Part Number 800-1185-01
Revision 50 of 28 September 1984

Engineering Manual

for the

Sun-2/120 CPU Board

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

Trademarks

Multibus® is a registered trademark of Intel Corporation.

Ethernet® is a registered trademark of Xerox Corporation.

UNIX™ is a trademark of Bell Laboratories.

Sun Workstation® is a registered trademark of Sun Microsystems, Incorporated.
Sun-2/120™, **Sun-2/170™** and **DVMA™**
are trademarks of Sun Microsystems, Incorporated.

Revision History

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

Contents

Contents

Tables

Figures

Preface

Purpose and Audience

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

Organization

This manual contains an overview of the CPU board, including a block diagram. Detailed design information is contained in the Theory of Operation chapters. We also include schematics, PAL listings, and a list of documents you might want to consult for additional information.

Chapter 1 — *Overview* — contains a block diagram of the CPU board, and describes the board connectors and pinouts for the serial and parallel ports.

Chapter 2 — *General Description* — describes the microprocessor, the power requirements, and summarizes the CPU performance.

Chapter 3 — *Control* — explains the generation of control signals.

Chapter 4 — *MMU* — summarizes the Memory Management Unit (MMU).

Chapter 5 — *I/O Space* — describes the I/O devices accessible by the CPU.

Chapter 6 — *P2 Bus Interface* — describes the function of the P2 bus and its decoding and parity logic. This chapter also includes a table of P2 pin assignments.

Chapter 7 — *P1 Bus Interface* — describes the bus master interface and compliance with the Multibus specification.

Chapter 8 — *DVMA Operation* — explains Sun's direct virtual memory access operation.

Appendix A — *CPU Board Schematics*

Appendix B — *PALs* — contains PALASM source code.

Appendix C — *Reference Documents* — lists some documents to refer to for additional information.

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

Notations Used In This Manual

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

Conventions used for hardware signal names in this manual are:

- Both active-high and active-low signals are used. A signal name that is followed by a minus sign (−) indicates that the signal is active LOW (<0.4V). For example, the Column Address Strobe, M.CAS0−, is such a low-active signal.
- A signal that is ***not*** followed by a minus sign is understood to be a HIGH active signal (>2.0V). An example of such a signal would be the parity error signal, PARERR.
- For signals with multiple meanings or synonyms, the signal names are listed as separated by a slash (/). An example of this would be the on-board, memory expansion select signal, PM.OB−/P2. A high signal is understood to be an assertion of the P2 bus, while a low signal is an assertion of on-board status.
- Bus signals are indicated by a common prefix followed by a number. For example, a 16-bit data bus might be labelled D0, D1, D2, and so on until D15.
- A group of signals that is part of a signal vector is denoted by a common prefix separated from its suffix by a period. For example, all P1 signals start with the prefix "P1.", and P1 bus address signals are P1.A00, P1.A01, etc.
- Connector signals are distinguished by a suffix of "[]" with an optional string enclosed inside the square brackets identifying the connector name.

Components

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

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

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

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

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

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

Definitions

Definitions of abbreviations used in the text:

DVMA	Direct Virtual Memory Access
CPU	Central Processing Unit
MMU	Memory Management Unit
PMEG	Page Map Entry Group
RES	Reserved
POR	Power-On Reset

Chapter 1

Overview

1.1. Introduction

The Sun-2/120™ CPU board is the main component of the Sun-2/120 and Sun-2/170™ workstations. It provides on a single Multibus® card, most of the components of a high-performance engineering and scientific workstation: processor, high-speed memory interface, virtual memory management, 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 multi-process virtual memory management. The board addresses one to four megabytes of memory with zero wait-state access. With optional memory boards, each supporting one megabyte of main memory, systems can be configured with up to four megabytes of memory. All main memory is equipped with byte parity error detection.

Input/output includes two high-speed serial lines with full modem control. An interface to the P1 Multibus with master and slave capabilities is provided. For a block diagram illustrating the major sections of the CPU, refer to Figure 1-1 on the following page.

1.2. Block Diagram

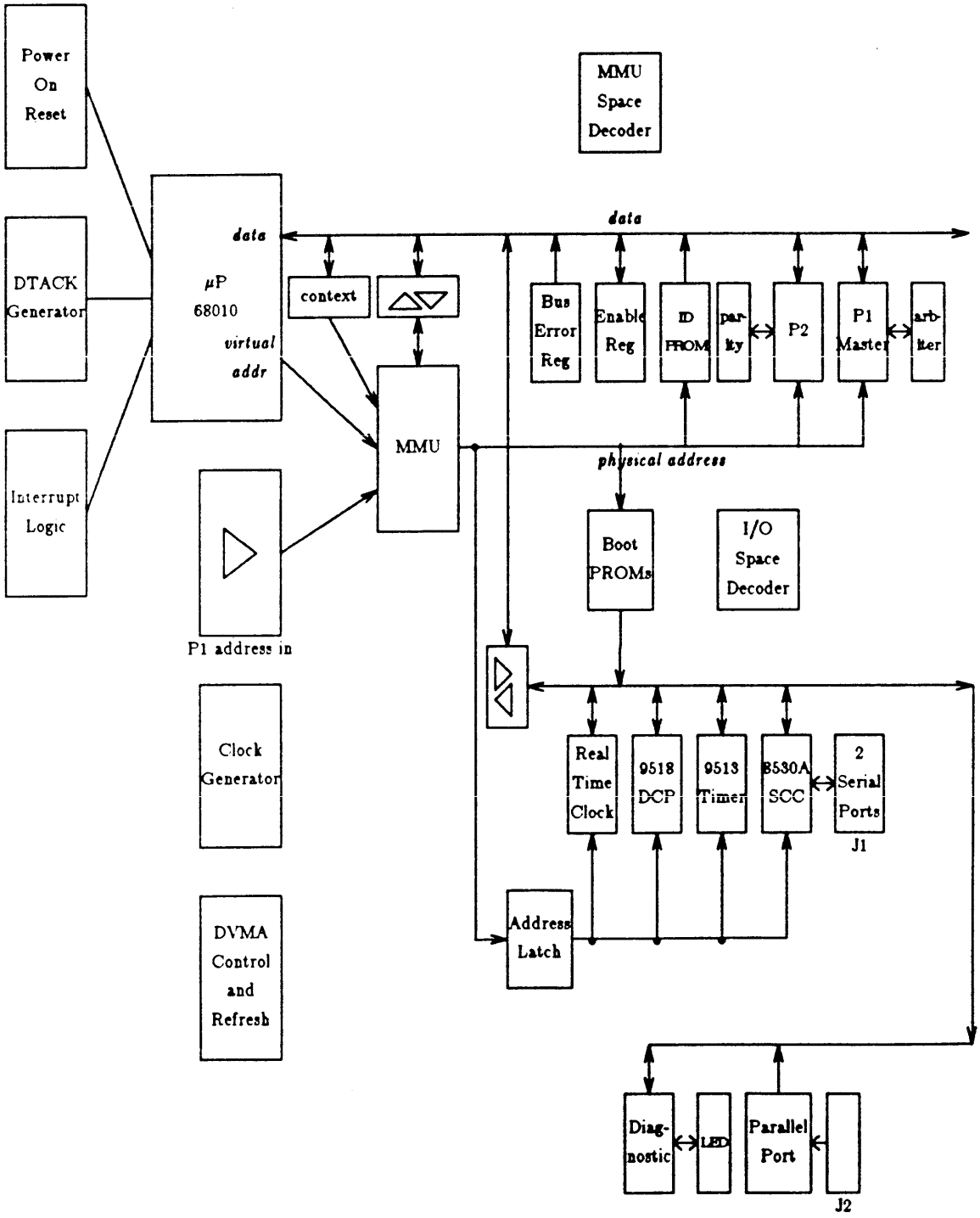


Figure 1-1: 120 CPU Block Diagram

1.3. CPU Board Connectors

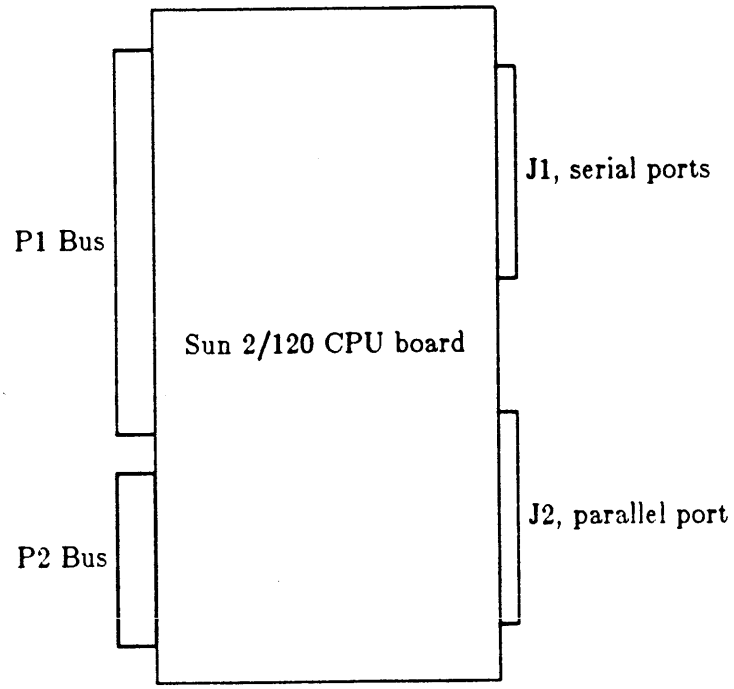
The connectors on the backplane edge of the CPU board are the P1 and P2 connectors:

- P1 — carries the P1 Bus.
- P2 — serves for the memory expansion bus, or the P2 bus.

The connectors on the input/output side of the board are:

- J1 — Serial ports A and B
- J2 — Parallel input port (keyboard)

Figure 1-2: Physical Layout of the Sun-2/120 CPU Board (Component side up)



1.3.1. Connector Pin-outs

Note: on the J1 and J2 connectors, only those pins actually connected to something are listed in the following tables; open pins are not documented.

Table 1-1: J1 Connector Pins

Serial Ports			
Pin	Signal	Pin	Signal
3	TXDA	28	TXDB
4	DBA	29	DBB
5	RXDA	30	RXDB
7	RTSA	32	RTSB
8	DDA	33	DDB
9	CTSA	34	CTSB
11	DSRA	36	DSRB
13	GND	38	GND
14	DTRA	39	DTRB
15	DCDA	40	DCDB
22	DAA	47	DAB
24	SPAREA	49	SPAREB

Table 1-2: J2 Connector Pins

Parallel Port			
Pin	Signal	Pin	Signal
1	IN0	25	IN12
2	GND	26	GND
3	IN1	27	IN13
4	GND	28	GND
5	IN2	29	IN14
6	GND	30	GND
7	IN3	31	IN15
8	GND	32	GND
9	IN4	34	GND
10	GND	36	GND
11	IN5	37	VCC
12	GND	38	GND
13	IN6	39	VCC
14	GND	40	GND
15	IN7	41	INT7–
16	GND	42	GND
17	IN8	43	POR–
18	GND	44	GND
19	IN9	45	P.RESET–
20	GND	46	GND
21	IN10	47	P.HALT–
22	GND	48	GND
23	IN11	49	VCC
24	GND	50	GND

Chapter 2

General Description

This manual describes the theory of operation of the Sun-2 CPU board. The discussion assumes that the reader is familiar with the architecture, the installation, and the programming of the Sun-2 CPU Board. In addition, 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.

2.1. Microprocessor

The Sun-2/120 CPU board uses a 10-MHz Motorola 68010 microprocessor. For more information on its timing and programming, refer to the M68000 16/32-Bit Microprocessor Programmer's Reference Manual.

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 4.2BSD version of the UNIX† operating system. It provides multiple, simultaneous process contexts each with up to 16 megabytes virtual memory space. In addition, the MMU provides separate address spaces for the system and for the user.

2.2. Power

The 120 CPU board uses +5V for most of its on-board logic. It requires -5V for the RS423 drivers. The -5V is generated from the -12V supply by on-board regulator LM337 (U137), or can be selected directly from the P1 bus via jumper J102. Signal -5VR connects to pins 24 and 49 on J1 to terminate that line.

2.3. Performance

The CPU speed is summarized below:

```
CPU clock cycle:   101.72 nsec
CPU basic cycle:   406.90 nsec
```

† UNIX is a trademark of Bell Laboratories.

2.3.1. Memory Access Times

The Sun-2/120 CPU board addresses one to four megabytes of main memory. With Sun-2/120 optional memory expansion boards, one to three additional megabytes of main memory can be added. Memory is equipped with byte parity error detection.

All accesses to main memory run with zero wait states, except when copy mode is enabled. Refer to the Sun-2/120 Video Manual for an explanation of copy mode. When copy mode is enabled, accesses to video memory will overlap with 128 kbytes of main memory, and the accesses to either video or main memory will have the same performance as the video. Refresh is done via a DVMA channel, so there is no direct impact of refresh overhead on main memory access time.

2.3.2. Video Memory Access Time

Write accesses to the video memory are buffered. Thus, a single write will complete without wait states. A subsequent operation, whether read or write, will have to wait until the video memory has completed the requested operation. Write accesses to the video memory by copy mode cause the same behavior as direct write accesses. Read accesses to the video memory are not buffered and must wait until the cycle completes.

2.3.3. Multibus Access Times

Multibus I/O devices are identical to Multibus memory except that they are located in a separate address space. The timing of Multibus accesses depends on two factors: the access time of the Multibus device, and the cost of Multibus acquisition if the Sun-2 processor currently does not own Multibus mastership. Once Multibus mastership is acquired, it is retained and given up only on demand if another master requests it.

The total number of wait states for a Multibus access can be computed by the following formula: 2 WS (overhead) + 3 WS (if board is currently not Multibus master) + access time of Multibus device divided by the clock period of the CPU rounded up to the nearest integer number. Another limitation on Multibus access time is cycle time. Some Multibus devices have slower cycle times than access times which will increase effective access time in cycle-time-limited transfers.

2.3.4. Multibus DVMA Access Time

DVMA cycles from the Multibus 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 typically ranges from 750 nanoseconds to 1050 nanoseconds with an average of about 900 nanoseconds.

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 a 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.

2.4. Serial I/O

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.

2.5. Other features

The Sun-2/120 board includes a bidirectional interface to the P1 bus with master and slave capabilities. The board provides 20-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 power-on self tests.

2.6. Initialization and Power Circuitry

Three types of reset need to be distinguished:

2.6.1. Power-On/Power-Off Reset

The 120 CPU 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 **K100**, and resistor network **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 volts.

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.

2.6.2. Watchdog Reset

The Sun-2 architecture provides a watchdog circuit that generates a signal equivalent to power-on reset whenever the CPU halts with a double bus fault. The result of a watchdog reset is identical to a power-on reset, as far as the CPU and the system are concerned.

2.6.3. CPU reset

When the CPU executes a reset instruction, it resets all on-board and off-board I/O devices that offer an external reset function. No other devices are affected. Specifically, MMU devices such as the system enable register and the diagnostic register are not affected by CPU reset.

2.7. Configuration Jumpers

The jumpers on the CPU Board should be wired as follows:

- J700 (5-6) Bus Priority In (BPRN−) (not installed)
- J700 (7-8) Common Bus Request (CBRQ−) (not installed)
- J701 (1-2) Bus Clock (BCLK) (installed)
- J701 (3-4) Constant Clock (CCLK) (installed)
- J400 (1-2) 27128 EPROMs (installed)
- J400 (3-4) 27256 EPROMs (not installed)

Only one of J400 (1-2) and J400 (3-4) must be installed at a time.

The following table is a summary of the jumpers on the CPU board.

120 CPU Jumper Configuration

Label	Pins	Description	Normal
J100	1-2	connect P1.INT0 to INT0	in
	3-4	connect P1.INT1 to INT1	in
	5-6	connect P1.INT2 to INT2	in
	7-8	connect P1.INT3 to INT3	in
	9-10	connect P1.INT4 to INT4	in
	11-12	connect P1.INT5 to INT5	in
	13-14	connect P1.INT6 to INT6	in
	15-16	connect P1.INT7 to INT7	in
J102	1-2	connect P1.-5V to -5V	in
	3-4	connect -5V on-board regulator to -5V	out
J200	1-2	connect 39.3216 MHz crystal to clock generator	in
J400	1-2	select 27128-type EPROMS	in
	3-4	select 27256-type EPROMS	out
J700	1-2	CPU drives P1 bus with reset	in
	3-4	P1.INIT drives CPU with reset	out
	5-6	enable serial arbitration option	out
	7-8	force arbiter to relinquish bus after each transfer	out
J701	1-2	CPU drives P1.BCLK-	in
	3-4	CPU drives P1.CCLK-	in

Chapter 3

Control

3.1. Clock Generation

All system clocks are derived from a 39.3216-MHZ crystal oscillator K1114A:U200. The oscillator's output frequency, C(25), is divided into clocks C50 and C50– by flip-flop 74F74:U203-0. C50 is then divided to generate C100 and C100–. C50– is used to generate C.S3 which controls the multiplexing of the row and column addresses on the P2 bus. (C.S3 is actually halfway between state 3 and state 4 of the processor clock.) C50 has exactly one 50% duty cycle. C100 (and its inverse) drives the 68010 CPU, the timing flip-flops, as well as synchronizers and state machines on the board.

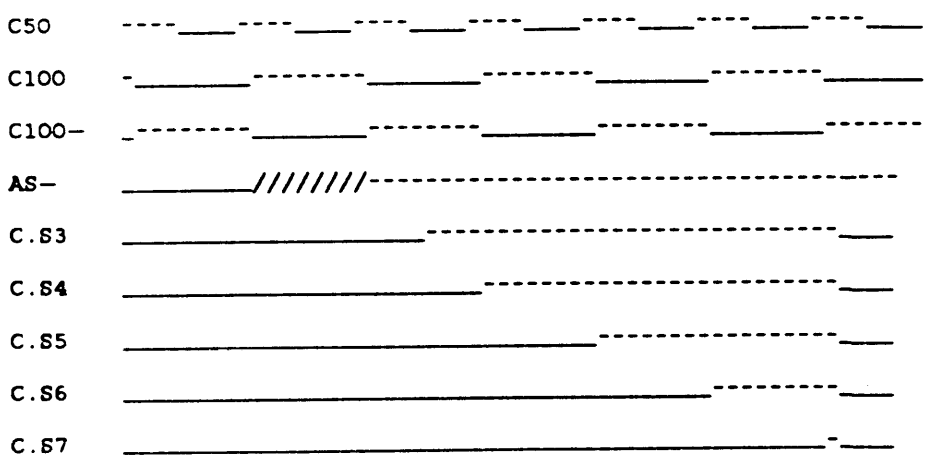
Clocks C.S3, C.S4, C.S5, C.S6, and C.S7 are activated during the corresponding 68010 states S3 through S7. However, clocks C.S4 and C.S6 are only enabled for standard cycles. They are not asserted for cycles that do not have the VALID bit asserted or for non-standard or disabled cycles. Disabled cycles are either Boot Enable cycles (supervisor program access in boot state) or FC0 and FC1 active cycles (MMU accesses and interrupt acknowledge cycles). These conditions are decoded by PAL P16R4:U316 and are indicated by asserting signal DIS.

Table 3-1: Clock Signal Definitions

Signal	Function	Comments
P.AS–	RAM-RAS	Asserted for all 68010 and all DVMA cycles. This means that memory is cycled in sync with P.AS–.
C.S3	RA/CA MUX	Row Address to Column Address multiplexor Asserted on every cycle. Occurs at C.S3.5 .
C.S4	CAS Enable	Asserted only on standard cycles with VALID bit set.
C.S5	Strobe Enable	Asserted on every cycle, but decoder 74F138:U400 is only enabled when ERR indicates there are no errors pending.
C.S6	Statistic Bit Enable	Enables writing of accessed/modified bits Only asserted on standard cycles with VALID bit set.
C.S7	XACK Enable	Enables XACK generation in PAL Q-U212.

The timing of clock C.S3 is special in that it is not asserted on the C(100.50-0) falling edge of C100, but rather 25 nanoseconds later (on the falling edge of C50). The timing diagram below

illustrates when C.S3 and the other clocks become active.



3.2. DTACK Generation

68010 DTACK, or data transfer acknowledge, is generated by multiplexor 74F251:U111 as follows.

On non-standard cycles (see previous section), C.S4 is not asserted selecting C.S5 as P.DTACK, thereby causing one wait state.

On standard cycles, TYPE0 and TYPE1 select DTACK as shown in the following table:

Type	Meaning	DTACK Source
0	P2 Bus Memory	P2.WAIT–
1	Local I/O	IOACK
2	P1 Bus Memory	XACK
3	P1 Bus I/O	XACK

P2.WAIT– indicates that an asychronous P2-bus device (120 video memory) has not yet completed the transfer. Signal IOACK is generated by PAL16R6:U415 causing two wait states for on-board I/O accesses. XACK comes from the P1 bus P1.XACK qualified with AEN via gate 74F32:U108. Thus XACK is only asserted when the CPU board is bus master.

3.3. BERR Generation

68010 BERR, or bus error, can occur under four conditions:

1) Protection Error (PROTERR),
2) Timeout Error (TIMEOUT),
3) Parity Error Low Byte (PARERRL),
4) Parity Error Upper Byte (PARERRU).

These four conditions are combined with gate 74LS20:U114-0.

Bus errors are only recognized during standard cycles, via gate **74F00:U101** and they are suppressed during non-CPU cycles. via signal **P.BACK–** setting synchronization flip-flop **74F74:U105**.

In addition, P.BERR is asserted during rerun cycles via signal **XBERR**. See description of rerun conditions in Chapter 8 for further information.

3.4. VPA Generation

68010 VPA, or valid peripheral address, is asserted on only one condition. On interrupt acknowledge cycles as decoded by **74ALS138:U321**, the assertion of **P.VPA** causes the 68010 to execute autovectored interrupts as defined by 68010 operation.

Chapter 4

Memory Management Unit (MMU)

4.1. CPU and MMU Space

The Sun-2 architecture has three major sections: CPU space, MMU space, and I/O device space (Chapter 5). The following table describes how the different CPU address spaces are mapped to the CPU and the MMU spaces. By using separate address spaces for MMU and CPU space, the full virtual address space is retained for supervisor and user processes.

Figure 4-1: CPU Board Function Codes

FC2	FC1	FC0	Function Code Reference Classification
0	0	0	undefined/reserved
0	0	1	user data
0	1	0	user program
0	1	1	access to MMU
1	0	0	undefined/reserved
1	0	1	superviser data
1	1	0	superviser program
1	1	1	interrupt acknowledge

4.2. CPU Space

The CPU space is composed of the 68010 central proccessing unit (CPU) and DVMA masters, such as the P1 slave interface.

4.3. MMU Space

The MMU space is the core of the Sun-2 architecture. It includes the Memory Management Unit and other Sun-2 architecture extensions to the CPU. The extensions include 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. All devices in the MMU space are accessed by function code 3.

The Sun-2 Memory Management Unit provides translation, protection, sharing, and memory allocation for a multi-process environment. All CPU accesses to memory, on-board I/O, and to the system bus (P1 bus) are translated and protected identically. DVMA accesses by I/O channels also pass through the virtual memory management and thus operate in a fully protected environment.

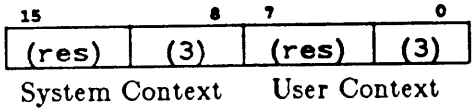
The MMU consists of a context register, a segment map, and a page map. Virtual addresses from the processor are translated into intermediate addresses by the segment map then into physical addresses by the page map.

The MMU has a page size of 2048 (2K) bytes and a segment size of 32K bytes (yielding 16 pages per segment). Up to eight contexts can be mapped concurrently. The maximum virtual address space for each context is 16M bytes.

4.3.1. Contexts

The Sun-2 MMU is divided into eight distinct address spaces or ***contexts***. The current context is selected by means of a three-bit ***context register*** as illustrated in the following figure.

Figure 4-2: Context Register Attributes



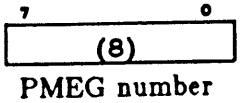
(res): reserved

To allow different address spaces for the supervisor and user, separate context values for each are provided. The MMU automatically uses the system context register whenever the CPU issues a supervisor function code. The supervisor can address the user context via the CPU MOVS instruction using a non-supervisor function code, by mapping the pages of interest into its own system context, or by sharing address space with the user by setting the two context values equal. The two context registers can be accessed together as a word, or separately as the odd or even byte within a word. When read, the reserved bits are not defined.

4.3.2. Segment Map

The segment map has 4096 (4K) entries. It is indexed by the nine most-significant bits of the virtual address and three bits of the current context register. Thus, the segment map is divided into eight sections of 512 entries each, with one section per context. Segment map entries are eight bits wide, pointing to a page map entry group (PMEG) as illustrated in the following figure.

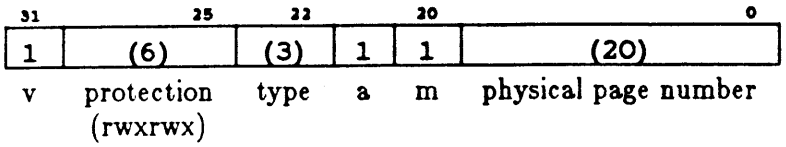
Figure 4-3: Segment Map Attributes



4.3.3. Page Map

The page map contains 4096 (4K) entries each mapping a 2K-byte page. Page map entries are composed of a valid bit, a protection field, a type field, accessed and modified bits (statistics bits), and a page number. The page map is divided into 256 sections of 16 entries each. Each section is pointed to by a segment map entry and is called a page map entry group (PMEG). The following figure illustrates the page map.

Figure 4-4: Page Map Attributes



v: valid bit
a: accessed bit
m: modified bit

The valid bit determines whether a page map entry is valid or not. A valid bit value of 1 means that the page map entry is valid and that the other fields of the page map entry determine how the reference is to be translated and protected. A valid bit value of 0 means that an access to this page will be aborted, while the rest of the page map entry is ignored. In this case, the remaining bits of the page map entry may contain arbitrary information.

Access to pages can be controlled with the six-bit protection field. From left (MSB) to right (LSB), the six bits correspond to "supervisor-read-write-execute" and "user-read-write-execute" priveleges. This provides all 64 combinations of supervisor and user "rwxrwx". A "1" entry enables the corresponding read, write, or execute capability; a "0" entry disables the capability.

The three-bit page type field provides for multiple physical address spaces, each starting at a physical address of 0. At the same time, the page type field can indicate which busses and bus synchronization are used for a particular physical address space. The assignment of the page type field is described in the MMU implementation section.

The accessed and modified bits are set, as the name implies, whenever a page is accessed or modified (written into). The statistics bits (a and m) will not be updated when the page is invalid or when the protection code does not allow the attempted operation. In addition, these bits are not updated in a cycle that aborts due to a parity error in the previous cycle. However, the statistics bits will be updated on all other cycles, including cycles that terminate due to timeout or cycles that cause parity errors.

The page map contains a 20-bit physical page number field. In conjunction with the 11-bit physical byte number (address bits A01..A10), the page map can generate physical addresses of up to 31 bits. However, the Sun-2 architecture does not define how many physical address bits are actually stored in the map, or how many physical address bits are decoded when accessing specific physical devices. For the 120 CPU, the page map field uses 12 bits of the 20-bit physical page number field, allowing physical address of 23 bits, or eight megabytes. The other physical address bits in the page map are unimplemented. When read, the unimplemented bits are not defined.

The decoding of the three-bit page type field, together with the number of address bits the page types use or decode, is described in the table below.

Table 4-1: Physical Address Assignments

Type	Address	Device	Wait States
0	23-bit	P2 Memory	
	[0x000000]	Physical Memory 1-4 Mbytes	0
	[0x700000]	Black-and-white frame buffer	0 (Write), 4-8 (Read)
	[0x780000]	Keyboard and mouse UART	0 (Write), 4-8 (Read)
	[0x781800]	Video control register	0 (Write), 4-8 (Read)
1	14-bit	On-board I/O	
	[0x000000]	EPROM	2
	[0x000800]	Reserved	
	[0x001000]	Encryption processor	2-8
	[0x001800]	Parallel port	2
	[0x002000]	Serial port	2
	[0x002800]	Timer	2
	[0x003800]	Real-time clock	12
2	20-bit	P1 bus memory	
	[0x000000]	0 - 1 Mbyte 796-bus memory space	2 + device access time
3	20-bit	P1 bus I/O	
	[0x000000]	0 - 1 Mbyte 796-bus memory space	2 + device access time

Note: accesses to the Multibus incur at least an additional 2 wait states access time if the bus mastership must be acquired.

4.4. MMU Implementation

The Memory Management Unit is composed of the following:

Device	Type	Location
User Context Register	LS2518	U300
System Context Register	LS2518	U301
User/System CX Multiplexor	74F257	U302
Segment Map	2168	U303, U304
Segment Map R/W Buffer	AM2949	U314
Page Map	2168	U305, U306, U307, U308, U309, U310
Page Map R/W Buffers	AM2949	U317, U318, U319, U320
Protection Decoder	74F151	U315
Statistic Bit Logic	P16R4	U316
ID PROM	74S288	U411
Bus Error Register	74LS534	U412
System Enable Register	74LS273	U413
System Enable Readback	74LS244	U414
Diagnostic Register	74LS273	U802

Accesses to these devices are decoded via MMU strobe decoder **74ALS138:U322, U323, U324** in MMU Function Code, as decoded by Function Code Decoder **74ALS138:U321.**

4.5. MMU Summary

page size:	2K bytes
segment size:	32K bytes
process size:	16M bytes
# of contexts:	8
# of segments per context:	512
# of pages per segment:	16
# of PMEGs:	256
# of pages total:	4096
# of segments total:	4096

Chapter 5

On-board I/O Devices (I/O Space)

The I/O space (or device space) of the Sun-2 architecture contains the devices accessed by the CPU with data or program space instructions. These devices include main memory, the system bus, I/O devices, and so on. All elements of device space are accessed through the MMU. This allows all devices to be protected, shared, and managed in a uniform manner in a multi-process environment.

The on-board I/O devices are as follows:

Device	Type	Location
Boot PROMs	27128/27256	U406, U407
Data Ciphering Processor	9518	U601
Timer Controller	9513	U604
Input Port	74LS244	U800, U801
Serial Communication Controller	Z8530A	U605
Real-Time Clock	58167	U420

Accesses to these I/O devices are decoded via type decoder 74F138:U400, read decoder 74ALS138:U401, and write decoder 74ALS138:U402. The only exception to this is the decoding of the PROMs for supervisor program accesses in boot state. These cycles enable the PROM via gates 74F32:U409 and 74LS08:U408.

5.1. ID PROM

The purpose of the ID PROM is to provide basic information on the machine type and a unique serial number for software licensing, distribution, and access. In addition, the ID PROM stores the Ethernet® address, the date of manufacturing, and a checksum.

The ID PROM is implemented as a 32-byte PROM mapped as shown in the following table.

Register	Address	Size	Type
ID PROM 0	0x0008	byte	read only
ID PROM 1	0x0808	byte	read only
ID PROM 2	0x1008	byte	read only
...	. . .	...	...
ID PROM 31	0xf808	byte	read only

The content of the ID PROM is as follows:

Entry	Field	
(1)	Format	1 byte
(2)	Machine Type	1 byte
(3)	Ethernet address	6 bytes
(4)	Date	4 bytes
(5)	Serial number	3 bytes
(6)	Checksum	1 byte
(7)	Reserved	16 bytes

Explanation:

(1) Format — the format of the ID PROM.

(2) Machine Type — a number specifying an implementation of the architecture.

(3) Ethernet address — the unique 48-bit ethernet address assigned by Sun to this machine. The ethernet address stored in the ID PROM is the primary ethernet address of the CPU, replacing any additional ethernet addresses that might be stored on peripheral boards.

(4) Date — the date the ID PROM was generated. It is in the form of a 32-bit long word which contains the number of seconds since January 1, 1970.

(5) Serial number — a three-byte serial number.

(6) Checksum — defined so that the longitudinal XOR of the first 16 bytes of the PROM including the checksum yields 0.

(7) Reserved — for future expansion.

5.2. Diagnostic Register

The diagnostic register drives an eight-bit LED for displaying error messages. Although the diagnostic register is a word device, only bits 0 through 7 are actually displayed. Bits 8 through 15 are unused. A "0" bit written will cause the corresponding LED to light up, a "1" bit to be dark. Upon power-on reset, the diagnostic register is initialized to 0 causing all LEDs to light up. The no-fault state is defined to be all ones, with no LEDs lit up.

Register	Address	Data	Type
Diagnostic LED	0xA	Word	Write only
Initialization:	none		

5.3. Bus Error Register

When a bus error occurs, the bus error register latches the cause of the bus error to allow software to identify the source of the error. The bus error register always latches the cause of the most recent bus error. In case of multiple bus errors, the information relating to the earlier bus errors is lost.

The bus error register is a read-only register. It is not initialized or cleared upon reset. Without a corresponding bus error, the content of the bus error register is undefined.

Register	Address	Data	Type
Bus Error	0xC	Word	Read only
Initialization:	none		

The fields of the bus error register are defined as follows:

Bit	Name	Meaning
D00	PARERRL	Parity error low byte (D00-D07)
D01	PARERRU	Parity error upper byte (D08-D15)
D02	TIMEOUT	Timeout error
D03	PROTERR	Protection error
D04	AEN–	P1 access
D05		High
D06		High
D07	PAGEVALID	1 = valid page, 0 =invalid
D08..D15	reserved	

In more detail, the bus error conditions are as follows:

- Page invalid (PAGEVALID=0) means that the page referenced was not tagged as a valid page in the MMU.
- Protection error (PROTERR) means that the page protection logic did not allow the kind of operation attempted.
- Parity errors (PARERRL and PARERRU) can occur only during P2 memory (page type 0) accesses. Since parity errors are detected too late in the cycle to abort the current cycle, they abort the following cycle instead. If the following CPU cycle does not recognize bus errors then the parity error will abort the next cycle that does recognize bus errors. CPU

cycles that do not recognize bus errors include CPU accesses to the MMU, interrupt acknowledge cycles, trap cycles, and supervisor program accesses in boot state. In any event, the address at which the CPU receives the bus error is unrelated to the address of the parity error, which is not available.

- Timeout results from a non-completed reference. This can occur when accessing non-existent devices on cycles that use a positive handshaking mechanism. The bus error address can be used to determine which device did not respond.

5.4. System Enable Register

The system enable register enables system facilities, provides soft interrupts, and controls booting. The system enable register can be read and written under software control and is cleared on power up (hardware reset) and watchdog reset, but not upon CPU reset. Bits are assigned as follows:

Register	Address	Data	Type
System Enable	OxE	Word	Read or write
Initialization:	cleared on power-on reset		
Interrupt:	level 1, 2, and 3, autovector		

Figure 5-1: System Enable Register Fields

Bit	Name	Meaning
D00	EN.PAR	Enable parity generation Generate bad parity (for testing)
D01	EN.INT1	Autovector interrupt on level 1
D02	EN.INT2	Autovector interrupt on level 2
D03	EN.INT3	Autovector interrupt on level 3
D04	EN.PARERR	Enable parity error reporting
D05	EN.DVMA	Enable Direct Virtual Memory Access
D06	EN.INT	Enable all interrupts
D07	BOOT–	Boot state (0 = boot, 1 = normal)
D08..D15	reserved	

When cleared after power-on or watchdog reset, all bits are initialized to zero. In this state, boot state is active, parity generation and reporting are disabled; DVMA, soft interrupts and all other interrupts are disabled.

The EN.INT fields cause interrupts on the corresponding level. For example, an interrupt request caused by an EN.INT bit stays active until software clears the corresponding bit.

Upon power-on reset or watchdog reset, the system enable register is cleared, forcing boot state active and disabling all interrupts and parity errors. Boot state forces all supervisor program fetches to access the on-board boot PROM device independently from the setting of the MMU.

All other types of references are unaffected and will be mapped as during normal operation of the processor.

5.5. Boot PROMs

The boot PROM I/O device is a pair of 28-pin sockets for 128K or 256K boot PROMs. Unlike all other devices, the boot PROM is addressed directly with the low-order, non-translated (virtual) address bits from the CPU. Thus, even though each 2K page must be enabled with its own entry in the page map, the physical page number in the page map is ignored and the low-order bits of the virtual address are used instead.

Register	Address	Data	Type	
Word 0	0	Word	Read only	
Word 1	2	Word	Read only	
....				
Word 0x3FFF	0x7FFE	Word	Read only	(27128s)
Word 0x7FFF	0xFFFF	Word	Read only	(27256s)
Reference:	none			
Initialization:	none			
Interrupt:	none			

The boot PROM device is also accessed in boot state. In boot state, all supervisor program fetches are forced from the boot PROM device, independently from the setting of the MMU.

5.6. Parallel Port

The parallel port is a non-latching, 16-bit input port. Since the input data is non-latched, the data may change during the moment it is being read. For best results, the data should be re-read until stable data is obtained.

Register	Address	Data	Type
Parallel Port	0	Word	Read only
Initialization:	none		
Interrupt:	none		
Reference:	none		

5.7. Serial Ports

Serial ports are implemented with the Zilog 8530A SCC (serial communications controller). The SCC features two high-speed, fully-symmetrical, and highly-programmable serial channels with built-in baud-rate generators. The clock input to the SCC is a 4.9152-MHz clock, derived from

the basic system clock C100. The SCC is mapped as follows:

Register	Address	Data	Type
CH B control	0	Byte	Read or write
CH B data	2	Byte	Read or write
CH A control	4	Byte	Read or write
CH A data	6	Byte	Read or write

Interrupt:	level 6 autovector
Initialization:	needs to be initialized in software
Reference:	Zilog 8530 SCC data sheet
Recovery Time:	1.6 microseconds

5.8. Timer

An AMD 9513 timer chip with five 16-bit timers is provided. The clock input to the 9513 is a 4.9152-MHz clock, derived from the basic system clock. The 9513 GATE1 input is wired to the 9513 FOUT output. The timer is mapped as follows:

Register	Address	Data	Type
Timer data	0	Word	Read or write
Timer Command	2	Word	Read or write

Interrupt:	level 7 for timer 1, level 5 for timer 2-5, autovector
Initialization:	internal reset whenever power suppy drops below 3.0V
Reference:	AMD 9513 programming book

Note the synchronization requirements of the 9513 timer. Before writing into a counter, the counter's clock source must be disabled first.

Initialization of the 9513 timer is special in that the chip has an on-chip power-on reset that initializes the chip whenever the power supply voltage is less than 3V. The chip is not affected by power-on resets, watchdog resets, or CPU resets.

5.9. Encryption Processor

The encryption processor is an AMD 9518/8068 data ciphering processor providing high-speed NBS DES encryption. To access an internal register in the 9518/8068, the address register must be written first. Once the address register is set up, the selected register can be accessed repeatedly. The encryption processor is mapped as follows:

Register	Address	Data	Type
Data register	0	Byte	Read or write
Address register	2	Byte	Write only

Interrupt:	none
Initialization:	none
Reference:	AMD 9518/8068 data sheet
Recovery time:	1.6 microseconds (minimum time between successive accesses)

5.10. Real-Time Clock

The real-time clock maintains the time of day and the date. A battery powers the clock when the main power is off. The real-time clock is based on the National 58167 chip which is addressed as 32 registers.

Register	Address	Data	Type
Register 0	0	Byte	Read or write
Register 1	2	Byte	Read or write
....			Read or write
Register 0x1F	0x3E	Byte	Read or write

Interrupt:	none
Initialization:	none
Reference:	National Semiconductor 58167 data sheet

Chapter 6

P2 Bus Interface

6.1. P2 Bus Description

The P2 bus connects the Sun CPU board to Sun memory boards and the Sun video board. It is a single-master bus; only one CPU board can be connected to it. It is a synchronous bus; all timing is generated by the processor board. The protocol does not provide for a handshaking capability in the traditonal sense, rather it provides a negative acknowledge or "WAIT" capability that can hold the current cycle until it is deasserted. In its signals and timings, the P2 bus follows very closely the characteristics of 64k dynamic RAMs.

6.2. P2 bus Decoding

Note that 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. Illegal page reference accesses are turned into read cycles.

6.3. Parity Error Logic

The P2 bus and the memory boards on the P2 bus are equipped with byte parity. Parity errors abort the 68010 via the bus error mechanism.

All bus errors except parity errors are recognized by the 68010 in the same 68010 cycle as they occur. Parity errors cannot be recognized in the same cycle because they are only detected at the very end of the cycle, when it is too late to abort the current cycle. Since they are not recognized in the current cycle, parity errors need to be latched until they are recognized by the CPU. This is done in the parity error flip-flop 74F74:U512, providing a separate flip-flop for upper and lower parity errors, at the end of a memory read cycle.

In order to recognize a bus error, the 68010 must execute a "non-disabled" cycle or a cycle in which C.S4 is asserted. The parity error flip-flop will be cleared if BERR is true and DS is deasserted, which indicates a CPU cycle terminated by BERR. Note that the BERR will not be set on non-CPU cycles because P.BACK will keep the bus error flip-flop 74F74:U105 in its idle state.

Parity error reporting can be disabled without affecting system operation. To initialize parity in main memory, all of memory needs to be written with parity generation enabled. Separate enable bits are provided for parity generation and parity checking to allow software testing of the parity function.

6.3.1. P2 Bus Signal Definition

The Sun P2 bus consists of the following signals:

Signal Name	Type	Description
P2.A0..7	O	Multiplexed address lines that transmit the row-address during the leading edge of RAS and the column address during the leading edge of CAS
P2.A17..22	O	High-order address bits, valid at leading edge of CAS
P2.DI0..15	O	Data lines from processor to memory
P2.DIL	O	Lower Byte Parity from processor to memory
P2.DIU	O	Upper Byte Parity from processor to memory
P2.DO0..15	I	Data lines from memory to processor
P2.DOL	I	Lower Byte Parity from memory to processor
P2.DOU	I	Upper Byte Parity from memory to processor
P2.R/W–	O	Read/Write– Signal
P2.REN–	O	Refresh Enable, current cycle is a refresh cycle
P2.RAS–	O	Row Address Strobe
P2.CAS–	O	Column Address Strobe
P2.WEL–	O	Lower Byte Write Strobe
P2.WEU–	O	Upper Byte Write Strobe
P2.WAIT–	I	Wait line, holds current cycle until deasserted

Type O means Output (signal direction from Processor to Memory)
Type I means Input (signal direction from Memory to Processor)

6.3.2. P2 Bus Timing Diagram

```
Semantics:
                x := Signal Unstable
                = := Signal Stable
                - := High Level Signal
                _ := Low Level Signal


P2.AO..7        xxxxx====xxxxx================xxxxxxx

P2.A17..22      xxxxxxxxxxxxxx================xxxxxxx

P2.R/W-         xxxxx========================xxxxxxx

P2.REF-         xxxxx========================xxxxxxx

P2.DOO..15      xxxxxxxxxxxxxxxxxxxxxxxxx====xxxxxxx

P2.DIO..15      xxxxxxxxxxxxxxxxxxx==========xxxxxxx


P2.RAS-         -------_______________________-------

P2.CAS-         -------------_________________-------

P2.WEL-         --------------------------____-------

P2.WEU-         --------------------------____-------
```

Table 6-1: Pin Assignments on the P2 Connector

Pin	*Component Side*		*Pin*	*Circuit Side*	
	Mnemonic	*Description*		*Mnemonic*	*Description*
1	A09	Address bit 9	2	A18	Address bit 18
3	A19	Address bit 19	4	A20	Address bit 20
5	CAS–	Column Address Strobe	6	RAS–	Row Address Strobe
7	WAIT–	DTACK WAIT signal	8	WEL–	Write Enable — lower byte
9	DI00	Data In 0	10	DO00	Data Out 0
11	DI01	Data In bit 1	12	DO01	Data Out bit 1
13	A01	Address bit 1	14	A21	Address bit 21
15	DI02	Data In bit 2	16	DO02	Data Out bit 2
17	DI03	Data In bit 3	18	DO03	Data Out bit 3
19	A02	Address bit 2	20	A22	Address bit 22
21	DI04	Data In bit 4	22	DO04	Data Out bit 4
23	DI05	Data In bit 5	24	DO05	Data Out bit 5
25	A03	Address bit 3	26	REFR–	Refresh
27	DI06	Data In bit 6	28	DO06	Data Out bit 6
29	DI07	Data In bit 7	30	DO07	Data Out bit 7
31	A04	Address bit 4	32	GND	Signal GND
33	DIL	Parity In — lower byte	34	DOL	Parity Out — lower byte
35	DIU	Parity In — upper byte	36	DOU	Parity Out — upper byte
37	A05	Address bit 5	38	GND	Signal GND
39	DI08	Data In bit 8	40	DO08	Data Out bit 8
41	DI09	Data In bit 9	42	DO09	Data Out bit 9
43	A06	Address bit 6	44	GND	Signal GND
45	DI10	Data In bit 10	46	DO10	Data Out bit 10
47	DI11	Data In bit 11	48	DO11	Data Out bit 11
49	A07	Address bit 7	50	R/W–	Read — high, Write — low
51	DI12	Data In bit 12	52	DO12	Data Out bit 12
53	DI13	Data In bit 13	54	DO13	Data Out bit 13
55	A08	Address bit 8	56	WEU–	Write Enable — upper byte
57	DI14	Data In bit 14	58	DO14	Data Out bit 14
59	DI15	Data In bit 15	60	DO15	Data Out bit 15

Chapter 7

P1 Bus Interface

7.1. Bus Master Interface

The 120 CPU board supports the IEEE 796-Bus (Multibus) standard. The CPU can be either a master or slave as needed, but not both at once. The P1 bus is used primarily for supporting I/O devices, but there is nothing in the implementation to prevent it from being used for other purposes.

7.2. Compliance

With one known exception, the P1 interface complies completely with the IEEE 796-bus specification. The exception is the length of the P1.INIT– assertion time. When the RESET instruction is executed by the microprocessor configured as CPU0, the length of the P1.INIT– pulse is only 12.8 microseconds. The specification calls for a minimum pulse of five (5) milliseconds. P1.INIT– meets all other conditions of the specification.

The P1 bus provides for multiple masters via the standard multimaster bus exchange arbitration scheme. The P1, as implemented, is a mode 2 master. Mode 2 masters are unlimited in bus control. Bus timeouts are allowed, and conformance with the maximum busy period is not required. A consequence of this is that there is no guaranteed latency for bus acquisition.

The arbitration is configured for both the serial and parallel priority schemes, although the 120 backplane supports only the parallel scheme.

The following is the formal notation for indicating the level of compliance in the other areas:

Table 7-1: Multibus Compliance

Area	Level	Meaning
Data Path	D16	8 or 16-bit data path
Memory Address Path	M20	20-bit memory address path
I/O Address Path	I16	8 or 16-bit I/O address path
Interrupt Attributes	V0	Non-bus-vectored interrupt requests
	L	Level triggering

One consequence of the way interrupts are implemented in the 120, is that the priority scheme is backwards from P1 to CPU. The highest priority P1 interrupt is mapped into the lowest 68010 autovector, and the lowest priority P1 interrupt is mapped into non-maskable interrupt (NMI). This is not a problem because a system can still be configured with the relative levels spelled out.

7.3. P1 Bus Address and Data Drivers

Inverting flow-through latches 74LS533:U700, U701, and U702 latches 68010 address bits P.A1...P.A10 and translated address bits MA11..MA19 during clock C.S3-6. In addition, IOLDS and IOUDS are latched to form P1.BHEN, and P.LDS– is latched to form P1.A0. (See Table 7-2.)

Inverting bidirectional drivers 74LS640:U712, U713, and U714 exchange data bits D0...D15 between the on-board data bus and the P1 data bus. Buffer 74LS640:U714 is the byte swap buffer that is enabled whenever the 68010 or DVMA performs a byte-swap transfer. Both address and data buffers are enabled with AEN– from bus arbiter P16R4A:U718 indicating P1 bus mastership.

7.4. Byte order and A0 Address Generation

The Sun-2 CPU Board uses 68010 byte order on the P1 bus to offer a consistent memory model for the 68010. Notice that the 68010 byte order is incompatible with the P1-bus byte order: the 68010 numbers the *upper* byte (Data bits 8 through 15) the even byte, whereas the P1 bus calls the *lower byte* (Data bits 0 thru 7) the even byte.

	D15 .....D8	D7 .....D0
68010 Byte Order	Byte 0	Byte 1
P1 Bus Byte Order	Byte 1	Byte 0

In addition, the 68010 differs from the P1 bus in that it transfers even bytes on data lines D8 through D15 and odd bytes on D0 through D7. The P1 bus transfers both odd and even bytes via D0 through D7. Finally, the 68010 does not output address bit A0. This address bit needs to be reconstructed from the 68010 data strobes. To achieve 68010 byte-order on the P1 bus, the A0 on the P1 bus must be the inverse of 68010 A0 for byte transfers.

The following table summarizes the state of different signal lines for word and byte transfers between 68010 and P1 bus.

Table 7-2: P1 Buffer Logic

68010 Transfer		68010	68010 P.UDS–	68010 P.LDS–	P1-Bus P.A0	Word P1.A0	Byte Buffer	P1-Bus Buffer
16-bit	D0..15	0	0	0	0	0	1	D0..15
8-bit	D0..7	1	0	1	0	0	1	D0..7
8-bit	D8..15	0	1	0	1	1	0	D0..7

From the table it can be seen that Word Buffer = 68010 P.LDS– and that P1.A0 = P.LDS–.

7.5. P1-Bus Multimaster Logic

The P1 bus provides for multiple masters on the bus, exchanging bus mastership via a standard protocol (see IEEE-796 Bus standard). The Sun 2 CPU board uses a 16R4A Pal (U718) as an arbiter to implement this protocol.

The arbiter works as follows. The arbiter remains in the idle state until the processor requests the bus by asserting SYSB. SYSB is syncronized to the bus arbitration clock P1.BCLK–. This is necessary since one can select an alternate source for P1.BCLK– by pulling J701 (1-2). The arbiter then requests the bus by asserting P1.CBRQ– and P1.BREQ–. The arbiter will remain in the request bus state until it receives control of the bus, which is signified by P1.BPRN– being asserted, and P1.BUSY– being deasserted by the previous bus master. If the 68010 reruns the bus cycle, due to refresh or P1 deadlock, the arbiter will keep the request for the bus in, until it receives the bus. It will then hold the bus till the 68010 reruns the cycle. When the arbiter receives bus mastership, it asserts AEN–, and P1.BUSY– to indicate that the bus is currently in use. It will retain bus mastership until another bus master requests the bus via P1.BPRN– or P1.CBRQ–. This will minimize the transfer time for sucessive P1 bus accesses, since the arbitration delay time is potentially eliminated.

When the arbiter obtains mastership it asserts AEN–. This enables the address drivers immediately and the data drivers with DATAEN– or P20L10:U212. The driver for the bus control signals, 74F244:U717 is enabled after a minimum 100-nsec delay created by synchronizer 74F74:U709, but no earlier than processor state C.S7 since C.S4 must be active before the synchronizer can recognize AEN–. This is necessary because on write cycles 68010 signals P.LDS– and P.UDS– are only valid at the beginning of state S5.

DATAEN– will be deasserted, which disables the data drivers, as soon as SYSB is deasserted. AEN– will be deasserted after SYSB is synchronized to the arbiter.

7.6. P1 Bus Clocks

The Sun 68010 CPU board normally generates P1-bus BCLK and CCLK via driver 74F244:U717. In a multimaster system, only one master may drive these clocks. To configure the Sun 68010 board for such a system, BCLK can be disconnected by removing jumper J701-2 and CCLK can be

disconnected by removing jumper J703-4.

7.6.1. P1 Address Map

Table 7-3: Sun-2 Multibus Memory Map

Address	Device
0x0000	DVMA Space
	.
	.(256 Kbytes)
	.
0x03f800	DVMA Space
0x40000	Sun Ethernet Memory (#1)
	.
	.(256Kbytes)
	.
0x7f800	Sun Ethernet Memory (#1)
0x80000	SCSI (#1)
	.
	.(16 Kbytes)
	.
0x83800	SCSI (#1)
0x84000	SCSI (#2)
	.
	.(16Kbytes)
	.
0x87800	SCSI (#2)
0x88000	Sun Ethernet Control Info (#1)
	.
	.(16Kbytes)
	.
0x8b800	Sun Ethernet Control Info (#1)
0x8c000	Sun Ethernet Control Info (#2)
	.
	.(16Kbytes)
	.
0x8f800	Sun Ethernet Control Info (#2)
0x90000	--- FREE ---
	.
	.(64 Kbytes)
	.
0x9f800	--- FREE ---
0xa0000	Sun Ethernet Memory (#2)
	.
	.(64 Kbytes)
	.
0xaf800	Sun Ethernet Memory (#2)
0xb0000	--- FREE ---
	.
	.(64 Kbytes)
	.
0xbf800	--- FREE ---

Address	Device
0xc0000	Sun Frame Buffer
	.
	.(128 Kbytes)
	.
0xdf800	Sun Frame Buffer
0x30000	3COM Ethernet (#1)
0xe0800	3COM Ethernet (#1)
0xe1000	3COM Ethernet (#1)
0xe1800	3COM Ethernet (#1)
0xe2000	3COM Ethernet (#2)
0xe2800	3COM Ethernet (#2)
0xe3000	3COM Ethernet (#2)
0xe3800	3COM Ethernet (#2)
0xe4000	--- FREE ---
	.
	.(16Kbytes)
	.
0xe7c00	--- FREE ---
0xe8000	Sun Color
	.
	.(64 Kbytes)
	.
0xf7800	Sun Color
0xf8000	--- FREE ---
	.
	.(16 Kbytes)
	.
0xff800	--- FREE ---

Chapter 8

DVMA Operation

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:

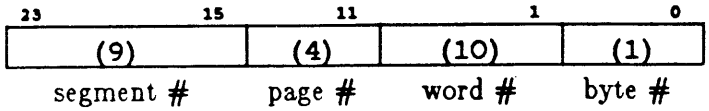
- Refresh
- P1 bus

8.1. Direct Virtual Memory Access (DVMA)

Input/output devices capable of direct memory access are implemented in the Sun-2 architecture with direct virtual memory access, or DVMA. DVMA means that bus masters use virtual addresses rather than physical addresses to access their target device, typically memory.

DVMA translates and protects all accesses identically. This avoids the dual mapping problems associated with physical address DMA in a virtual memory environment.

The following table summarizes virtual address layout.



DVMA is implemented as follows:

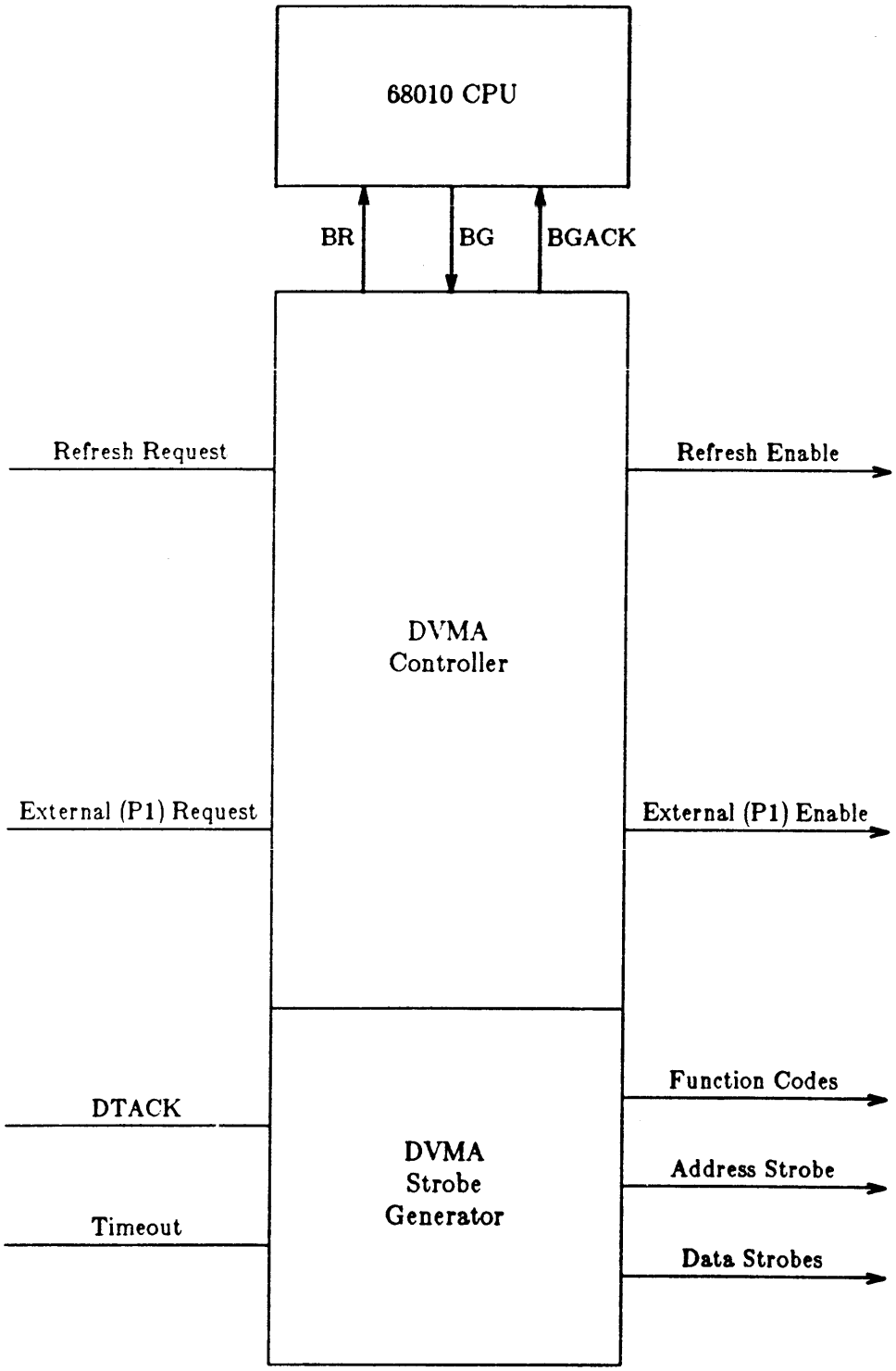
- Address space. DVMA accesses are performed as data read-write operations in the supervisor function code.
- Protection. Protection applies to DVMA the same way as to the CPU. The supervisor read or write capability in the page map has to be enabled to allow the corresponding type of access. If the respective capability is not set, the attempted DVMA cycle is aborted.
- Parity errors. DVMA read cycles that cause parity errors are aborted.
- Statistics bits. The update and modify bits are set on successfully executed DVMA cycles the same way as on CPU cycles.
- Deadlock. For DVMA devices that can cause deadlock with the CPU, such as a CPU access to the system bus conflicting with a DVMA access from the system bus, deadlock is resolved by rerunning the CPU cycle.
- Self-reference. DVMA cycles that are self-referential, such as a system bus DVMA transfer attempting to reference the system bus, will be aborted.

- Error handling. When a DVMA cycle is aborted, the error is signalled to the controlling master for error handling. The master typically will stop transferring.

Further details and limitations of DVMA operation are described later in this chapter under each particular DVMA device.

The following figure shows how the DVMA Controller and Strobe Generator interface to the 68010.

Figure 8-1: DVMA and Strobe Generator Interface to 68010



8.2. DVMA and Refresh

The Sun-2 CPU Board offers hardware refresh and DVMA operation from the P1 bus. These features are implemented primarily by Timer Controller P20X10:U211, Refresh Counter 74LS491:U210, DVMA Decoder P20L10:U212, DVMA Controller P16R4:U213, and DVMA Address Drivers 74LS244:U706, U707, and U708.

The following table illustrates the primary signals generated by these components and their enable conditions. CPU, REN, and XEN indicate a CPU, a Refresh, and a DVMA cycle, respectively.

Component	Enable	Signals
68010 CPU	CPU	P.AS–, P.LDS–, P.UDS–, P.R/W–
	CPU	P.FC0, P.FC1, P.FC2
	CPU	P.A1..P.A23
DVMA Controller U213	---	REN–, XEN–, BR–, BGACK–
	XEN,REN	P.AS–
	XEN	P.FC1
DVMA Decoder U212	XEN	P.LDS–, P.UDS–, P.R/W–
DVMA Address U705,U707,U708	XEN	P.A1..P.A23
	XEN	P.FC2
Refresh Counter U210	REN	P.A1..P.A11

8.3. Driving and Terminating 68010 Bus Signals

All three-statable 68010 signals are terminated via pull-ups R9.SIP:S103, S104, and S105. This causes these signals to assume a defined state when they are not driven, that is, when the 68010 is being reset or when bus mastership is exchanged between the 68010 and the DVMA controller.

During a *refresh* cycle, the 68010 bus signals are driven as follows:

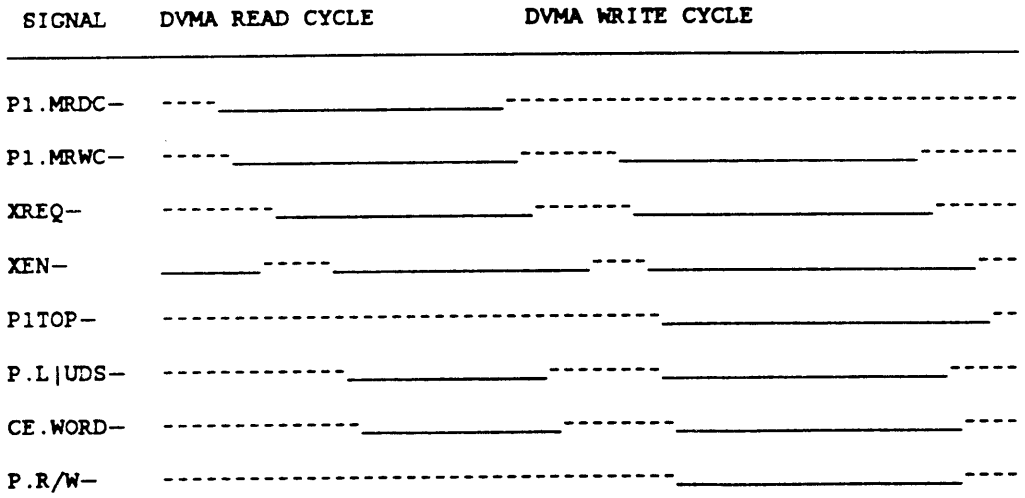
Signal	Driven by	
P.AS–	PAL	U213
P.UDS–, P.LDS–, P.R/W–	Pull-Up	S103
P.FC0, P.FC1, P.FC2	Pull-Up	S103
P.A1 through P.A10	Counter	U210
P.A11 through P.A23	Pull-Up	S104, S105

During a *DVMA* cycle, the 68010 bus signals are driven as follows:

Signal	Driven by	
P.AS–	PAL	U213
P.UDS–, P.LDS–, P.R/W–	PAL	U212
P.FC0	Pull-Up	S103
P.FC1	PAL	U213
P.FC2	Driver	U705
P.A1 through P.A23	Drivers	U705, U707, U708

8.4. P1 Bus DVMA Decoder

The DVMA Decoder P16L8:U212 recognizes P1 bus DVMA requests and generates the signals P.LDS–, P.UDS–, and P.R/W– during DVMA cycles. In addition, the DVMA Decoder controls the enable and the direction of the P1 bus data buffers LS640:U712, U713, and U714 for both DVMA cycles and CPU cycles via signals P1TOP–, CE.BYTE–, and CE.WORD–. See the following timing diagram for DVMA cycle timing.



8.4.1. DVMA Decoder Signals

DATAEN–, or data enable, is generated by bus arbiter P16R4A:U718 and signals bus mastership for the CPU Board. XEN–, or external enable, is driven from DVMA Controller P16R4:U213 and signals that an external cycle is enabled. XEN– enables the external address buffers 74LS533:U705, U707, and U708 and four control signals generated by the DVMA Decoder PAL: P.R/W–, P.LDS–, P.UDS– and P1.XACK–.

CE.WORD– enables the 16-bit data buffer between the processor and the P1 bus. It is asserted on DATAEN– word and low-byte transfers, on DATAEN– writes, and on XEN– word transfers. The DATAEN– write case guarantees data hold on the P1 bus if the write operation was directed to the P1 bus.

CE.BYTE– enables the eight-bit swap buffer between the processor and the P1 bus for upper byte DATAEN– read cycles from P1 to the processor and for upper byte XEN– transfers.

P1TOP– means enable the direction of the data bus buffers from P1 to the processor. It is asserted on non-XEN P1 memory read MRDC– and input/output read IORC cycles, as well as on XEN write cycles.

XREQ– is asserted when an external request is recognized and stays asserted while the external strobe P1.MRWC– is active. To recognize an external request the following conditions must be met: EN.DVMA asserted, P1.A19 and P1.A18 deasserted, P1.MRWC– active, and DATAEN– and XEN– deasserted. The XEN condition guarantees that no new XREQ– can become active while the XEN– associated with a previous request is still asserted.

The following signals are three-stated and only enabled to be driven by the DVMA decoder when XEN– is asserted.

P.R/W– is asserted on DVMA write cycles. The signal is latched before state C.S7 and held until P1.MRDC– comes true or until XEN– goes away.

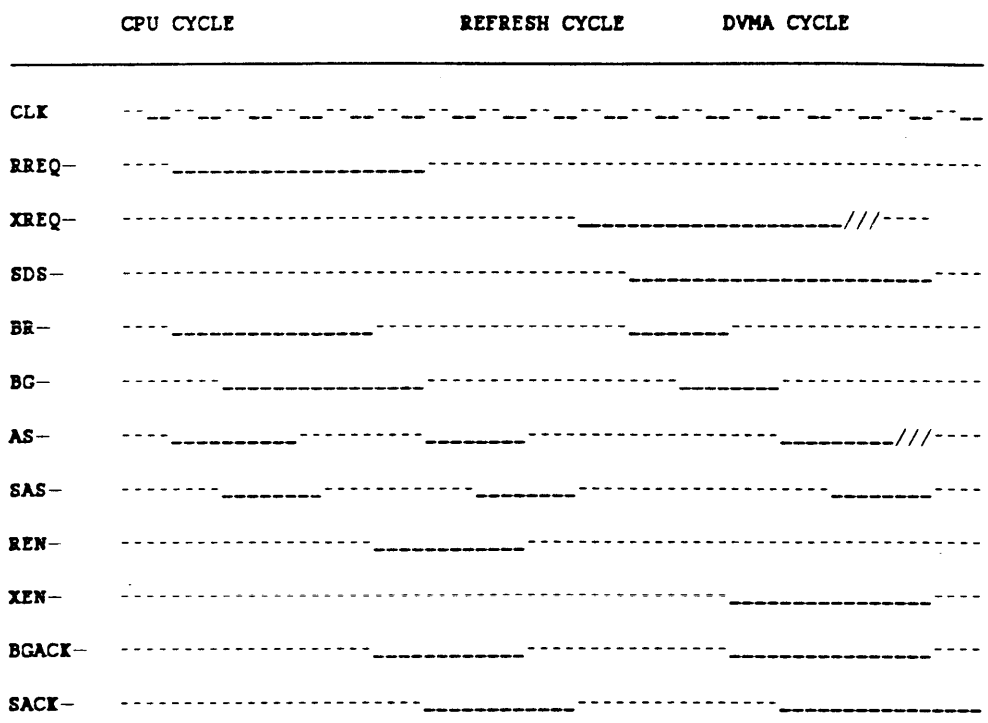
P.LDS– is asserted for even byte and word transfers.

P.UDS– is similar to P.LDS– except that it is asserted for odd byte and word transfers.

P1.XACK signals to the external DMA device that the on-board cycle successfully completed. This is the case when C.S7 is asserted, protection error PROTERR is false, and, in the case of read cycles, parity error PARERR is false.

8.5. DVMA Controller

The DVMA controller P16R4:U213 is at any time in one of three states: IDLE, REN, or XEN. REN state is active while executing refresh cycles, XEN state while executing P1-bus DVMA cycles. The state machine is in IDLE state if it is not in REN or XEN state.



CLK : is the 100-nanosecond clock to the DVMA Controller. All inputs to the DVMA controller state machine are synchronous to this clock except input XREQ- which is used asynchronously only.

RREQ- : indicates a refresh request from the timer controller P20X10:U211.

XREQ- : indicates an asynchronous external request from the DVMA decoder P20L10:U212.

SDS- : is the synchronized version of XREQ- via flip-flop 74F74:U207-0.

BR- : bus request, is asserted from the DVMA Controller to the 68010 when XREQ- or RREQ- is pending but BGACK- is inactive.

BGACK- and SACK- : when the machine enters state XEN or REN, it asserts BGACK- one PAL delay after entering the state. In the next state after BGACK- is asserted, the synchronous version of BGACK-, SACK-, is asserted. BGACK- stays asserted during the entire refresh or DMA cycle, and causes the AS- and FC1 to be three-state enabled. When XEN- or REN- is deasserted, BGACK- is deasserted one PAL delay later and then SACK- is deasserted one clock later.

AS- : asserted one PAL delay after SACK-. AS- remains asserted while in the REN state and SACK- is asserted, or while in XEN state and SACK- and XREQ- is asserted.

XEN state is entered when the state machine is in IDLE state, a GRANT is issued, no refresh request is pending, and synchronous data strobe or SDS- is pending. The state machine will stay in XEN state until SDS- goes away.

REN state is entered when the state machine is in IDLE state, a GRANT is issued, and a refresh request is pending. Note that if a refresh request and a synchronous data strobe are pending at the same time, refresh request will take priority over the synchronous data strobe. The state machine will stay in REN state for two additional states; the first while SACK- is not asserted, the second while SAS- is not asserted.

FC1 : driven low in XEN state. Since FC0 and FC2 are pulled up by external pull-up resistors, the effective function code in XEN state is five, or supervisor data. FC1 is driven high in REN

state thus the effective function code for REN state is seven, or supervisor-reserved.

XBERR : external bus error, is asserted if SDS–, XHALT, and SYSB are active. This condition indicates that the CPU is attempting to access the P1 bus while a DMA device is attempting to access the processor bus. XBERR will stay asserted until SYSB becomes inactive.

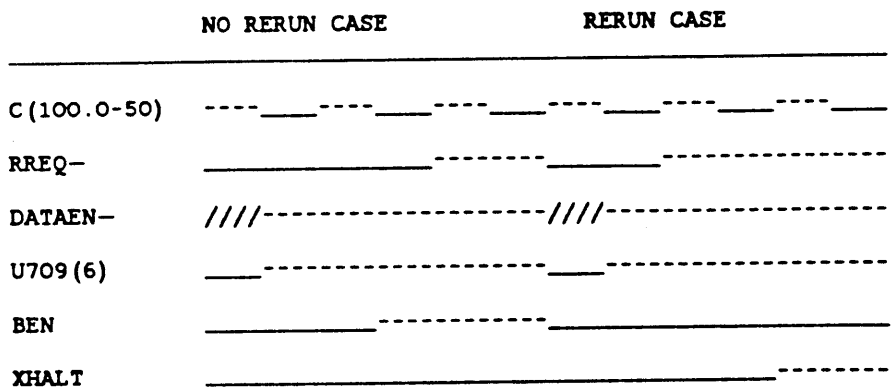
XHALT : external halt, is asserted if SDS– is active and SYSB is active, indicating CPU access to the P1 bus. It will stay asserted while XBERR is asserted.

8.6. Rerun Conditions

CPU cycles are rerun under two conditions: bus deadlock and refresh deadlock. Bus deadlock occurs when the CPU attempts to access the P1 bus while a master on the P1 bus attempts to access the CPU board via DVMA. Refresh deadlock occurs if a refresh request is pending while the CPU is waiting for P1-bus access.

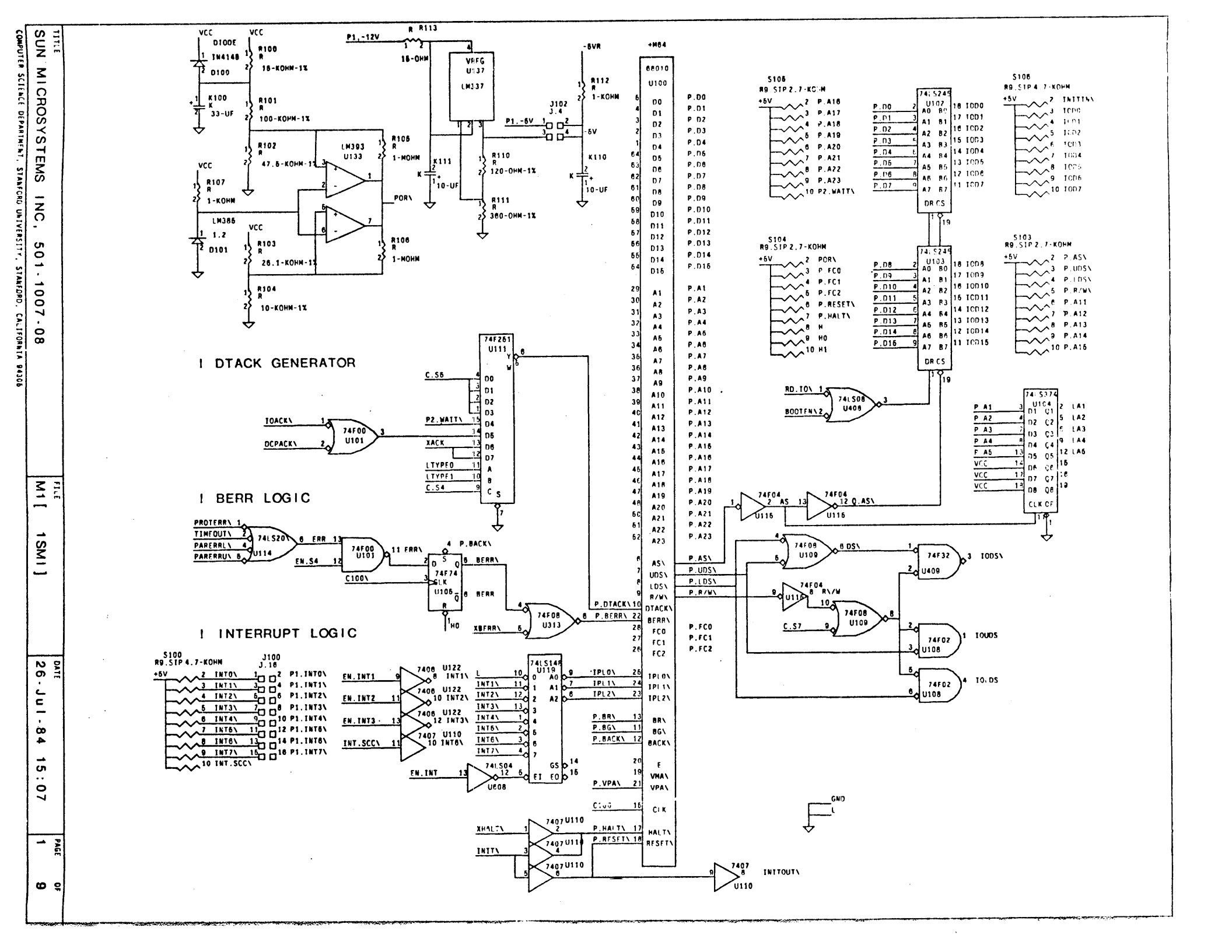
In both cases, the current CPU cycle is aborted via the 68010 bus cycle rerun mechanism. This is done by asserting HALT and BERR and keeping them asserted until the current 68010 bus cycle is terminated. The 68010 then performs normal bus arbitration, letting the pending DVMA or refresh cycle proceed, and after regaining the bus, will retry the previously-aborted cycle.

Refresh deadlock cycles will only be rerun if signal BEN has not been asserted yet. BEN enables the P1 bus strobes. If BEN is already active, rerun is no longer possible because P1 bus cycles, once begun, cannot be restarted. However, if BEN is not yet asserted and the rerun condition is true then BEN will not be asserted subsequently. This is guaranteed because the rerun condition, caused by signal RREQ–, is simultaneously recognized by DVMA controller Q-U213 and inhibits assertion of enable flip-flop 74F74:U709 via gate 74F00:U101. Also, RREQ– will stay asserted until after C.S4 is deasserted, clearing enable flip-flop 74F74:U709 via gate 74F08:U703. The timing diagram below illustrates this exchange.



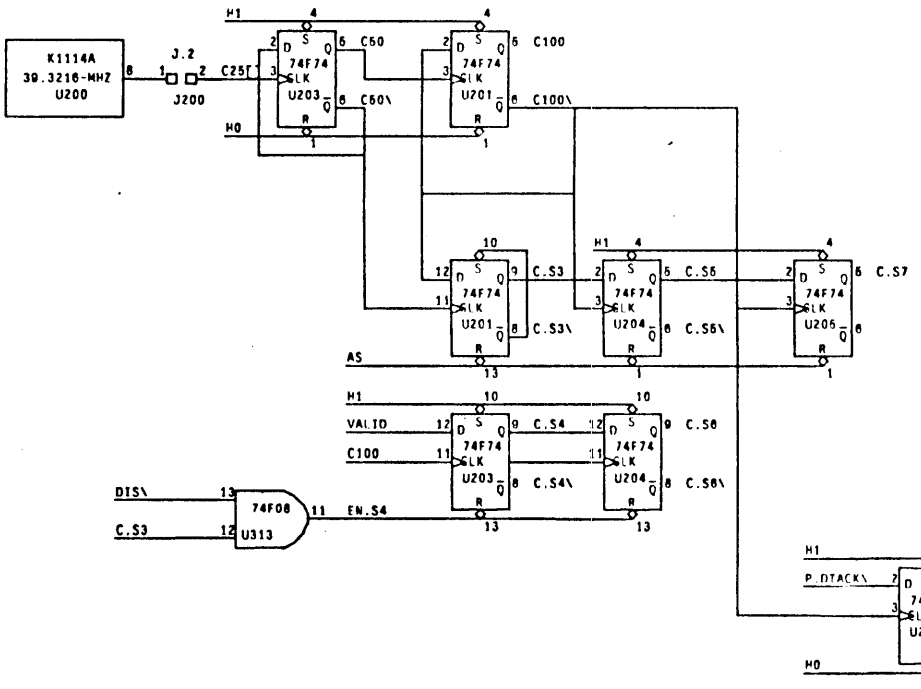
Appendix A

CPU Board Schematics

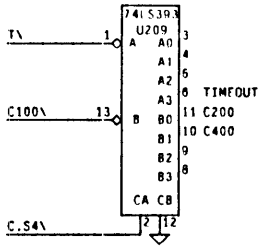

TITLE
SUN MICROSYSTEMS INC. 501-1007-08
FILE
M1[1SM1]
DATE
26-Jul-84 15:07
PAGE
1 OF 9
COMPUTER SCIENCE DEPARTMENT, STANFORD UNIVERSITY, STANFORD, CALIFORNIA 94305
! DTACK GENERATOR
! BERR LOGIC
! INTERRUPT LOGIC
VCC
D100E
IN4148
D100
R100
18-KOHM-1%
K100
33-UF
R101
100-KOHM-1%
R102
47.5-KOHM-1%
R107
1-KOHM
LM386
1.2
D101
R103
26.1-KOHM-1%
R104
10-KOHM-1%
LM393
U133
R105
1-MOHM
R106
1-MOHM
POR\
P1.-12V
R R113
18-OHM
VREG
U137
LM337
K111
10-UF
R110
120-OHM-1%
R111
360-OHM-1%
J102
J.4
P1.-5V
-5V
R112
1-KOHM
K110
10-UF
-5VR
+M64
68010
U100
P.D0
P.D1
P.D2
P.D3
P.D4
P.D5
P.D6
P.D7
P.D8
P.D9
P.D10
P.D11
P.D12
P.D13
P.D14
P.D15
P.A1
P.A2
P.A3
P.A4
P.A5
P.A6
P.A7
P.A8
P.A9
P.A10
P.A11
P.A12
P.A13
P.A14
P.A15
P.A16
P.A17
P.A18
P.A19
P.A20
P.A21
P.A22
P.A23
AS\
UDS\
LDS\
R/W\
DTACK\
BERR\
FC0
FC1
FC2
IPL0\
IPL1\
IPL2\
BR\
BG\
BACK\
E
VMA\
VPA\
CLK
HALT\
RESET\
S108
R9.SIP2.7-KOHM
P.A16
P.A17
P.A18
P.A19
P.A20
P.A21
P.A22
P.A23
P2.WAIT\
74LS245
U107
DR CS
IOD0
IOD1
IOD2
IOD3
IOD4
IOD5
IOD6
IOD7
S106
R9.SIP4.7-KOHM
INTTIN\
S104
R9.SIP2.7-KOHM
POR\
P.FC0
P.FC1
P.FC2
P.RESET\
P.HALT\
H
H0
H1
74LS245
U103
IOD8
IOD9
IOD10
IOD11
IOD12
IOD13
IOD14
IOD15
S103
R9.SIP2.7-KOHM
P.AS\
P.UDS\
P.LDS\
P.R/W\
P.A11
P.A12
P.A13
P.A14
P.A15
RD.IO\
BOOTFN\
74LS08
U408
74LS374
U104
LA1
LA2
LA3
LA4
LA6
74F261
U111
C.S6
IOACK\
DCPACK\
74F00
U101
XACK
LTYPE0
LTYPE1
C.S4
PROTERR\
TIMEOUT\
PARERRL\
PARERRU\
74LS20
U114
ERR
EN.S4
C100\
74F74
U106
P.BACK\
BERR
HO
XBERR\
74F08
U313
P.DTACK\
74F04
U116
74F08
U109
74F32
U409
74F02
U108
IODS\
IOUDS
IOLDS
EN.INT1
EN.INT2
EN.INT3
INT.SCC\
7406
U122
7407
U110
74LS148
U119
INT1\
INT2\
INT3\
INT4\
INT5\
INT6\
INT7\
IPL0\
IPL1\
IPL2\
74LS04
U608
EN.INT
P.BR\
P.BG\
P.BACK\
P.VPA\
XHALT\
INIT\
P.HALT\
P.RESET\
INTTOUT\
GND
S100
R9.SIP4.7-KOHM
J100
J.16
INT0\
INT1\
INT2\
INT3\
INT4\
INT5\
INT6\
INT7\
INT.SCC\
P1.INT0\
P1.INT1\
P1.INT2\
P1.INT3\
P1.INT4\
P1.INT5\
P1.INT6\
P1.INT7\

COMPUTER SCIENCE DEPARTMENT, STANFORD UNIVERSITY, STANFORD, CALIFORNIA 94305

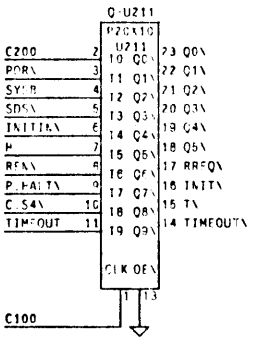
SYSTEM CLOCK GENERATOR



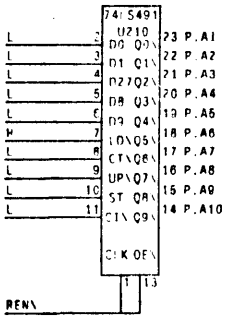
TIMEOUT COUNTER



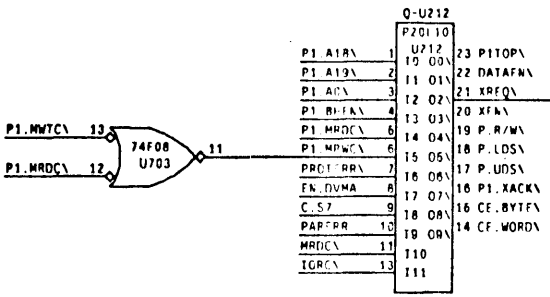
TIMER CONTROLLER



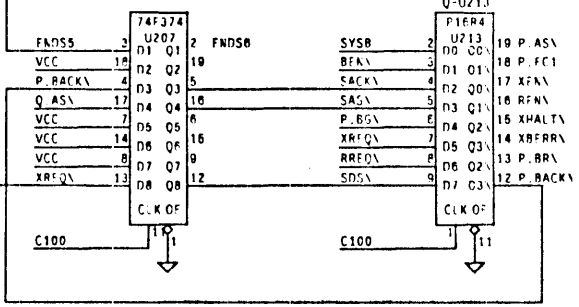
REFRESH COUNTER



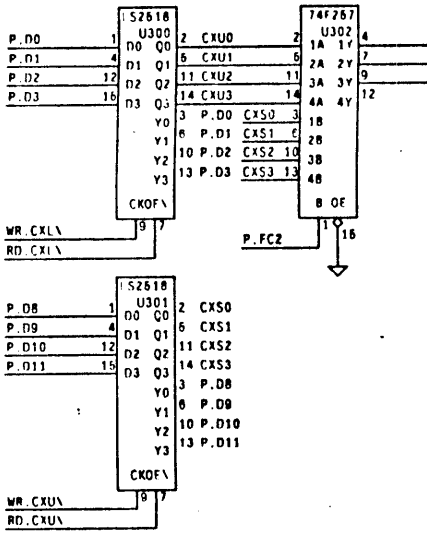
DVMA DECODER



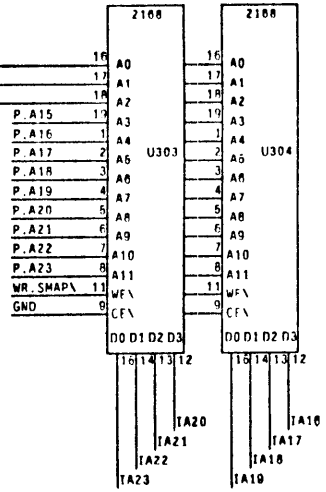
DVMA CONTROLLER



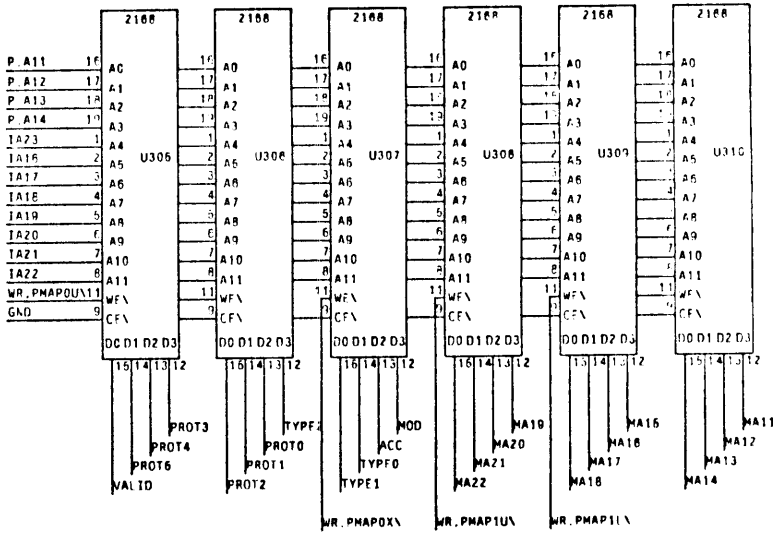
! CONTEXT REGISTER



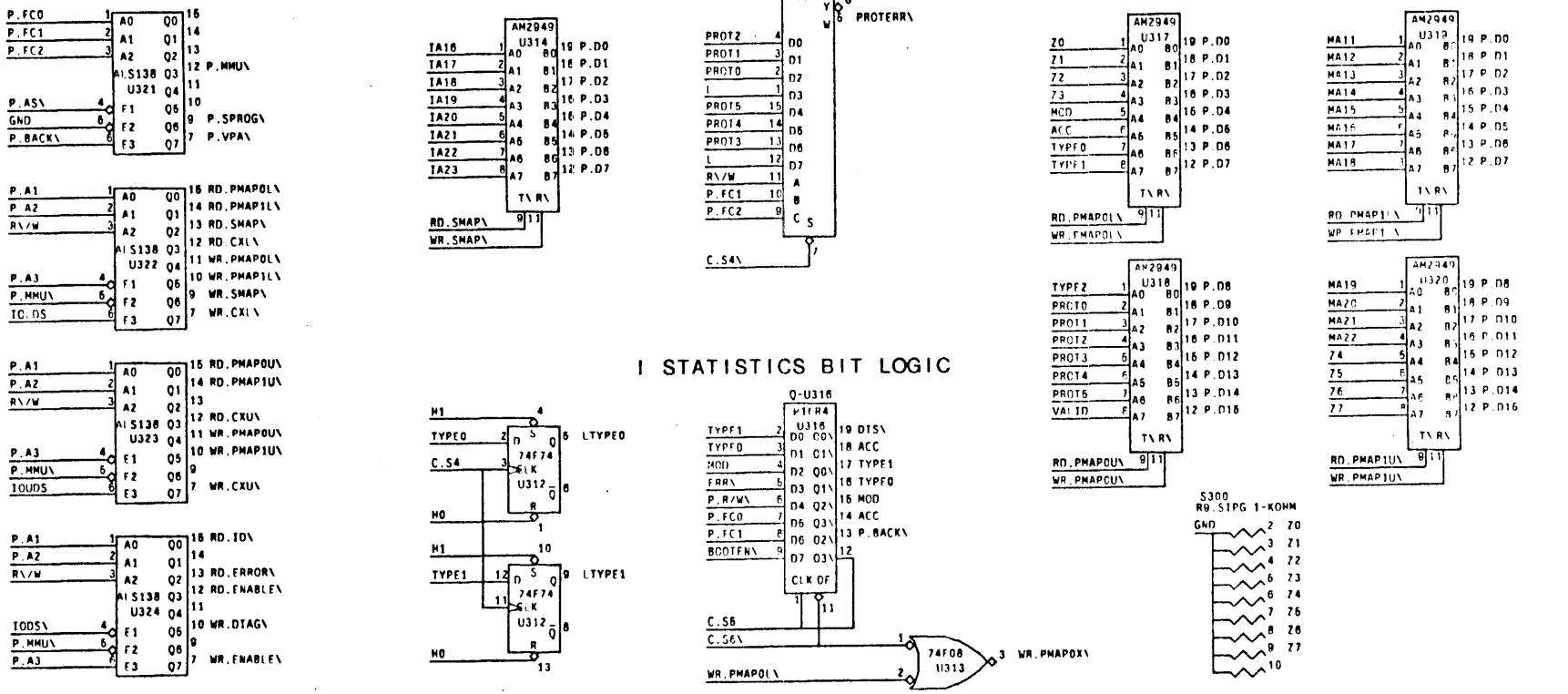
! SEGMENT MAP



! PAGE MAP



! PROTECTION DECODER



! STATISTICS BIT LOGIC

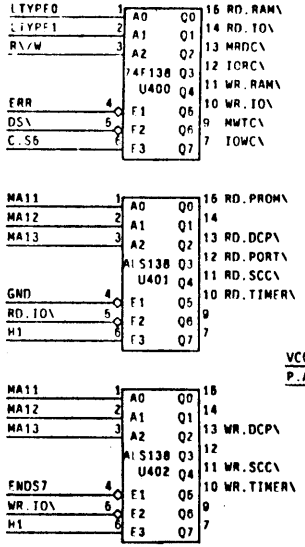
TITLE: SUN MICROSYSTEMS INC, 501-1007-08

FILE: M4[1SM1]

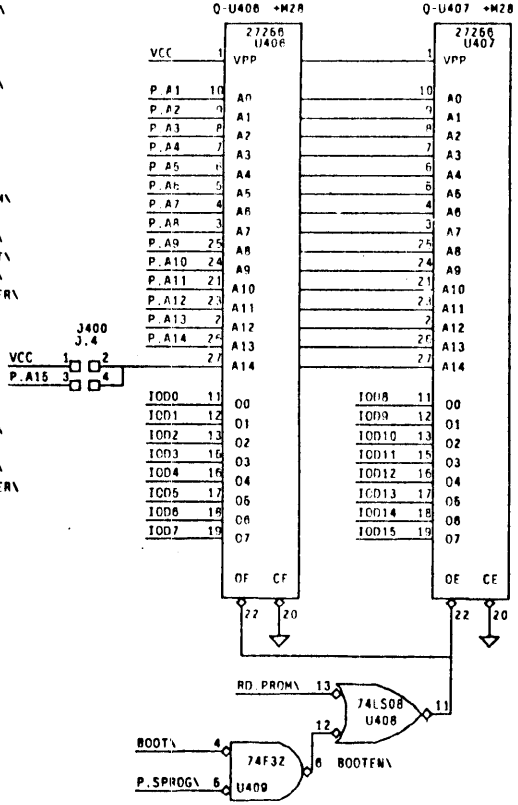
DATE: 26-Jul-84 14:41

PAGE: 4 OF 9

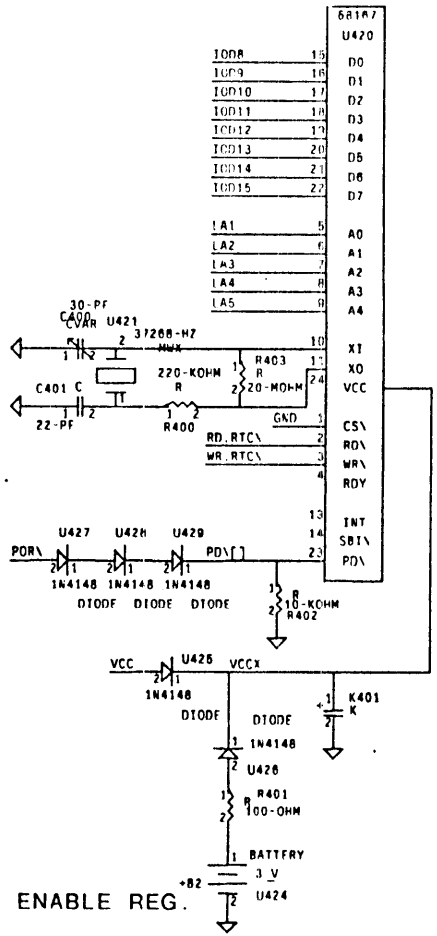
! STROBE DECODERS



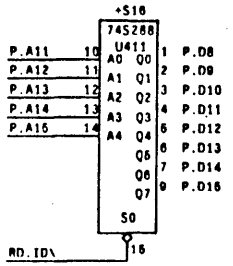
! BOOT PROMS



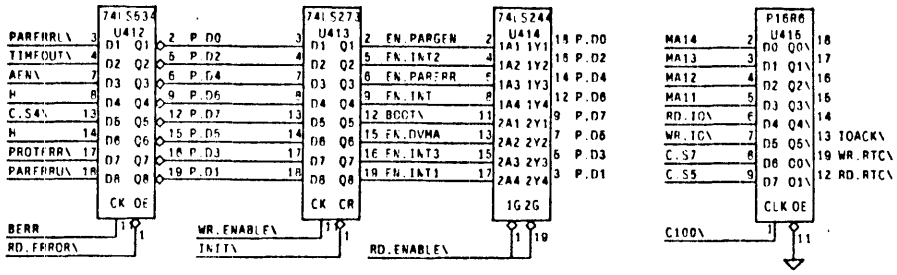
! REAL-TIME CLOCK



! ID PROM



! BUS ERROR REG. ! SYSTEM ENABLE REG.

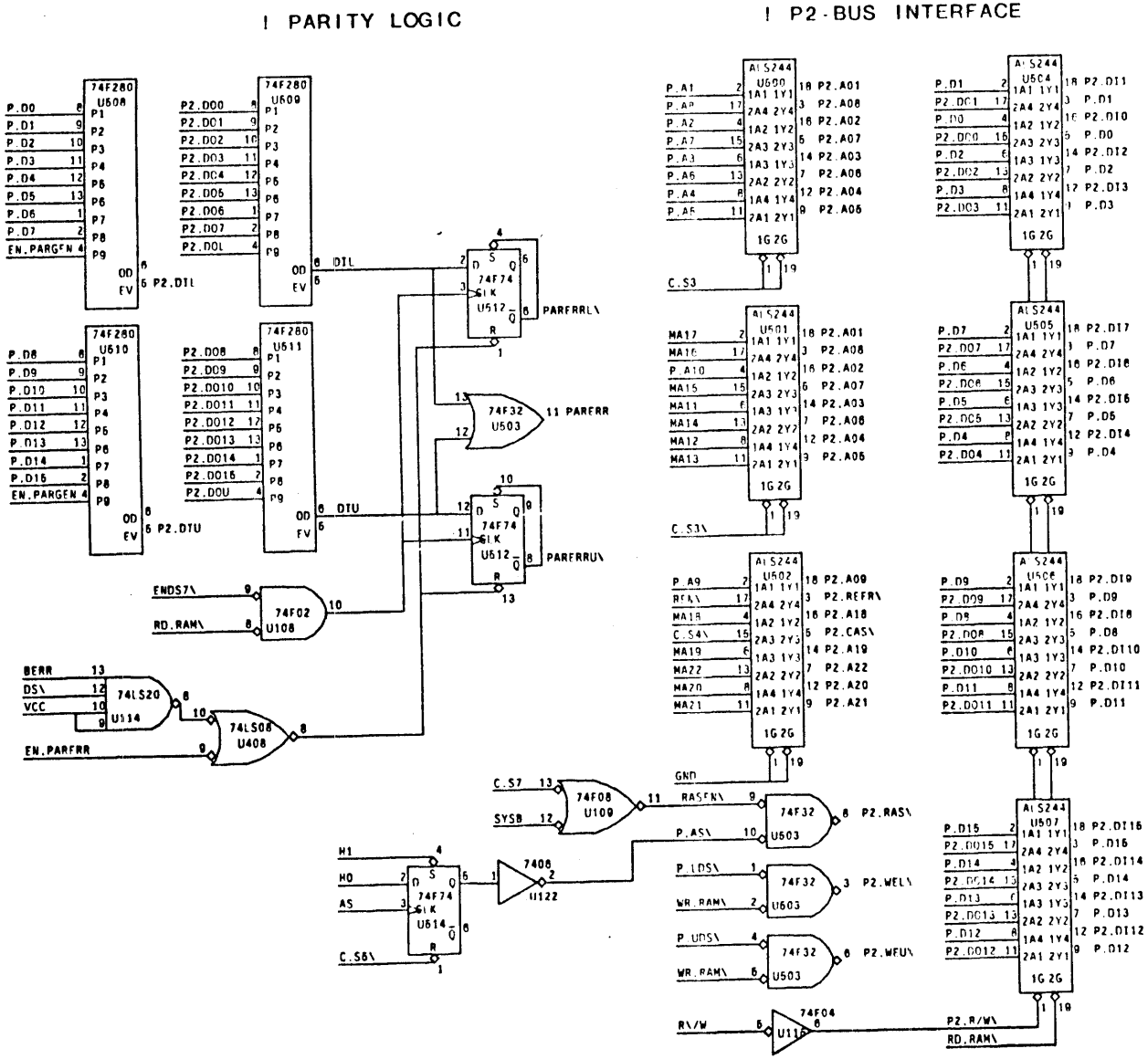


TITLE: SUN MICROSYSTEMS INC, 501-1007-08

FILE: M5[1SMI]

DATE: 26-Jul-84 14:45

PAGE: 5 OF 9



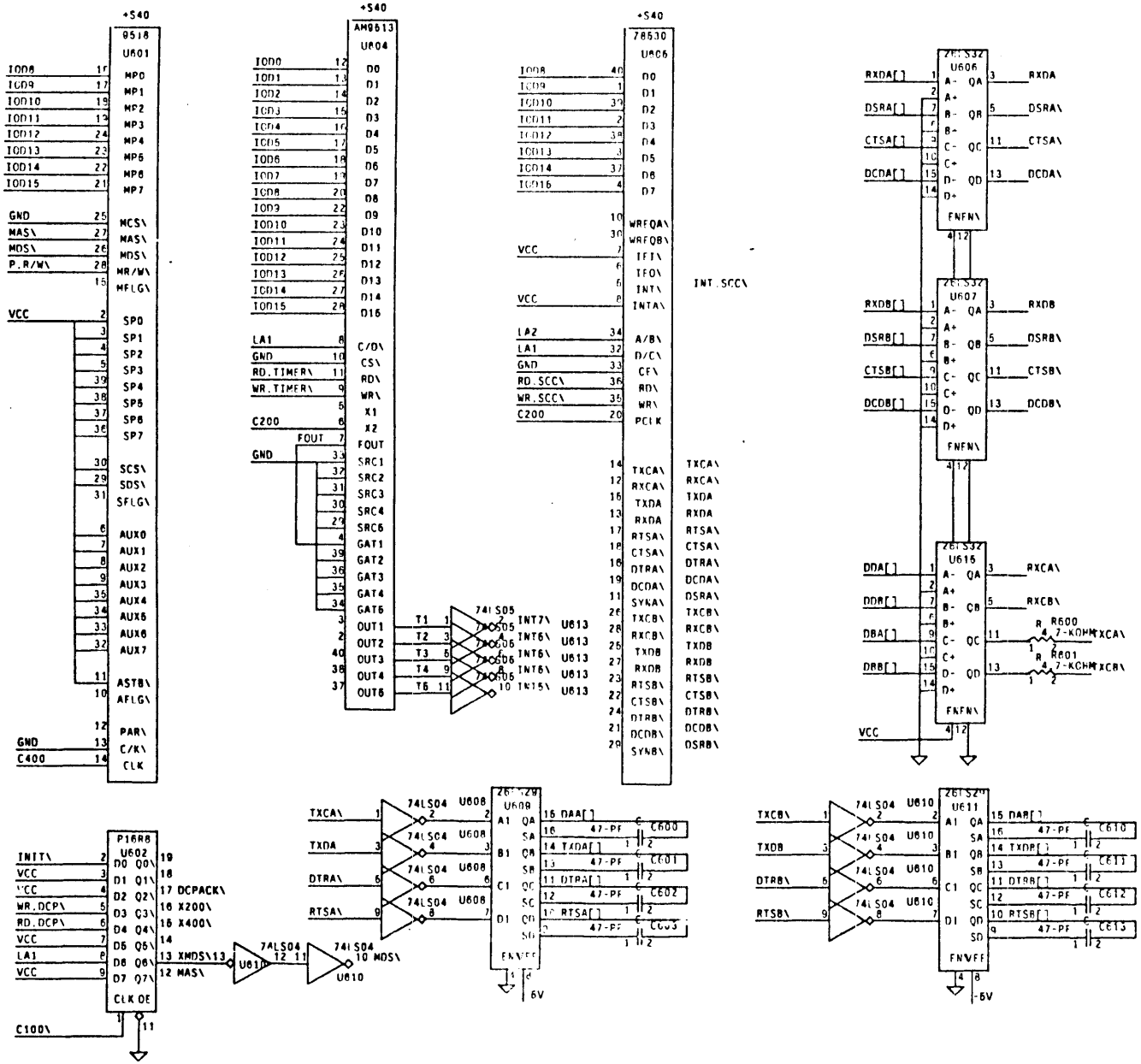
COMPUTER SCIENCE DEPARTMENT, STANFORD UNIVERSITY, STANFORD, CALIFORNIA 94306

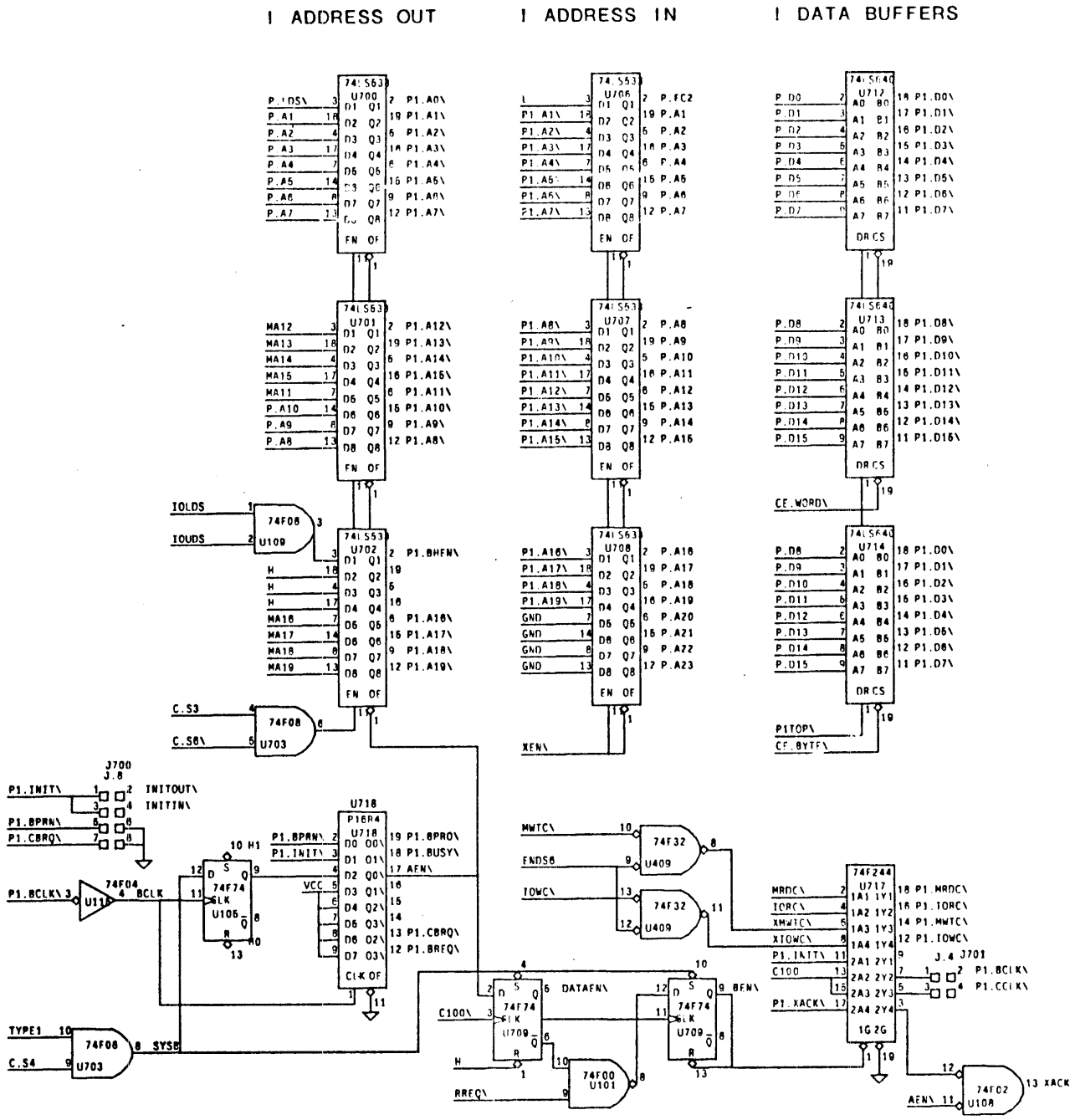
! DCP

! TIMER

! SCC

! RS423 DRIVERS





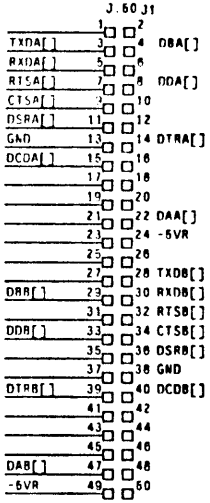
COMPUTER SCIENCE DEPARTMENT, STANFORD UNIVERSITY, STANFORD, CALIFORNIA 94306

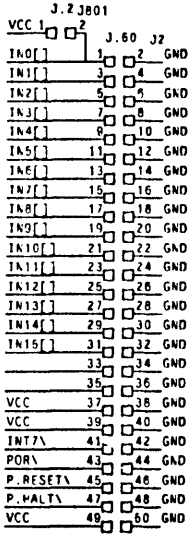
TITLE: SUN MICROSYSTEMS INC, 501-1007-08

FILE: M7[1SM1]

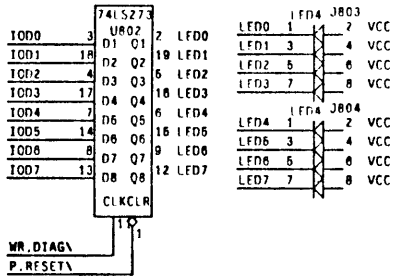
DATE: 27-Jul-84 11:51

PAGE 7 OF 9

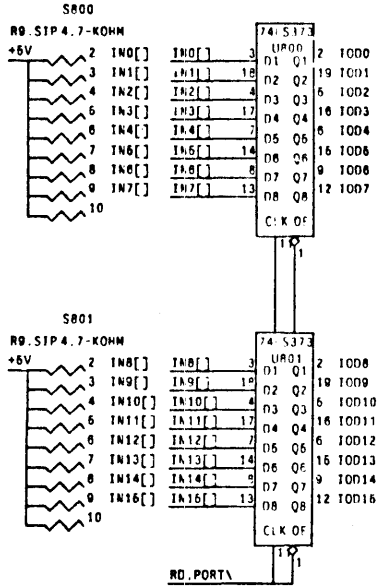


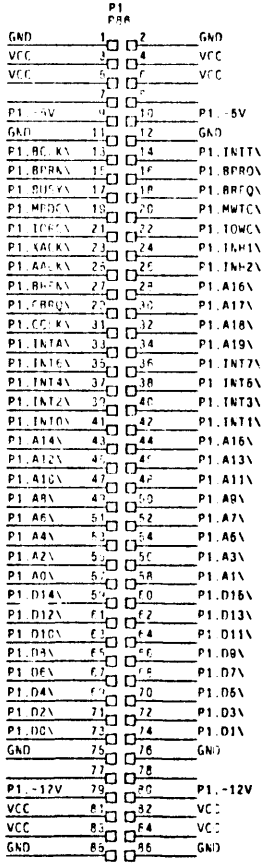


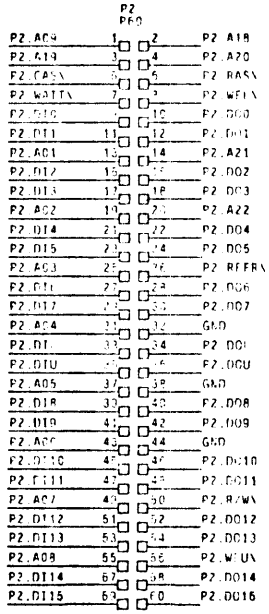
I DIAGNOSTIC REGISTER

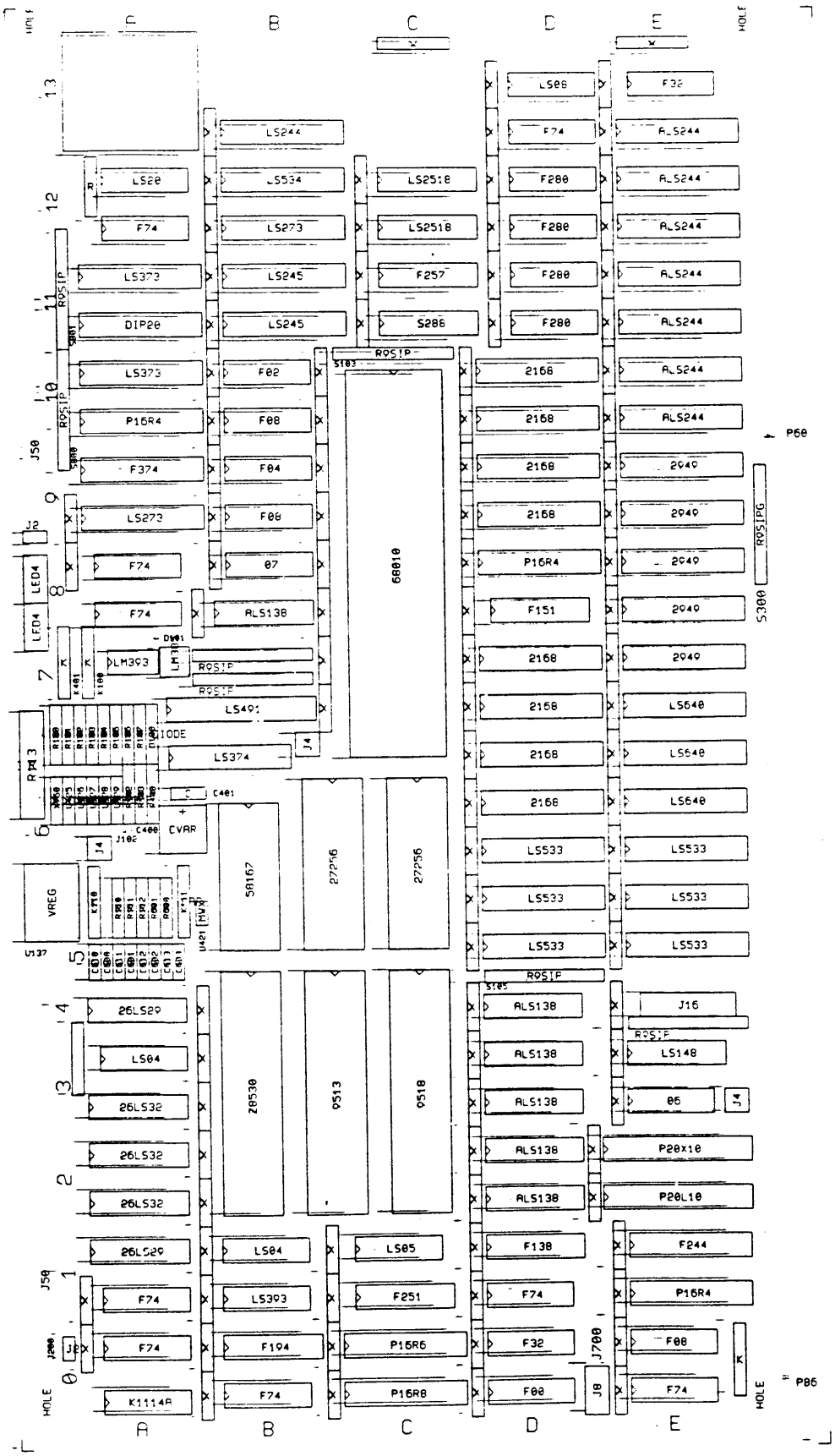


I INPUT PORT





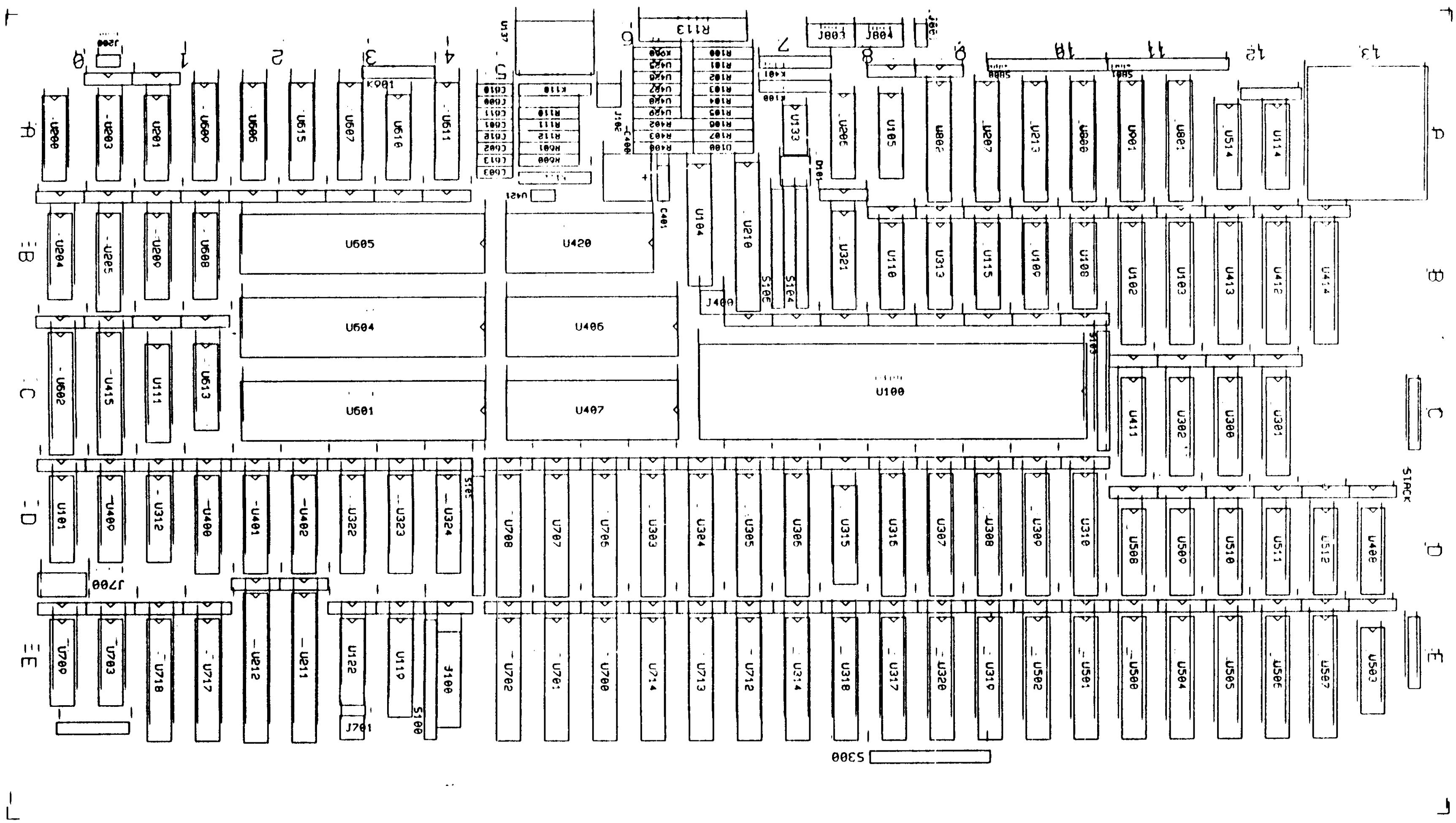




SUN MICRO 09-Aug-84 105102

SILK1 BPL=0

SUN MICRO
-SILK2
00-Aug-84 105102



Appendix B

PALs

For information on PALASM, refer to the ***MMI PAL Programmable Array Logic Handbook, 3rd edition.***

Source code from PALASM with pinout is on the following pages.

Table B-1: CPU PALs SUMMARY

U #	type	checksum	function	part #	page #
U211	20X10	7A93	Timer Controller	520-1123-01	B-2
U212	20L10	7F80	DVMA Decoder	520-1122-01	B-4
U213	16R4	512A	DVMA Controller	520-1119-01	B-6
U316	16R4	2A01	Statistics Bit Logic	520-1120-01	B-8
U415	16R6	65B0	RTC/Waitgen	520-1125-01	B-10
U602	16R8	47A2	DCP Interface	520-1121-01	B-13
U718	16R4A	524B	P1 Arbitration	520-1124-01	B-15

```
pal20x10
Rev 1.1                  Wed Jul  4 1984                          JM
U211            Timer Controller Pal for 120 cpu board
Sun Microsystems Inc, Mt View CA
c100 /c200 /por sysb /sds /initin notused /ren /p.halt /as tin gnd
/oe /timeout /t /init /rreq /q5 /q4 /q3 /q2 /q1 /q0 vcc

;
;       Macros
;
#define Q7-     /t
#define Q6-     /rreq
#define CY4     c200
#define CY8     CY4 * q0
#define CY16    CY8 * q1
#define CY32    CY16 * q2
#define CY64    CY32 * q3
#define CY128   CY64 * q4
#define CY256   CY128 * q5
#define CY512   CY256 * /Q6-
#define CY1024  CY512 * /Q7-

q0      := q0 + por :+:         ; C400
           CY4 * /por

q1      := q1 + por :+:         ; C800
           CY8 * /por

q2      := q2 + por :+:         ; C1600
           CY16 * /por

q3      := q3 + por :+:         ; C3200
           CY32 * /por

q4      := q4 + por :+:         ; C6400
           CY64 * /por

q5      := q5 + por :+:         ; C12800
           CY128 * /por

rreq    := rreq * /ren +
           CY256 * /ren

init    := init + por :+:
           ;
           ; watchdog reset!
           ;
           CY256 * p.halt * /sds * /sysb
             * /por

t       := t * as :+:
           CY128 * as

timeout := timeout * as +
           tin * as

function table
c100
  /c200 /por sysb /sds notused
        notused /ren /p.halt /as tin
```

```
                /oe
                        /timeout /t /init /rreq
                                /q5 /q4 /q3 /q2 /q1 /q0
-----------------------------------------
c 11xhx x11h1   1       hh1h    111111
c 11xhx xh1h1   1       hh11    111111
c 11xhx xh111   1       h111    111111
c 1hxhx xh11h   1       1hh1    hhhhhh
c 11xhx x1111   1       1h1h    111111
c 1hx1x xh111   1       1111    hhhhhh
c 11xhx x1111   1       111h    111111
c 1hxhx x1h11   1       1h1h    hhhhhh
c 1hxhx x1111   1       1h1h    hhhhh1
c 1hxhx x11h1   1       hh1h    hhhh1h
-----------------------------------------

description
```

```
pal20110
Rev 1.1         Aug 15 1984             JM
U212            DVMA Decoder Pal for 120 cpu board
Sun Microsystems Inc, Mt View CA
/p1.a18 /p1.a19 /p1.a0 /p1.bhen /p1.mrdc /p1.mrwc
/proterr en.dvma c.s7 parerr /mrdc gnd
/iorc /ce.word /ce.byte /p1.xack /p.uds /p.lds
/p.wr /xen /xreq /aen /pltop vcc

ce.word = aen * p.lds +                 ; CPU CYCLE (R/W LOW BYTE/WORD)
          aen * p.wr +                  ; CPU (WRITE)
          xen * p.lds                   ; DVMA CYCLE (R/W LOW BYTE/WORD)

        ;
ce.byte = aen * /p.lds * p.uds * /p.wr +        ; CPU CYCLE (READ UPPER BYTE)
          xen * /p.lds * p.uds                  ; DVMA CYCLE (R/W UPPER BYTE)

pltop   = /xen * mrdc +                         ; NON-DVMA MRDC CYCLE
          /xen * iorc +                         ; NON-DVMA IORC CYCLE
          xen * p.wr                            ; DVMA WRITE CYCLE CONDITION

; ASSERTED ON DVMA CYCLES ONLY
if ( xen ) p.wr = p1.mrwc * /p1.mrdc * /c.s7 +  ; SET
                  p.wr * /p1.mrdc               ; HOLD

; ASSERTED ON DVMA CYCLES ONLY
if ( xen ) p.lds = /p1.a0 * xreq * xen +        ; EVEN BYTE
                   p1.bhen * xreq * xen +       ; WORD
                   p.lds * p1.mrwc              ; HOLD

; ASSERTED ON DVMA CYCLES ONLY
if ( xen ) p.uds = p1.a0 * xreq * xen +         ; ODD BYTE
                   p1.bhen * xreq * xen +       ; WORD
                   p.uds * p1.mrwc              ; HOLD

; ASSERTED ON DVMA CYCLES ONLY
if ( xen ) p1.xack = c.s7 * p1.mrdc * /proterr * /parerr + ; DVMA READ CYCLE
                     c.s7 * p1.mrwc * /p1.mrdc * /proterr  ; DVMA WRITE CYCLE

xreq    = en.dvma * /p1.a19 * /p1.a18 * p1.mrwc * /aen * /xen +         ; SET
          xreq * p1.mrwc                                                ; HOLD

function table
/p1.a18 /p1.a19 /p1.a0 /p1.bhen /p1.mrdc /p1.mrwc
        /proterr en.dvma c.s7 parerr /mrdc /iorc
                /ce.word /ce.byte /p1.xack /p.uds /p.lds
                        /p.wr /xen /xreq /aen /pltop
-------------------------------------------------------------------------
l1xlhh  hlllll  hhzzz   zhhhl
l1xlhl  hlllll  lhzll   hhhll
hlxlhl  hlllhl  lhzhl   hhhll
hhxlhl  hlllll  hlzlh   hhhll
hhxlhl  hhllhh  hhzhh   hhhlh
hhxlhl  hhllxx  lhzll   lhhlx
hhxlhl  hhllxx  lhzhl   lhhlx
hhxlhl  hhllxx  lhzlh   lhhlx
hhxlhl  hhllxx  hhzhh   lhlhx
hhhlhl  hhllxx  hhzzz   zhlhx
hhhlhl  hhllxx  lhhll   lllhl
hhhlhl  hhhlxx  lhlll   lllhl
```

```
hhhlhl  hhhlxx  lhlll   lllhl
llxlhh  hhllxx  hhhhh   llhhl
llxlhh  hhllxx  hhzzz   zhhhx
hhlhll  hhllxx  hhzzz   zhlhx
hhlhll  hhllxx  hlhlh   hllhh
hhlhll  hhllxx  hlhlh   hllhh
hhlhll  hhhlxx  hllll   hllhh
hhlhll  hhhlxx  hllll   hllhh
llhhhh  hhllxx  hhhhh   hlhhh
llhhhh  hhllxx  hhzzz   zhhhx
-------------------------------------------------------------------------
description
```

```
pal16r4
Rev 1.1          Wed Aug 15 1984                  JM
U213             DVMA Controller Pal for 120 cpu board
Sun Microsystems Inc, Mt View CA
c100 sysb /ben /sack /sas /p.bg /xreq /rreq /sds gnd
/oe /p.back /p.br /xberr /xhalt /ren /xen fc1 /p.as vcc


p.back  = ren +
          xen

p.br    = xreq * /p.back +
          rreq * /p.back

if ( p.back ) /fc1 = xen                          ; SUPERVISOR DATA FOR XDMA

if ( p.back ) p.as = sack * ren +
                     sack * xen * xreq

xen     := /xen * /ren * p.bg * /sas * /rreq * sds +     ; SET
           xen * sds                                     ; CLEAR

ren     := /xen * /ren * p.bg * /sas * rreq +    ; STATE 0
           ren * /sack +                         ; STATE 1
           ren * /sas                            ; STATE 2

xberr   := sds * sysb * xhalt +                  ; SET ON MULTIBUS DEADLOCK
           rreq * /ben * sysb * xhalt +          ; SET ON REFRESH DEADLOCK
           xberr * sas                           ; HOLD

        ;ASSERT XHALT-
        ;
xhalt   := sds * sysb +                          ; SET ON MULTIBUS DEADLOCK
           rreq * /ben * sysb +                  ; SET ON REFRESH DEADLOCK
           xberr                                 ; XBERR PLUS ONE STATE

function table

c100
  sysb /ben /sack /sas
       /p.bg /xreq /rreq /sds /oe
             /p.back /p.br /xberr /xhalt
                     /ren /xen fc1 /p.as
-------------------------------------------
c lhhh  hhhhl   xhhx    xhzz
c lhhh  hhhhl   xhhh    xhzz
c lhhl  hllll   xxhh    xhzz
c lhhl  lllll   xxhh    xhzz
c lhhh  lllll   lhhh    lhhh
c lhhh  lllll   lhhh    lhhh
c lhlh  llhll   lhhh    lhhl
c lhll  hlhll   hlhh    hhhh
c lhll  hlhll   hlhh    hhhh
c lhlh  hlhll   hlhh    hhzz
c lhhh  llhll   lhhh    hllh
c lhhh  llhll   lhhh    hllh
c lhlh  hlhll   lhhh    hlll
c lhll  hlhll   lhhh    hlll
c lhll  hhhhl   hhhh    hhhh
c lhlh  hhhhl   hhhh    hhzz
```

```
c lhhh  hhhhl   hhhh    hhzz
c lhhh  hhhll   hhhh    hhzz
c hhhh  hhhll   hhhl    hhzz
c hhhh  hhhll   hhll    hhzz
c hhhh  hhhll   hhll    hhzz
c lhhh  hhhll   hhhl    hhzz
c lhhh  hhhll   hhhh    hhzz
c lhhh  hhlhl   hlhh    hhzz
c hhhh  hhlhl   hlhl    hhzz
c hhhh  hhlhl   hlll    hhzz
c hhhh  hhlhl   hlll    hhzz
c lhhh  hhlhl   hlhl    hhzz
c lhhh  hhhhl   hhhh    hhzz
c hlhh  hhlhl   hlhh    hhzz
c lhhh  hhhhl   hhhh    hhzz
c hhhh  hhhhl   hhhh    hhzz
c hhhh  hhhll   hhhl    hhzz
c lhhh  hhhhl   hhhh    hhzz
c lhhl  hllll   hlhh    hhzz
c lhhl  lllll   hlhh    hhzz

-----------------------------------------------

description
```

```
pal16r4
Rev 1.0          Thu Jun 28 1984                          JM
U316             Statistic Bit Logic Pal for 120 cpu board
Sun Microsystems Inc, Mt View CA
c.s5c type1 type0 mod en read p.fc0 p.fc1 /booten gnd
/c.s6 c.s5 /p.back acc. mod. type0. type1. acc /dis vcc


/type1. := /type1                          ; write back

/type0. := /type0                          ; write back

/acc. := /acc * /en * /dis +               ; keep old acc when not enabled
         /acc * dis                        ; keep old acc when disabled

/mod. := /mod * read * en * /dis +         ; keep mod value on read cycles
         /mod * /en * /dis +               ; old value if not enabled
         /mod * dis                        ; old value if disabled

dis =    p.fc0 * p.fc1 * /p.back +         ; mmu reference
         p.fc1 * p.back +                  ; refresh reference
         booten


function table

c.s5c type1 type0 mod en
         read p.fc0 p.fc1 /booten /p.back
                  /c.s6 c.s5 acc. mod.
                           type0. type1. acc /dis
-----------------------------------------------------
lxxxh    xxxlx    hlzz     zzxl
lxxxh    xxxlx    hhzz     zzxl
lxxxh    xxxlx    hlzz     zzxl
lxxxh    xllhx    hlzz     zzxh
lxxxh    xllhx    hhzz     zzxh
lxxxh    xhlhx    hlzz     zzxh
lxxxh    xhlhx    hhzz     zzxh
lxxxh    xlhhx    hlzz     zzxl
lxxxh    xlhhx    hhzz     zzxl
lxxxh    xhhhx    hlzz     zzxl
lxxxh    xhhhx    hhzz     zzxl
lxxxh    xhhhx    hlzz     zzxl
lxxxh    xllhx    hlzz     zzxh
cllih    hllhx    hlzz     zzlh
llllh    hllhx    hhzz     zzlh
llllh    hllhx    lhhl     lllh
clhhh    hllhx    hlzz     zzhh
llhhh    hllhx    hhzz     zzhh
llhhh    hllhx    lhhh     hlhh
chllh    lllhx    hlzz     zzlh
lhllh    lllhx    hhzz     zzlh
lhllh    lllhx    lhhh     lhlh
chhhh    lllhx    hlzz     zzhh
lhhhh    lllhx    hhzz     zzhh
lhhhh    lllhx    lhhh     hhhh
cllll    hllhx    hlzz     zzlh
lllll    hllhx    hhzz     zzlh
lllll    hllhx    lhll     lllh
clhhl    lllhx    hlzz     zzhh
```

```
llhhl   lllhx   hhzz    zzhh
llhhl   lllhx   lhhh    hlhh
lhllh   hhhhx   hlzz    zzll
chllh   hhhhx   hlzz    zzll
lhllh   hhhhx   hhzz    zzll
lhllh   hhhhx   lhll    lhll
lhhhh   lllhx   hlzz    zzhh
lhhhh   lhhhx   hlzz    zzhl
chhhh   lhhhx   hlzz    zzhl
lhhhh   lhhhx   hhzz    zzhl
lhhhh   lhhhx   lhhh    hhhl
lhhhh   lllhx   hlzz    zzhh
-------------------------------------------------
```

description

```
pal16r6                          u415                PAL DESIGN SPECIFICATION
120 CPU BOARD                                        KB 06/12/84
I/O ACKNOWLEDGE AND TOD RD/WR CONTROL SIGNAL GENERATOR
SUN MICROSYSTEMS

/CLK100 MA14 MA13 MA12 MA11 /RDIO /WRIO CS7 CS5 GND
GND /RDRTC /IOACK NC IQ0 IQ1 IQ2 IQ3 /WRRTC VCC


IF (VCC) RDRTC = /MA14 * MA13 * MA12 * MA11 * RDIO * CS7 ;58167 read strobe

IF (VCC) WRRTC = /MA14 * MA13 * MA12 * MA11 * WRIO *     ;58167 write strobe
                                       CS7 * /IOACK      ;which goes inactive
                                                         ;when IOACK active

                 ;RD / WR ACK for the Parallel Port ( 2 wait states )
         IOACK:= /MA14 * /MA13 * MA12 * MA11 * RDIO * CS5 +
                 /MA14 * /MA13 * MA12 * MA11 * WRIO * CS5 +

                 ;RD / WR ACK for PROM, SCC, Timer  ( 2 wait states )
                 /MA14 * /MA12 * RDIO * CS5 +
                 /MA14 * /MA12 * WRIO * CS5 +

                 ;RD /WR ACK for 58167 when the counter is equal to 10 or 11
                 ; ( 12 wait states )
                 /MA14 * MA13 * MA12 * MA11 * RDIO * IQ3 * /IQ2 * IQ1 * CS5 +
                 /MA14 * MA13 * MA12 * MA11 * WRIO * IQ3 * /IQ2 * IQ1 * CS5

         /IQ0 := /CS5 +                                          ; reset
                  CS5 *  IQ0 * /IOACK +                          ; toggle
                  CS5 * /IQ0 *  IOACK                            ; hold at eleven

         /IQ1 := /CS5 +                                          ; reset
                  CS5 * /IQ1 * /IQ0 +                            ; hold
                  CS5 *  IQ1 *  IQ0 * /IOACK +                   ; toggle
                  CS5 * /IQ1 *  IOACK                            ; hold at eleven

         /IQ2 := /CS5 +                                          ; reset
                  CS5 * /IQ2 * /IQ0 +                            ; hold
                  CS5 * /IQ2 * /IQ1 +                            ; hold
                  CS5 *  IQ2 *  IQ1 *  IQ0 * /IOACK +            ; toggle
                  CS5 * /IQ2 *  IOACK                            ; hold at eleven

         /IQ3 := /CS5 +                                          ; reset
                  CS5 * /IQ3 * /IQ0 +                            ; hold
                  CS5 * /IQ3 * /IQ1 +                            ; hold
                  CS5 * /IQ3 * /IQ2 +                            ; hold
                  CS5 *  IQ3 *  IQ2 * IQ1 * IQ0 * /IOACK +       ; toggle
                  CS5 * /IQ3 *  IOACK                            ; hold at eleven

function table

/CLK100 /RDIO /WRIO MA14 MA13 MA12 MA11 CS7 CS5                  ; inputs
/IOACK /RDRTC /WRRTC IQ3 IQ2 IQ1 IQ0                             ; outputs

;/  / /                  / / /
;C  R W  M M M M  C C    I R W  I I I I
;L  D R  A A A A  S S    O D R  Q Q Q Q
;K  I I  1 1 1 1  7 5    A R R  3 2 1 0
;   O O  4 3 2 1         C T T
```

```
;                                K C C
-------------------------------------------------------------------------------
c  x x  x x x x  1 1     h h h  1 1 1 1    ; idle state
c  h h  1 1 h h  1 h     h h h  1 1 1 h    ;setup parallel port rd or wr ioack
c  l h  1 1 h h  1 h     l h h  1 1 h l    ;read port
c  l h  1 1 h h  1 l     h h h  1 1 1 1    ;end read port
c  h l  1 1 h h  1 h     l h h  1 1 1 h    ;write port
c  h l  1 1 h h  1 l     h h h  1 1 1 1    ;end write port
c  h h  1 x 1 x  1 h     h h h  1 1 1 h    ;setup prom scc timer ioack
c  l h  1 x 1 x  1 h     l h h  1 1 h l    ;read prom scc timer
c  h l  1 x 1 x  1 h     l h h  1 1 h l    ;write prom scc timer
c  x x  x x x x  1 1     h h h  1 1 1 1    ; idle state
c  h h  1 h h h  1 1     h h h  1 1 1 1    ;setup for 58167 operation
c  l h  1 h h h  1 h     h h h  1 1 1 h    ;read wait 1
c  l h  1 h h h  h h     h l h  1 1 h l    ;wait 2
c  l h  1 h h h  h h     h l h  1 1 h h    ;3
c  l h  1 h h h  h h     h l h  1 h 1 1    ;4
c  l h  1 h h h  h h     h l h  1 h 1 h    ;5
c  l h  1 h h h  h h     h l h  1 h h l    ;6
c  l h  1 h h h  h h     h l h  1 h h h    ;7
c  l h  1 h h h  h h     h l h  h 1 1 1    ;8
c  l h  1 h h h  h h     h l h  h 1 1 h    ;9
c  l h  1 h h h  h h     h l h  h 1 h l    ;10
c  l h  1 h h h  h h     l l h  h 1 h h    ;hold 11 ioack
c  h h  h h h h  h h     h h h  h 1 h h    ;let counter continue
c  h h  h h h h  h h     h h h  h h 1 1    ; to fully test the
c  h h  h h h h  h h     h h h  h h 1 h    ; counter
c  h h  h h h h  h h     h h h  h h h l    ;
c  h h  h h h h  h h     h h h  h h h h    ;
c  h h  h h h h  h h     h h h  1 1 1 1    ;
c  h h  h h h h  h h     h h h  1 1 1 h    ;
c  h h  h h h h  1 1     h h h  1 1 1 1    ;clear counter with cs5
c  h h  1 h h h  1 1     h h h  1 1 1 1    ;setup for 58167 operation
c  h l  1 h h h  1 h     h h h  1 1 1 h    ;write wait 1
c  h l  1 h h h  h h     h h l  1 1 h l    ;wait 2
c  h l  1 h h h  h h     h h l  1 1 h h    ;3
c  h l  1 h h h  h h     h h l  1 h 1 1    ;4
c  h l  1 h h h  h h     h h l  1 h 1 h    ;5
c  h l  1 h h h  h h     h h l  1 h h l    ;6
c  h l  1 h h h  h h     h h l  1 h h h    ;7
c  h l  1 h h h  h h     h h l  h 1 1 1    ;8
c  h l  1 h h h  h h     h h l  h 1 1 h    ;9
c  h l  1 h h h  h h     h h l  h 1 h l    ;10
c  h l  1 h h h  h h     l h h  h 1 h h    ;hold 11 - remove wrrtc
c  h l  1 h h h  1 1     h h h  1 1 1 1    ;end operation
c  x x  x x x x  1 1     h h h  1 1 1 1    ; idle
-------------------------------------------------------------------------------
```

description:

IQO:3 - This is a four bit counter that is enabled when CS5 is active. The counter is used to issue the 58167 IOACK signal on state 11 (68010 state S27 and S28) which is used to negate the WRRTC write strobe signal from state 11 on. The WRRTC signal is negated at this time to allow for data and address hold times.

IOACK - This is the DTACK signal for the 58167, 9513, 8530A, PROM, and Parallel Port. IOACK is issued at S27:28 for the 58167 and at S7:8 for the other devices. This adds 12 wait-states for the 58167 and two for all others.

WRRTC - 58167 write strobe. Begins at S7 and ends at S27.

RDRTC - 58167 read strobe. Begins at S7 and ends at S0 of the next cycle.

```
pal16r8                 u602                    dcpctl
        Rev 0.1         Aug 15 1984             JM
                120 CPU DCP control pal
 JM     Sun Microsystems Inc            Mt View, CA
clk /sanity vcc vcc /wrdcp /rddcp vcc la1 vcc gnd
/oe /mas /mds q0 /x400 /x200 /ack q1 q2 vcc

ack := /q2 * q1 * /sanity +
       q2 * /q1 * q0 * x200 * /sanity +
       q1 * /q0 * /x200 * /sanity

mds := q2 * /q1 * q0 * x200 * /sanity +
       q2 * q1 * /q0 * /sanity +
       q2 * q1 * /x200 * /x400 * /la1 * rddcp * /sanity +
       q2 * q1 * /x200 * /x400 * /la1 * wrdcp * /sanity

mas := /q2 * q1 * q0 * /sanity +
       q1 * q0 * la1 * rddcp * /sanity +
       q1 * q0 * la1 * wrdcp * /sanity

/q2 := /q2 * q1 * q0 * /sanity +
       q1 * q0 * la1 * rddcp * /sanity +
       q1 * q0 * la1 * wrdcp * /sanity

/q1 := q2 * /q1 * q0 * x200 * /sanity +
       q2 * q1 * /q0 * /x200 * /sanity

/q0 := /q2 * q1 * q0 * /sanity +
       q2 * q1 * /q0 * x200 * /sanity +
       q1 * q0 * /x200 * /x400 * /la1 * rddcp * /sanity +
       q1 * q0 * /x200 * /x400 * /la1 * wrdcp * /sanity

x400 := x400 * x200 * /sanity +
        /x400 * /x200 * /sanity

x200 := /x200 * /sanity

function table

clk /oe /sanity
        /rddcp /wrdcp la1
                /x400 /x200
                        q2 q1 q0
                                /mas /mds /ack

;c c i  i i i   s s    s s s   o o o
-------------------------------------------------------------------
                                        ;reset
c l l   x x x   h h    h h h   h h h   ;STATE idle
l h x   x x x   z z    z z z   z z z   ; check tristate enable
;
;       MAS- Read Cycle
;
c l h   l h h   l l    l h h   l h h   ;STATE strobemas
c l h   x x x   l h    l h l   l h l   ;STATE mas_dtack
c l h   x x x   h l    h h h   h h l   ;STATE idle
c l h   h h x   h h    h h h   h h h   ;STATE idle
;
;       MAS- Write Cycle
;
```

```
c l h   h l h   l l     l h h   l h h   ;STATE strobemas
c l h   x x x   l h     l h l   l h l   ;STATE mas_dtack
c l h   x x x   h l     h h h   h h l   ;STATE idle
c l h   h h x   h h     h h h   h h h   ;STATE idle
;
;       MDS- Read Cycle
;
c l h   l h l   l l     h h l   h l h   ;STATE strobemds
c l h   x x x   l h     h h l   h l h   ; stay here for 1 more clock
c l h   x x x   h l     h l h   h l l   ;STATE mas_dtack
c l h   x x x   h h     h l h   h l l   ; stay here for 1 more clock
c l h   h h x   l l     h h h   h h h   ;STATE idle
;
;       MDS- Write Cycle
;
c l l   x x x   h h     h h h   h h h   ;STATE idle (reset)
c l h   h l l   l l     h h l   h l h   ;STATE strobemds
c l h   x x x   l h     h h l   h l h   ; stay here for 1 more clock
c l h   x x x   h l     h l h   h l l   ;STATE mas_dtack
c l h   x x x   h h     h l h   h l l   ; stay here for 1 more clock
c l h   h h x   l l     h h h   h h h   ;STATE idle

-------------------------------------------------------------------
;c c i   i i i   s s     s s s   o o o

;ll -> lh -> hl -> hh

description:
```

```
pal16r4         u718                        arb
        Rev 1.2         Jun  6 1984             JM
                120 CPU P1 Bus (MULTIBUS) Arbiter
        Sun Microsystems Inc                Mt View, CA
clk /plbprn /plinit sysb /test vcc vcc vcc vcc gnd
/oe /plbreq /plcbrq /cbrqo q0 q1 /aen /plbusy /plbpro vcc

plbpro  = q1 * q0 * plbprn * /plinit

if( aen ) plbusy = aen

if( cbrqo ) plcbrq = cbrqo

plbreq  = /q1 * /plinit +
          /q0 * /plinit

cbrqo := q1 * /q0 * plbusy * /plinit +
         q1 * /q0 * /plbprn * /plinit +
         q1 * q0 * sysb * /plinit

aen := /q1 * q0 * /plinit +
       /q1 * sysb * /plinit +
       /q0 * /test * sysb * /plbusy * plbprn * /plinit

/q1 := /q1 * /plcbrq * plbprn * /plinit +
       /q1 * q0 * /plinit +
       /q1 * sysb * /plinit +
       q1 * /q0 * /plbusy * plbprn * /plinit

/q0 := /q1 * /q0 * /plcbrq * plbprn * /plinit +
       q1 * /q0 * plbusy * /plinit +
       q1 * /q0 * /plbprn * /plinit +
       sysb * /plinit

function table

clk /oe /test
        /plinit /plbprn sysb
                /plcbrq /plbusy
                        /plbreq /plbpro
                                /cbrqo /aen
                                       q1 q0

-------------------------------------------------------------------
                                                    ;reset
c l l   l x x   z z     h h     h h     h h     ;STATE idle
l h l   x x x   z z     h h     z z     z z     ; check tristate enable
l l l   h h l   z z     h h     h h     h h     ; check tristate outs
l l l   h h l   z z     h h     h h     h h     ; check bprn-
l l l   h l l   z z     h l     h h     h h     ; check bprn-
c l l   h h l   z z     h h     h h     h h     ; check sysb
;
c l l   h h h   l l     l h     l h     h l     ;STATE requestbus
c l l   h l l   l l     l h     l h     h l     ; busy, got priority
c l l   h h l   l h     l h     l h     h l     ; not busy, no priority
;
c l l   h l h   z h     l h     h h     l l     ;STATE havebus
c l l   h l h   l l     l h     h l     l l     ; busy is driven
c l l   h l l   h l     l h     h h     l l     ; busy is driven
c l l   h h l   h h     h h     h h     h h     ; check bprn- . sysb
```

```
;
c l l   h h l   z z     h h     h h     h h     ;STATE idle
c l l   h h h   l l     l h     l h     h l     ;STATE requestbus
c l l   h l h   z h     l h     h h     l l     ;STATE havebus
c l l   h l h   z l     l h     h l     l l     ; busy is driven
c l l   h l l   l h     h l     h h     h h     ; check cbrq- . sysb
;
c l l   h h l   z z     h h     h h     h h     ;STATE idle
c l l   h h h   l l     l h     l h     h l     ;STATE requestbus
c l l   h l l   z h     l h     h h     l h     ;STATE holdbus
c l l   h l l   z l     l h     h l     l h     ; busy is driven
c l l   h l h   z l     l h     h l     l l     ;STATE havebus
c l l   h l l   l h     h l     h h     h h     ;STATE idle

-------------------------------------------------------------------
;c c   i i i   b b     o o     n n     s s

description:
```

Appendix C

Reference Documents

Interested readers may consult the following documents for additional information.

Sun-2 Architecture Manual — 22 May 1984 or latest edition, Sun Microsystems Inc., Mountain View, CA

Intel Multibus Specification — June 1982 or latest edition, Intel Corporation

Motorola M68000 16/32-Bit Microprocessor Programmer's Reference Manual — fourth edition (1984) or latest edition, Prentice-Hall, Inc., Englewood Cliffs, NJ

Am9513 System Timing Controller — August 1983 or latest edition, Advanced Micro Devices Inc., Sunnyvale, CA

MOS Microprocessors and Peripherals — June 1983 or latest edition, Advanced Micro Devices Inc., Sunnyvale, CA. This publication includes information about the 8530A Serial Communication Controller, the 9518 Data Ciphering Processor, and the 9513 System Timing Controller.

MM58167 Microprocessor Compatible Real Time Clock — data sheet, latest edition, National Semiconductor Corporation, Santa Clara, CA

PAL Programmable Array Logic Handbook — 3rd or latest edition, Monolithic Memories Incorporated, Santa Clara, CA

READER COMMENT SHEET

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

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

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

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