exist in the architecture and how they are accessed. These devices include main memory, the system bus, and I/O devices.

All CPU accesses to device space pass through the MMU and thus are translated and protected in an identical fashion. In addition, direct memory accesses by I/O devices also pass through the memory memory management and thus operate in a fully protected environment.

1.4. 2050 Board Block Diagram

Figure 1-1 illustrates how the CPU, MMU, and devices are interconnected on the 2050 Board.

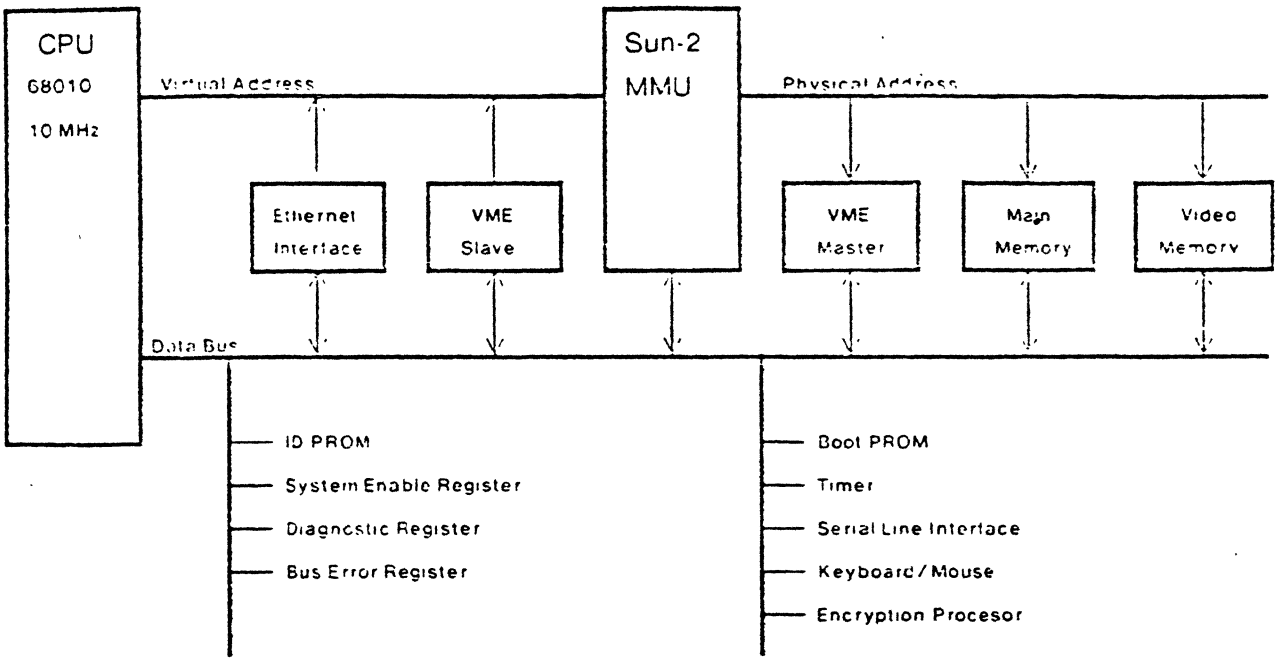


Figure 1-1: Sun 2050 Board Architecture

The CPU sends out a virtual address that is translated by the MMU into a physical address. The CPU, Ethernet Interface, and VME Slave Interface arbitrate for and share the virtual address bus on the left side of the MMU. The VME Master Interface, Main Memory, Video Memory, and I/O Devices are addressed with physical addresses on the right side of the MMU.

1.5. Sun-2 MMU Overview

The Sun-2 Memory Management Unit provides address translation, protection, sharing, and memory allocation for multiple processes executing on the CPU.

The memory management consists of a context register, a segment map, and a page map. Virtual addresses from the processor are translated into intermediate addresses by the segment map and then into physical addresses by the page map.

The memory management uses a page size of 2K Bytes and a segment size of 32K Bytes (giving 16 pages per segment). Up to 8 contexts can be mapped concurrently. The maximum virtual address space for each context is 16M Bytes.

Figure 1-2 shows how virtual addresses are translated into physical ones.

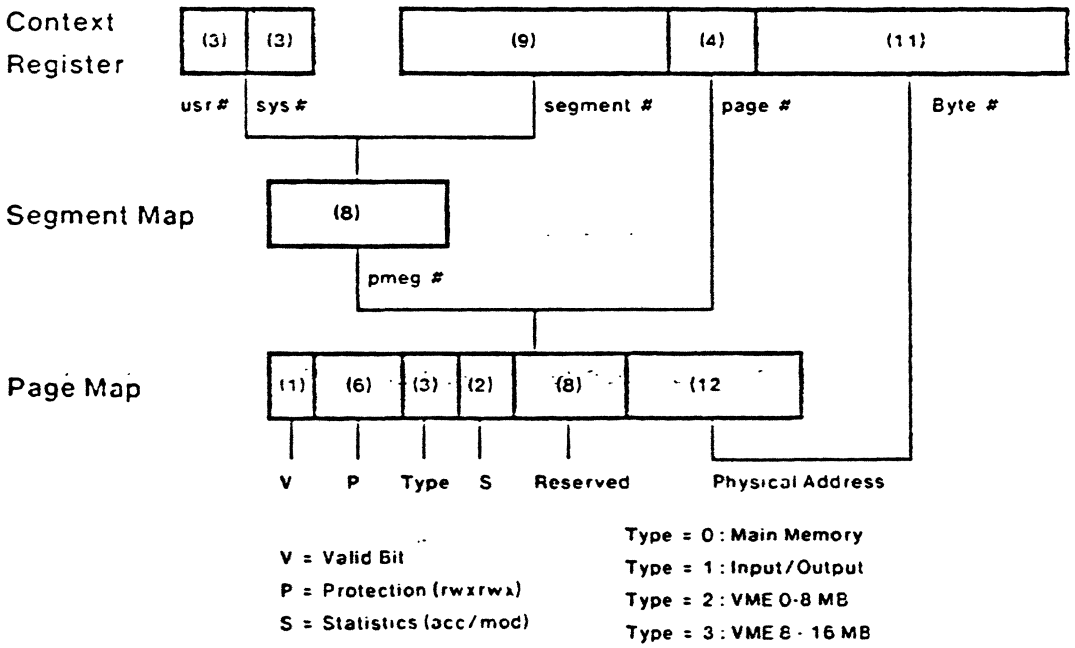


Figure 1-2: Sun-2 Memory Management

1.6. 2050 Board FloorPlan

Figure 1-3 gives an overview of layout of the 2050 Board, which is the main CPU board.

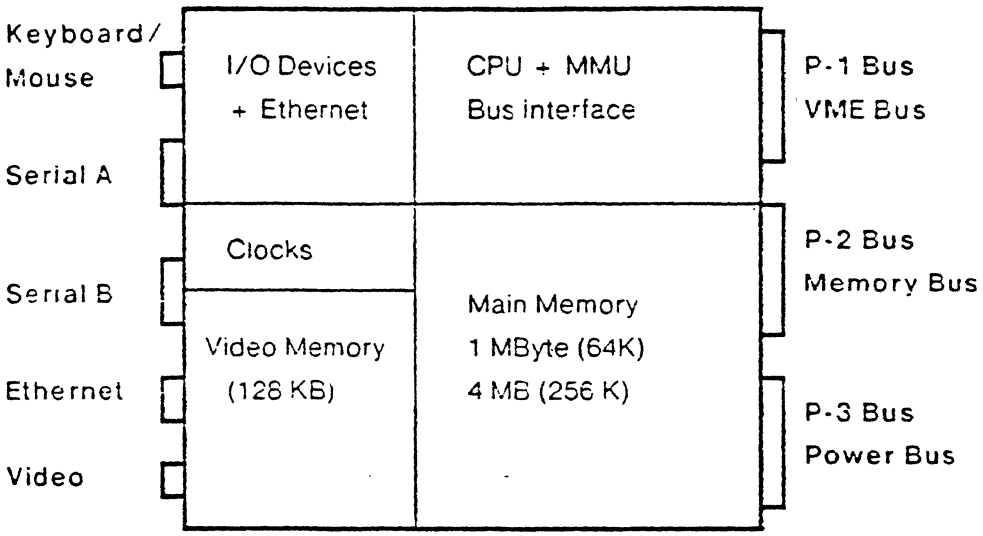


Figure 1-3: Sun 2050 Board Floor Plan

The connectors on the backplane of the board are called the P1, P2, and P3 connectors. The P1 Connector carries the VME Bus, also referred to as the P1-Bus. The P2 Connector serves for the memory expansion bus, or the P2-Bus. The P3 Connector powers the board.

The connector on the input/output side of the board are, in sequence from top to bottom: [J605] Keyboard/Mouse Connector, [J603] Serial Port A, [J604] Serial Port B, [J700] Ethernet Port, and [J1800] Video Connector.

1.7. 2051 Board FloorPlan

Figure 1-4 gives an overview of layout of the 2051 Board, which is the memory expansion board.

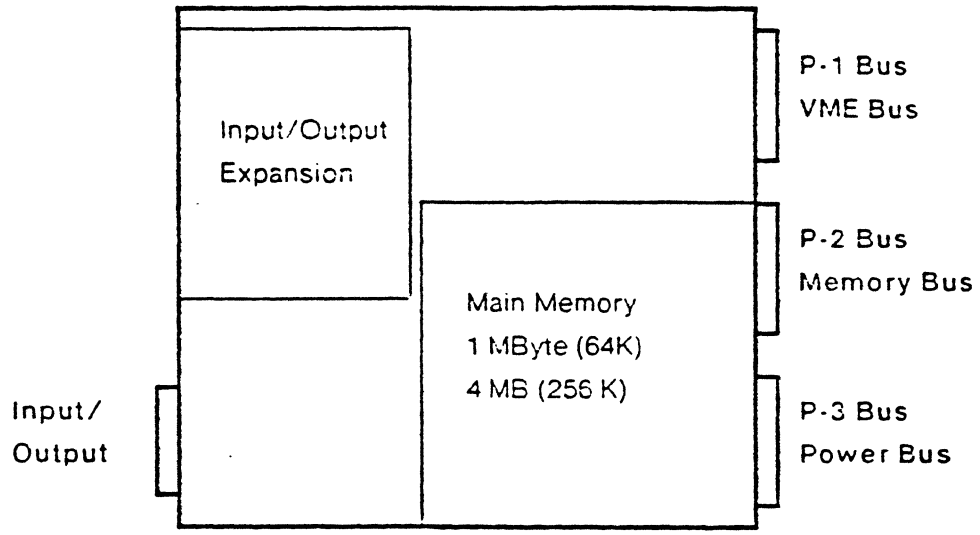


Figure 1-4: Sun 2051 Board Floor Plan

The connectors on the backplane of the board are called the P1, P2, and P3 connectors. The P1 Connector carries the VME Bus, also referred to as the P1-Bus. The P2 Connector serves for the memory expansion bus, or the P2-Bus. The P3 Connector powers the board.

The piggy-back connectors on the board provide for expansion with one input/output board, i.e., a floating point processor board. The connector at the lower left of the board provides an interface from the input/output board to the outside.

1.8. Specification Summary

1.8.1. CPU

- M68010 CPU, 10 MHz

1.8.2. Memory

- 1M Bytes (64K) or 1/2/3/4M Bytes (256K) of main memory
- 1M Bytes (64K) or 1/2/3/4M Bytes (256K) of expansion memory
- high-speed, no-wait state operation
- transparent hardware memory refresh
- byte parity error detection

1.8.3. Memory Management Unit

- Sun-2 memory management unit
- two-level, multiprocess virtual memory management
- full support for demand paging
- 16M Bytes virtual address space per process
- separate address spaces for supervisor and user
- valid, accessed, and modified tags to assist paging algorithms
- separate read, write, and execute tags for user and supervisor accesses

1.8.4. Display Subsystem

- dedicated dual-ported video memory
- 1152 by 900 display format
- 100 MHz video clock
- 67 Hz non-interlaced video refresh

1.8.5. Ethernet Interface

- VLSI Ethernet controller (82586)
- digital phase decoder
- packets transferred directly in and out of main memory
- extensive diagnostic capabilities

1.8.6. Serial I/O Ports

- two programmable serial I/O ports
- based on synchronous communication controller (8530)
- software programmable baud rates (75 baud to 19.2 kilobaud)
- asynchronous, synchronous, and bit-stuffing protocols
- two serial ports for keyboard and mouse

1.8.7. Other Features

- VME System bus interface
- DVMA (direct virtual memory access) from VME Bus
- optional DES encryption processor (AMD 9518)
- up to 128K Bytes EPROM (27128, 27256, 27512)
- five programmable 16-bit timers (AMD 9513)
- software interrupt capability
- software readable identification PROM (storing serial number and other information)

1.8.8. Diagnostic Features

- diagnostic LED display
- bus error register
- watchdog reset timer
- bus timeout timer

1.9. VME-bus Specification

1.9.1. Master Capabilities

- Data Bus Size: D16 MASTER 16-bit/8-bit data
- Address Bus Size: A24 MASTER 24-bit/16-bit addresses
- Timeout Option: TOUT(100 USEC) 100 microsecond timeout period
- Sequential Access: None
- Interrupt Handler: IH(1-7) STAT Level 1 through 7, jumperable
- Requestor Option: ROR R(3) Release on Request, level 3

1.9.2. Slave Capabilities

- Data Bus Size: D16 SLAVE 16-bit/8-bit data
- Address Bus Size: A24 SLAVE 24-bit-only addresses
- Sequential Access: None
- Interrupter Options: None

1.9.3. System Controller Capabilities

- Clock Option: SYSCLK 16 MHz, jumperable
- Arbiter Option: ONE Bus Request Level 3 Only
- Note: The 2050 Board must be the System Controller in a VME System.

1.9.4. Power Monitor Capabilities

- ACFAIL Option: ACFAIL asserted when VCC < 4.5V
- SYSRESET Option: SYSRESET asserted during CPU Reset
- SYSFAIL Option: SYSFAIL not used

1.9.5. Environmental Characteristics

- Operating Temperature: 10 - 55 C
- Humidity: 0 - 90 %, non-condensing

1.9.6. Power Characteristics

- 12 Amp max at +5 Volt +- 5%
- 0.5 Amp max at +12 Volt +- 5%
- 0.5 Amp max at -12 Volt +- 5%

1.9.7. Physical Characteristics

- Height: 366.67 mm (14.44")
- Width: 400.00 mm (15.75")
- Depth: 40.64 mm (1.6")
- Weight: 1788 g (64 oz)

2. User Guide

2.1. Programming

The 2050 Board implements the Sun-2 Architecture, Machine Type 2. The full architecture is documented in the Sun-2 Architecture Manual and no attempt is made to repeat this information here. However, this section does describe the features specific to this implementation of the architecture.

2.2. MMU Implementation

The MMU of this machine type implements a page number field of 12 bits. It thus supports a physical address of 23 bits, capable of addressing 8M Bytes. The other physical address bits in the page map are not implemented. When read, those bits not implemented remain undefined.

2.3. Physical Address Assignments

Type	Address	Device	Wait States
0	23-bit	Memory Bus	
	[0x000000]	Physical Memory 1..8M Bytes	0
1	23-bit	I/O Bus	
	[0x000000]	BW-Video Memory	1 (Write), 4..8 (Read)
	[0xC20000]	Video Control Register	2
	[0x7F0000]	EPROM	2
	[0x7F0800]	Ethernet Interface	2
	[0x7F1000]	Encryption Processor	2..8
	[0x7F1800]	Keyboard/Mouse Interface	2
	[0x7F2000]	Serial Port	2
	[0x7F2800]	Timer	2
	[0x7F3000]	Reserved	2
	[0x7F3800]	Reserved	2
2	23-bit	P1-Bus or System Bus	
	[0x000000]	0..8M Bytes VME 24-bit address	1 + device access time
3	23-bit	P1-Bus or System Bus	
	[0x000000]	8..16M Bytes VME 24-bit address	1 + device access time
	[0x7F0000]	64K Bytes VME 16-bit address	1 + device access time

Accesses to the VME Bus incur an additional 2 wait states access time if the 2050 board is not currently bus master.

2.4. Interrupt Assignments

The following table summarizes the interrupt level assignments for the devices that have been described in this manual. All these interrupts are autovectored.

7	TIMER1
6	Serial Port
5	TIMER2..5
4	VIDEO
3	Ethernet or system enable register EN.INT3
2	System enable register EN.INT2
1	System enable register EN.INT1

In addition, the VME-bus can cause vectored interrupts on all levels. Individual VME-bus interrupt levels can be disabled with jumpers.

2.5. Performance Data

2.5.1. CPU Speed

```
CPU clock cycle:        101.72 nsec (9.8304 MHz)
CPU basic cycle:        406.90 nsec
```

2.5.2. Video Memory Access Time

Read accesses are unbuffered and will cause 4 to 8 wait states. Write accesses to the video memory are buffered. However, subsequent read or write accesses will have to wait until the video memory has completed the requested operation. Write accesses to the video memory via the copy mode will cause the same behavior as direct write accesses.

2.5.3. P1-Bus Access Times

This section describes the access times of the P1-Bus. The time to complete a P1-Bus access consists of three elements: overhead, the cost of P1-Bus acquisition if the 2050 Board is not currently P1-Bus master, and the actual access time of the P1-Bus device.

The total number of wait states for a P1-Bus access can be computed by the following formula:

1 WS (overhead) + 2 WS (bus acquisition time if board does not have bus mastership and bus is idle) + access time of P1-Bus device divided by the clock period of the CPU rounded up to the nearest integer number.

2.5.4. DVMA Access Time

DVMA cycles from the P1-Bus are serviced after the current CPU cycle completes and after pending memory refresh cycles are executed. Thus DVMA cycles exhibit a variable access time that ranges from 0.7 microseconds in the best case to 1.5 microseconds worst case with an average of about 1.0 microseconds.

After a DVMA cycle has executed, a CPU cycle will start before another DVMA cycle is granted. This means that the cycle time for DVMA is one DVMA cycle plus at least one CPU cycle. Thus the DVMA cycle time will be in the range of 1.1 to 1.9 microseconds with an average of 1.4 microseconds, as long as the DVMA master can generate transfers at this rate.

2.5.5. P1-Bus Reset

The 2050 Board can be configured either as a P1-Bus Reset Master or Slave.

As a P1-Bus Reset Master, the 2050 Board issues Reset to the VME Bus. Power-On Reset, Watchdog Reset, and 68010 Reset will all assert P1-Bus Reset. Other P1-Bus devices may also assert P1-Bus Reset, but this will have no effect on the on-board CPU and devices.

As a P1-Bus Reset Slave, the 2050 Board receives Reset from the VME Bus, but does not drive Reset to the VME Bus. The VME Bus Reset has the same effect as an on-board power-on-reset.

2.6. Connectors

This section documents the pinout of all the connectors used on the Sun 2050 board.

2.6.1. J603: Serial Port A

PIN	SIGNAL	PIN	SIGNAL
1	----	14	----
2	TXDA[]	15	DBA[]
3	RXDA[]	16	----
4	RTSA[]	17	DDA[]
5	CTSA[]	18	----
6	DSRA[]	19	----
7	GND	20	DTRA[]
8	DCDA[]	21	----
9	----	22	----
10	----	23	----
11	----	24	DAA[]
12	----	25	VEE
13	----	--	----

2.6.2. J604: Serial Port B

PIN	SIGNAL	PIN	SIGNAL
1	----	14	----
2	TXDB[]	15	DBB[]
3	RXDB[]	16	----
4	RTSB[]	17	DDB[]
5	CTSB[]	18	----
6	DSRB[]	19	----
7	GND	20	DTRB[]
8	DCDB[]	21	----
9	----	22	----
10	----	23	----
11	----	24	DAB[]
12	----	25	VEE
13	----	--	----

2.6.3. J605: Keyboard/Mouse

PIN	SIGNAL	PIN	SIGNAL
1	RXD0[]	9	GND
2	GND	10	VCC
3	TXD0[]	11	VCC
4	GND	12	VCC
5	RXD1[]	13	----
6	GND	14	VCC
7	TXD1[]	15	VCC
8	GND	--	----

2.6.4. J700: Ethernet

PIN	SIGNAL	PIN	SIGNAL
1	----	9	E.COL-
2	E.COL+	10	E.TXD-
3	E.TXD+	11	----
4	----	12	E.RXD-
5	E.RXD+	13	+12V
6	GND	14	----
7	VCC	15	----
8	----	--	----

2.6.5. J1800: Video

PIN	SIGNAL	PIN	SIGNAL
1	VIDEO+	6	VIDEO-
2	----	7	GND
3	HSYNC	8	GND
4	VSYNC	9	GND
5	----	-	----

2.7. Jumpers

This section describes all the jumpers used on the board. In the following listing, each group of jumpers denotes exclusive combinations. That means, within each group only one jumper combination may be active at a time.

2.7.1. Configuration Jumpers

These jumpers allow configuration of the 2050 Board for specific applications. Default Jumpers are marked with an asterisk (*).

LABEL	PINS	DESCRIPTION IN/OUT
*J702	1-2	Enable/Disable 5 Volt to Ethernet
*J704	1-2	Level 2/Level 1 Ethernet Transceiver
*J800	1-2	Enable/Disable VME Interrupt Level 1
*J800	3-4	Enable/Disable VME Interrupt Level 2
*J800	5-6	Enable/Disable VME Interrupt Level 3
*J800	7-8	Enable/Disable VME Interrupt Level 4
*J800	9-10	Enable/Disable VME Interrupt Level 5
*J800	11-12	Enable/Disable VME Interrupt Level 6
*J800	13-14	Enable/Disable VME Interrupt Level 7
*J900	1-2	DVMA Address Comparator A20=0/1
*J900	3-4	DVMA Address Comparator A21=0/1
*J900	5-6	DVMA Address Comparator A22=0/1
*J900	7-8	DVMA Address Comparator A23=0/1
*J900	9-10	Enable/Disable VME Arbiter
*J900	11-12	Enable/Disable VME Reset Master
J900	13-14	Enable/Disable VME Reset Slave
*J900	15-16	Enable/Disable VME System Clock
*J1600	1-2	Video Register Sense Bit 0
*J1600	3-4	Video Register Sense Bit 1
*J1600	5-6	Video Register Sense Bit 2
*J1600	7-8	Video Register Sense Bit 3

2.7.2. Permanent Jumpers, 2050 Board

The following jumpers are factory installed and are normally not modified. Those installed normally are indicated with an asterisk (*).

LABEL	PINS	DESCRIPTION IN/OUT
*J200	1-2	Enable/Disable UART Clock
*J200	3-4	10/12 MHZ CPU operation
J200	5-6	12/10 MHZ CPU operation
J200	7-8	Reserved
J200	9-10	Reserved
*J200	11-12	Enable/Disable Ethernet Clock
*J200	13-14	Enable/Disable Memory Refresh
*J200	15-16	Enable/Disable Timeouts
J500	1-2	PROM TYPE = 27128
*J500	3-4	PROM TYPE = 27256 or 27512
*J500	5-6	PROM TYPE = 27128 or 27128
J500	7-8	PROM TYPE = 27512
J1201	1-2	Enable/Disable 2nd megabyte (256K RAM)
J1201	3-4	Enable/Disable 3/4 megabyte (256k RAM)
*J1201	5-6	64K/256K RAMs
J1201	7-8	256K/64K RAMs
*J1201	9-10	64K/256K RAMs
J1201	11-12	256K/64K RAMs
*J1201	13-14	64K/256K RAMs
J1201	15-16	256K/64K RAMs
J1600	9-10	Reserved
J1600	11-12	Reserved
*J1600	13-14	10/12 MHZ CPU operation
J1600	15-16	12/10 MHZ CPU operation
*J1801	1-2	Enable/Disable 100 MHZ Video Clock

The jumper positions for different PROM sizes are summarized in the table below.

PROM	JUMPER	JUMPERED PINS
27128	J600	1-2 and 5-6
27256	J600	3-4 and 5-6
27512	J600	3-4 and 7-8

2.7.3. Permanent Jumpers, 2051 Board

On the memory expansion board, there are jumpers for different memory sizes. The jumpers are factory installed and are normally not modified. For a 1 megabyte base board, the jumpers for the memory expansion board are as follows, organized by memory size on memory expansion board:

SIZE	JUMPER	JUMPERED PINS
1 MB	J2200	3-4
1 MB	J2201	5-6, 9-10, 13-14
2 MB	J2200	3-4, 5-6
2 MB	J2201	3-4, 7-8, 11-12
3 MB	J2200	3-4, 5-6, 7-8
3 MB	J2201	7-8, 11-12, 15-16
4 MB	J2200	3-4, 5-6, 7-8, 9-10
4 MB	J2201	7-8, 11-12, 15-16

In addition, the 2051 Board has a jumper block for the daisy chained VME-bus grant lines and the VME-bus interrupt acknowledge chain. These jumpers are installed in systems that need to daisy-chain those VME-bus signals.

JUMPER	PINS	FUNCTION
J2100	1-2	BUS GRANT 0
J2100	3-4	BUS GRANT 1
J2100	5-6	BUS GRANT 2
J2100	7-8	BUS GRANT 3
J2100	15-16	IACK CHAIN

3. Theory of Operations

This chapter describes the theory of operations of the 2050 Board and the conventions that are used in the schematics.

3.1. Conventions

This section describes the conventions employed in the schematics and the documentation of this board. The discussion assumes that the reader has a working knowledge of digital electronics and has access to descriptions of the components used on the board.

3.1.1. Schematics

The schematics is contained in the file with extension ".PRE". The suffix of the schematics file names reflects the drawing page number.

3.1.2. Signal Conventions

Whenever possible, standard drawing conventions are employed. Signal flow is shown from left to right, and top to bottom.

Both active-high and active-low signals are used. A signal name that is followed by a minus ("-") indicates that the signal is asserted active low (<0.4V), e.g. OE-. Conversely, a signal that is not followed by a minus is an active high signal (>2.0V).

For signals with multiple meanings or synonyms, the synonyms are listed separated by a slash "/". For example, the signal name for a read-write signal that is active low for write is "READ/WRITE-".

Signals that are part of busses 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 to "D15". A group of signals that are part of a signal vector are denoted by a common prefix separated by the suffix with ".". For example, all P1 signals start with the prefix "P1.".

Connector signals are distinguished by a suffix of "[]" with an optional string enclosed inside the square brackets identifying the connector name.

3.1.3. Component Conventions

Components are identified by component name (e.g. 74LS00), component location (e.g. U100), and properties if required (e.g. 100-OHM).

Component names (also referred to as Body Name in the wirelist) indicate the type of component being used. The component name is derived from the "generic" or industry standard name. Component names are translated into Diptypes that specify the physical component associated with the component name. There is only one diptype for components that are sections of the same physical package (e.g. four 74LS00 gates form one 74LS00 diptype). Diptypes are translated by the parts list into manufacturer codes and part names.

Component locations provide a unique designator for the component. They are chosen to indicate the schematics page on which the component is located. For example, component U100 is most likely positioned on page 1. Component locations consist of one letter followed by one to four digits. The letter indicates the type of component and is one of:

Letter	Component Type
C	Standard Capacitor
D	Diode
K	Electrolytic Capacitor
L	Inductance
X	Decoupling Capacitor
J	Jumper or Connector
R	Resistor
S	single-in-line component
U	dual-in-line component

Location labels are cross-indexed in the wirelist into diptype and component names and locations on the schematics.

Component Properties help to further specify a generic component. Three types of properties are used:

Property	Meaning	Example	Interpretation
:	Value Specification	:10-UF	This capacitor has a value of 10 UF
=	Reference	=A50C	This part is referred as part A50C
+	Additive Property	+S40	Add a 40-pin socket to this component

3.1.4. State Diagrams

State Diagrams are drawn to the following conventions:

1. Left to right with incrementing state numbers along the horizontal axis.
2. Signal transitions represent the actual logic levels of the named signal.
3. Signals are represented without propagation delays.

3.2. Major Blocks

The major logic blocks are:

- Power
- Initialization
- Clocks
- CPU
- IO-Bus Interface
- MMU
- I/O Devices
- P2-Bus Interface
- Memory
- Video Subsystem
- Ethernet Interface

3.3. Power

Reference: Schematics Page 19

The 2050 Board uses +5V for all of its onboard logic. It also requires a +12V for the Ethernet transceiver and a -5V for the RS423 drivers and the Video ECL circuitry. The -5V is generated from the -12V supply by on-board regulator [LM337:U137]. Signal [-5VR] connects to the UART connectors pin 25 to terminate that line.

3.4. Initialization

Reference: Schematics Page 19

The 2050 Board includes a power-on/power-off reset generator that provides an accurate reset pulse. The circuit uses a dual comparator [LM393:U133], a 1.2 Volt reference voltage [LM385.D101], charge capacitor [K:K100], and resistor network [R:R100..R107].

The first comparator forms a power-on reset generator by comparing the voltage from the charge capacitor with the reference. This comparator asserts its output until the voltage across the charge capacitor corresponds to a VCC of 4.5 Volt. The second comparator forms a power-off reset generator by comparing the +5V supply with the reference. This comparator asserts its output when the +5V supply voltage is below 4.5 Volt without the charge delay incurred by the first comparator. The output of both comparators is wire ORed so that signal power-on-reset [POR] is active when either comparator asserts its output.

3.5. Clock Oscillators

Reference: Schematics Page 2

The 2050 Board has 4 independent clock oscillators on board. They are:

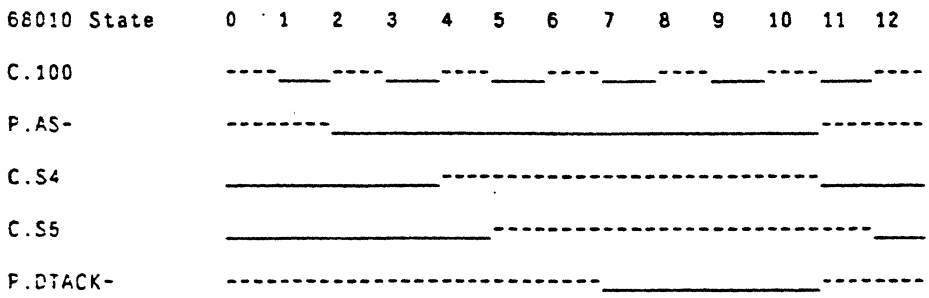
- 10-MHz CPU clock and constant clock (19.6608 MHz) [K1114A:U200].
- 12-MHz CPU clock (24.0000 MHz) [K1114A:U201].
- Ethernet clock and VME system clock (16.0000 MHz) [K1114A:U202].
- Video clock (100.0000 MHz) [K1114A:U1800].

In addition, the Ethernet PLL [MB502:U701] features its own crystal oscillator with a frequency of 100 MHz. All clock oscillators have disconnect jumper for ATE test purposes. The 12-MHz CPU Clock is installed only if the board is configured for that frequency.

3.6. Derived Clocks

The system clock is the particular CPU clock for which the board is configured divided by two in flipflop [74F74:U203-0]. Counter [74LS590:U211] divides the system clock into clocks for the data ciphering processor, the timeout counter, and the refresh clock. Counter [74LS393:U212-1] divides the constant clock [C.51A] into clock [C.204] for the UART and the Timer.

The clock strobes [C.S4, C.S5, ETC] are derived from transitions on the system clock and enabled with processor address strobe [P.AS] as illustrated in the figure below for a 12-state cycle.



3.7. CPU

Reference: Schematics Page 1

3.7.1. Reset

CPU Reset is generated by PAL [P16R4:U109] under three conditions:

- *Power-On-Reset:* The power-on-reset generator asserts [POR] that causes PAL [P16R4:U109] to assert processor reset.
- *External Reset:* If the 2050 Board is configured as a reset slave, then VME System Reset [P1.SYSR] asserts RESIN that causes PAL [P16R4:U109] to assert processor reset
- *Watchdog Reset:* If the CPU halts it asserts [P.HALT]. In this case, PAL [P16R4:U109] automatically generates processor reset to continue processing.

3.7.2. Special Cycles

In the discussion below, reference will be made to *special cycles*. A special cycle is one in which the 68010 function code is neither program or data. Special cycles include CPU space cycles (FC = 7) and MMU space cycles (FC = 3). Supervisor program fetches in Boot state, which are forced to read from the Boot PROM, are also treated as special cycles.

Special cycles are recognized in PAL [P16L8:U101] and cause signal [Q.SPECIAL] to be asserted. [Q.SPECIAL] inhibits the assertion of [Q.CAS] in flipflop [74F74:U204], inhibits the assertion of [P.BERR] in PAL [P16L8:U103], and selects signal [SPWAIT] as source for DTACK in selector [74F151:U118]. Thus during special cycles no bus errors can occur, and the source for the [DTACK] is [SPWAIT].

3.7.3. DTACK

The CPU uses a number of handshake signals to generate the timing required by the devices it is accessing. The following table gives the source of the DTACK for the different page types and special cycles.

Condition	Device	DTACK
TYPE=0	Main Memory	CS4 * (READ + ~P2.WAIT)
TYPE=1	Video Memory	CS7 * ~P2.WAIT
TYPE=1	I/O	CS7 * ~P2.WAIT
TYPE=2	VME	P1.DTACK
TYPE=3	VME	P1.DTACK
FC=6 ^ BOOT	EPROM	CS7
FC=3	MMU Access	CS7
FC=7 ^~A19	Breakpoint	Internal
FC=7 ^ LOCAL	Autovector	Internal
FC=7 ^~LOCAL	VME Interrupt	P1.DTACK

The handshaking is implemented with selector [74F151:U118] in conjunction with PAL [A101 = P16L8:U101] and PAL [A106 = P16L8:U106].

3.7.4. BERR

Bus error can occur under the following conditions:

- Invalid Page Entry
- Protection Error
- Parity Error Lower Byte
- Parity Error Upper Byte
- VME Bus Error
- Timeout

These error conditions, together with signal [Q.SPECIAL] are ORed in gate [74ALS30:U130] asserting [Q.ERROR]. Whenever [Q.ERROR] is active it disables all read-write strobes by disabling strobe decoder [74F138:U400] and I/O decoder [LS2521:U403].

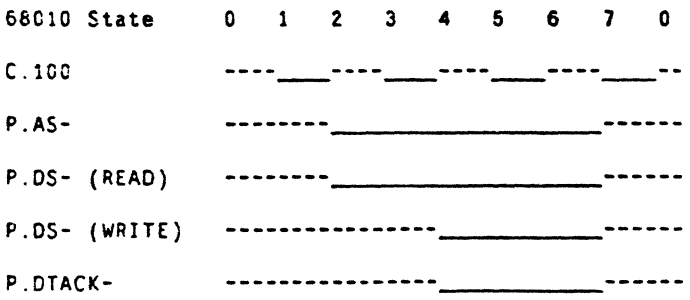
If [Q.ERROR] occurs in a non-special 68010 cycle, three things happen. First, PAL [A103 = P16L8:U103] asserts [Q.BERR] after state [C.S5], thereby aborting the current cycle. Second, the PAL generates [Q.BERRCLK] which latches the error condition into the bus error register [ALS534.U511]. Third, the PAL cleares the parity error flipflops [74F74:U424] with signal [Q.PARCLR] in case they were set.

3.7.5. Address Error Cycles

During address error cycles, the 68010 asserts address strobe but no data strobes. The effect of this is that a normal cycle is executed; however, since no data strobes are active no read or write strobes are asserted via decoder [74F138:U400]. The statistic bits in the MMU are updated on address error cycles.

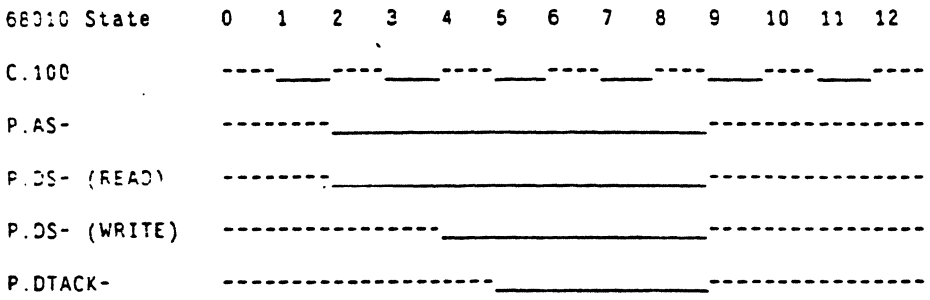
3.7.6. 68010 Cycle to Memory

68010 cycles to memory execute normally without wait states by asserting DTACK at state 4. The only exception to this is write cycles to memory that are shadowed to the video memory (in video copy mode). In that case, signal [P2.WAIT] will delay subsequent write cycles until it is deasserted, indicating that the video memory is ready to accept additional write requests.



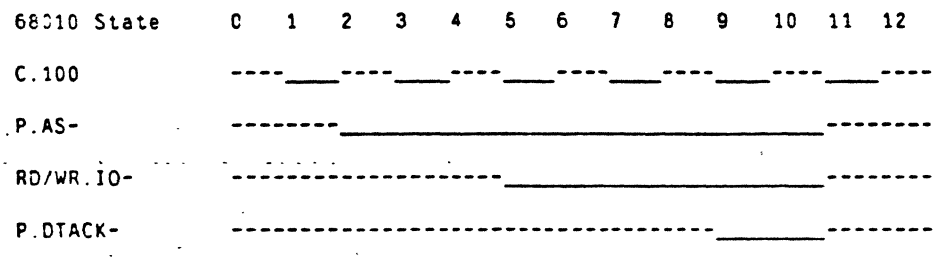
3.7.7. 68010 Write Cycle to Video Memory, Best Case

68010 write cycles to video memory generate DTACK at state 5, causing 1 wait state. Cycles will be longer if a previous video memory cycle has not completed yet.



3.7.8. 68010 Cycle to I/O

68010 Cycles to I/O generate DTACK at state 9, causing 2 wait states. Cycles to the data ciphering processor use special timing described in the data ciphering chip section.



3.8. DVMA Logic

Reference: Schematics Page 2, Motorola 68010 Data Sheet.

3.8.1. Overview

The DVMA Controller takes requests from DVMA devices, obtains the processor bus from the 68010, and performs a read/write cycle for the device, generating appropriate function codes and strobes.

The DVMA Devices in their order of priority are:

- Refresh
- Ethernet
- VME-bus

Figure 3-1 shows how the DVMA Controller and Strobe Generator interface to the 68010.

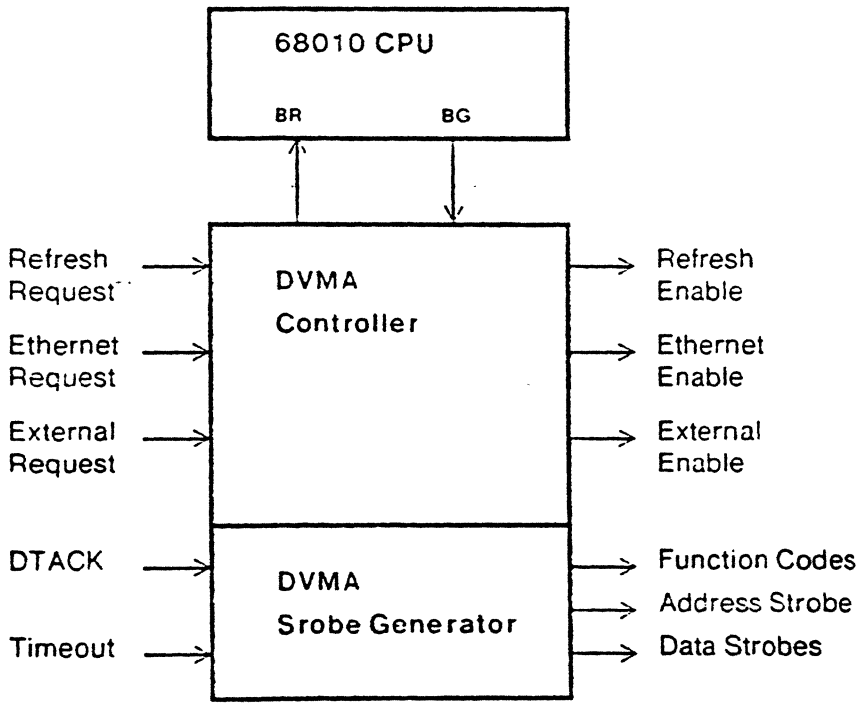


Figure 3-1: DVMA Controller

3.8.2. DVMA Cycles

DVMA requests are posted in the request flipflops [74F74:U207-0, U207-1, U203-1] with the rising edge of the signals [R.REF, E.AS, X.DMA], respectively. The request flipflops are reset by signals [R.DMAEN, E.CLR, X.DMAEN] respectively.

Posted DVMA requests are synchronized with register [74F374:U213] before entering the DVMA controller PAL [P16R8:U214]. The DVMA controller PAL prioritizes the incoming requests, issues a bus request to the 68010 [S.BR], then waits for the 68010 to release the processor bus by watching 68010 bus grant [P.BG] and the end of 68010 address strobe [P.AS], before asserting the DVMA enable corresponding to the request.

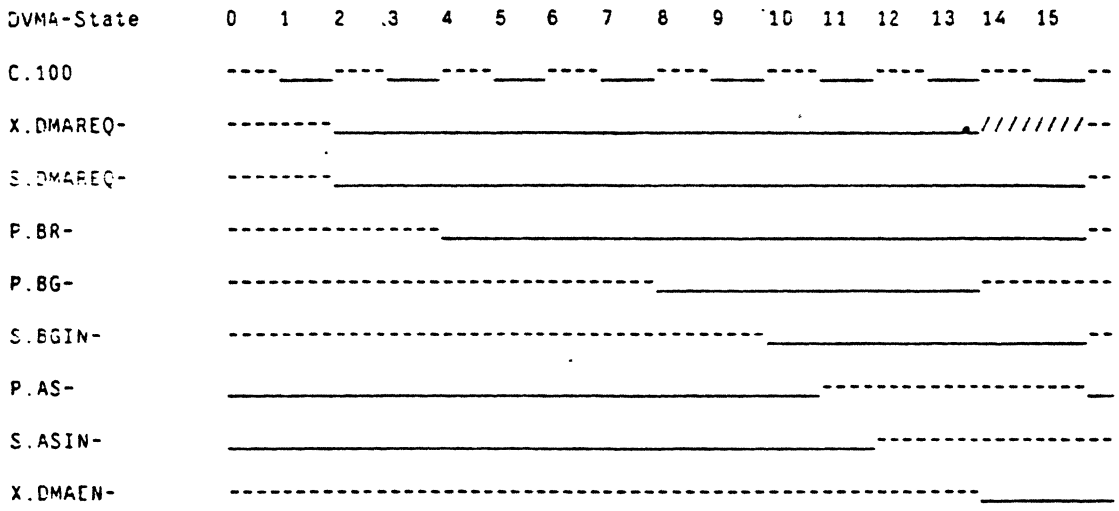
In addition, the DVMA controller PAL generates a DMA-cycle signal [S.DMA] that enables the tri-state buffers in the DVMA strobe PAL [P16L8:U215] to drive the function codes [P.FC0..2], address strobe [P.AS], and data strobes [P.UDS, P.LDS] and Ethernet read/write strobes [E.WE, E.OE]. Function Codes, data strobes, and device codes are asserted as follows:

DVMA	DMA	LDS	UDS	FC	Space
REFRESH	3	0	0	7	CPU Space
ETHERNET	2	1	~E.A0	5	System Data
EXTERNAL	1	X.LDS	X.UDS	5	System Data

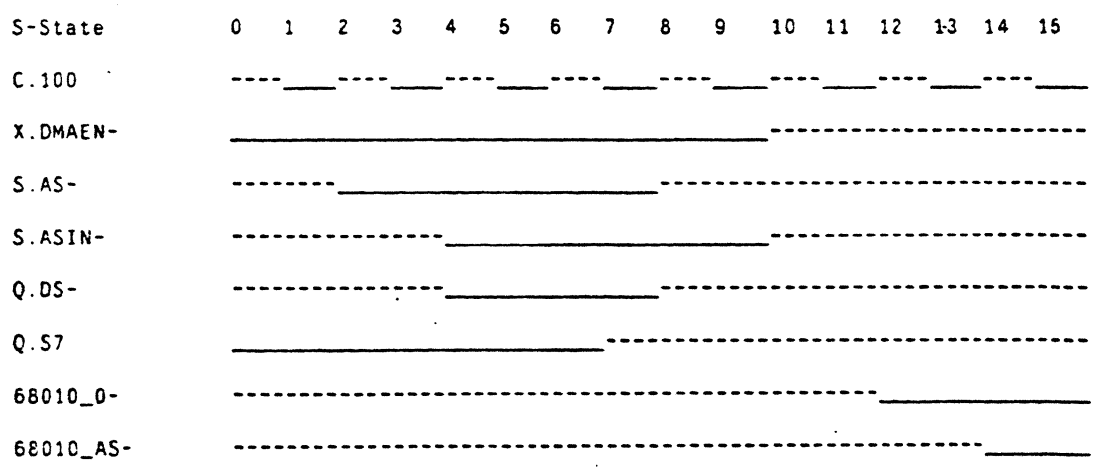
Address strobe [P.AS] is asserted one state after [S.DMA], being enabled by signal [S.ASON] that is one clock cycle delayed from [S.DMA]. Address strobe is terminated with [S.ASOFF] from the DVMA controller PAL. [S.ASOFF] is asserted on refresh cycles when [S.ASIN] is received, and for all other cases when signal [Q.S7] was received, indicating normal completion, or signal [S.ERR] indicating timeout. Since [Q.S7] is derived from the bus handshake signal [P.DTACK] the DVMA controller is able to perform transfers to asynchronous bus devices.

3.8.3. DVMA Arbitration Cycle

Arbitration occurs concurrently with ongoing bus activity. The 68010, after receiving a bus request [P.BR-] issues a bus grant [P.BG-]. When the DVMA controller sees bus grant and address strobe [P.AS] deasserted, it acquires the bus and asserts the DMA Enable.



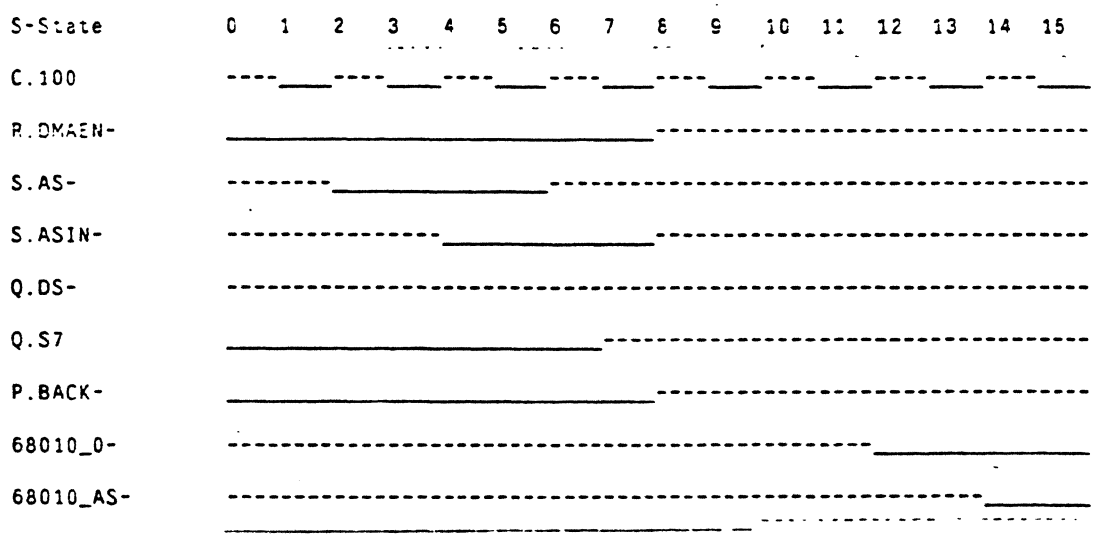
3.8.4. DVMA Cycle, Synchronous Memory



3.8.5. DVMA Cycle, Memory Refresh

Memory refresh requests are generated every 12.8 microseconds by a low-to-high transition of output [C.12800] of synchronous counter [74LS590:U211]. This transition sets signal [R.DMAREQ] in flipflop [74F74:U207-0], which in turn is sychronized in register [74F374:U213] and causes a refresh cycle in DVMA controller [P16R8:U214]. During the refresh cycle [R.DMAEN] is asserted which output-enables refresh counter [74LS590:U210] and with its trailing edge advances the refresh counter to its next state. The refresh counter drives address lines [P.A02..09] which constitute the row-address refresh addresses of the RAM chips. During refresh cycles, both banks of memory are enabled. This is done via PAL [P16L8:U108] asserting both [Q.BANK0, Q.BANK1] causing the RAS generation logic to assert both [P2.RAS0, P2.RAS1].

Refresh cycles are shorter than other DVMA cycles in that they only last for eight states. This is shown in the timing diagram below.



3.9. I/O-Bus

Reference: Schematics Page 1

The I/O data bus [IO.D00..15] connects to the following devices:

- all input/output devices,
- all MMU devices, and
- the VME data port.

The I/O data bus is connected to processor data bus [P.D00..15] via bidirectional transceivers [8308:U110, U111]. These bus buffers are enabled via PAL [P16R4:U102] as follows. The I/O Bus is driven from the processor data bus on all processor write cycles and all DVMA read cycles. The processor data bus is driven from the I/O-bus on all DVMA write cycles and all processor read cycles from I/O devices, MMU devices, and VME Bus.

3.10. MMU and MMU Space Devices

Reference: Schematics Page 3

3.10.1. Overview

The MMU consists of user context register [LS2518:U300], system context register [LS2518:U301], user/system context multiplexor [74F158:U302], segment map RAM [2168:U303, U304], and page map RAM [2168:U305 THROUGH U310].

Other MMU space devices are the bus error register [ALS534:U511], the system enable register [ALS534:U511] with readback [ALS244:U513], the diagnostic register [ALS273:U514] with LEDs [LED4,J515, J516], and the ID PROM [P5X8:U510].

3.10.2. Decoding

The MMU and MMU space devices are accessed via decoders [ALS138:U322, U323, U324]. Decoder [ALS138,U324] is the read decoder, decoder [ALS138:U323] is the upper byte write decoder, and decoder [ALS138,U322] is the lower byte write decoder. All MMU space devices are connected to the lower byte.

3.10.3. MMU Operation

During a normal address translation cycle, the processor system function code [P.FC2] selects between the user [P.FC2 = 0] and supervisor [P.FC2 = 1] context. The selected context value, together with address lines [P.A15..23] index the segment map RAM which produces a page-map-entry-group [IA16..23]. The page-map-entry-group, in conjunction with address lines [P.A11..15] index the page map RAM, producing as its output the valid bit [VALID], protection codes [PROT0..5], type field [TYPE0..1], accessed bit [ACC], modified bit [MOD], and mapped address lines [MA11..22].

The protection field is checked with decoder [74F151:U315]. If the protection bit corresponding to the state of the read/write line [0.R/W] an the processor function codes [P.FC1, P.FC2] is not set, then output [PROTERR] will be asserted.

The accessed and modified bits are updated on all non-special cycles. For this operation, the current value of the type field, which is in the same nibble as the accessed and modified bit, is latched into register [ALS374:U316] with clock [C.S5].

The actual update starts with [C.S5] and ends with [C.S7]. During this time, PAL [P16L8:U103] asserts both [WR.UPDATE], which turns on write enable to RAM [2168:U307], and [WR.STAT] which output enables register [ALS374:U316] with the new data to be written into RAM [2168:U307].

3.11. I/O Devices

Reference: Schematics Page 4, 5, 6, 7

3.11.1. Overview

Input/Output devices comprise the PROMs, the Ethernet Control Register, the Keyboard/Mouse UART, the Serial Communication Controller, and the Timer chip. All input/output devices connect to the IO-Bus.

3.11.2. Decoding

Input/output devices are selected with a MMU type field of 1 [TYPE1 = 0], [TYPE0 = 1], and address lines [MA16..22] all ones. This condition is decoded with comparator [LS2521:U403] in conjunction with gate [74F32:U433-3] producing signal [CE.IO].

[CE.IO] disables the P1/P2-Bus decoder [74F138:U400] and in conjunction with [Q.R-/W] enables I/O read decoder [ALS138:U401] and in conjunction with [Q.R/W-] I/O write decoder [ALS138:U402]. Both the I/O read and write decoder decode mapped address lines [MA11..13] to select one of eight possible devices.

3.11.3. PROMs

Since the PROMs are larger than a single 2K page, they are addressed directly with the low-order bits of the non-translated (virtual) address from the CPU, [P.A01..A14]. The PROMs are constantly chip enabled with CE tied to ground. The PROMs are output-enabled with signal [OE.PROM] generated in PAL [P16L8:U101] during boot cycles [BOOT = 1,FC = 6] and during read-PROM cycles [RD.PROM = 1].

3.11.4. Timer

Timer chip [AM9513:U504] provides five 16-bit timers. The timer is driven by a 4.9152 MHz input clock [C.204], generated from the 19.6608 MHz clock oscillator [K1114A:U200] via binary counter [74LS393:U212], independent of the CPU clock.

Gate input 1 of the timer chip is wired to the timer [FOUT] signal. Output [OUT1] is connected to interrupt request 7 [IRQ7], and outputs [OUT2..5] drive interupt request 5 [IRQ5] via open-collector inverters [74LS05:U505].

3.11.5. Keyboard/Mouse

The serial keyboard/mouse UART [8530:U600] are implemented with a SCC (serial communication controller). The SCC features two high-speed, highly programmable serial channels with built-in baud-rate generators. The clock input to the SCC is a 4.9152 MHz input clock [C.204], independent of the CPU clock.

The serial lines to and from the keyboard/mouse are driven and received via inverters [74LS04:U608, U610].

3.11.6. Serial Communication Controller

The RS-423 UARTS [8530:U601] are implemented with the same type SCC as the keyboard/mouse interface.

Serial port A occupies channel A of the UART, in conjunction with inverter [74LS04:U608], driver [26LS29:U609], and receiver [26LS32:U606].

Serial port B occupies channel B of the UART, in conjunction with inverter [74LS04:U610], driver [26LS29:U611], and receiver [26LS32:U607].

Receiver [26LS32:U615] is shared between channel A and B for synchronous UART applications. Purpose of resistors [R4.SIP:S601] is to provide RS-232 compatible fail-safe line termination.

3.11.7. Ethernet Control Register

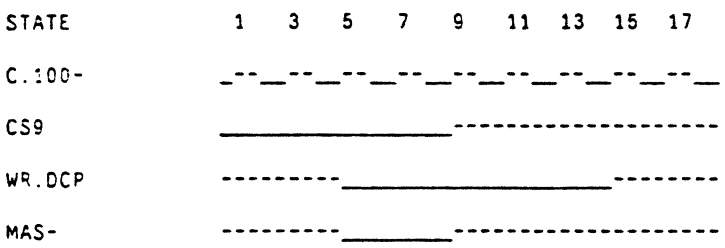
The Ethernet Control Register [ALS273:U716, ALS244:U717] controls the overall operation of the Ethernet interface. Register [ALS273:U716] is reset with processor reset. Further information on the Ethernet operation is contained in the section on Ethernet.

3.11.8. Data Ciphering Processor

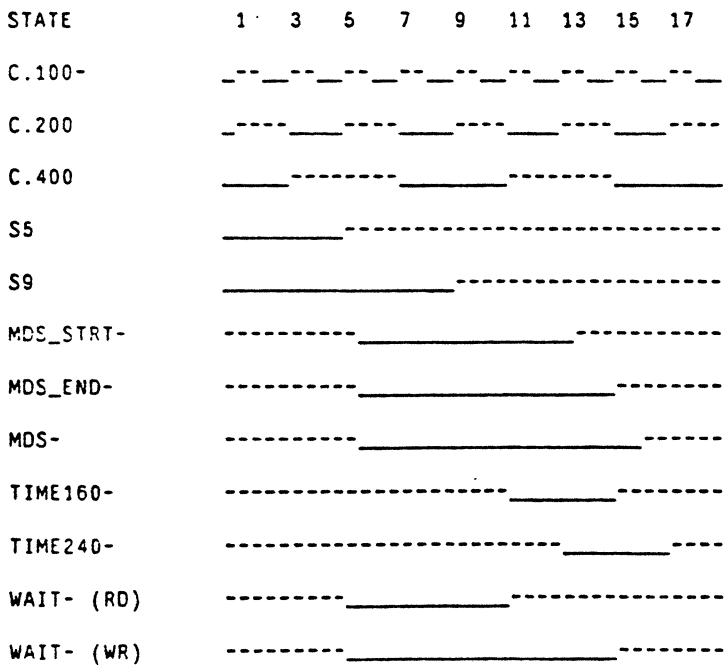
Reference. Schematics Page 5

The Data Ciphering Processor [9518:U506] has special timing requirements implemented by PAL [P16L8:U507]. One requirement of the DCP is that its data strobe [MDS] may only be deasserted 20-70 nsec after trailing edge of its clock [C.400]. Other requirements of the DCP are long hold times on data and read/write; those are achieved by turning off the DCP data strobe early before the end of the cycle. The state diagrams below illustrate these timings.

3.11.9. 68010 Address Load to DCP



3.11.10. 68010 Read/Write to DCP, Best Case



3.12. P2-Bus Interface

Reference: Schematics Page 4, 10 ,11.

3.12.1. Introduction

The P2-Bus is the internal bus which interconnects the CPU to main memory, the video memory, and expansion memory. Going off-board, the P2-Bus is brought out on connector [P96:P1102], pins 1 through 32 and 65 through 96.

3.12.2. P2 Signals

The P2-Bus consists of a number of address lines, bidirectional data lines, parity lines, timing signals, enable signals, and a handshake line.

Address Lines, Data Lines, Read/Write Line, and Handshake Line:

P2.Signal	Description
P2.A00.23	Address Lines (24)
P2.D00.15	Data Lines (16)
P2.DIL, DIU	Parity from CPU to Memory (2)
P2.DOL, DOU	Parity from Memory to CPU (2)
P2.R/W	Read/Write Strobe
P2.WAIT	Negative Handshake

Control Signals:

P2.Signal	Description	Asserted on
P2.RAS	Row-Address-Strobe	C.S3
P2.RAS0	Row-Address-Strobe 0	C.S3 ∧ ¬P.A01 ∨ C.S3 ∧ REFRESH
P2.RAS1	Row-Address-Strobe 1	C.S3 ∧ P.A01 ∨ C.S3 ∧ REFRESH
P2.CAS	Column-Address-Strobe	C.S4 ∧ ¬Q.SPEC
P2.RD	P2-Bus Read Strobe	C.S5 ∧ ¬ERROR ∧ ¬CE.IO ∧ ¬TYPE1
P2.WEU	P2-Bus Write Strobe	C.S5 ∧ ¬ERROR ∧ ¬CE.IO ∧ ¬TYPE1
P2.WEL	P2-Bus Write Strobe	C.S5 ∧ ¬ERROR ∧ ¬CE.IO ∧ ¬TYPE1

The memory control signals [Q.RAS, Q.RAS0, Q.RAS1, Q.CAS, Q.WEL AND Q.WEU] are generated centrally on the CPU side of the P2-Bus.

RAS is generated by and-or gates [74F64:U218] in conjunction with inverter [74F04:U922]. [Q.RAS] is asserted when processor address strobe is active (P.AS = 1) and the clock is low (C.100 = 0). This is the case at the beginning of processor state 3. After RAS is first asserted, it is latched via inverter [74F04:U922] until the later of [C.S7] or [P.AS] being deasserted. [Q.RAS0, Q.RAS1] have the same timing as [Q.RAS] except they are only asserted if [Q.BANK0, Q.BANK1] are asserted, respectively.

[CAS] is generated by flipflop [74F74:U204-1]. It is asserted at time [C.S4] on non-special cycles [Q.SPECIAL = 0]. [CAS] is inhibited during special cycles because the column address is not guaranteed to be stable during memory management updates and thus would cause invalid decoding in memory.

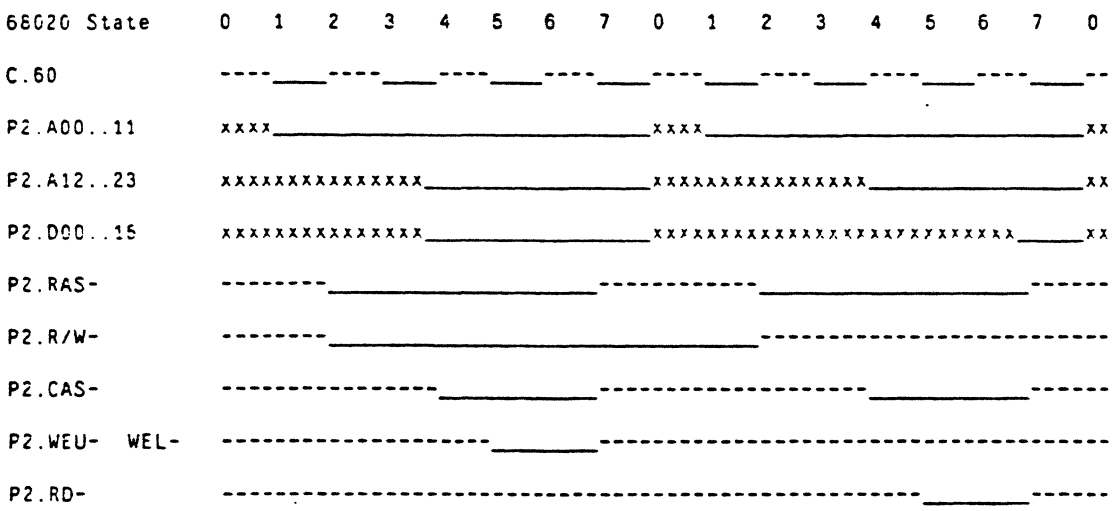
The upper and lower write enable to memory, [P2.WEU-, P2.WEL-] are generated in decoder [74F138:U400] in conjunction with gates [74F32:U433-1, U433-2]. The write strobes are asserted with [C.S5] and data strobe [Q.DS] active, with no error condition [Q.ERROR] present. They are turned off

with the processor upper and lower data strobe, [P.UDS, P.LDS].

Accesses to the P2-Bus are decoded in decoder [F138:U400]. A read or write reference to the P2-Bus [RD.P2, WR.P2] is generated when: the page type is 0 or 1 [TYPE1 = 0], data strobe is asserted [Q.DS = 1], no bus error condition exists [Q.ERROR = 0], clock state 5 is asserted [C.S5 = 1], and the reference is not to an I/O Device [CE.IO = 0].

3.12.3. P2-Bus Cycle

The timing of a P2-Bus cycle is illustrated in the figure below for a standard memory write cycle followed by a memory read cycle.



During read-modify-write cycles, processor address strobe and thus [Q.RAS] and [Q.CAS] stay asserted for the entire length of the cycle.

Note that both [P2.RAS] and [P2.CAS] are asserted before the page map type field is decoded and before the protection field is evaluated. Thus [P2.CAS] indicates a valid address, but not necessarily a valid reference. Only the read/write strobes qualify a reference.

The timing shown above applies to main memory read cycles and main memory write cycles that do not have the negative handshake [P2.WAIT] asserted. Accesses to video memory are similar to main memory read cycles, except that signal [P2.WAIT] is active. [P2.WAIT] is asserted by the video memory interface when it needs to delay the completion of the current cycle. On read cycles to video memory, [P2.WAIT] is asserted to delay the current cycle until valid read data is available. On write cycles to the video memory, [P2.WAIT] is asserted whenever the video memory is completing a buffered operation and is thus not yet ready to accept a new cycle. The same mechanism is used for write operations to main memory shadowed by video memory (video copy mode).

3.12.4. Parity Error Logic

Reference: Schematics Page 4

The Parity Error Logic generates parity for memory write operations and checks parity for memory read operations. Note that the parity error logic is only used for memory accesses (page type 0). It is not used for any other cycles, such as video memory cycles (page type 1).

On writes, parity is generated with parity generators [74F280:U420, U421]. When signal [EN.PARGEN] is asserted, *odd* parity is generated and stored in memory. Odd parity means that the sum of all data bits and the parity bit is odd.

On reads, parity is checked with parity checkers [74F280:U422, U423]. If the even output of the parity checkers is true, then a parity error has occured. This parity error information is clocked into the parity flipflops [74F74:U424-0, U424-1] on memory read cycles [TYPE0 = 0, RD.P2 = 1] with the leading edge of [C.S7], delayed by two inverter delays [74F04:U404-3, U404-4].

The parity error flipflops are self-latching. This means that they remain set until they are cleared by signal [Q.PARCLR]. The parity error flipflops remain cleared if parity checking is disabled [EN.PARERR = 0].

The outputs of the parity error flipflops [PARERRL, PARERRU] are ORed with the other bus error conditions in gate [74LS30:U130]. This generates signal [Q.ERROR] which in turn generates bus error [Q.BERR] to the 68010 via PAL [P16L8:U103].

Parity errors are different from other bus errors in that they cannot abort the 68010 cycle in which they occur. This is because they are only detected at the end of a read cycle, after a point at which the 68010 can abort the current cycle. The parity error flipflops provide the function of latching parity errors until they are recognized by the CPU.

In order to recognize the bus error caused by a pending parity error, the 68010 must execute a "non-special" cycle [Q.SPECIAL = 0]. Under this condition, PAL [P16L8:U103] generates signal [Q.BERRCLK] which clocks the parity error flipflop state into the bus error register [ALS534:U511] and signal [Q.PARCLR] which clears the parity error flipflops.

Parity generation and checking can be disabled for testing purposes. To initialize parity in main memory, all of memory needs to be written with parity generation enabled. When signal [EN.PARGEN] is not asserted, then *even* parity is generated. This allows the parity error function to be tested.

3.13. Ethernet Interface

Reference: Schematics Page 7, Intel 82586 Ethernet Controller Manual.

3.13.1. Overview

The Ethernet Interface is built around the Intel 82586 VLSI Ethernet Controller [U700] and the Fujitsu MB502 Phase Lock Loop Decoder [U701], as shown in Figure 3-2. The Ethernet Control Register [ALS273:U716, ALS244:U717] controls the overall operation of the Ethernet interface.

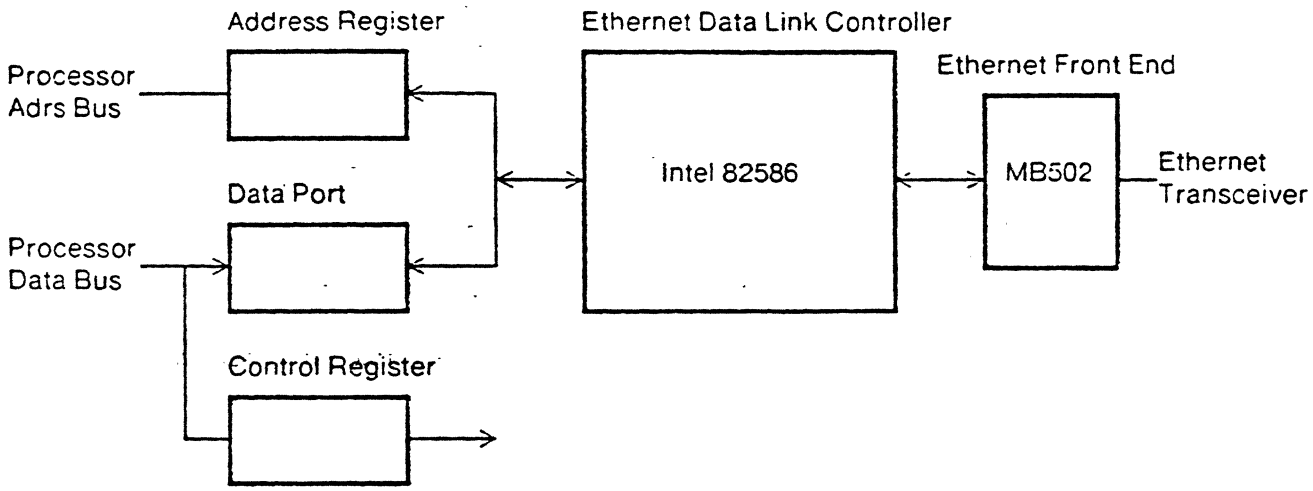


Figure 3-2: Ethernet Interface

3.13.2. Ethernet Data Link Controller

The Intel 82586 Ethernet Data Link Controller is configured as follows: Maximum Mode [MN/MX- = 0], asynchronous ready [READY = 0], directly enabled [HLDA = HOLD], and always clear to send [CTS- = 0]. The 82586 receives an 8 MHz clock from flipflop [74F74:U713-1]. Pullup [R9-SIP:S700] supports the VOH-level required by the 82586. For a complete description of this part, refer to the Intel 82586 Data Sheet.

3.13.3. Ethernet DVMA Cycle

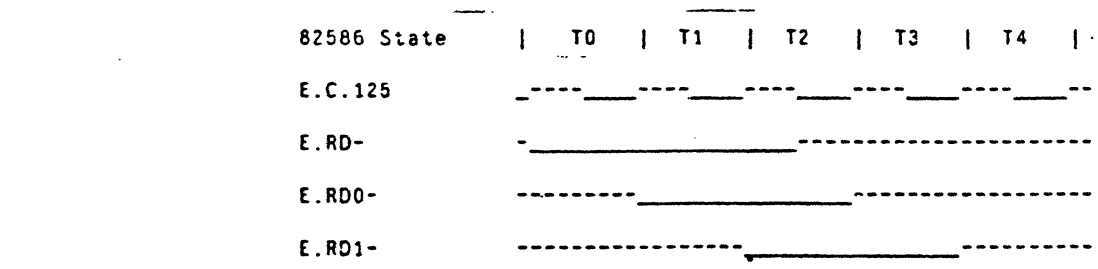
When the Ethernet controller wants to access main memory, it asse [E.RD] or write control [E.WR]. Ethernet read and write controls [74LS00:U715-3] to generate Ethernet data strobe [E.DS]. The leading [E.DS] then sets the Ethernet DVMA request flipflop [74F74:U207-1]. request [E.HOLD]. ANDed with Ethernet error inactive [E.ERR-] in gate [[E.REQ] to the DVMA Arbiter latch [74F374:U213]. This will cause the A the bus from the CPU until the 82586 drops [E.HOLD].

Ethernet Data Strobe is also clocked via [74F74:U713-0] at the ne Ethernet clock [E.C.125] to generate Ethernet address strobe [E.AS] address strobe latches the 24-bit Ethernet address into the Ethernet a U703, U704] to generate Processor Address [P.A01 THROUGH P.A23] whe enable [E.DMAEN]. In addition, Ethernet write control [E.WR] is latch generate Processor Read/Write strobe [P.R/W-] when enabled.

At this point, the Ethernet has requested a DVMA cycle and is waiting a write cycle (Ethernet to Memory), the DVMA controller will enable th [ALS244:U707, U708] with Ethernet output enable [E.OE]. On a read cyc DVMA controller will latch the data read from memory into the [ALS373:U705, U706] at the trailing edge of Ethernet write enable [E.WE] are output enabled by Ethernet read-1 [E.RD1] to the Ethernet controller in the diagram below.

The Ethernet read and write buffers are byte swapped between the Ethernet data bus. This means that the processor data bits 0..7 are con 8..15 and vice versa.

If a bus error is encountered during an Ethernet DVMA cycle, the Ethe [ALS74:U719-1] causing the Ethernet Error signal to be asserted [E.ERR]. Ethernet DVMA requests to be set in the Ethernet DVMA request flipflop [7 bus error flipflop can only be reset by an Ethernet reset command [E.RESE



3.13.4. Ethernet Phase Lock Loop Decoder - U701

The Fujitsu Ethernet Encoder/Decoder [MB502:U701] connects the board directly to an external Ethernet transceiver. The MB502 uses a digital phase lock loop with 10 samples per bit cell. An internal oscillator with external crystal X700 together with tank circuit [C:C700, C703, C704, L:L700] supplies the 100 MHz input frequency to the PLL chip. Jumper [J.2:J704] selects between Ethernet Level 1 and Level 2 interface characteristics (Level 2 if jumpered).

The Ethernet frontend is interfaced to the Ethernet data link controller with inverters [74F04:U709] and flipflops [74F74:U710, U712]. Pullup [R9.SIP:S700] raises the signal levels to those required by the EDLC.

3.13.5. Ethernet Transceiver Interface - J700

The Ethernet Connector [J700] follows the standard Ethernet definition. Jumper [J.2:J702] supplies +5V to the Ethernet connector for transceivers that require this voltage. The Ethernet transceiver drop cable is terminated with resistor networks [R4.SIP:R704, R705].

3.14. VME Bus Interface

Reference: Schematics Page 8, 9, 10, VME Bus Manual.

The VME Bus interface consists of the following functions:

- VME Bus Utility Functions
- VME Arbiter
- VME Master Interface
- VME Slave Interface
- VME Interrupt Handler

Figure 3-3 shows how these functions are interconnected.

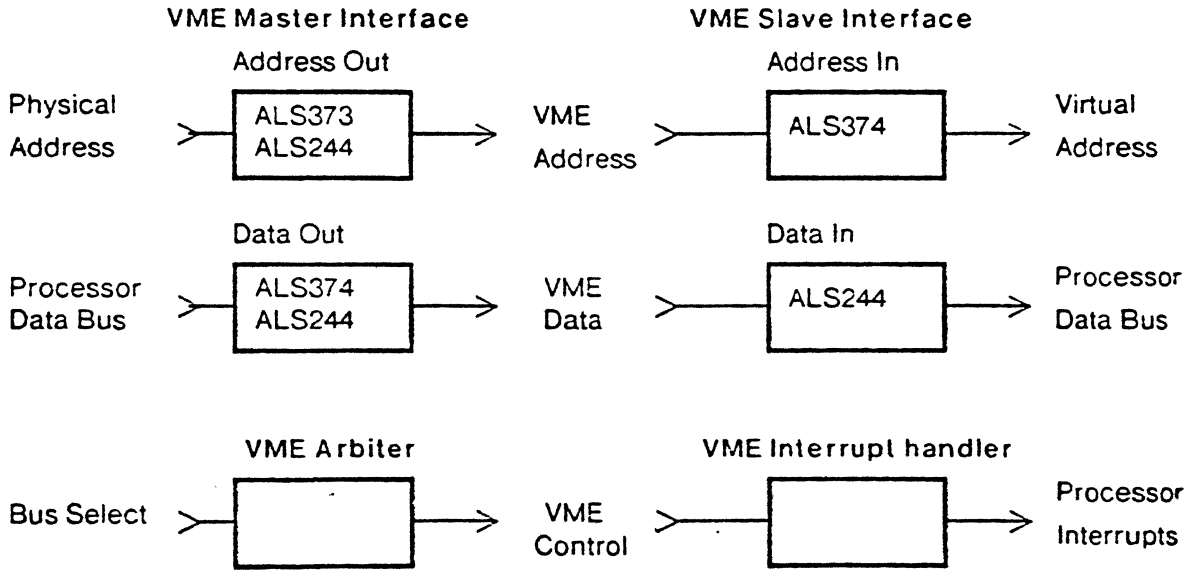


Figure 3-3: VME Interface

3.14.1. VME Bus Utility Functions

The VME Bus Utility functions are implemented by these four utility lines: System Clock [P1.SYSCLK], AC Fail [P1.ACFAIL], System Reset [P1.SYSR], and System Fail [P1.SYSF].

System Clock is driven from the 16 MHz oscillator signal [C.62A] via a high-current driver [74F244:U817]. System Clock has no phase relationship with any other VME signals. It can be disconnected from the VME Bus by removing jumper [J.16:J900-15.16].

AC Fail is driven to the VME Bus by open collector driver [74ALS6411:U818]. It is asserted while Power-On-Reset is active. It cannot be disconnected from the VME Bus.

System Reset is driven to the VME Bus by open collector driver [74ALS6411:U818] It is asserted whenever Processor-Reset is active. It cannot be disconnected from the VME Bus.

The 2050 Board can be configured either as a P1-Bus Reset Master or Slave.

As a P1-Bus Reset Master, the 2050 Board issues Reset [B.RESOUT] to the VME Bus. Power-On Reset, Watchdog Reset, and 68010 Reset will all assert P1-Bus Reset. Other P1-Bus devices may also assert P1-Bus Reset, but this will have no effect on the on-board CPU and devices.

As a P1-Bus Reset Slave, the 2050 Board receives Reset [B.RESIN] from the VME Bus, but does not drive Reset to the VME Bus. The VME Bus Reset has the same effect as an on-board power-on-reset.

System Fail is not used or generated by the 2050 Board.

3.14.2. VME Arbiter and Requestor

The VME Arbiter and Requestor functions are implemented in one state machine [74F374:U812, U813, P9X4:U811, P16L8:U814]. Out of the options possible within the VME Bus Spec, the arbiter implements the ONE ROR arbiter option. ONE means that the arbiter monitors bus request level 3 [P1.BR3] only and accomplishes arbitration via the level 3 daisy chain [P1.BG3IN, P1.BG3OUT]. ROR or *release on request* means that the arbiter only releases the bus when a request from another master is pending.

When the CPU wants to access the VME Bus, either for a standard read/write cycle or for a interrupt acknowledge cycle, it asserts signal Bus Select [B.BSEL] via PAL [P16L8:U810].

If the arbiter currently does not own VME Bus mastership, it requests bus mastership by asserting VME Bus request [P1.BREQ] and going through the normal VME Bus arbitration sequence. If the arbiter already owns bus mastership, it will keep the bus mastership until another VME Bus master requests it.

3.14.3. VME Master Interface

Once the 2050 Board obtains VME Bus mastership, the VME Master Interface allows the 2050 Board to access VME Slaves on the VME Bus. The interface consists of address and address modifier latches [ALS374:U940, U941, U942, U943] and drivers [ALS244-1:U900, U901, U902, U903]; write data latches [ALS374:U910, U911] and drivers [ALS2441:U908, U909]; read data buffers [ALS244:U908, U909]; and control signal driver [74F244:U817-0]. The VME Slave Device being addressed will respond to the transfer by asserting either data transfer acknowledge [P1.DTACK] or bus error [P1.BERR]. These two signals pass through flow-through latch [74F373:U815] and are qualified in PAL [P16L8:U816] before reaching the 68010 CPU.

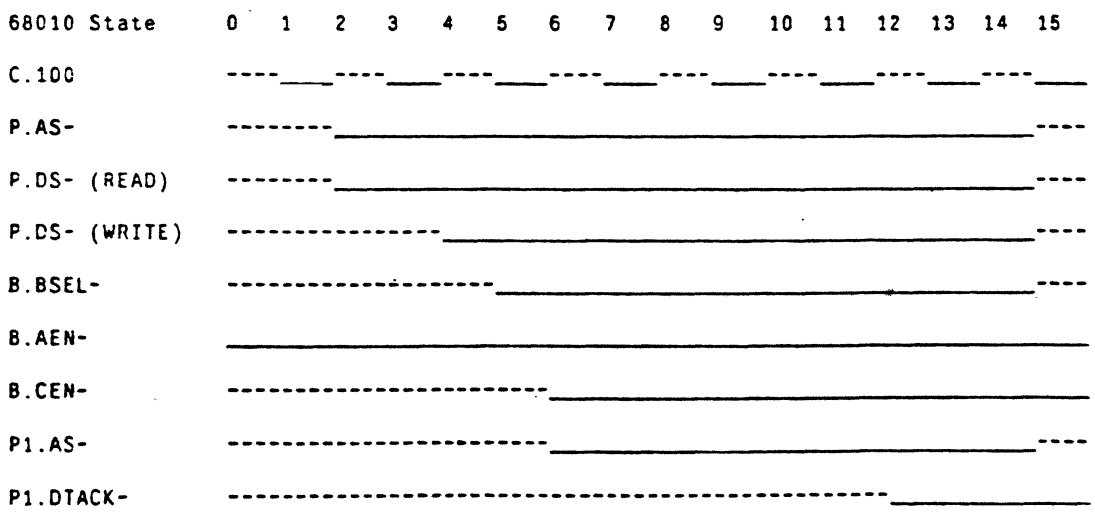
The VME Master interface supports complete backoff/rerun capability. This capability is utilized for VME accesses that take longer than 2 usec. In case of a VME access that is not completed within 2 usec, the state of the VME interface is frozen and a CPU rerun cycle is performed. During the rerun cycle, the CPU can give the on-board bus to the Ethernet interface or to the refresh logic to allow these devices to perform their functions. After these devices complete their activities, the rerun is terminated and the CPU continues with its VME access. This operation is transparent to the VME Bus. Notice that rerun cycles are also executed while the board is waiting for VME Bus mastership.

The VME rerun operation in detail proceeds as follows. Starting with [C.S4] of a cycle, counter [74LS393:U212] starts counting with the falling edges of [C.400]. After eight input transitions the counter asserts [B.C2]. This signal is inverted via [74F04:U221-1] and reclocked in register [74FC74 U206] thereby generating x(B.C3). Signal [B.C3] closes the [BERR, DTACK] flow-through latch [74F373:U815]. If a [BERR, DTACK] arrives before the latch closes the CPU will complete the cycle as normal and the rerun sequence is aborted at that point.

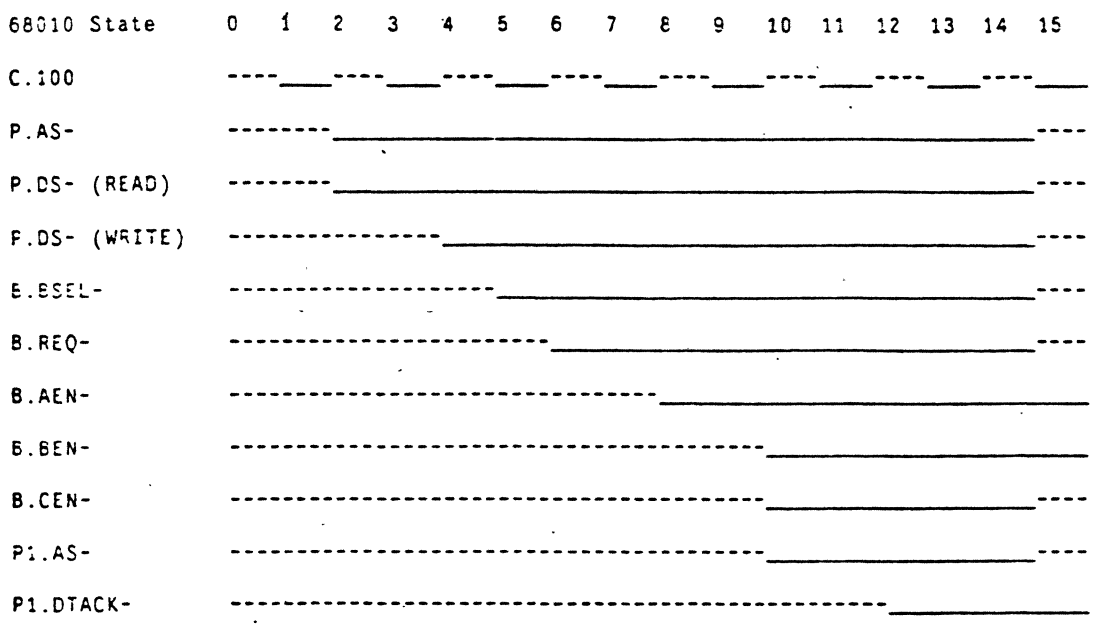
If the CPU has not received a [BERR, DTACK] at this point the rerun sequence continues. After two additional transitions of clock [C.400], counter [74LS393:U212] asserts both [B.C2] and [B.C1]. This event is decoded in PAL [P16L8:U814] asserting output [B.TO3]. [B.TO3] enters PAL [P16L8:U810] which in turn generates [B.FREEZE] and initiates the actual rerun cycle to the CPU with signal [B.RERUN]. [B.FREEZE] causes the VME control signals, write data, and addresses to be latched until the CPU resumes the cycle after completion of the rerun.

The timeout counter [74LS393:U809] counts the number of bus reruns. Each assertion of [B.C2] increments the counter by one. When the counter reaches 128 it asserts timeout.

3.14.4. 68010 Cycle to VME Bus, Currently Busmaster



3.14.5. 68010 Cycle to VME Bus, Not Currently Busmaster



3.14.6. VME Slave Interface

The VME Slave Interface allows the 2050 Board to be accessed by other VME Masters on the VME Bus. It uses the address comparator [LS2521:U930] to match the four high-order address lines from the bus [P1.A20..A23] against the base address bits [X.A0..A3] selected by switches [J.16:J900]. In addition, the VME address modifiers 4 and 5 must be set, [P1.AM4 = 1, P1.AM5 = 1], the VME interrupt acknowledge must be not set [P1.IACK = 0], and the 2050 Board must not be bus master [B.AEN = 0]. If all these conditions are met and VME address strobe [X.AS] and a VME data strobe [X.UDS OR X.LDS] is asserted then signal [X.DMA] is asserted, indicating that a VME Slave Interface request is pending.

The rising edge of [X.DMA] sets the external DMA request flipflop [74F74:U203-1] posing an external DMA request [X.DMAREQ] to the DVMA controller. In response, the DVMA controller requests the on-board bus from the CPU and executes an on-board cycle using the external DVMA address stored in register [ALS374:U904, U905, U906]. On a write cycle, the DVMA controller uses the data buffer [ALS244:U908, U909] to enable data from the VME bus. On a read cycle, the data read from memory is stored in register [ALS374:U910, U911] before it is driven to the VME bus with data buffer [ALS2441:U912, U913].

At the end of the VME DVMA cycle the handshaking flipflops [74F74:U931] are set with the trailing edge of [X.DMAEN]. If a bus error is present at this time, the bus error flipflop [74F74:U931-0] is set, asserting [X.BERR]. Otherwise, the DTACK flipflop [74F74:U931-1] is set, asserting [X.DTACK]. Both [X.DTACK] and [X.BERR] are driven to the VME bus with open-collector driver [ALS6411:U818]. Signal [X.DMA] stays asserted until the VME Master drops its data strobes. This in turn clears the Bus Error and DTACK flipflops [74F74:U931].

3.14.7. VME Interrupt Handler

The VME Interrupt Handler responds to Interrupts on the VME Bus. The 2050 Board does not generate any interrupts to the VME Bus. Jumper J800 can individually connect and disconnect all VME Interrupt levels. Priority decoder [74LS148:U800] prioritizes the enabled VME interrupt requests and generates encoded interrupt lines [B.IPL0..2]. These encoded interrupt lines together with the onboard interrupt requests are combined in PROM [27S33A:U105] which in turn drives the 68010 interrupt lines [IPL0, IPL1, IPL2].

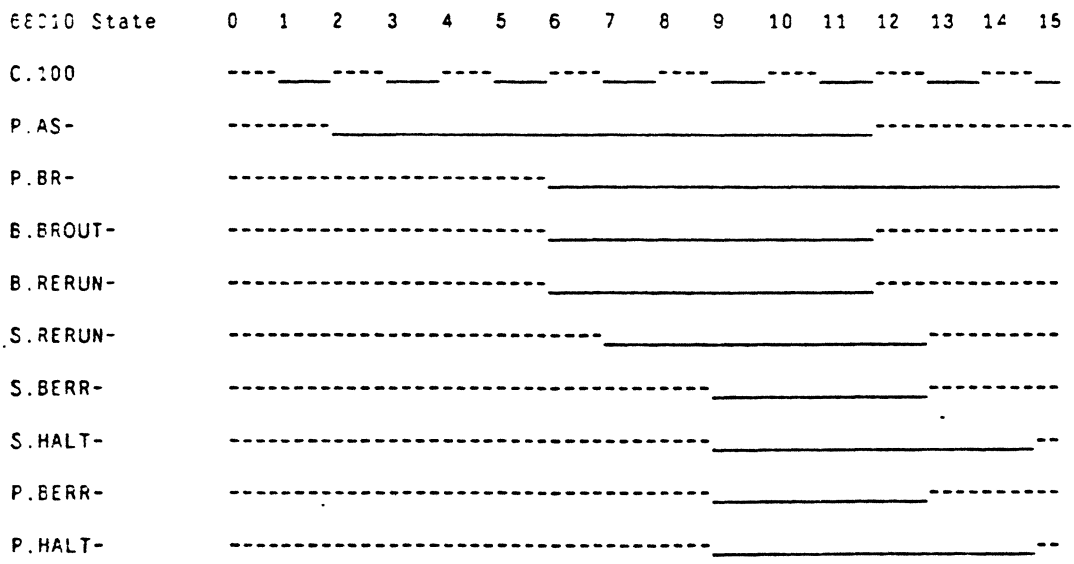
When the 68010 recognizes an interrupt request, it issues function code 7 and sends out the interrupt level being acknowledged on address lines [A01..A03]. The on-board/offboard interrupt selector [74F151:U802] decodes whether an external interrupt request is pending at the level the 68010 acknowledges. The output of this selector [B.IRQ] is sampled at state 4 in the onboard/offboard interrupt flipflop [74F74:U205-0]. Output [Q.AUTOV] is asserted if no offboard interrupt request was pending or if address bit [A19] is deasserted, indicating a non-interrupt cycle. Output [Q.AUTOV] is deasserted if an onboard interrupt request is pending at the level being acknowledged and if address bit A19 is asserted, indicating a valid interrupt cycle.

3.14.8. 68010 Rerun Cycles

Rerun cycles are executed on two conditions: VME Bus deadlock and long VME accesses. These two rerun conditions are recognized in PAL [P16L8:U810] which generates [B.RERUN]. Signal [B.RERUN] is synchronized in flipflop [74F374:U206] before driving PAL [P16R4:U102] which in turn generates the required [Q.BERR, P.HALT] signals to cause a CPU rerun.

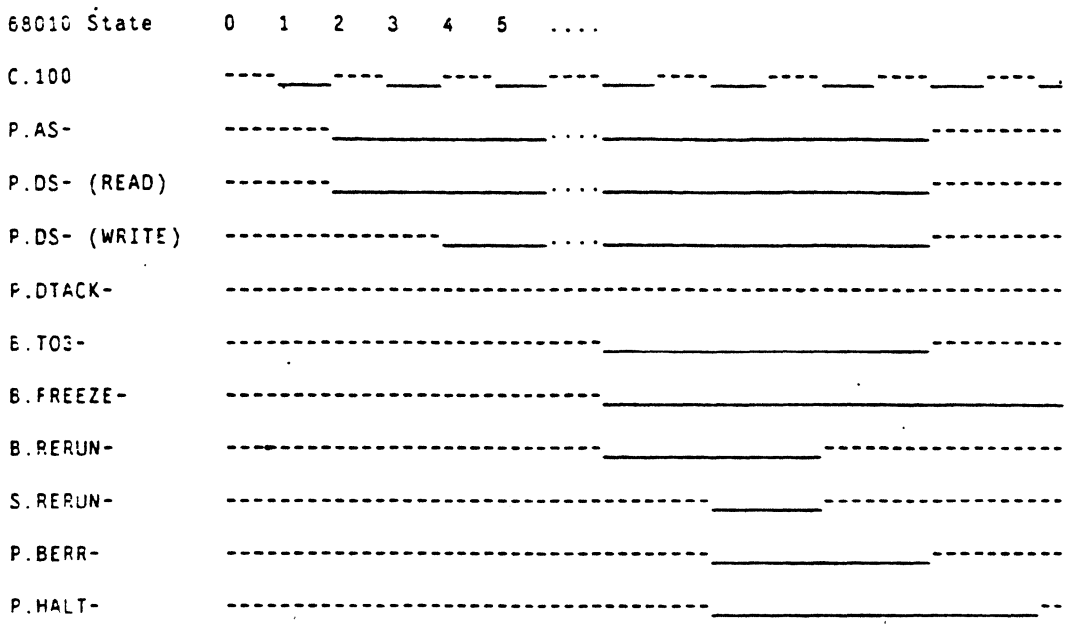
3.14.9. Rerun, VME Deadlock Case

The condition here is that the CPU is attempting to access the VME Bus while another master on the VME Bus is attempting to access the 2050 Board as a slave device. Since the VME Bus has no rerun capability, the 68010 must yield to the VME Bus request to resolve the deadlock. The condition is present if [B.SEL] and [X.DMA] are simultaneously valid.



3.14.10. Rerun, VME Rerun Case

The VME Rerun case is initiated for VME cycles that are not completed within the short timeout time, including VME accesses waiting for VME Bus mastership.


68010 State 0 1 2 3 4 5 ....
C.100
P.AS-
P.DS- (READ)
P.DS- (WRITE)
P.DTACK-
B.TOS-
B.FREEZE-
B.RERUN-
S.RERUN-
P.BERR-
P.HALT-

3.15. Memory

Reference: Base Board Schematics Page 11, 12, 13, 14, 15

Reference: Expansion Board Schematics Page 21, 22, 23, 24, 25

3.15.1. Introduction

The description of the memory applies in the same way to the memory contained on the 2050 Base Board as to the memory on the 2051 Expansion Board.

The memory design consists of the following functions:

- memory array (1/2/3/4M Bytes)
- address multiplexor and driver
- control signal driver
- bank decoder and driver
- data drivers

The interconnection of these pieces is shown in the Figure 3-4.

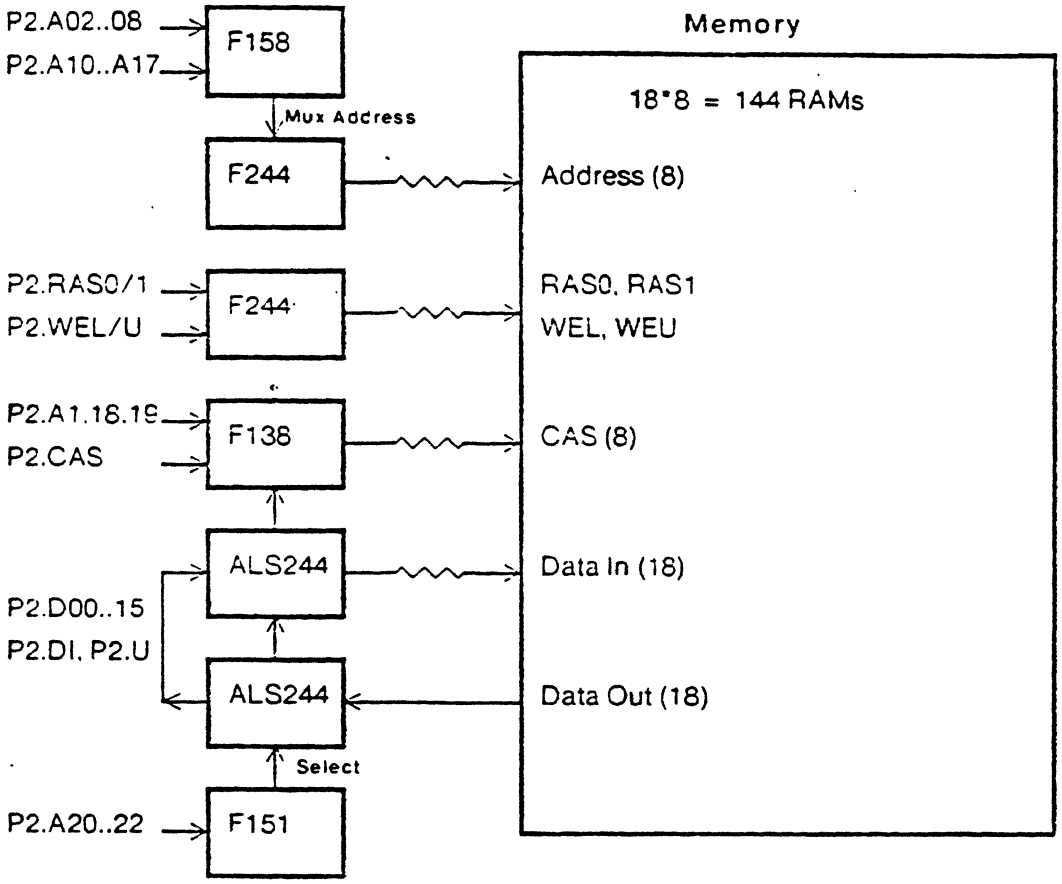


Figure 3-4: Memory Interface

3.15.2. Memory Interface

The CPU interfaces to the memory via the P2.Bus. This means that all interface signals are available on the P2-connector [P96:P1102], allowing a memory expansion board to be interfaced to the same bus. The following description applies equally to the memory on the CPU Board as well as to the expansion memory.

3.15.3. Memory Organization

Memory is organized as 8 banks of 18 RAM chips each, making a total of 144 chips. Each bank stores 16 data bits and 2 parity bits.

RAM chips can be either 64K or 256K Bits. With 64K RAMs, each bank stores 128K Bytes plus parity, and all of memory stores one Megabyte.

With 256K RAMs, each bank stores 512K Bytes plus parity, making total memory capacity four Megabytes.

3.15.4. Memory RAM and Bank Decoding

Due to the pipelined RAS-CAS access, memory is CAS decoded because the translated address bits that select which bank of memory is accessed are only available in time for the CAS address strobe. For special cycles [Q.SPECIAL = 1], such as MMU updates, CAS is not asserted.

Decoding for 64K and 256K RAM chips is as follows:

Decoding	64K RAMs	256K RAMs
RAS Bank	A01	A01
RAS Address	A02..A09	A02..A10
CAS Address	A10..A17	A11..A19
CAS Bank	A01,18,19	A01,A20,A21

3.15.5. Memory Section Decoding

To allow the memory to respond to arbitrary 1 megabyte sections within the 8 megabyte memory address space, memory select decoder [74F151:U1200] decodes the three high-order address bits [P2.A20..A22] and reads from the select jumper [J16:J1200] whether the addressed 1 megabyte section of memory is enabled or not. If enabled, the Memory Select signal [M.SEL] enables CAS decoder [74F138:U1201] and read/write buffers [ALS244:U1210..U1214] via decoder [ALS138:U1202].

The first megabyte of memory is always enabled. The second megabyte of memory is enabled with jumper [J1201:1-2] installed. The third and fourth megabyte are enabled as a pair if jumper [J1201:3-4] is installed.

3.15.6. Memory Drivers

The RAM signals are driven as follows:

[RAS, WEL, WEU], and the Address Lines are driven by 74F244 drivers with 33 Ohm series termination. Each bank of memory has its own set of drivers for these signals.

CAS is driven directly by the CAS decoder [74F138:U1201] with 33 Ohm series termination. Data to the RAMs is driven by [ALS244] drivers with 68 Ohm series resistors [R:R1200-R1217].

3.16. Video

Reference: Schematics Page 16, 17, 18, 19

3.16.1. Overview

The video subsystem consists of the following functions:

- video memory (128K Byte)
- video memory controller
- data multiplexor
- P2-Bus interface
- video sync controller
- video shifter

The interconnection of these pieces is shown in the Figure 3-5.

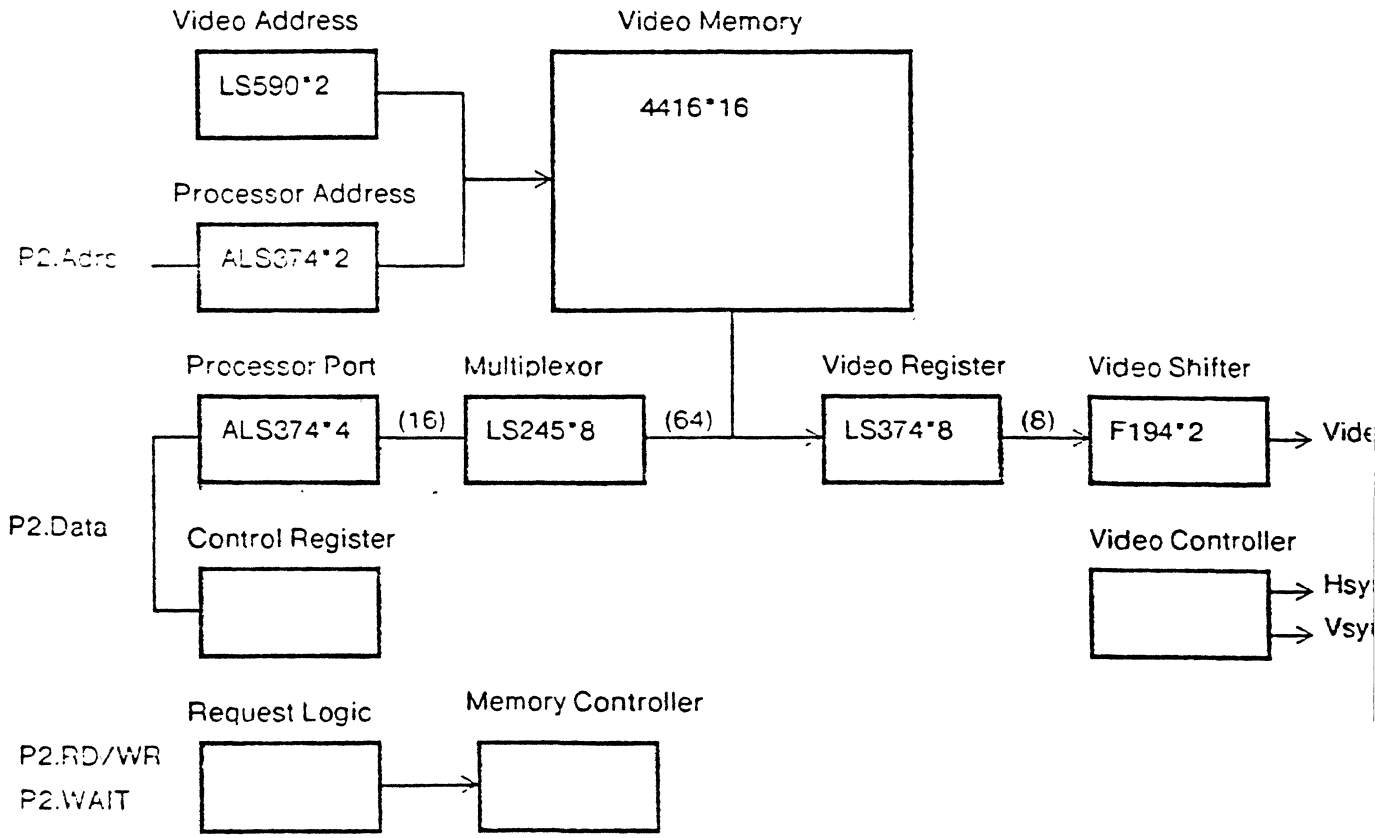


Figure 3-5: Memory Interface

3.16.2. Video Memory and Addressing

The 128 kilobyte video memory on the board, chips [4416:U1700-U1707, U1710-U1717], is dual ported for processor access and video refresh. The memory is organized as 16K Words of 64 bits each. Processor update cycles read 16 bits at a time or write 8 or 16 bits at a time. Video refresh cycles read 64-bits at a time.

The address for processor cycles is stored in register [ALS574:U1632] for the row-address and in register [ALS574:U1633] for the column address. Register [ALS373:U1634] demultiplexes the multiplexed memory addresses.

The address for video cycles comes from counters [74LS590:U1630, U1631] for row and column address, respectively. Notice that the organization of the 4416 RAM chips requires an 8-bit row address and a 6-bit column address. Address lines [V.A0] and [V.A7] are not used for column addressing.

The video refresh counters are incremented every 640 nsec with the rising edge of [V.OE1-] except during states without display enable. They are reset to 0 with signal [V.RESET-].

3.16.3. Video Memory Controller

The video memory controller state machine generates the timing for the video memory and other basic timing strobes for the video subsystem. It consists of PROMs [P5X8:U1604, U1605] and latches [74F374:U1606, U1607]. 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 three basic types of cycles: Idle cycles, Processor update cycles, and Video refresh cycles. The memory controller executes an idle cycle or a processor update cycle between states 0 through 7 and a video refresh cycle between states 8 through 15.

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

Processor Update Cycles: Processor Update Cycles are executed between states 0 and 7 if synchronous request is asserted [V.SREQ = 1] and if the register select 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.

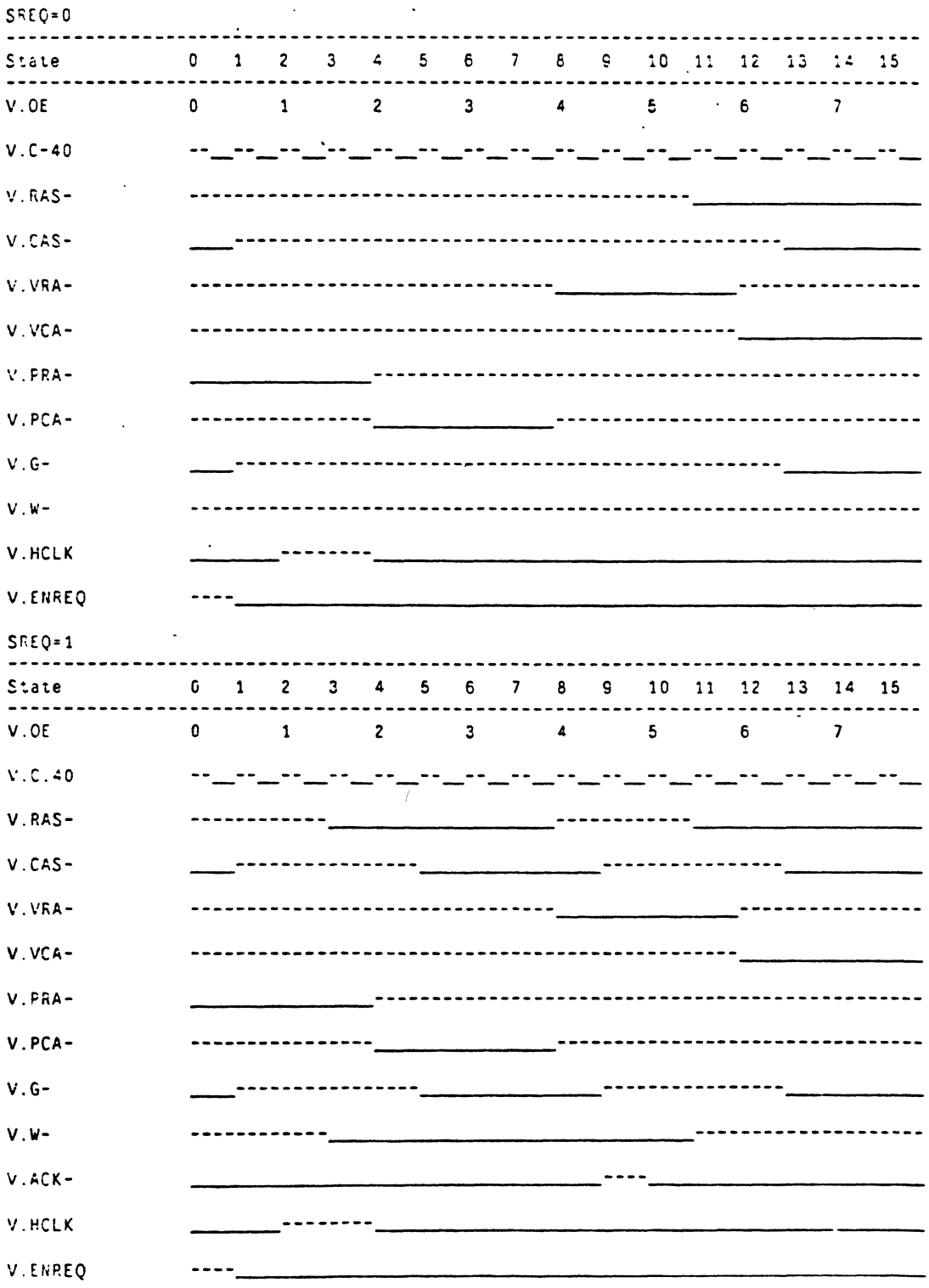
Read Cycle: A read cycle is executed if no external write strobes [V.LDS, V.UDS] are pending in the request latch [ALS374:U1615]. The memory word addressed by bank selects [V.BS1] and [V.BS2] is enabled via the RAS decoder PAL [P16R4:U1616]. The read data passes from the video RAM chips onto the internal data bus [V.B00..15] through buffers [74LS245:U1730..U1737] and is latched in the data output register [ALS374:U1602, U1603] at the rising edge of signal [V.ACK].

Write Cycle: A write cycle is executed if a external write strobes [V.LDS, V.UDS] is pending in the request latch [ALS374: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.

Video Refresh Cycle Video refresh cycles are 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 contained in registers [74F374:U1640] and [74F374.U1641], respectively. These registers are loaded from counters [74LS590: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 video output bus [V.00-V.07] via control lines [V.0E0-..V.0E7-]. Starting with [V.0E0-] 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].

A processor cycle is executed if the synchronous request [V.SREQ] is active (the generation of [V.SREQ] is described below under request logic). During a processor cycle, signals [V.PRA] and [V.PCA] enable the processor row and column address from the processor address latches [ALS374.U1632] and [ALS374:U1633], respectively, in time for [V.RAS] and [V.CAS], the row and column address strobe.

3.16.4. Video State Machine



3.16.5. Video Interface to P2-Bus

3.16.6. P2-Bus Address Decoding

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

For direct reads and direct writes, the video logic is selected if the four most significant P2 address bits [P2.A20..A23] are all ones. In that case, decoder [ALS138:U1621] produces signal [V.BSEL-], which generates a video request via PAL [P16L8:U1620].

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:U1623] generates [V.CSEL-] which generates a video request via PAL [P16L8:U1620].

PAL [P16L8:U1620] also decodes [P2.A17] in direct mode [V.BSEL = 1] to generate the read/write strobes for the video control register [ALS273:U1610, U1611].

3.16.7. P2-Bus Request Generation

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 1-deep FIFO. This means that on a write cycle the processor does not need 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.

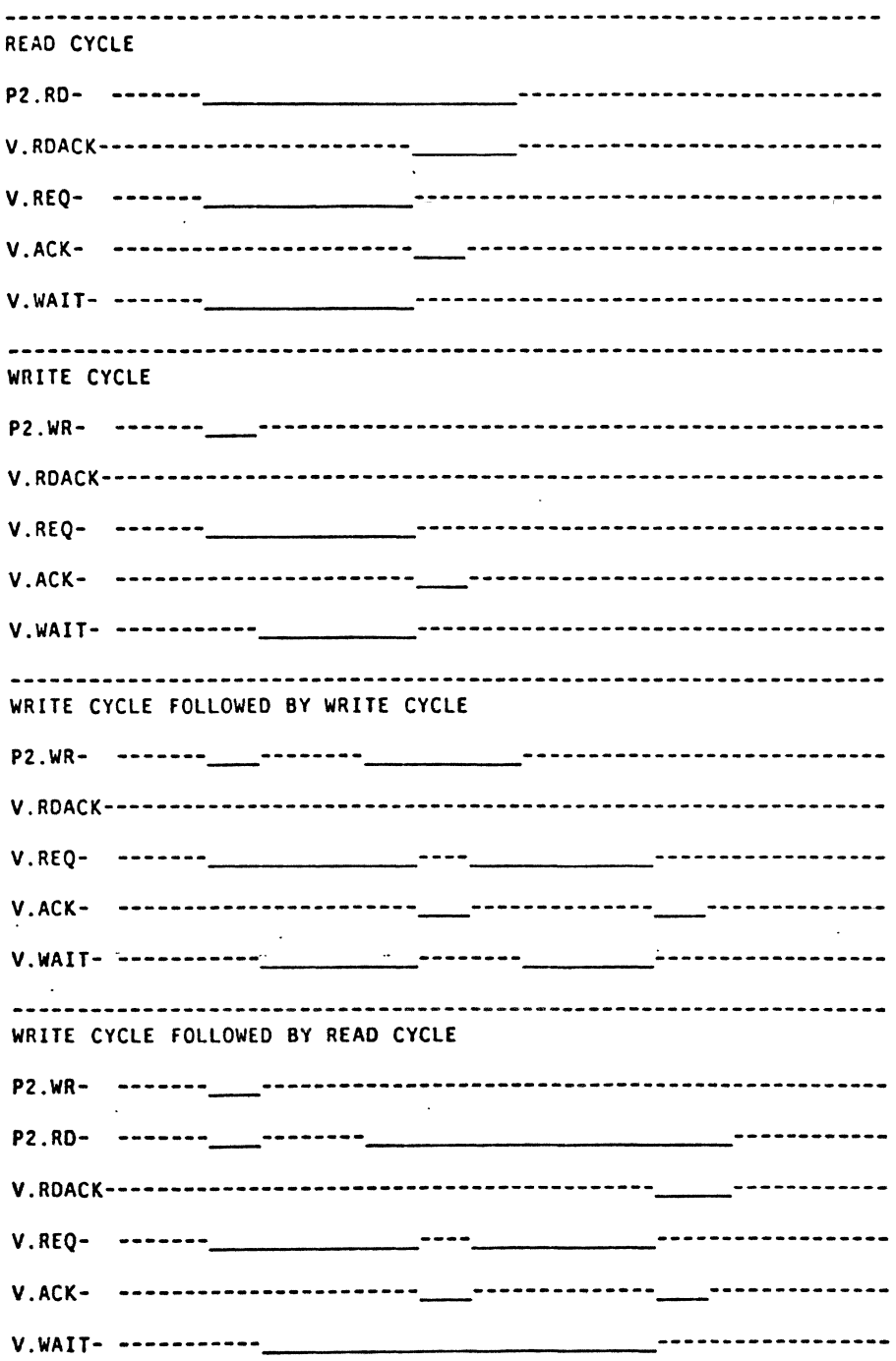
An interesting case occurs if a write cycle is immediately followed by a read cycle. In this case, the write cycle is still in progress while the new read cycle is pending. The design of the request logic assures that the read cycle is only begun after the write cycle has been completed.

This read/write cycle handshaking is implemented in PAL [P16L8:U1620]. A request is set when the video section is addressed in with a read or write cycle in direct mode [V.BSEL] or with a write cycle in copy mode [V.CSEL]. Signal [BUSY], causing [P2.WAIT], is set while a request is in progress.

The leading edge of the request signal [V.REQ] clocks the demultiplexed processor address into the processor address register [ALS534:U1632, U1633]. It also clocks the low-order address bits [P2.A01, P2.A02] and the write enable bits [P2.WEL, P2.WEU] into register [ALS374:U1615].

[V.REQ] is sampled with signal [V.ENREQ] into flipflop [74F74:U1624-0]. The sampled signal is reclocked on the next clock edge of [V.C40] into flipflop [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].

3.16.8. P2-Bus Interface Timing



3.16.9. Video Controller

The video controller generates the timing for the video monitor. The following description applies to the "standard Sun-2 video monitor". This video monitor has the following attributes:

```
Visible Display     1152 pixels by 900 lines
Video Clock:        10 nsec          100 MHz
Horizontal Cycle:   16.00 usec       62.5 kHz
Vertical Cycle:     15000 usec       66.66 Hz
Horizontal Retrace: 4.48 usec
Vertical Retrace:   600 usec
```

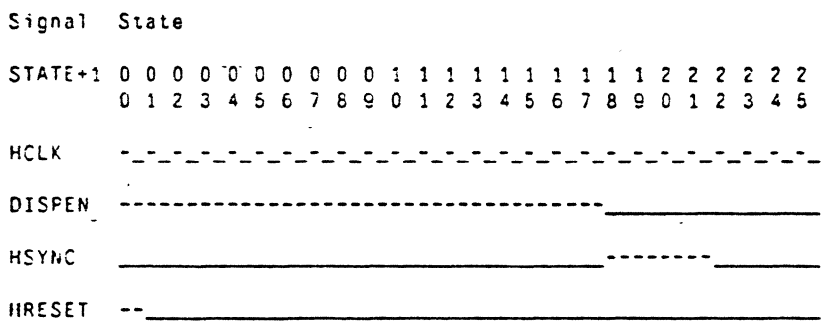
Video controller latch [74F374:U1812] latches the outputs of horizontal and vertical decoding PROM on the rising edge of [V.HCLK].

3.16.10. 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.HCLR] generated by video controller latch.

Horizontal decode PROM [P9X4:U1811] decodes horizontal counter inputs [V.HS0] through [V.HS6], plus vertical blank [VBLANK] from the vertical state machine. Horizontal decode PROM outputs are [V.HCLR, V.HSYNC, AND V.DISPEN].

3.16.11. Horizontal State Machine Timing Diagram

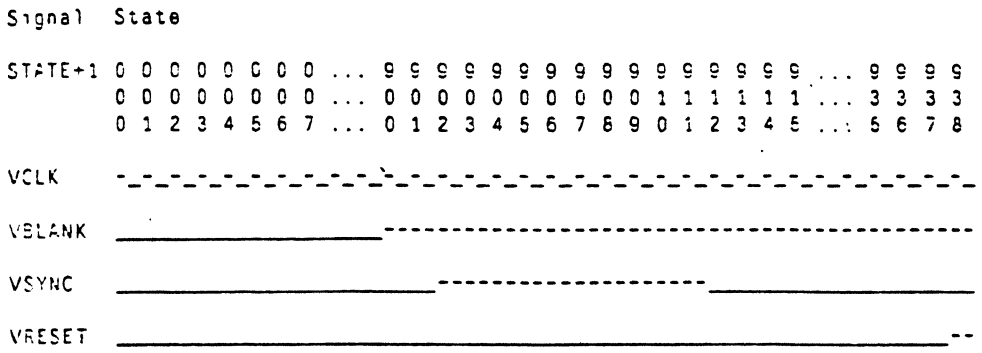


3.16.12. Vertical State Machine

Vertical counter [74LS393:U1813, U1814] is advanced on falling edge of horizontal sync [V.HSYNC]. Vertical counter is reset with [V.VCLR] 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.CLR, V.VBLANK, AND V.RESET]. The vertical decode PROM function is defined in PROM A1815.

3.16.13. Vertical State Machine Timing Diagram



3.16.14. Video Interrupt Logic

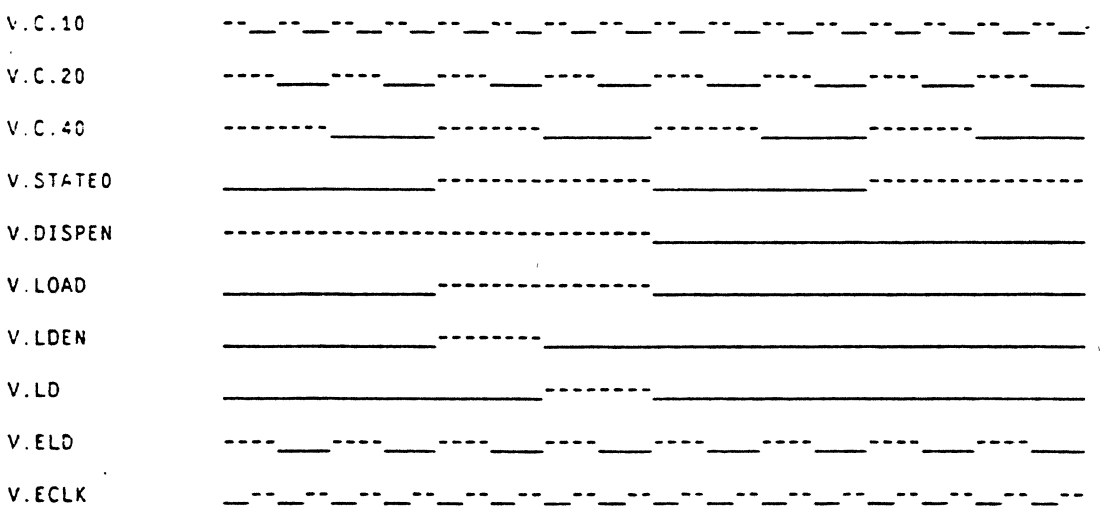
Interrupt flipflop (74F74:U1803-1) is set at the leading edge of [V.VBLANK] as long as interrupt enable [V.INTEN] is enabled.

3.16.15. Video Clock and Shifter

The 100 MHz video clock [V.C.10] that is generated by crystal oscillator [K1114A:U1800] is buffered by gate [74F08:U1808-0] and is then divided into a 50 MHz clock by flipflop [74F112:U1801-0].

The video data [V.00..7] is loaded into two 50 MHz shift register. [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 10 nanosecond clock [V.C.10] and 20 nanosecond clock [V.C.20] is converted from TTL to ECL levels by converter [10H124:U1807] and drives 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 [R1800. R1801] to [-5V] and are intended to drive differential ECL terminated with an impedence of 100 Ohm.

The timing is illustrated in the figure below.



```
paltype pal1618
palname A101
palid 1.24 84/07/31

PALBEGIN

% Inputs

1  INPUT P.FC0
2  INPUT P.FC1
3  INPUT P.FC2
4  INPUT BOOT-
5  INPUT Q.AUTOV-
6  INPUT RD.PROM-
7  INPUT Q.AS
8  INPUT C.S7
9  INPUT PIN9
11 INPUT B.DTACK

10 GND
20 VCC

% Outputs

19 OUTPUT Q.MMU-
18 OUTPUT C.S7AS
17 OUTPUT OE.PROM-
16 OUTPUT Q.SPECIAL-
15 OUTPUT Q.INTVEC-
14 OUTPUT Q.VPA-
13 OUTPUT FC7-
12 OUTPUT SPWAIT-

EQUATIONS

ASSERT Q.MMU-                          % Function code 3. MMU access.
ENABLE  ALWAYS
OR      / P.FC2 & P.FC1 & P.FC0 & Q.AS

ASSERT C.S7AS                          % Demorganize: C.S7 & AS
ENABLE  ALWAYS
OR      / C.S7
OR      / Q.AS

ASSERT OE.PROM-                        % Boot and FC=6; Fetch inst from EPROM.
ENABLE  ALWAYS
OR      P.FC2 & P.FC1 & / P.FC0 & BOOT & Q.AS
OR      RD.PROM

ASSERT Q.SPECIAL-                      % Special Cycle. Inhibit CAS.
ENABLE  ALWAYS
OR      P.FC1 & P.FC0                  % FC=7 or FC=3. INTA or MMU cycle.
OR      P.FC2 & P.FC1 & / P.FC0 & BOOT % BOOT cycle.

ASSERT Q.INTVEC-                       % VME interrupt acknowledge cycle.
```

```
ENABLE  ALWAYS
OR      P.FC2 & P.FC1 & P.FC0 & C.S7 & / Q.AUTOV & Q.AS

ASSERT Q.VPA-                           % Local interrupt acknowledge cycle.
ENABLE  ALWAYS
OR      P.FC2 & P.FC1 & P.FC0 & C.S7 & Q.AUTOV & Q.AS

ASSERT SPWAIT-                          % DTACK for SPECIAL cycles.
ENABLE ALWAYS
OR      P.FC2 & P.FC1 & / P.FC0 & BOOT & / C.S7         % Boot cycles
OR      P.FC2 & P.FC1 & P.FC0 & / Q.AUTOV & / B.DTACK   % VME Vector
OR      P.FC2 & P.FC1 & P.FC0 &   Q.AUTOV               % VPA cycles. No DTACK.

ASSERT FC7-
ENABLE ALWAYS
OR      P.FC2 & P.FC1 & P.FC0


TIMING: NO-CLOCK
%               1234567890123456789012345678901234567890123456789012345678901234567890
Q.MMU-          ------- ---------------- ---------------- ----------
Q.AS            _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
P.FC0           __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __
P.FC1           ____ ____ ____ ____ ____ ____ ____ ____ ____ ____ ____ ____ ____ ____ ____
P.FC2           ________ ________ ________ ________ ________ ________ ________ ________ ____
C.S7            ________________ ------------------------------------------------------ ____
C.S7AS          ________________ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ ____
BOOT-           ________________ ---------------- ---------------- ----------
RD.PROM-        ---------------------------------- ----------------------------------------
OE.PROM-        ---------------- ---------------- ----------------------------------------
Q.SPECIAL-      ------ ____ ____ ------ ------ ------ ------ ____ ____ ----------
Q.AUTOV-        ---------------------------------- ----------------------------------------
Q.INTVEC-       ---------------------------------- ----------------------------------------
Q.VPA-          ---------------------------------------------------------------------------
SPWAIT-         ---------------- ---------------- ---------------- ----------
B.DTACK         _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
FC7-            ---------------- ---------------- ---------------- ----------
PIN9            xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

PALEND
```

```
paltype pal16r4
palname A102
palid 1.16 84/07/30

PALBEGIN

% Inputs

2  INPUT Q.SPECIAL-
3  INPUT CE.IO-
4  INPUT Q.R/W-
5  INPUT S.DMA-
6  INPUT POR1-
7  INPUT TC_20480-
8  INPUT S.RERUN-
9  INPUT TYPE1

1  CLOCK C-80-
10 GND
11 OUTPUT-ENABLE OE-
20 VCC

% Outputs

19 OUTPUT RD.IO-
18 OUTPUT WR.IO-
17 OUTPUT S.BERR-
16 OUTPUT S.HALT-
15 OUTPUT X.HALT-
14 OUTPUT INIT-
13 OUTPUT P.HALT-
12 OUTPUT P.RESET-

EQUATIONS

: WRITE Q.R/W ;                          % Define WRITE
: READ Q.R/W- ;                          % Define READ

ASSERT RD.IO-                            % Enable data buffers from IO.Dx->P.Dx
ENABLE  ALWAYS
OR      / S.DMA & Q.SPECIAL & READ       % MMU read cycle
OR      / S.DMA & CE.IO & READ           % I/O read cycle
OR      / S.DMA & TYPE1 & READ           % VME read cycle
OR      S.DMA & WRITE                    % DVMA write cycle

ASSERT WR.IO-                            % Enable data buffers from P.Dx->IO.Dx
ENABLE  ALWAYS
OR      / S.DMA & WRITE                  % MMU + I/O + VME write cycle
OR      S.DMA & / CE.IO & / TYPE1 & READ % DVMA read cycle from P2-bus

ASSERT S.BERR-                           % Clocked output. Syncronized bus error
OR      S.RERUN

ASSERT S.HALT-                           % Clocked output. Syncronized halt.
OR      S.RERUN
```

```
paltype pal1618
palname A103
palid 1.15 84/07/31

PALBEGIN

% Inputs

1  INPUT Q.ERROR
2  INPUT Q.SPECIAL-
3  INPUT C.S5
4  INPUT Q.S7
5  INPUT Q.AS
6  INPUT WR.PMAPOL-
7  INPUT Q.R/W-
8  INPUT MOD
9  INPUT S.DMA-
11 INPUT S.BERR-
18 INPUT TIMEOUT-

10 GND
20 VCC

% Outputs

19 OUTPUT Q.BERRCLK
17 OUTPUT Q.BERR-
16 OUTPUT Q.PARCLR-
15 OUTPUT WR.STAT-
14 OUTPUT WR.UPDATE-
13 OUTPUT WR.PMAPOX-
12 OUTPUT MOD1

EQUATIONS

: WRITE Q.R/W ;                                  % Confusing once '-' gone

ASSERT Q.BERRCLK                          % Demorganize:
ENABLE  ALWAYS                            % Q.BERR & /S.DMA & /S.BERR & Q.AS
OR      / Q.BERR
OR      S.DMA
OR      S.BERR
OR      / Q.AS

ASSERT Q.BERR-                                   % Processor bus error
ENABLE  ALWAYS
OR      Q.ERROR & C.S5 & / Q.S7 & Q.AS & / Q.SPECIAL    % Set only btwn CS5-QS7
OR      Q.BERR & Q.AS                            % Hold while Q.AS
OR      Q.BERR & / S.DMA & / Q.PARCLR            % Clear parerr at end of cycle
OR      S.BERR & C.S5 & Q.AS                     % Rerun cycle
OR      TIMEOUT & C.S5 & / Q.S7 & Q.AS           % Handle timeout on Q.SPECIAL

ASSERT Q.PARCLR-                                 % Pulse at end of Q.BERR
ENABLE  ALWAYS
OR      Q.BERR & / S.DMA & / Q.AS
```

```
ASSERT WR.UPDATE-                                   % Wr strobe for pmap nibble
ENABLE  ALWAYS                                      % ...containing acc/mod bits
OR      / Q.ERROR & C.S5 & / Q.S7 & / Q.SPECIAL     % Update for normal cycles
OR      WR.PMAPOL & Q.SPECIAL & / Q.S7              % Written by CPU

ASSERT WR.STAT-                                     % OE stat bits to pmap entry
ENABLE  ALWAYS
OR      / Q.ERROR & C.S5 & / Q.S7 & / Q.SPECIAL         % Turn-on quickly
OR      WR.UPDATE & / Q.SPECIAL                         % Turn-off slowly

ASSERT WR.PMAPOX-                                   % OE IO bus to pmap entry
ENABLE  ALWAYS
OR      WR.PMAPOL & Q.SPECIAL                           % Turn-on quickly
OR      WR.UPDATE & Q.SPECIAL                           % Turn-off slowly

ASSERT MOD1                                         % Pmap page modified bit
ENABLE  ALWAYS                                      % De-morganize: MOD or WRITE
OR      / MOD & / WRITE

TIMING: NO-CLOCK
%               12345678901234567890123456789012345678901234567890
Q.AS            -_--_____--_---_-----_----_-_____________----------
C.S5            __--__--__--__--__--__--__--__--__--__--__--__--__
Q.S7            ____----___---____----____----____----____----____
Q.ERROR         --------_______--------_______--------_______--------
Q.SPECIAL-      ----------------_______________----------------_____
S.BERR-         ---------___-------___-----------------------------
S.DMA-          ----_-------------------------_____________________
TIMEOUT-        ---------------_----------------------___-----------

Q.BERRCLK       __--_______-__--____-_____________________________
Q.BERR-         --__-------__--__--____-------------------_______
Q.PARCLR-       ------------------------------------------------
WR.PMAPOL-      --__---------_--------____----------_----------_
WR.UPDATE-      ----------_---_--------__--------------_-----_
WR.PMAPOX-      ----------------------____--------------------_
WR.STAT-        ----------__---_------------------------_------

MOD             ----___----___----___----___----___----___----___--
Q.R/W-          --------_______--------_______--------_______--------
MOD1            ----____------------____------------____----------

PALEND
```

```
static char* sccs_id = "1.3 84/07/20";

/* This information proprietary to Sun Microsystems Inc */

/* ============================================================
 * Author: Model-50 Group
 * Date :  March 9, 1984
 * Prom Name: A105
 * Prom Type: 1024 x 4.
 * Speed: 35 nsec.
 * Purpose: Interrupt level priority encoding. VME has precedence.
 *
 * ============================================================
 */

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

/* Define inputs to 1K x 4 prom. */

#define cpu_int1  (a0)
#define cpu_int2  (a1)
#define cpu_int3  (a2)
#define cpu_int4 !(a3)
#define cpu_int5 !(a4)
#define cpu_int6 !(a5)
#define cpu_int7  (a6)
#define vme_ipl2 !(a7)
#define vme_ipl1 !(a8)
#define vme_ipl0 !(a9)

/* Define vme interrupt levels */

#define vme_int1        (!vme_ipl2 && !vme_ipl1 &&  vme_ipl0)
#define vme_int2        (!vme_ipl2 &&  vme_ipl1 && !vme_ipl0)
#define vme_int3        (!vme_ipl2 &&  vme_ipl1 &&  vme_ipl0)
#define vme_int4        ( vme_ipl2 && !vme_ipl1 && !vme_ipl0)
#define vme_int5        ( vme_ipl2 && !vme_ipl1 &&  vme_ipl0)
#define vme_int6        ( vme_ipl2 &&  vme_ipl1 && !vme_ipl0)
#define vme_int7        ( vme_ipl2 &&  vme_ipl1 &&  vme_ipl0)

/* Perform priority encoding */
intlevel()
{  int level;
   level = 0;                                   /* Lowest priority */
   if (vme_int1 || cpu_int1) level = 1;
   if (vme_int2 || cpu_int2) level = 2;
   if (vme_int3 || cpu_int3) level = 3;
   if (vme_int4 || cpu_int4) level = 4;
   if (vme_int5 || cpu_int5) level = 5;
   if (vme_int6 || cpu_int6) level = 6;
   if (vme_int7 || cpu_int7) level = 7;         /* Highest priority */
   return(level);
}
```

```
/* Define Outputs */
#define ipl2            (intlevel() & 0x04)     /* msb */
#define ipl1            (intlevel() & 0x02)
#define ipl0            (intlevel() & 0x01)     /* lsb */

main()
{
prom1024x4;

prombegin
prom(0,d0,!ipl0)
prom(0,d1,!ipl1)
prom(0,d2,!ipl2)
prom(0,d3,1)
promend;

writeprom('A105',0);
}
```

```
paltype pal1618
palname A106
palid 1.10 84/07/31

PALBEGIN

% Inputs

1  INPUT C.S5
2  INPUT C.S7
3  INPUT C.S9
4  INPUT CE.IO-
5  INPUT PIN5
6  INPUT PIN6
7  INPUT Q.R/W-
8  INPUT P2.WAIT-
9  INPUT P.A01
11 INPUT R.DMAEN-

10 GND
20 VCC

% Outputs

  OUTPUT PIN19
.8 OUTPUT PIN18
17 OUTPUT PIN17
16 OUTPUT PIN16
15 OUTPUT TOWAIT-
14 OUTPUT IOWAIT-
13 OUTPUT Q.BANK0
12 OUTPUT Q.BANK1

EQUATIONS

: WRITE Q.R/W ;                 % For ease of reading

ASSERT TOWAIT-
ENABLE ALWAYS
OR     WRITE & P2.WAIT

% At 10 MHz, 2 wait I/O, 1 wait frame buffer.
: IOWAIT_10MHZ
  OR    CE.IO & / C.S7
  OR    / C.S5 ;
% At 12 MHz, 3 wait I/O, 2 wait frame buffer.
: IOWAIT_12MHZ
  OR    CE.IO & / C.S9
  OR    / C.S7 ;

ASSERT IOWAIT-
ENABLE  ALWAYS
IOWAIT_10MHZ                            % Initial product at 10 mhz

ASSERT Q.BANK0                          % Select first ram bank
```

```
ENABLE  ALWAYS                          % De-morganize: / P.A01 or R.DMAEN
OR      P.A01 & / R.DMAEN

ASSERT Q.BANK1                          % Select Second Ram Bank
ENABLE  ALWAYS                          % De-morganize:   P.A01 or R.DMAEN
OR      / P.A01 & / R.DMAEN

TIMING: NO-CLOCK
%               12345678901234567890123456789012345678901234567890
Q.R/W-          __--__--__--__--__--__--__--__--__--__--__--__--__--__
P2.WAIT-        ____----____----____----____----____----____----____----__
TOWAIT-         __------__------__------__------__------__------__------__

C.S5            _-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-
C.S7            __--__--__--__--__--__--__--__--__--__--__--__--__--__--
C.S9            ___----___----___----___----___----___----___----___----__
CE.IO-          ________--------________--------________--------________--
IOWAIT-         ___-___-_-_-_-_-___-___-_-_-_-_-___-___-_-_-_-_-___-__

P.A01           __--__--__--__--__--__--__--__--__--__--__--__--__--__--__
R.DMAEN-        ____----____----____----____----____----____----____----__
Q.BANK0         ------__------__------__------__------__------__------__--
Q.BANK1         ----__------__------__------__------__------__------__----

PIN5            XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PIN6            XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
PIN19           ??????????????????????????????????????????????????????????
PIN18           ??????????????????????????????????????????????????????????
PIN17           ??????????????????????????????????????????????????????????
PIN16           ??????????????????????????????????????????????????????????

PALEND
```

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

/* This information proprietary to Sun Microsystems Inc */

/* ================================================================
 * Author: Model-50 Group
 * Date : March 9, 1984
 * Prom Name: A1604 and A1605
 * Prom Type: 32 x 8.
 * Speed: 35 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.
 *
 * XREQ=0 (No CPU read/write access)
 * ----------------------------------------------------------------------
 * State   0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15
 * ----------------------------------------------------------------------
 * Clock   --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
 * VOE\    0000000011111111222222223333333344444444555555556666666677777777
 * RAS\    ----------------------------------------_____________________----
 * RAS0\   ------------------------------------------_____________________
 * CAS\    ____----------------------------------------_____________------
 * G\      ____----------------------------------------_____________------
 * WE\     ----------------------------------------------------------------
 * OEVRA\  --------------------------------_________________-----------------
 * OEVCA\  ----------------------------------------------_________________---
 * OEPRA\  _________________-----------------------------------------------
 * OEPCA\  ----------------_________________-------------------------------
 * HCLK    ________--------________________________________________________
 * ENREQ   ----____________________________________________________________
 * LOAD    ____----____----____----____----____----____----____----____----
 * LD0     ____--____--____--____--____--____--____--____--____--____--____
 * LD1     ____--____--____--____--____--____--____--____--____--____--____
 *
 * XREQ=1 (CPU read/write access)
 * ----------------------------------------------------------------------
 * State   0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15
 * ----------------------------------------------------------------------
 * Clock   --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
 * VOE\    0000000011111111222222223333333344444444555555556666666677777777
 * RAS\    --------_____________________-----------_____________________----
 * CAS\    ____----------------_____________----------------_____________---
 * G\      ____----------------_____________----------------_____________---
 * W\      ------------_____________________-------------------------------
 * OEVRA\  --------------------------------_________________-----------------
 * OEVCA\  ----------------------------------------------_________________---
 * OEPRA\  _________________-----------------------------------------------
 * OEPCA\  ----------------_________________-------------------------------
 * ACK     ____________________________________----____________________----
 * HCLK    ________--------________________________________________________
```

```
 * ENREQ   ----_____________________________________________________________
 * LOAD     ___----___----___----___----___----___----___----___----___----___
 * LD0      ____--____--____--____--____--____--____--____--____--____--____--_
 * LD1      _____--____--____--____--____--____--____--____--____--____--____--
 *
 * ============================================================================
 */

#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,10)))

#define pra     range( 0,nstate, 3)
#define pca     range( 4,nstate, 7)
#define vra     range( 8,nstate,11)
#define vca     range(12,nstate,15)

#define ack     (xreq && (nstate==9))
#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("A1604",0);
writeprom("A1605",1);

}
```

```
paltype pal16r4
palname A1616
palid 1.6 84/07/30

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.WE-
8  INPUT V.RCO-
9  INPUT V.STATE3
12 INPUT V.DISPEN-

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-
13 OUTPUT V.CEN-

EQUATIONS

: VIDEO_CYCLE   V.STATE3 ;                    % Cycle is for video
: CPU_CYCLE   / V.STATE3 ;                    % Cycle is for CPU

% 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
```

```
paltype pal1618
palname A1620
palid 1.11 84/08/01

PALBEGIN

% Inputs

1  INPUT V.BSEL-
2  INPUT V.CSEL-
3  INPUT P2.RD-
4  INPUT P2.WEU-
5  INPUT P2.WEL-
6  INPUT V.WL-
7  INPUT P2.A17
8  INPUT V.WU-
9  INPUT V.ACK
11 INPUT Q.AS

10 GND
20 VCC

% Outputs

19 OUTPUT V.RD-
18 OUTPUT V.RDC-
17 OUTPUT V.WLC
16 OUTPUT V.WUC
15 OUTPUT V.REQ-
14 OUTPUT V.RDACK-
13 OUTPUT V.WAIT-
12 OUTPUT VREQ                          % NAME OK, we gen both V.REQ and V.REQ-

EQUATIONS

ASSERT V.RD-                            % Enable frame buffer read data to P2
ENABLE ALWAYS
OR      V.BSEL & P2.RD & / P2.A17

ASSERT V.RDC-                           % Enable control reg read data to P2
ENABLE ALWAYS
OR      V.BSEL & P2.RD & P2.A17

ASSERT V.WUC                            % Write upper byte control reg
ENABLE ALWAYS                           % De-morganize: V.BSEL & AS & A17 & WEU
OR      / V.BSEL
OR      / Q.AS
OR      / P2.A17
OR      / P2.WEU

ASSERT V.WLC                            % Write lower byte control reg
ENABLE ALWAYS                           % De-morganize: V.BSEL & AS & A17 & WEL
OR      / V.BSEL
OR      / Q.AS
OR      / P2.A17
```

```
OR      / P2.WEL

ASSERT V.REQ-                           % Frame buffer request
ENABLE ALWAYS
OR      V.BSEL & Q.AS & / P2.A17 & P2.RD  & / V.ACK & / V.RDACK
OR      V.BSEL & Q.AS & / P2.A17 & P2.WEU & / V.ACK
OR      V.BSEL & Q.AS & / P2.A17 & P2.WEL & / V.ACK
OR      V.CSEL & Q.AS            & P2.WEU & / V.ACK
OR      V.CSEL & Q.AS            & P2.WEL & / V.ACK
OR      V.REQ                              & / V.ACK          % Hold til ack

ASSERT V.RDACK-                         % Used to hold deassertion of V.WAIT
ENABLE ALWAYS
OR      V.ACK & / V.WL & / V.WU         % Set at end of read req
OR      P2.RD & V.RDACK                 % Hold till RD gone

ASSERT V.WAIT-                          % Inhibit DTACK
ENABLE ALWAYS
OR      V.BSEL & P2.RD & / P2.A17 & / V.RDACK   % Set on RD
OR      V.WAIT & P2.RD & / V.RDACK              % Hold until V.RDACK
OR      / P2.WEL & / P2.WEU & V.REQ             % Set at end of write
OR      V.WAIT & V.REQ & / V.ACK                % Hold till V.ACK

ASSERT VREQ                             % Inverter
ENABLE ALWAYS
OR      V.REQ-

TIMING: NO-CLOCK
%               123456789012345678901234567890123456
P2.RD-          --___----___----------------------
P2.WEL-         ------_-------__--__---------------
P2.WEU-         -------_---------------__--___-----
Q.AS            ______-------_--__--__--__---_____
V.REQ-          ---------__---_---_--_---_---
VREQ            _________--___----_---_----_---___
V.ACK           -_________-_-____-_--_-____-___--_
V.WAIT-         ----_----_-----_--_---__--_---
V.RDACK-        _----------_--------------------_-
V.WL-           ---------------____-___-----------
V.WU-           ----------------------_________--
V.BSEL-         ---_______---___-----------___---
V.CSEL-         ------------------___--___-------

P2.A17          __--_----_________:----------_______
V.RD-           ----_----_------------------------
V.RDC-          ---_-------------------------------
                ------_-------------------------------
                ------_-------------------------------

PALEND
```

```
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 */

/* ==========================================================================
 * Author: Model-50 Group
 * 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 */

/* ================================================================
 * Author: Model-50 Group
 * 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
 * invisble    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
 * invisble   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);
```

```
}
```

```
paltype pal16r8
palname A214
palid 1.36 84/08/26

PALBEGIN

% Inputs

2 INPUT S.RREQ-
3 INPUT S.EREQ-
4 INPUT S.XREQ-
5 INPUT S.EHOLD
6 INPUT P.BG-
7 INPUT S.ASIN
8 INPUT S.ERR-
9 INPUT Q.S7

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

% Outputs

19 OUTPUT R.DMAEN-
18 OUTPUT E.DMAEN-
17 OUTPUT X.DMAEN-
16 OUTPUT E.CLR-
15 OUTPUT S.DMA-
14 OUTPUT S.ASOFF-
13 OUTPUT PIN13-
12 OUTPUT S.BR-

EQUATIONS

: IDLE          / R.DMAEN & / E.DMAEN & / X.DMAEN ;

% Non-contigous DVMA
: GRANT1        P.BG & / S.ASIN  & IDLE  & S.BR & / S.ERR ;

% Back-to-Back DVMA (Last state on current DMAEN cycle)
: GRANT2        P.BG & S.ASIN & S.ASOFF & S.DMA & S.BR & / S.ERR ;

ASSERT S.BR-
OR      / P.BG & S.RREQ                         % Set
OR      / P.BG & S.EREQ & S.EHOLD               % Set
OR      / P.BG & S.XREQ                         % Set
OR      S.BR & S.RREQ                           % Hold. Pending S.RREQ
OR      S.BR & S.EHOLD & / R.DMAEN              % Hold. Pending S.EHOLD /R.DMAEN
OR      S.BR & S.XREQ                           % Hold. Pending S.XREQ
OR      S.BR & R.DMAEN & / S.ASIN               % Hold while R.DMAEN
OR      S.BR & S.DMA & / R.DMAEN & / Q.S7       % Hold while E.DMAEN and X.DMAEN

ASSERT R.DMAEN-                                 % Refresh dvma
OR      S.RREQ & GRANT1
OR      S.RREQ & GRANT2
```

```
OR      R.DMAEN & S.DMA & / S.ASOFF
OR      R.DMAEN & S.DMA & / S.ASIN
OR      S.DMA & / S.ASOFF & / E.DMAEN & / X.DMAEN       % Deal with S.DMA on POR

ASSERT E.DMAEN-                                         % Ethernet dvma
OR      S.EREQ & S.EHOLD & / S.RREQ & GRANT1
OR      S.EREQ & S.EHOLD & / S.RREQ & GRANT2
OR      E.DMAEN & S.DMA & / S.ASOFF
OR      E.DMAEN & S.DMA & / S.ASIN

ASSERT X.DMAEN-                                         % VME-bus dvma
OR      S.XREQ & / S.RREQ & / S.EREQ & GRANT1
OR      S.XREQ & / S.RREQ & / S.EREQ & GRANT2
OR      X.DMAEN & S.DMA & / S.ASOFF
OR      X.DMAEN & S.DMA & / S.ASIN

ASSERT S.DMA-                                           % R.DMAEN or E.DMAEN or X.DMAEN
OR      S.RREQ & GRANT1                                 % Refresh turnons
OR      S.RREQ & GRANT2
OR      S.EREQ & S.EHOLD & / S.RREQ & GRANT1            % Ethernet turnons
OR      S.EREQ & S.EHOLD & / S.RREQ & GRANT2
OR      S.XREQ & / S.RREQ & / S.EREQ & GRANT1           % VME turnons
OR      S.XREQ & / S.RREQ & / S.EREQ & GRANT2
OR      S.DMA & / S.ASOFF
OR      S.DMA & / S.ASIN

ASSERT S.ASOFF-
OR      R.DMAEN & S.ASIN
OR      S.DMA & Q.S7 & S.ASIN
OR      S.ERR

% Assert E.CLR to clear Ethernet request flipflop.
ASSERT E.CLR-
OR      S.EREQ & E.DMAEN & / S.ASIN

TIMING:
%           1122334455667788990011223344556677889900112233445566778899001122
CLK         _-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-
P.BG-       ------___------_____----_____--------
S.BR-       ?----___------_____----_____----------
S.RREQ-     ----___-----------------------_____----------
S.EREQ-     ----------_____----------_____----------
S.EHOLD     _____------_____----------_____
S.XREQ-     ----------___----_____----------

S.ASOFF-    ?_____--___--------_----_----------_----_----_____-----
S.ASIN      ----___----_____----_----_____----_----_----_____
Q.S7        ----_____--___--_____--___-_____
X.DMAEN-    ???-----------_____------_____----------
E.DMAEN-    ???----------_____----------_____----------
R.DMAEN-    ???----___-----------------------_____----------

S.DMA-      ???----___------_____------_____----------
E.CLR-      ?----------_----------_----------
S.ERR-      _____-----------------------------___-------
OE-         ____________________________________________
```

```
PIN13-        ??????????????????????????????????????????????????????????????????????

PALEND
```

```
paltype pal16l8
palname A800
palid 1.11 84/07/30

PALBEGIN

% Inputs

1  INPUT R.DMAEN-                    % Refresh DMA enable
2  INPUT E.DMAEN-                    % Ethernet DMA enable
3  INPUT X.DMAEN-                    % VME (eXternal) DMA enable
4  INPUT Q.R/W-                      % Read or Write
5  INPUT S.DMA-
6  INPUT S.ASOFF-
7  INPUT S.ASON-
8  INPUT E.A0
9  INPUT X.LDS-
11 INPUT X.UDS-

10 GND
20 VCC

% Outputs

19 OUTPUT E.WE
18 OUTPUT E.OE-
17 OUTPUT P.FC0
16 OUTPUT P.FC1
15 OUTPUT P.FC2
14 OUTPUT P.AS-
13 OUTPUT P.UDS-
12 OUTPUT P.LDS-

EQUATIONS

: WRITE Q.R/W ;

ASSERT E.WE                         % Demorganize:
ENABLE  ALWAYS                      % E.DMAEN & S.ASON & / S.ASOFF & / WRITE
OR      / E.DMAEN
OR      / S.ASON
OR      S.ASOFF
OR      WRITE

ASSERT E.OE-
ENABLE  ALWAYS
OR      E.DMAEN & / S.ASOFF & WRITE

% Function codes for DVMA cycles:
%       R.DMAEN: FC=7. CPU space.
%       E.DMAEN: FC=5. Supervisor data.
%       X.DMAEN: FC=5. Supervisor data.

ASSERT P.FC0
ENABLE  S.DMA
```

```
OR      NEVER

ASSERT P.FC1
ENABLE  S.DMA
OR      / R.DMAEN

ASSERT P.FC2
ENABLE  S.DMA
OR      NEVER

ASSERT P.AS-                    % Address strobe for DVMA cycles
ENABLE  S.DMA
OR      S.ASON & / S.ASOFF

ASSERT P.UDS-                   % Data strobes for DVMA cycles
ENABLE  S.DMA
OR      E.DMAEN & / E.A0 & S.ASON & / S.ASOFF
OR      X.DMAEN & X.UDS & S.ASON & / S.ASOFF

ASSERT P.LDS-
ENABLE  S.DMA
OR      E.DMAEN & S.ASON & / S.ASOFF
OR      X.DMAEN & X.LDS & S.ASON & / S.ASOFF

TIMING: NO-CLOCK
%               12345678901234567890123456789012345678901234567890
Q.R/W-          _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
S.ASON-         __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __
S.ASOFF-        ___ ____ ____ ____ ____ ____ ____ ____ ____ ____ ____
E.DMAEN-        _____ ________ _____ ________ _____ ________ _____ __
E.WE            _____-_____________-_____________-_____________-_____
E.OE-           ____-_-___________-_-___________-_-___________-_-____

R.DMAEN-        _____________-----------_____________--
S.DMA-          _____________________________-----------------
P.FC0           ------------------------------xxxxxxxxxxxxxxxx
P.FC1           ----------------_______________xxxxxxxxxxxxxxxx
P.FC2           ------------------------------xxxxxxxxxxxxxxxx
P.AS-           ____-______-______-______-__xxxxxxxxxxxxxxxxxx

E.A0            _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
X.DMAEN-        _____________-----------_____________--
X.UDS-          ___ __ _____ __ _____ ________ _____ ________ ____ __
X.LDS-          ____ ________ _ ____ ________ _____ ________ ____ __
P.UDS-          ____-________-______-----------xxxxxxxxxxxxxxxxxx
P.LDS-          ____-______-_______-----------xxxxxxxxxxxxxxxxxx

PALEND
```

```
paltype pal16r4
palname A507
palid 1.18 84/08/01

PALBEGIN

% Inputs

2  INPUT WR.DCP-                        % Write strobe from I/O decoders
3  INPUT C-160
4  INPUT C-320
5  INPUT C.S9
6  INPUT P.A01
7  INPUT RD.DCP-
8  INPUT C.S6
9  INPUT C-80-

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

% Outputs

19 OUTPUT MAS-                          % Address strobe to DES chip
18 OUTPUT D.START-
17 OUTPUT D.END-
16 OUTPUT T160-
15 OUTPUT T240-
14 OUTPUT WAIT-
13 OUTPUT MDS-
12 OUTPUT P2.WAIT-

EQUATIONS

ASSERT MAS-                             % DCP addr strobe. Unclocked.
ENABLE ALWAYS
OR      P.A01 & WR.DCP & / C.S9

ASSERT D.START-                         % DCP data strobe. Unclocked.
ENABLE ALWAYS
OR      / P.A01 & RD.DCP & / C.S9
OR      / P.A01 & WR.DCP & / C.S9 & C.S6
OR      D.START & / T240

ASSERT D.END-                           % DCP data strobe. Clocked.
OR      D.START & / T240

ASSERT T160-                            % Internal clocked edge
OR      D.END & C.S9 & / C-320 & C-160
OR      T160 & / T240

ASSERT T240-                            % Internal clocked edge
OR      T160
```

```
ASSERT WAIT-                                    % Clocked
OR      D.START & RD.DCP & / D.END & / T160 & / T240
OR      D.START & RD.DCP & / C.S9  & / T160 & / T240
OR      D.START & RD.DCP &   C-320 & / T160 & / T240
OR      D.START & RD.DCP & / C-160 & / T160 & / T240
OR      D.START & WR.DCP                  & / T240

ASSERT MDS-                                     % Actual data strobe to chip
ENABLE  ALWAYS
OR      D.START
OR      D.END
OR      MDS & / C-80

ASSERT P2.WAIT-                                 % Open-collector WAIT
ENABLE WAIT
OR      WAIT

TIMING:
%           1122334455667788990011223344556677889900112233445566778899001122
CLK         _-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-
C-80-       -_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
C-160       __--__--__--__--__--__--__--__--__--__--__--__--__--__--__--__--__
C-320       ____----____----____----____----____----____----____----____----__
C.S6        ________----------____----____----------____----------____--------
C.S9        ____________----------____--____----------____----------____------
P.A01       ____________________----------____________________________________
MAS-        ----------------------------___-------------------------------------
WR.DCP-     ------____________------___-----____-------------------------------
RD.DCP-     ----------------------------------____----------____----------------
WAIT-       ?????????_____----------____----------____----------____----------
P2.WAIT-    ?????????_____xxxxxxxxxxxxxx____----------____xxxxxxxxxxxx
D.START-    ????????_____----------____----------____----------____----------
D.END-      ?????????_____----------____----------____----------____----------
MDS-        ????????_____----------____----------____----------____----------
T160-       ???----------___----------____----------____----------____--------
T240-       ?????----------___----------____----------____----------____------
OE-         ____________________________________________________________________

PALEND
```

```
paltype pal1618
palname A810
palid 1.42 84/10/04

PALBEGIN

% Inputs

1  INPUT RD.P1-
2  INPUT WR.P1-
3  INPUT B.AEN-
4  INPUT B.TO3-
5  INPUT X.DMA-
6  INPUT P.RESET-
7  INPUT X.DMAEN-
8  INPUT P1.WRITE-
9  INPUT C.S4-
11 INPUT Q.INTVEC-
13 INPUT P.R/W-

10 GND
20 VCC

% Outputs

19 OUTPUT VME.DWEN-
18 OUTPUT B.BSEL-
17 OUTPUT B.RERUN-
16 OUTPUT B.FREEZE-
15 OUTPUT VME.DOE-
14 OUTPUT XHOLD-
12 OUTPUT VME.DIE-

EQUATIONS

ASSERT B.BSEL-                          % CPU wants vme mastership
ENABLE ALWAYS
OR      RD.P1 & / X.DMAEN               % Read cycle
OR      WR.P1 & / X.DMAEN               % Write cycle
OR      Q.INTVEC & / X.DMAEN            % VME interrupt acknowledge cycle
OR      B.BSEL & C.S4                   % Hold during read-modify-write cycles
OR      B.BSEL & B.FREEZE               % Hold during any rerun condition

ASSERT B.FREEZE-
ENABLE ALWAYS
OR      B.RERUN & / X.DMAEN & / P.RESET % Set
OR      B.FREEZE & / C.S4 & / P.RESET   % Hold starting on RERUN turn-off
OR      B.FREEZE & / RD.P1 & / WR.P1 & / Q.INTVEC & / P.RESET
OR      B.FREEZE & X.DMAEN & / P.RESET

ASSERT B.RERUN-
ENABLE ALWAYS
OR      B.TO3                           % 2.5 usec short timeout
OR      RD.P1    & X.DMA & / XHOLD      % Bus deadlock
OR      WR.P1    & X.DMA & / XHOLD      % Bus deadlock
```

```
OR      Q.INTVEC & X.DMA & / XHOLD      % Bus deadlock
OR      B.RERUN  & C.S4                 % Hold until CPU accepts Rerun.

ASSERT XHOLD-
ENABLE ALWAYS
OR      X.DMAEN                         % Set
OR      XHOLD & X.DMA                   % Hold till end of XDMA
OR      XHOLD & B.FREEZE                % Or Hold till rerun cycle

ASSERT VME.DWEN-
ENABLE ALWAYS
OR      C.S4 & / X.DMA & / B.FREEZE & / B.BSEL
OR      C.S4 & / X.DMA & XHOLD
OR      C.S4 & X.DMAEN

ASSERT VME.DOE-                         % Enable output data to VME
ENABLE ALWAYS
OR      B.AEN & / X.DMAEN & / P.R/W- & C.S4   % Set on CPU write.
OR      B.AEN & P1.WRITE & B.FREEZE     % Hold during CPU rerun
OR      X.DMA & / P1.WRITE              % DVMA read

ASSERT VME.DIE-                         % Enable input data from VME
ENABLE ALWAYS
OR      RD.P1
OR      Q.INTVEC
OR      X.DMAEN & / P.R/W-

TIMING: NO-CLOCK
%               12345678901234567890123456789012345678
P.RESET-        _______________________________________
B.TO3-          ________________ ______________________
                               _

B.BSEL-         _ _ _ _ _ _ _ _ _____________ _______ ___
B.FREEZE-       ________________ _____________ _________ _
B.RERUN-        ________________ _____________ _________ _
RD.P1-          _ __________ _ _ ______ ___ ___ ___ ___ _
WR.P1-          _____ ______ _ _____ _ _____ _____ _____
Q.INTVEC-       _________ __ _ ____ _____ _____ _____ ___
C.S4-           __ __ __ __ _ __ __ __ __ __ __ _ __ _ __

XHOLD-          ____________________ _____ ___ ____ _____
X.DMA-          ____________________ _____ ____ ____ ____

B.AEN-          __ _________ _____ _ _____ _____ _____ ___
X.DMAEN-        ____________________ ___ ______ ____ _____
P1.WRITE-       _____ ___ ____ ___ _____ _____ _____ ____
P.R/W-          _____ ___ _________ _____ _____ _____ ____

VME.DWEN-       ____________________ ________ _ _ ___ ___
VME.DOE-        ______ __ __________ _____ _ ___ ____ ____
VME.DIE-        _ ______ __ __ _ _ _ ___ __ ___ ___ ___ _

PALEND
```

```
static char* sccs_id = "1.17 84/10/02";

/* This information proprietary to Sun Microsystems Inc */

/* ============================================================
 * Author: Model-50 Group
 * Date : March 9, 1984
 * Prom Name: A811
 * Prom Type: 1024 x 4.
 * Speed: 25 nsec.
 * Purpose: VME-Bus Arbiter and Master functions
 * Timing:
 *   The arbitration state machine supports the CPU as either the Arbiter,
 *   a mere Bus Master, or Both. The state machine supports overlapped
 *   data transfers and bus arbitration. If we are the bus arbiter and
 *   no one wants the bus and the bus is not busy then we will wait in
 *   Bus Master State under the assumption that the next bus transfer will
 *   be from the CPU to the VME-Bus. Likewise, if are the current bus
 *   master, we will remain in Bus Master State until someone specifically
 *   requests the bus. These steps remove arbitration overhead on CPU to
 *   VME accesses and do not slow bus arbitration on requests by other
 *   bus masters.
 *
 *   The arbitration state machine has 9 potential states.  These states are:
 *      IDLE            Bus in use by another master
 *      BUSREQ          We are issuing a bus request
 *      WAITFORBUS      Next bus cycle is ours. Wait for deassertion of AS.
 *      WAITREQ         Next bus cycle is ours. Hold bus request one more state.
 *      BUSGRANT        We are issuing a bus grant out.
 *      MASTER_REQ      We just became bus master after asserting bus request.
 *      MASTER          We are bus master asserting bus busy.
 *      MASTER_NB       We are bus master not asserting bus busy.
 *      MASTER_G        We are bus master asserting bus grant out.
 *
 * ============================================================
 */

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

/* Define inputs to 1K x 4 Prom. */
#define bgout    !(a0)
#define bbout    !(a1)
#define brout    !(a2)
#define aen      !(a3)
#define bbin     !(a4)
#define brin     !(a5)
#define as       !(a6)
#define sel      !(a7)
#define bgin     !(a8)
#define arb      !(a9)

/* Define Bus Busy. "bbin" suppressed externally if "bbout" asserted. */
/* Prevents transition from MASTER to BUSGRANT to IDLE on our own "bbout". */
#define busy (bbin || bbout)
```

```
/* Define Bus Req. "brin" suppressed externally if "brout" asserted. */
/* Prevents transition from BUSREQ to MASTER to MASTER_G on our own "brout". */
#define breq (brin || brout)

/* Define potential states */

#define IDLE            (!bgout && !bbout && !brout && !aen)
#define BUSREQ          (!bgout && !bbout &&  brout && !aen)
#define WAITFORBUS      (!bgout &&  bbout && !brout && !aen)
#define WAITREQ         (!bgout &&  bbout &&  brout && !aen)
#define MASTER_REQ      (!bgout &&  bbout &&  brout &&  aen)
#define MASTER          (!bgout &&  bbout && !brout &&  aen)
#define MASTER_NB       (!bgout && !bbout && !brout &&  aen)
#define MASTER_G        ( bgout && !bbout && !brout &&  aen)
#define BUSGRANT        ( bgout && !bbout && !brout && !aen)

/* Define state transitions */

n_IDLE()
{   int value;
        value  =         arb && IDLE && (busy && !sel);
        value |=         arb && IDLE && (!busy && !sel && !breq && as);
        value |=         arb && BUSGRANT && (busy && !sel);
        value |=         arb && BUSGRANT && (!busy && !breq);   /* Error */
        value |=        !arb && IDLE && (!sel && !bgin);
        value |=        !arb && MASTER && (!sel && !bgin && breq);
        value |=        !arb && MASTER_NB && (!sel && !bgin);
        value |=        !arb && BUSGRANT && (!sel && !bgin);
        return(value);
}

n_BUSREQ()
{   int value;
        value  =         arb && IDLE && (busy && sel);
        value |=         arb && BUSREQ && (busy);
        value |=         arb && BUSGRANT && (busy && sel);
        value |=        !arb && IDLE && (busy && sel && bgin);
        value |=        !arb && IDLE && (sel && !bgin);
        value |=        !arb && BUSREQ && (busy);
        value |=        !arb && BUSREQ && (!bgin);
        value |=        !arb && BUSGRANT && (sel && !bgin);
        return(value);
}

n_WAITFORBUS()
        value;
        value  =         arb && IDLE && (!busy && sel && as);
        value |=         arb && WAITFORBUS && (as);
        value |=         arb && WAITREQ && (as);
        value |=        !arb && IDLE && (!busy && sel && as && bgin);
        value |=        !arb && WAITFORBUS && (as);
        value |=        !arb && WAITREQ && (as);
        return(value);
}
```

```
n_WAITREQ()
{   int value;
        value  =         arb && BUSREQ && (!busy && as);
        value |=        !arb && BUSREQ && (!busy && as && bgin);
        return(value);
}

n_MASTER_REQ()
{   int value;
        value  =         arb && BUSREQ && (!busy && !as);
        value |=        !arb && BUSREQ && (!busy && !as && bgin);
        return(value);
}

n_MASTER()
{   int value;
        value  =         arb && IDLE && (!sel && !busy && !breq && !as);
        value |=         arb && IDLE && (sel && !busy && !as);
        value |=         arb && WAITFORBUS && (!as);
        value |=         arb && WAITREQ && (!as);
        value |=         arb && MASTER_REQ;
        value |=         arb && MASTER && (sel && !breq);
        value |=         arb && MASTER && (sel && breq && !as);
        value |=         arb && MASTER && (!sel && !breq);
        value |=        !arb && IDLE && (sel && !busy && !as && bgin);
        value |=        !arb && WAITFORBUS && (!as);
        value |=        !arb && WAITREQ && (!as);
        value |=        !arb && MASTER_REQ;
        value |=        !arb && MASTER && (sel && !breq);
        value |=        !arb && MASTER && (sel && breq && !as);
        value |=        !arb && MASTER && (!sel && !breq);
        value |=        !arb && MASTER && (bgin);
        return(value);
}

n_MASTER_NB()
{   int value;
        value  =        !arb && MASTER && (sel && !bgin && breq && as);
        value |=        !arb && MASTER_NB && (sel && !bgin);
        return(value);
}

n_MASTER_G()
{   int value;
        value  =         arb && MASTER && (sel && breq && as);
        value |=         arb && MASTER_G && (sel);
        value |=        !arb && MASTER_NB && (sel && bgin);
        value |=        !arb && MASTER_G && (sel);
        return(value);
}

n_BUSGRANT()
{   int value;
        value  =         arb && IDLE && (!sel && !busy && breq);
        value |=         arb && MASTER && (!sel && breq);
        value |=         arb && MASTER_G && (!sel);
```

```
        value |=         arb && BUSGRANT && (!busy && breq);
        value |=        !arb && IDLE && (!sel && bgin);
        value |=        !arb && MASTER_NB && (!sel && bgin);
        value |=        !arb && MASTER_G && (!sel);
        value |=        !arb && BUSGRANT && (bgin);
        return(value);
}

/* Define new outputs */

#define n_bgout (n_MASTER_G() || n_BUSGRANT())
#define n_bbout (n_WAITFORBUS() || n_WAITREQ() || n_MASTER_REQ() || n_MASTER())
#define n_brout (n_BUSREQ() || n_WAITREQ() || n_MASTER_REQ())
#define n_aen   (n_MASTER_REQ() || n_MASTER() || n_MASTER_NB() || n_MASTER_G())

main()
{
prom1024x4;

prombegin

prom(0,d0,!n_bgout)
prom(0,d1,!n_bbout)
prom(0,d2,!n_brout)
prom(0,d3,!n_aen)

promend;

writeprom("A811",0);
}
```

```
paltype pal1618
palname A814
palid 1.10 84/10/04

PALBEGIN

% Inputs

1  INPUT B.C1
2  INPUT B.C3-
3  INPUT C.RAS
4  INPUT B.XBSY-
5  INPUT B.BBOUT-
6  INPUT B.BROUT-
7  INPUT P1.BR0-
8  INPUT P1.BR1-
9  INPUT P1.BR2-
11 INPUT P1.BR3-
15 INPUT B.BRIN-
17 INPUT P1.SYSR-

10 GND
20 VCC

% Outputs

19 OUTPUT B.TO3-
18 NOTUSED
16 OUTPUT B.SYSR
14 NOTUSED
13 OUTPUT B.BBSY-
12 OUTPUT B.BR-

EQUATIONS

ASSERT B.TO3-
ENABLE ALWAYS
OR      B.C3 & B.C1

ASSERT B.SYSR                                   % Inverter
ENABLE ALWAYS
OR      P1.SYSR-

ASSERT B.BBSY-
ENABLE ALWAYS
OR      B.XBSY & / B.BBOUT

ASSERT B.BR-
ENABLE ALWAYS
OR      P1.BR0 & / B.BROUT
OR      P1.BR1 & / B.BROUT
OR      P1.BR2 & / B.BROUT
OR      P1.BR3 & / B.BROUT
OR      B.BRIN & / B.XBSY                       % Hold to correct RC delay
```

```
TIMING: NO-CLOCK
%                 12345678901234567890123456789012345678901234567890
B.C1              ____----____----____----____----____----____----____
B.C3-             --------________--------________--------________----
B.TO3-            ------------____------------____------------____----

P1.SYSR-          ________--------________--------________--------____
B.SYSR            --------________--------________--------________----
B.XBSY-           ________--------________--------________--------____
B.BBOUT-          ____----____----____----____----____----____----____
B.BBSY-           ----____------------____------------____------------

B.BR-             _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
B.BROUT-          _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
B.BRIN-           _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
Q.RAS             ----____----____----____----____----____----____----
P1.BR0-           ________--------________--------________--------____
P1.BR1-           ________________--------------------------------------
P1.BR2-           ----------------________----------------------------
P1.BR3-           ------------------------------------____------------

PALEND
```

```
paltype pal1618
palname A816
palid 1.9 84/10/08

PALBEGIN

% Inputs

1  INPUT Q.AS
2  INPUT P.LDS-
3  INPUT P.UDS-
4  INPUT B.BERRIN-
5  INPUT B.DTACKIN-
6  INPUT B.AEN-
7  INPUT B.BEN-
8  INPUT B.FREEZE-
9  INPUT C.S6
11 INPUT B.BSEL-
14 INPUT B.BBOUT-

10 GND
20 VCC

% Outputs

19 OUTPUT P1.AEN-
18 OUTPUT B.AS-
17 OUTPUT B.LDS-
16 OUTPUT B.UDS-
15 OUTPUT B.DTACK
13 OUTPUT B.DEN-
12 OUTPUT B.BERR-

EQUATIONS

: XEN B.AEN & B.BEN & B.DEN ;

ASSERT B.AS-
ENABLE ALWAYS
OR      XEN & B.BSEL & C.S6 & Q.AS & / B.FREEZE
OR      XEN & B.AS & B.FREEZE                    % Hold between C.S4 and C.S6
OR      XEN & B.AS & Q.AS                        %    after FREEZE deassertion
OR      XEN & B.AS & / B.BBOUT & B.LDS  % Add 2 delay on last vme cycle
OR      XEN & B.AS & / B.BBOUT & B.UDS  % Add 2 delay on last vme cycle

ASSERT B.UDS-
ENABLE ALWAYS
OR      XEN & B.AS & B.BSEL & Q.AS & P.UDS & / B.FREEZE
OR      XEN & B.UDS & B.FREEZE
OR      XEN & B.UDS & Q.AS & P.UDS               % Hold between C.S4 and C.S6

ASSERT B.LDS-
ENABLE ALWAYS
OR      XEN & B.AS & B.BSEL & Q.AS & P.LDS & / B.FREEZE
OR      XEN & B.LDS & B.FREEZE
```

```
OR      XEN & B.LDS & Q.AS & P.LDS                  % Hold for rerun cycle

ASSERT B.DEN-
ENABLE ALWAYS
OR      B.AEN & B.BEN & B.BSEL & C.S6 & Q.AS
          & / B.DTACKIN & / B.BERRIN & / B.FREEZE
OR      XEN & B.FREEZE
OR      XEN & B.AS
OR      XEN & B.LDS
OR      XEN & B.UDS

ASSERT B.DTACK                                      % Demorganize:
ENABLE ALWAYS                                       % DTACKIN DEN / FREEZE & CS6
OR      / B.DTACKIN
OR      / B.DEN
OR      B.FREEZE
OR      / C.S6

ASSERT B.BERR-
ENABLE ALWAYS
OR      B.BERRIN & B.DEN & / B.FREEZE & C.S6

ASSERT P1.AEN-
ENABLE ALWAYS
OR      B.AEN & B.BBOUT
OR      B.AEN & B.DEN & B.FREEZE
OR      B.AEN & B.LDS & P.LDS                       % Off at P.DS + 1 pal
OR      B.AEN & B.UDS & P.UDS                       % Off at P.DS + 1 pal

TIMING: NO-CLOCK
%               123456789012345678901234
B.BBOUT-        ____________________----
B.BEN-          --____--________________
B.AEN-          -______--________________
P1.AEN-         -______--____________--
C.S6            ---_---__----__---__
Q.AS            _-_-----_-----_----_
B.AS-           -----__---____----__--

B.FREEZE-       -------_-------------
B.BSEL-         ----_____-____---___--
B.DTACKIN-      _____--------_____----
B.DEN-          -----_---____----_--
B.DTACK         _____________-________
B.BERRIN-       ------------------_-
B.BERR-         ------------------_--

P.UDS-          -----___------_____-
B.UDS-          -----_----------_--
P.LDS-          -----___------_____-
B.LDS-          -----_----------_--

PALEND
```