



**Communication
Automation
Corporation**

1180 McDermott Dr ♦ West Chester, PA 19380-4022

Tel: (610) 692-9526 ♦ Toll-free: (800) 367-6735 ♦ Fax: (610) 436-8258 ♦ <http://www.cacdsp.com> ♦ Email: sales@cacdsp.com

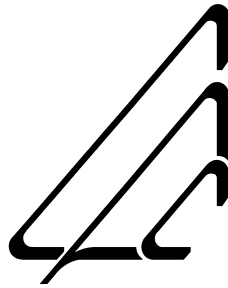
V6SSDL

Solid State
Digital
Delay Line
FIFO

Technical Reference
Version FIFO.3

email: support@cacdsp.com

May 16 2002



© 2002 Communication Automation Corporation
West Chester, PA (USA)

License Agreement

This Software/Hardware is protected by the copyright laws that pertain to computer software and hardware. It is illegal to duplicate the design and implementation of the Hardware and/or to make copies of the Software except for backups. It is illegal to give copies to another person, or to duplicate the Software by any other means, including electronic transmission. The Software contains trade secrets and, to protect them, you may not decompose, reverse engineer, disassemble, or otherwise reduce the applicable portions of the Software to human perceivable form. You are permitted, however, to freely distribute your own derived software that communicates to the CAC DSP boards through these drivers, free of any royalty.

Warranty

Communication Automation Corporation (CAC) reserves the right to make changes to these products, including any software and/or hardware described herein, without notice. No warranty of merchantability or fitness for a particular purpose is expressed or implied. CAC shall not be held liable for incidental or consequential damages in connection with, or arising out of, the use of this Software. CAC does not recommend the use of any of its products, Software or Hardware, for medical or life support applications wherein a failure or malfunction of the product may threaten life or cause injury and will not knowingly license or sell its products for either such use. No rights under any patent accompany the sale of any such products.

Trademarks

MIPS is a registered trademark of MIPS Technologies, Inc.

POSIX is a registered trademark of the Institute of Electrical and Electronics Engineers, Inc.

VME and VMEbus are registered trademarks of Motorola, Inc.

SCSA is a registered trademark of Dialogic Corporation.

Unix is a registered trademark of Santa Cruz Operations.

GNUPro is a registered trademark of Cygnus Solutions.

SPARC, SunOS and Solaris are registered trademarks of Sun Microsystems Computer Corporation and SunSoft.

MS-DOS, Windows95, Windows98, WindowsNT and Windows 2000 are registered trademarks of Microsoft Corporation in the United States of America and other countries.

Use of a term in this manual should not be regarded as affecting the validity of any trademark or service mark.

TABLE OF CONTENTS

V6SSDLF Quick Tour	8
1.1 Functional Description	8
1.2 Major Components	9
1.3 Hardware Options.....	10
1.4 VMEBus Interface	10
1.5 Host and Module Interrupts.....	10
1.6 FPDP Interface.....	10
1.7 Performance.....	11
1.8 Hardware and Configuration Versions.....	11
V6SSDLF Hardware Description.....	13
1.9 Front Panel.....	13
1.10 On-Board Controls.....	13
1.11 Processor	13
1.12 Interrupts	13
1.13 Memory	14
1.14 V6SSDLF Registers.....	17
1.15 CPLD Registers.....	31
1.16 Feature Descriptions.....	34
1.17 Board Setup	39
1.18 Operating Concepts.....	41
1.19 Flash Memory Data Formats.....	45
V6SSDLF Software Support.....	49
1.20 Host Control Libraries	49
1.21 Utility Programs	60

TABLE OF FIGURES

Figure 1: V6SSDLF Data Paths	8
Figure 2: Hardware Revisions.....	11
Figure 3: CPLD Versions	12
Figure 4: FPGA FIFO Configurations	12
Figure 5: A16 Address switches.....	15
Figure 6: Register Map	17
Figure 7: Status Command Register.....	18
Figure 8: Command Field Bits.....	18
Figure 9: Delay Line State.....	18
Figure 10: Status Field Bits	19
Figure 11: Base Address Register.....	21
Figure 12: VME IRQ Register	21
Figure 13: Snapshot Length/Address Register	21
Figure 14: Output Delay Register or Tap Delay Register	22
Figure 15: LED Control Register Display	22
Figure 16: LED Run Mode Display	23
Figure 17: VME Dump Offset Register	23
Figure 18: Snapshot Control Register	24
Figure 19: Snapshot Modes	25
Figure 20: FPGA Version Register.....	26
Figure 21: Input Address Register	26
Figure 22: Output Address Register.....	26
Figure 23: Dump Address Register.....	27
Figure 24: Memory Size Register	27
Figure 25: Memory Row Numbering.....	27
Figure 26: Memory Extent Values.....	27
Figure 27: FIFO Condition Register	29
Figure 28: Input Inhibit Register	30
Figure 29: CPLD Registers	31
Figure 30: CPLD Version Register	31
Figure 31: FPGA Configuration Control Register	32
Figure 32: FPDP Port Options Registers.....	32
Figure 33: Port Direction Controls.....	36
Figure 34: Relative Dump Details	38
Figure 35: FPDP Clock Termination Signals.....	39
Figure 36: Standard Jumper Configurations	40
Figure 37: Jumper Block Diagrams	40
Figure 38: Flash Memory Segments	45
Figure 39: FPGA Configuration Initial Segment Header	46
Figure 40: Microcode Segment Header	47
Figure 41: Microcode Version ID.....	47
Figure 42: Serial Number Segment Format.....	47
Figure 43: FPGA Configuration Addresses	54
Figure 44: VME ioctl() commands	55
Figure 45: dlyflash Parameters.....	60
Figure 46: dlysetup Options.....	61
Figure 47: dlysetup Commands.....	61
Figure 48: dlysetup Register Names	62

V6SSDL Solid State Digital Delay Line FIFO Version 3

VME Bus 6U Delay Line Product Family Architecture

The V6SSDL product family architecture provides a variety of 2-port and 3-port (tapped) Digital Delay Lines. The initial product in this family is the V6SSDLF, a 3-port delay line using Front Panel Data Port (FPDP) input and output interfaces.

CAC currently supports several functional products based on the V6SSDLF:

- Delay Line configurations, which provide programmable fixed delay of data for two output ports.
- FIFO buffer configurations, which accept data as presented and supply it to a consumer on demand.
- Data packing and unpacking configurations, which support multiple data items packed into, and across, the 32-bit FPDP data items.

Contact CAC for further information on available configurations.

V6SSDLF FPDP FIFO Buffer

This manual documents Release 3 of the V6SSDLF FIFO buffer and, where appropriate, the differences from later releases. ***Bold Italic print (like this sentence) identifies current release restrictions and differences.***

The V6SSDLF is a controllable digital delay device. It accepts 32-bit data items on one of its Front Panel Data Ports, and delivers the same data on an output FPDP on demand. An optional second output, separately controlled, is available on the third FPDP. ***The second output is not supported in this release.***

V6SSDLF Quick Tour

Figure 1 shows the major logical units and data paths in the V6SSDLF. The FPDP ports are on the left, the VMEBus on the right.

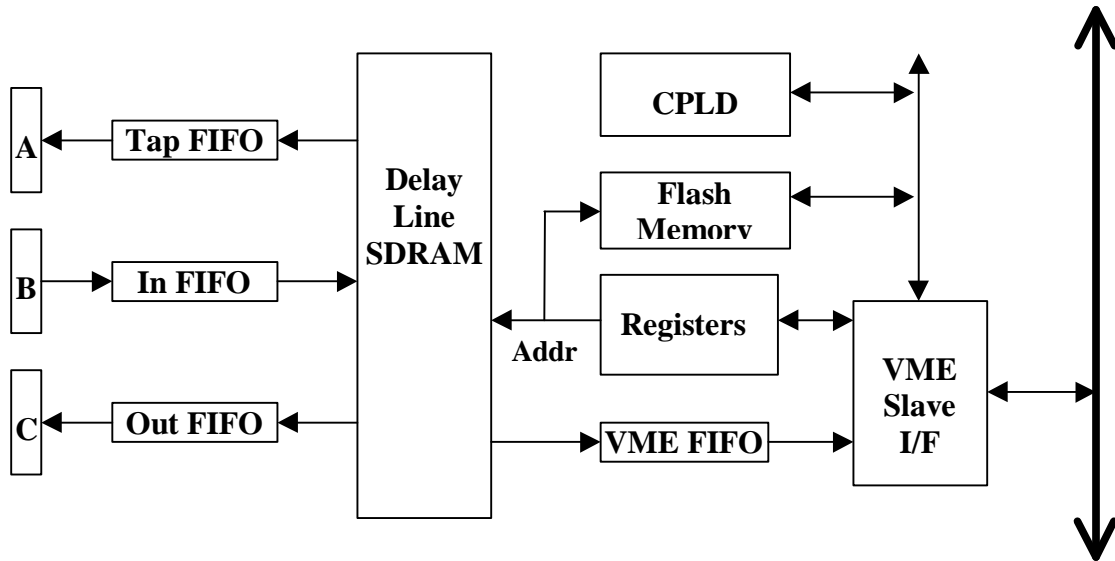


Figure 1: V6SSDLF Data Paths

1.1 Functional Description

The V6SSDLF accepts a continuous data stream, at rates up to 160 MB/sec (4-byte data samples at up to 40 MHz), and deliver it on demand to a second port. An SDRAM array holds a half billion data items (2 GB) with an optional memory extension doubling the memory. Full CRC protection provides single-bit error correction and multi-bit error detection; an error-prone memory element can be logically replaced with a redundant spare. A host interface allows controlling and monitoring the unit, including dumping a snapshot of the data stream for monitoring the data stream, or for later reloading and replaying. An optional second "tap" output can delay the data by an amount independent of the primary output delay.

The current release does not support the secondary (tap) output, port reconfiguration, CRC processing, or snapshot reloading.

The FIFO Buffer does not impose any delay on the data, allowing the consumer of the data to determine how long the data is held in the delay line. (An initial delay, effective only on startup of the delay line, may be set to prevent output until a specified amount of data has been read in.) Input and output are synchronously clocked, but are otherwise independent except when the delay line is empty of data:

- When the consumer asserts the Suspend signal, the V6SSDL de-asserts the Data Valid signal within 8 FPDP clocks.
- Output may continue after input has stopped until the data is exhausted.
- Some data left in an unfilled input buffer when input data stops will not be available for output until more input fills the buffer (128 32-bit data items).

The V6SSDL is a 6U-format VME assembly with three FPDP connectors, for input, output, and optional tap output. A front panel switch provides reset and automatic configuration control. Two front-panel LEDs display run status and host-driven indications.

A Field-Programmable Gate Array (FPGA) is configured from onboard flash memory to provide registers accessible from the VMEBus by the host, FIFO buffers for the FPDP ports and snapshot dump, and a microprogrammed processor, which loads its microcode from flash memory. The processor polls the status of the FIFOs, and transfers data as needed between the memory and the FIFOs. The host can program the flash memory with a new FPGA configuration or microcode over the VMEBus.

Baseboard clock generation can be set at the factory to a variety of clock speeds. The clock settings for a system depend on the system features and FPDP data rates it supports.

1.2 Major Components

Baseboard:

- Two gigabytes of delay line memory and associated buffers
- CPLD for initialization, port configuration, and clock control
- Large FPGA for primary control
- Flash memory for FPGA configuration, microcode, and error correction
- Clock generation and distribution chips
- Mezzanine connector
- JTAG connector for factory initialization

Optional Mezzanine:

- Two additional gigabytes of delay line memory and associated buffers
- CPLD for clock control
- Clock distribution chips

VMEBus interface:

- P1 and P2 backplane connectors
- Base address switches for A16 access

Front Panel Data Ports

- Three FPDP connectors (two standard, one optional)
- Interface chips
- PECL Clock Configuration Jumper Block

Other Front Panel Resources:

- Switch with Momentary/reset, Toggle/TBD positions
- Two three-color LEDs for status indications.

1.3 Hardware Options

- Baseboard memory size
- Memory extension mezzanine board
- Third FPDP
- System and memory clock speeds

1.4 VMEBus Interface

The VMEBus interface directly supports A16/D32 and A32/D32 slave access, including block transfers (BLT). D8 and D16 access is supported, but registers are updated only when all 4 bytes have been written; the volatile Dump Data Register changes when all 4 bytes have been read. There is no support for unaligned data access.

1.5 Host and Module Interrupts

The V6SSDLF module can generate a VME Interrupt for host interactions. The host can assign the IRQ to be used. Another IRQ can be used to synchronize snapshots on multiple V6SSDLF modules.

1.6 FPDP Interface

The V6SSDLF supports either TTL or PECL clocking on the FPDP interface. The output port clock reproduces the input port clock.

The FPDP interfaces support Unframed Data. Input data is accepted unless inhibited by the host. Not-Ready* and Suspend* signals received on the output

port inhibit output until de-asserted. Not-Ready* and Suspend* signals may be asserted on the input port to indicate not-running and FIFO-full conditions.

1.7 Performance

The throughput limits of the V6SSDLF depend on the FPGA configuration, the number of active ports, the memory clock rate, and the FPDP data rates. The current release with a 50 MHz memory clock supports 40 MHz FPDP data rates for two ports, and a concurrent VME data dump at data rates generally limited by the VME host (typically averaging about 1 MB/sec).

1.8 Hardware and Configuration Versions

Four parts of the V6SSDLF have version numbers that identify the capabilities and behavior of the delay line.

- The Hardware Revision (Figure 2) reflects changes in the layout and wiring of the PCB and the components installed on it. The PCB revision is a single digit printed on the top of the PCB. The second digit is the assembly revision; a value greater than 0 indicates a component change or assembly option.
- The CPLD Version Number (Figure 3), readable in the CPLD Version Register, identifies the version of the logic in the CPLD that controls board initialization, memory clock enables, and some FPDP signals. It is programmed at the factory.
- The FPGA Configuration Version (Figure 4) reflects the higher-level functionality of the delay line provided by the FPGA. The flash memory contains areas for three FPGA configurations, any of which can be selected or updated by the host. The FPGA Configuration Register identifies the currently active configuration and its microcode load flash address.
- The FPGA Microprogram version reflects the memory control sequencing and other functionality loaded by the FPGA microprogrammed engine from flash memory. Each FPGA configuration loads its microcode from a distinct area of flash. Microprogram versions may differ in functionality or options. The first few words of the flash memory area identify the microcode version.

H/W Rev	Description	CPLD Ver	Ports	Features
1.0	1st Production	0.1, 0.2, 1.1	3	40 or 50 MHz
3.1	2nd Production	1.3, 1.1	3	50 MHz, transmit jumper, larger CPLD

Figure 2: Hardware Revisions

The memory clock on the baseboard is easily modified to run at a variety of speeds. As built, Revision 1.0 board clocks are 40 MHz and Revision 3.1 board clocks are 50 MHz; many 1.0 boards run at 50 MHz. The memory clock speed

limits the aggregate FPDP data rate. Data errors will result if it is too slow for the data rate demanded by the FPDP behavior.

CPLD Ver	Description	H/W Revs	Features
0.1	Standard	1.0	
0.2	Reversed-order memory rows	1.0	For some partially populated boards
1.1	Recompiled 0.1	1.0, 3.0	PIO bug fixed
1.3	Rev 3 H/W	3.0	Transmit control jumpers

Figure 3: CPLD Versions

CPLD Versions 0.x have a bug affecting the behavior of PIO signals on the FPDP interfaces; the corresponding 1.x versions correct the problem. The new functionality of the Revision 3.0 board is masked by CPLD V1.1, allowing functional equivalence to Revision 1.0/CPLD V1.1 boards.

Release	Memory Clock Max	Ports			New Features
		In	Out	Tap	
FIFO.0	50 MHz	B	C	n/a	First FIFO release
FIFO.2	50 MHz	B	C	n/a	Input Inhibit register D8/D16 VME access
FIFO.3	50 MHz	B	C	n/a	reconfiguration reliability improved memory timing margins

Figure 4: FPGA FIFO Configurations

All hardware/CPLD configurations support all FPGA configurations within the memory clock limits noted.

V6SSDLF Hardware Description

The V6SSDLF consists of a VME 6U circuit board, up to 2 GB of SDRAM, a Field-Programmable Gate Array (FPGA), a Complex Programmable Logic Device (CPLD), and supplementary support devices. The FPGA implements a VMEBus interface, host-accessible registers, and control logic described in the remainder of this section. The CPLD provides initialization control and other support functions.

1.9 Front Panel

The Front Panel of the V6SSDL has the following resources:

- Three FPDP connectors, labeled A (at the top), B (center), and C (bottom). The center port is optional.
- A three-position center-off switch. The up position is momentary contact, which resets the FPGA if held less than two seconds, or reboots the board in its "safe" configuration if held long. The down position is a toggle, with no defined function beyond controlling a bit in the board status register.
- Two three-color LED indicators, L1 and L2, which are under host control through a register accessible from the VMEBus. LED L2 is also controlled by changes in the run status of the board: it turns green for running, yellow for idle, red for halted. A port configuration error will turn both LEDs red.

1.10 On-Board Controls

The V6SSDLF has the following controls and interfaces on the board itself:

- Two hexadecimal rotary switches for setting the VMEBus A16 address.
- A jumper block for each FPDP to determine the FPDP clock handling.
- A JTAG connector for factory initialization of the programmable devices on the board; it should not be needed in the field.

1.11 Processor

The FPGA implements a simple microprogrammable processor, which generates control signals for the Delay Line memory. The microcode is loaded from Flash Memory. The design of the processor and microcode is proprietary in nature and subject to revision. It is not publicly documented.

1.12 Interrupts

The V6SSDLF generates a selectable VME interrupt when it encounters any of the following error conditions:

- CRC failure on an output transfer from memory.
- Incorrect configuration of the FPDPs.

The current release does not generate either condition.

The interrupt indicates to the host that an error condition exists. The host can determine the error status from the Control and Status Register.

The VME interrupt generated (1 through 7) is selected by a field in the VME IRQ Register. If the field is zero, error interrupts are disabled.

An additional VME interrupt line can be dedicated to starting a synchronized snapshot on two or more V6SSDLF boards. The host configures all of the boards to set the dump start point when the interrupt signal is detected, then commands one board to signal the interrupt when a dump is desired. The host should be configured to ignore the interrupt line, since no board will respond to an interrupt acknowledge cycle. While a board is configured for a synchronized snapshot, it cannot interrupt the VME host.

1.13 Memory

This section describes the physical memory arrays and address spaces implemented by the V6SSDLF.

1.13.1 Delay Line RAM

The V6SSDLF baseboard includes up to 2 Gigabytes (GB) of Synchronous Dynamic RAM (SDRAM). The memory is organized in 16 physical rows, each providing 128 MB. The baseboard may be special-ordered with some rows not populated, providing memory sizes of 1 GB and 2 GB. The optional mezzanine provides another 2 GB in 16 rows. The baseboard and mezzanine will support denser memory chips when available, with up to 8 GB on the baseboard.

The Delay Line memory is not directly accessible from the VME Bus. The snapshot dump function allows a region of the memory to be read out to the VME host. An absolute dump starts at a host-specified address. A relative dump starts a host-specified length back from the current input address when the host invokes the snapshot; the host may read the resulting start address to identify the memory location. Relative snapshots of several delay lines in a VME backplane may be synchronized.

The delay line memory is organized into "sample blocks" of 128 FPDP data items (512 bytes). Host access for the snapshot dump uses smaller blocks of 32 samples (128 bytes). All accesses to memory transfer a block of data between

the delay line memory and a FIFO servicing the interface (FPDP or host). The block is aligned in memory with respect to its size (i.e., low address bits zero),

1.13.2 Flash Memory

A 16-megabit Flash Memory chip provides storage for three FPGA configurations, microcode for the control processor implemented in the FPGA, CRC error correction tables, and supplementary data. The memory is organized as 16-bit words, with a 20-bit address space. The memory is segmented, with 31 segments of 32KW (at addresses 0, 8000, ..., F0000) and 8 segments of 4KW (at addresses F8000, F9000, ..., FF000).

Host utility functions support flash memory loading, verification, and dumping. Section 1.19 describes the data formats for flash segments.

1.13.3 VME Address Space

The V6SSDLF is mapped into the VME Bus address space in two ways:

- For A16 access, two rotary hex switches on the baseboard specify the upper 6 bits of the A16 base address. The memory space is thus 1K bytes. The right-hand switch (further from the front panel) supplies the most significant four bits of the address. The high-order two bits of the left-hand switch (closer to the front panel) are the next two bits (i.e., settings 0 through 3 are equivalent). For instance, setting the right switch to B and left switch to 8 sets the base address to 0xB800; the maximum address with this setting is 0xBBFF. The switches are read after each FPGA or board reset. ***(The labels on the Revision 1 board are incorrect; the left-hand switch, labeled "MSB" provides the least-significant two bits of the 6-bit base.)***

Switch Position	Label		Address bits	Switch bits
	Rev 1	Rev 3		
Right	"LSB"	"SW 2"	15:12	3:0
Left	"MSB"	"SW 1"	11:10	3:2

Figure 5: A16 Address switches

- For A32 access, the base address and memory range are determined by a register that the host can set with an A16 access. The upper 16 bits of the register specify the most significant bits of the base address, and the lower 16 bits are a mask specifying which of the most significant bits must match the base address. A base address register value of 3430FFF0, for instance, sets the board up with an address range of 34300000 to 343FFFFF.

In both addressing modes, the VME Memory Space maps the VME Registers into the first 256 bytes, and provides access to the Delay Line memory dump (**and load**) function in the remainder. Any address above the registers (offset hex 100 or higher) is interpreted as the dump/**load** data register.

All registers are defined to be 32 bits wide, and 32-bit reads and writes are the native access methods. For host systems that cannot support 32-bit access, aligned 16-bit (word) and 8-bit (byte) access is supported with limitations reflecting the 32-bit internal structure. A register write takes effect only when all bytes of the new value have been written by sequential partial writes; incomplete partial writes to a register are invalidated by an intervening write to a different register. Similarly, the dump/load Data Register, which provides a new data item each time its value is read, changes only when all bytes have been read.

1.14 V6SSDLF Registers

This section describes a register as a source if the host can read it, and as a destination if the host can modify it by writing to it. Writing to a register that is undefined or read-only, or to bits that are not defined or read-only, has no effect. All undefined registers and bits are zero when read.

Addresses of registers in this section are hexadecimal register numbers. The VME address of a register is its number shifted left two bits to obtain the byte offset plus the base address. All registers are accessible in A16 and A32 address spaces. Figure 6 summarizes the register set.

Reg #	offset	Register Name	Access	In Release
0x00	0x00	Status/Command	Read/Write	Yes
0x01	0x04	Base Address	Read/Write	Yes
0x02	0x08	VME IRQ	Read/Write	Yes
0x03	0x0c	Snapshot Length/Address	Read/Write	Yes
0x04	0x10	FPDP Select	Read/Write	No
0x05	0x14	Output Delay	Read/Write	Yes
0x06	0x18	Tap Delay	Read/Write	No
0x07	0x1c	Chip Substitution	Read/Write	No
0x08	0x20	LED Control	Read/Write	Yes
0x09	0x24	VME Dump Offset	Read Only	Yes
0x0a	0x28	CRC Error Block	Read Only	No
0x0b	0x2c	CRC Error Syndrome	Read Only	No
0x0c	0x30	Debug Control	Reserved	Yes
0x0d	0x34	Snapshot Control	Read/Write	Yes
0x0e	0x38	Snapshot Initiate	Read/Write	Yes
0x0f	0x3c	FPGA Version	Read Only	Yes
0x10	0x40	Input Address	Read Only	Yes
0x11	0x44	Output Address	Read Only	Yes
0x12	0x48	Dump Address	Read Only	Yes
0x13	0x4c	Memory Size	Read/Write	Yes
0x14	0x50	Snapshot Start Address	Read Only	Yes
0x15	0x54	Snapshot End Address	Read Only	Yes
0x16	0x58	Tap Address	Read Only	Yes
0x17	0x5c	FIFO Condition	Read/Write	Yes
0x20	0x80	Flash Address	Read/Write	Yes
0x21	0x84	Flash Data	Read/Write	Yes
0x22	0x88	CPLD Address	Read/Write	Yes
0x23	0x8c	CPLD Data	Read/Write	Yes
0x2e	0xb8	FPDP Input Inhibit	Read/Write	Yes
0x40	0x100*	Dump/Load Data	Read/Write	Yes

Figure 6: Register Map

*All addresses within either assigned address space (A16 and A32) that are at least 0x100 bytes greater than the base address are equivalent, addressing the Dump/Load Data Register. This allows host VME controllers that increment the read address for multiple reads to transfer a snapshot dump as though it were stored in memory addressable from the VMEBus, incrementing the address for each read operation.

Other registers may be implemented for debugging and error diagnosis, but their definitions are subject to revision and users should not rely on values of registers not specified in this section.

1.14.1 Status/Command Register (0x00)

The Status/Command Register is a 32-bit source and destination, logically consisting of two 16-bit fields. Figure 7 shows the layout of the register.

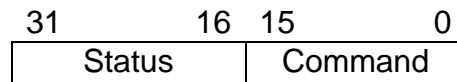


Figure 7: Status Command Register

The Command field is the least significant 16 bits of register 0. Figure 8 describes the interpretation of the bits. Initialization is as noted on FPGA Reset.

Bit #	Name	Binary Value(s)	Init
2	Global Halt	0 (enabled), 1 (halted)	1
0	Run	0 (idle), 1 (running)	0

Figure 8: Command Field Bits

Logic prevents both bits from being set, so there are three run states: halted (Global Halt set), idle (neither set), or running (Run set). Figure 9 summarizes the effects of each state.

Value	State	Input	Output	Memory	LED 2	VME Access
0x04	Halted	ignored	tri-stated	no refresh	red	CPLD, Flash
0x00	Idle	ignored	not valid	refresh only	yellow	snapshot
0x01	Running	accepted	valid (*)	running	green	snapshot

Figure 9: Delay Line State

The halted state allows initialization and maintenance of the delay line. The FPGA microprogram is stopped, allowing access to the CPLD and flash memory from the VMEBus interface. Transition out of the halted state is conditioned on

the output ports being configured consistently. Details of state characteristics and transitions are discussed in Section 1.16.4.

The idle state provides stable output port signals and snapshot dump support without running the delay line function. The delay line accepts no input data, and provides none to the output port. Data previously accepted may be dumped

In the running state, the delay line accepts input unless inhibited, and outputs data when data is available and the output port Not-Ready* and Suspend* signals allow it.

The Status field is the most significant 16 bits of register 0. Figure 10 describes the interpretation of the bits. If a bit is identified as writable, the hardware sets it to 1 when the corresponding condition occurs; and clears it to 0 when the host writes a 1 to the bit. The register is initialized as noted on Board Reset.

Bit #	Name	Writable	Binary Value(s)	Init
30	Bad Check State	No	0 (ok), 1 (error)	0
29	Bad Output Port State	No	0 (ok), 1 (error)	0
27	FIFO Empty	No	0 (data), 1 (empty)	1
26	FIFO Full	No	0 (space), 1 (full)	0
24	Front Panel Toggle	No	0 (ctr), 1 (down)	0
23	Micro Engine Error	No	0 (ok), 1 (error)	
22	Input Overflow	No	0 (ok), 1 (error)	0
21	FIFO Timeout Error	No	0 (ok), 1 (error)	0
20	FIFO Load Error	No	0 (ok), 1 (error)	0
19	Overflow Error	No	0 (ok), 1 (error)	0
18	Underflow Error	No	0 (ok), 1 (error)	0
17	CRC Interrupt	Yes	0 (no), 1 (yes)	0
16	Configuration Interrupt	Yes	0 (no), 1 (yes)	0

Figure 10: Status Field Bits

The port configuration test invoked when the host writes a zero to the Global Halt bit may set one of the error bits. The Bad Output Port State bit indicates the port is not correctly configured, and the Bad Check State bit indicates an internal failure of the test logic.

The FIFO Empty bit is true whenever the output has consumed all the data that has been written to memory from the input.

The FIFO Full bit is true when the input data fills the delay line memory to the output pointer.

The Front Panel switch has three positions: the central neutral position, a momentary contact position (up), which resets the FPGA or the board, and a

toggle position (down) which is reflected in bit 24. No further functionality is currently defined for the toggle.

The Micro Engine Error bit indicates failure of the microprogrammed engine to initialize properly. This may indicate that the microcode is not readable in flash memory at the load address (high 8 bits defined by the low byte of the Version Register, the rest 0), or the microcode and the FPGA version are incompatible.

The Input Overflow bit indicates a failure of the delay line to keep up with the input FPDP data. This usually indicates that the memory clock speed, FPDP clock speed, and board configuration (FPGA configuration and microcode versions) are not compatible, or that the microcode has not loaded correctly. The flag is set when a buffer is used for input before all its data is sent to memory; it may not indicate actual data errors. It is reset on restart (return to running state).

The FIFO Timeout Error bit is set if an access to the snapshot dump FIFO fails to complete before a watchdog timer goes off. It reflects failure to initiate the snapshot or an internal failure in the dump mechanism.

The FIFO Load Error bit is set by a VME write to the load/dump data register. ***The VME load function is not implemented in the current release.***

The Overflow Error bit is set if input data is written to memory when the FIFO Full bit is true, to indicate that the data pending for output has been overwritten. It is reset when the delay line is restarted (Run bit turned off and then on).

The Underflow Error bit is set if output data is read from memory when the FIFO Empty bit is true, to indicate that the output data stream has been corrupted. It is included as an indicator of internal logic failure, which should not occur. The bit is reset when the delay is restarted.

The CRC Interrupt bit is set if a CRC error causes an Interrupt. The Configuration Interrupt bit is set if an invalid port configuration is written to the Configuration Register. ***Neither function is implemented in the current release: both bits are therefore always 0.***

1.14.2 Base Address Register (0x01)

The Base Address Register is a source and destination that defines the region of the VME A32 address space that is serviced by the V6SSDLF. The upper 16 bits specify the base address and the lower 16 bits are a mask of the bits that are to be matched. The mask size determines the power-of-2 size of the address space, from a minimum of 64K bytes (all bits on). The mask should be ones from the most significant bit (15) to the least significant bit to be matched.

bits	field
31:16	base address value
15:0	base address mask

Figure 11: Base Address Register

1.14.3 VME IRQ Register (0x02)

The VME IRQ Register is a source and destination containing the VMEBus IRQ for the board, and an interrupt vector, returned by the board in an interrupt acknowledge cycle. Setting the IRQ field to zero disables the ability of the board to generate interrupts. ***The current release does not generate any interrupts and the effects of this register are undefined.***

bits	field
18:16	VME IRQ number
7:0	Interrupt vector

Figure 12: VME IRQ Register

1.14.4 Snapshot Length/Address Register (0x03)

The Snapshot Length/Address Register is a source and destination register that defines the start of a snapshot of delay line memory to be dumped. For relative dumps, its value is the desired offset from the current input pointer. For absolute dumps, its value is the starting address of the snapshot. In either case, the value is a number of dump blocks (32 data items), aligned as a byte offset. The way the dump is initiated determines the absolute/relative mode. The snapshot dump mechanism is discussed in sections 1.16.5 and 1.18.3.

bits	function
31:7	Snapshot Relative Offset or Absolute Address

Figure 13: Snapshot Length/Address Register

1.14.5 FPDP Select Register (0x04)

The FPDP Select Register (44) is a source and destination register containing configuration data for the Front Panel Data Ports. A field for each port specifies whether it is the input, output, tap, or unused. ***The current release has fixed port assignments and the FPDP Select Register is not implemented.***

1.14.6 Output Delay Register (0x05)

The Output Delay Register is a source and destination register containing the number of sample blocks (128 samples, or 512 bytes) the primary output is initially delayed behind the input. The value is aligned as a byte offset, with the low nine bits zero. The delay should be less than the memory size specified in the Memory Size Register. If the register is 0, output is available immediately after data starts reaching the memory.

bits	function
31:9	Output delay in sample blocks

Figure 14: Output Delay Register or Tap Delay Register

1.14.7 Tap Delay Register (0x06)

The Tap Delay Register is a source and destination register containing the number of sample blocks the secondary (tap) output is to be behind the input.

The tap is not implemented in this release, and the Tap Delay Register is always 0.

1.14.8 Chip Substitution Register (0x07)

The Chip Substitution Register is a source and destination register containing the physical location of the delay line memory chip to be replaced, and an flag to enable substitution of the spare chip for the one selected.

The chip substitution function has not been tested in the current release, and the Chip Substitution Register has an undefined effect.

1.14.9 LED Control Register (0x08)

The LED Control Register is a source and destination register containing control fields for the two front-panel LED indicators. Each LED is red if only its red bit is set, green if only its green bit is set, yellow if both are set, and off if neither is set.

bit	function
3	L2 Green
2	L2 Red
1	L1 Green
0	L1 Red

Figure 15: LED Control Register Display

LED L2 also changes with the board's run state, as shown in Figure 16.

halt	run	state	L2
1	0	halt	red
0	0	idle	yellow
0	1	run	green

Figure 16: LED Run Mode Display

L2 reflects the most recent change (either a write to the LED control register or a state change). L2 changes caused by state changes are intended for visual indications, and are not reflected in the LED control register. Both L1 and L2 turn red if a port configuration error is detected when attempting to leave the halt state.

1.14.10 VME Dump Offset Register (0x09)

The VME Dump Offset Register is a source register containing the byte offset of the start of a relative snapshot in the resulting dump. While Dump Blocks are aligned on 32-item (128 byte) boundaries, the start of a relative snapshot may occur at any sample within a block. The item at this offset in the first block is the one at the specified length from the input pointer when the snapshot signal was received. This register is not set on absolute dump initiation. The snapshot dump mechanism is discussed in sections 1.16.5 and 1.18.3.

bits	field
6:2	item offset

Figure 17: VME Dump Offset Register

1.14.11 CRC Error Block Register (0x0a)

The CRC Error Block Register is a source register containing the memory block index of the most recent CRC error. ***CRC checking is not supported in the current release.***

1.14.12 CRC Error Syndrome Register (0x0b)

The CRC Error Syndrome Register is a source register containing the syndrome associated with the most recent CRC error. ***CRC checking is not supported in the current release.***

1.14.13 Debug Control Register (0x0c)

The Debug Control Register is a source and destination register containing an index to select debugging data presented at test points on the board. It is not documented, as the information is subject to continual revision and is not a supported feature of the delay line. The register value generally has no other effect, but setting it should be avoided.

1.14.14 Snapshot Control Register (0x0d)

The Snapshot Control Register is a source and destination register specifying the initiation mode of the VMEBus snapshot facility. Absolute snapshots are not affected by this register.

bits	field	values
6:4	IRQ for snapshot sync	0: local initiation only 1-7: use IRQ to initiate
3	Dump/Load Mode	0: dump 1: load
0	Freeze on Snap	0: continue after snap 1: freeze on initiation

Figure 18: Snapshot Control Register

If the IRQ field is set to a non-zero value, the corresponding backplane interrupt request line is used to signal a synchronized multi-board relative snapshot. The host should be configured to ignore the line. All delay line boards with the same IRQ selected will trigger their relative snapshot dumps when the line goes low. Any of the boards may initiate the synchronized snapshot.

When the IRQ field and Snapshot Length (register 3) are non-zero, the delay line is unable to generate host interrupts for any other reason (CRC error or configuration error). In order to allow CRC error monitoring by the host, the snapshot control register should be kept zero except when a snapshot is required. **No host interrupts are implemented in the current release.**

The load function is not implemented in the current release, and the Dump/Load Mode bit is always 0.

If the Freeze on Snap bit is set, the delay line will stop when a snapshot is initiated: both the Freeze on Snap bit and the Run bit (in the Status/Control Register) will be zero after snapshot initiation. If the Freeze on Snap bit is zero, the delay line state will be unaffected by snapshot initiation.

1.14.15 Snapshot Initiate Register (0x0e)

The Snapshot Initiate Register is a destination register written by the host to define a snapshot of delay line memory to be dumped. The low bit of the value written and the value of the IRQ Selector in the Snapshot Control Register select one of three snapshot modes (summarized in Figure 19):

- If the low bit of the value written is one, the delay line initiates a local absolute snapshot. No other boards are affected.
- If the low bit of the value written is zero, and the IRQ Selector is zero, the delay line initiates a local relative snapshot. No other boards are affected.
- If the low bit of the value written is zero and the IRQ Selector is non-zero, the delay line initiates a synchronized relative snapshot by driving the selected IRQ line low.

There is no synchronized absolute snapshot, since the input pointers of delay lines are not synchronized.

Low Bit	IRQ Sel	Scope	Address Mode
1	any	local	absolute
0	0	local	relative
0	non-0	synchronized	relative

Figure 19: Snapshot Modes

The address modes differ in how they determine the snapshot start address:

- An absolute snapshot uses the value in the Snapshot Length/Address Register as the start address.
- A relative snapshot uses the value in the Snapshot Length/Address Register as an offset from the input sample address. The start address of the snapshot is the eventual memory address of the next sample received on the input port after the initiation signal minus the snapshot length.

The scope determines the snapshot initiation mechanism and timing:

- A local snapshot is initiated directly by the register write and starts only on the delay line whose Snapshot Initiate Register the host writes.
- A synchronized snapshot is initiated by the selected IRQ line going low; the snapshot is defined simultaneously on all delay lines waiting for the IRQ signal. The IRQ signal is clocked by the FPDP clock to minimize the effect of backplane propagation delays.

The synchronized snapshot mechanism does not use the IRQ line as a regular VME interrupt signal, and all other boards in the backplane should ignore it.

The snapshot dump mechanism is discussed in sections 1.16.5 and 1.18.3.

1.14.16 FPGA Version Register (0x0f)

The FPGA Version Register is a source register containing the version and revision number of the FPGA configuration, and the high eight bits of the microcode load address in flash memory. The configuration is identified by the high six hex digits, e.g., F1F0.03).

bits	field	Value
31: 16	FIFO version indicator	0xF1F0
15:8	revision index	0x03
7:0	MSB of microcode address in flash	0xfa

Figure 20: FPGA Version Register

1.14.17 Input Address Register (0x10)

The Input Address Register is a source register containing the address in delay line memory of the next sample block to be written from the input FPDP FIFO. It is typically several sample blocks behind the actual data at the input FPDP. The register is reset to zero each time the delay line enters running mode.

bits	function
31:9	Address in DL memory of next input sample block

Figure 21: Input Address Register

1.14.18 Output Address Register (0x11)

The Output Address Register is a source register containing the address in delay line memory of the next sample block to be read next to the output FPDP FIFO. It is typically several blocks ahead of the actual data being sent on the output FPDP. The register is reset to zero each time the delay line enters running mode.

bits	function
31:9	Address in DL memory of next output sample block

Figure 22: Output Address Register

1.14.19 Dump Address Register (0x12)

The Dump Address Register is a source register containing the address in delay line memory of the next dump block to be read to the VMEBus snapshot FIFO. It

is typically several blocks ahead of the data being read on the VMEBus by the host. The register is set to the dump start address each time a new snapshot is initiated.

bits	function
31:7	Address in DL memory of next VME dump block

Figure 23: Dump Address Register

1.14.20 Memory Size Register (0x13)

The Memory Size Register is a source and destination register specifying the base row and extent of the physical memory that the board uses. It allows the use of partially populated boards, and can be set up to exclude one or more contiguous rows of memory chips. Figure 24 shows the fields in the register.

Bits	Function	Description	Initial Value
12:8	base row	base row number	0x00
4:0	extent	number of rows - 1	0x0f

Figure 24: Memory Size Register

A fully populated V6SSDLF has 32 rows of memory, 8 rows each on the top and bottom of the baseboard and mezzanine. Each row is 128 Megabytes (32M 32-bit samples). Figure 25 shows the normal default row numbers. Specific FPGA or CPLD versions may alter the mapping.

Row	Top First	Top Last	Bottom First	Bottom Last
Baseboard	0 (0x00)	7 (0x07)	8 (0x08)	15 (0x0f)
Mezzanine	16 (0x10)	23 (0x17)	24 (0x18)	31 (0x1f)

Figure 25: Memory Row Numbering

All visible addresses are zero-based, and address calculations wrap around to zero when the high 5 bits would otherwise exceed the extent value. Figure 26 shows extent values for "power-of-2" memory sizes, the resulting delay line size, and corresponding maximum address.

Extent	Mem size	Max address	Application
1f	4096 MB	FFFFFFFF	Baseboard + Mezzanine
0f	2048 MB	7FFFFFFF	Baseboard Only
07	1024 MB	3FFFFFFF	Baseboard Top or Bottom
03	512 MB	1FFFFFFF	4 rows
01	256 MB	0FFFFFFF	2 rows
00	128 MB	07FFFFFF	single row

Figure 26: Memory Extent Values

Address 0 maps to the base row. The Base Row field can be set to a non-zero value to move the physical location of the zero address on the board. If the extent is less than the available memory, the base row selects the active subset of memory; it can be used to avoid a bad memory row or rows.

1.14.21 Snapshot Start Address Register (0x14)

The Snapshot Start Address Register is a source register containing the byte address of the first sample in the most recent relative snapshot. The first block in the snapshot dump contains the data at this address. The register is set as part of relative snapshot initiation, and is not affected by absolute snapshots.

The snapshot dump mechanism is discussed in sections 1.16.5 and 1.18.3.

1.14.22 Snapshot End Address Register (0x15)

The Snapshot End Address Register is a source register containing the byte address of the last sample in the most recent snapshot. The sample addressed is the first received on the Input FPDP after the relative snapshot initiation signal. The register is set as part of relative snapshot initiation, and is not affected by absolute snapshots.

The snapshot dump mechanism is discussed in sections 1.16.5 and 1.18.3.

1.14.23 Tap Address Register (0x16)

The Tap Address Register is a source register containing the address in delay line memory of the next sample block to be read next to the tap (secondary output) FPDP FIFO. ***The tap is not implemented in this release, and Tap Address Register is always 0.***

1.14.24 FIFO Condition Register (0x17)

The FIFO Condition Register is a source and destination register containing enable bits for upstream signaling on the input FPDP port, and a duplicate source of FIFO status bits. If enabled, the Not-Ready* signal is asserted when the delay line is halted or idle. If enabled, the Suspend* signal is asserted when the delay line is running and the FIFO Full condition is true. If not enabled, each signal remains unasserted.

bit	function	Initial value
29	Overflow Error Status	0
28	FIFO Full Status	0
25	Underflow Error Status	0
24	FIFO Empty Status	1
4	Enable Not-Ready	0 (disabled)
0	Enable Suspend	0 (disabled)

Figure 27: FIFO Condition Register

1.14.25 Flash Address Register (0x20)

The Flash Address Register is a source and destination register containing a twenty-bit flash memory address. Reading or writing the Flash Data Register with the delay line halted will read or write a 16-bit word the location specified.

1.14.26 Flash Data Register (0x21)

The Flash Data Register is a source and destination register containing a 16-bit data value read from or written to flash memory. If the delay line is halted, reading the Flash Data Register will return the value of the flash memory location addressed by the Flash Address Register; writing the Flash Data Register will write the data presented to that location. If the delay line is not halted, a read will return the hex value "BADACD1F" and a write will be ignored.

1.14.27 CPLD Address Register (0x22)

The CPLD Address Register is a source and destination register containing an 8-bit CPLD register address. Reading or writing the CPLD Data Register with the delay line halted will read or write the CPLD Register specified.

1.14.28 CPLD Data Register (0x23)

The CPLD Data Register is a source and destination register containing an 8-bit data value read from or written to a CPLD register. If the delay line is not halted, a read will return the hex value "BADACD1F"; a write will be ignored.

1.14.29 FPDP Input Inhibit Register (0x2e)

The FPDP Input Inhibit Register is a source and destination register containing a flag to keep the input FIFO from accepting any data from the input port. Output

may continue while data is available. This feature allows the host to stop the delay line temporarily, consume the data in the delay line, and restart it without resetting the input and output pointers. Figure 28 describes the register.

Bit	Field	Significance	Init
0	input inhibit	1: inhibit input 0: accept input	0

Figure 28: Input Inhibit Register

1.14.30 Dump/Load Data Register (> 0x3F)

The Dump/Load Data Register is a source register containing the next 32-bit word in a snapshot dump, **and a destination register for the next word of a snapshot load**. If the register is read before a dump is started, or the delay line is halted, or the internal FIFO mechanism fails, the hex value "DEADF1F0" is read. ***The snapshot load function is not implemented in the current release.***

1.15 CPLD Registers

The CPLD Registers are 8-bit (or smaller) registers within the CPLD on the baseboard. The CPLD registers are initialized during board initialization (power-up and "long" resets), and retain their values through FPGA resets ("short" resets) and reconfigurations. Figure 29 summarizes the CPLD Registers.

CPLD registers are accessible from the VMEBus when the delay line is halted. Access is indirect, through the CPLD Address Register and CPLD Data Register; the register number is the value to be written to the CPLD Address Register.

Reg #	Register Name	Access
0	CPLD Version	Read Only
1	Configuration Control	Read/Write
4	FPDP Clock Select	Read/Write
5	Port A Options	Read/Write
6	Port B Options	Read/Write
7	Port C Options	Read/Write

Figure 29: CPLD Registers

1.15.1 CPLD Version Register (0x0)

The CPLD Version Register is an 8-bit source register containing the CPLD version number. The CPLD Version is set at the factory and cannot be upgraded in the field. CPLD Versions are summarized in Section 1.8.

bits	field	meaning
7:4	version	significant change in CPLD
3:0	revision	varies, often board rev level

Figure 30: CPLD Version Register

1.15.2 FPGA Configuration Control Register(0x1)

The FPGA Configuration Control Register is a 3-bit source and destination register containing an FPGA configuration Select field and a Configure bit. The Select field identifies the starting address in flash memory of the configuration string for the FPGA. The Configure bit is transient, causing the FPGA to be reconfigured when set. The Select field survives into the new configuration, and may be read to confirm the source of the active configuration. (Some boards with CPLD Version 0.1 may not enter configuration 1 reliably.)

bits	name	value	meaning
2	Configure	1	start FPGA reconfiguration
		0	no operation
1:0	Select	00	Configuration 0 (80000)
		01	Configuration 1 (a0000)
		10	Configuration 2 (c0000)
		11	same as 00

Figure 31: FPGA Configuration Control Register

1.15.3 FPDP Clock Select Register (0x4)

The FPDP Clock Select Register is a one-bit source and destination register containing a flag selecting the FPDP PECL clock if 1, the FPDP TTL clock if 0. If the TTL clock is selected, the Port Option Register for the input port must enable the TTL clock input. The default value is 1, selecting the PECL clock. The selected clock determines input and output port clocking.

1.15.4 FPDP Port Options Registers (0x5-0x7)

The FPDP Port Options Registers are source and destination registers; each contains configuration controls and status for one FPDP:

- Register 5 controls FPDP A.
- Register 6 controls FPDP B.
- Register 7 controls FPDP C.

The bits in the registers are defined in Figure 32.

bit	name	if 0	if 1	default	Notes
7	TTL Clock	enable	disable	1	identifies input clock
6	Transmit Select (see note)	Rx	Tx	0	Rev 1, Rev 3 hardware
				input	Rev 3 hardware only
5	DIR* signal out	Tx	Rx	1	
4	DIR* signal in	Tx	Rx	input	read-only
3	PIO Data 1			0	bit 1 controls Tx/Rx
2	PIO Data 0			0	bit 0 controls Tx/Rx
1	PIO DIR 1	Rx	Tx	0	
0	PIO DIR 0	Rx	Tx	0	

Figure 32: FPDP Port Options Registers

The CPLD implements parts of the FPDP interface, including TTL input clock selection, DIR* and PIO signal handling, and buffer direction control. While the delay line is in its initial halted state, the host must set the options for each active port compatibly with its use by the FPGA configuration. Since the CPLD registers are not affected by FPGA reconfiguration, care should be taken to avoid

conflicts when shifting from one FPGA configuration to another with different port assignments or changing the assignments in the FPDP Select Register.

The TTL Clock bit should be 0 for output ports or if PECL clocking is desired. It should be 1 for the input port if TTL clocking is desired. If the FPDP Clock Select Register also selects TTL clocking, the input port TTL clock will then determine all FPDP clocking on the board. (PECL clock selection is covered in Sections 1.16.2 and 1.17.2.)

The Transmit Select bit determines the direction of the data buffers for the port. In CPLD versions 0.1, 0.2, and 1.1, intended for Revision 1 hardware, the host must write the Transmit Select bit to indicate whether the port is a transmit port or a receive port. Revision 3 hardware provides a jumper input to identify transmit ports; CPLD Version 1.3 makes the Transmit Select bit a read-only indication of the jumper configuration. CPLD Version 1.1 may be specified for Revision 3.0 hardware to provide functional compatibility with Revision 1.0 hardware.

The direction of each active port is set in the FPDP Select Register (***or assumed, in the current release***). For each output port, the Transmit Select bit in the corresponding FPDP Port Options Register must be 1 when the delay line is enabled (0 written to the Global Halt bit in the Control/Status Register). Otherwise (if Transmit Select is 0 for an output port), the port configuration check will keep the Global Halt bit set, preventing damage to the buffer chips for the port.

The DIR* signal for a transit port is driven low by the CPLD if the corresponding DIR* Signal Out bit is 0; otherwise it is pulled high by the termination circuitry unless driven low by the remote port. DIR* Signal In reflects the actual signal level; if it is 0 while DIR* Signal Out is 1, the remote port is configured to transmit on the port. The host can check for this condition before setting DIR* Signal Out.

A fault in CPLD Version 0.1 causes the PIO bits to behave in a non-standard fashion: when a PIO Direction bit is set to transmit (1), the corresponding signal changes value (0 or 1) with each FPDP clock, regardless of the PIO Data bit value specified. The fault is corrected in CPLD Version 1.1.

1.16 Feature Descriptions

This section provides topical information on features of the V6SSDLF, including some not yet available in standard configurations. For information on availability, contact CAC.

1.16.1 Clocks and Data Rates

The V6SSDLF supports three clocks:

- The system clock, generated on the board, is typically set to 40 MHz or 50 MHz. It drives the CPLD logic, configuration, and ancillary operations.
- The memory clock, or high-speed clock, is derived from the system clock, and is often the same frequency. It drives memory control and data lines, the microprogrammed engine, and VME access. For applications with very high data rates, the memory clock is designed to run up to 75 MHz.
- The FPDP clock is recovered from the clock received on the input port and propagated to output ports. The FPDP specification limits it to no more than 40 MHz. It drives FPDP transfers and synchronization across boards.

The memory throughput is determined by the memory clock speed, the width of the memory data bus (8 bytes), and the structure of the memory sequencing. Synchronous DRAMs transfer data at memory speeds after a start-up delay. The V6SSDLF blocks FPDP data to reduce the overhead from start-up, refresh, and VME dump data to less than 10% for current configurations. A 50 MHz memory clock typically supports a memory bandwidth of 350 MB/sec. A 75 MHz memory clock, which requires more pipelining of the data and control paths, would push the limit well over 500 MB/sec.

FPDP throughput is a function of the FPDP clock speed, the duty cycle of the data valid signal, and the number of active ports. A single-output configuration fed with a 40 MHz FPDP clock and 100% valid data requires a memory bandwidth of 320 MB/sec (2 ports * 4 bytes/cycle * 40 MHz). A tapped delay line with a 33 MHz FPDP clock and 90% valid data comes in at about 356 MB/sec.

A V6SSDLF will function properly as long as the total FPDP data rate does not exceed the available memory bandwidth. If the FPDP data rate exceeds the memory bandwidth, an Input Overflow error, reflected in the Status/Command Register, will typically result.

1.16.2 FPDP Clock Controls

FPDP clocking is determined by the strobe signals on the input port. At speeds up to 20 MHz, the TTL strobe signal may be used; higher speeds require the PECL strobes. Two levels of configuration on the board determine the clocking.

- At the top level, the CPLD Clock Select Register determines whether the input TTL clock or the input PECL clock drives the FPDP clock circuits.
- For the TTL clock at the second level, the CPLD Port Option Register for the input port must enable the TTL input clock (STROBE signal). If the PECL clock is selected at the top level, the input TTL clock, if enabled, is ignored.
- For the PECL clock at the second level, the jumpers for the input port must be configured to drive the clock selection with the input PSTROBE pair (R or RM in Figure 37). If the TTL clock is selected at the top level, the input PECL clock is ignored.

The clock selected at the top level drives all of the FPDP logic.

Output clocks are driven from the corresponding input clock:

- The TTL STROBE signal of each output port is driven from the selected FPDP clock if the Port Option Register does not enable its TTL STROBE as the input clock. All output ports drive the TTL strobe even if PECL clocking is used.
- The PECL PSTROBE pair is driven from the input PECL clock if the jumpers connect them to the port clock driver (TM in Figure 37). If the input port PSTROBE lines are not driven, or if they are not connected with the R or RM jumper configuration, the output PSTROBE lines will not have a useful clock signal.

1.16.3 FPDP Direction

Each port has a set of bi-directional buffers between the port connector and the FPGA or CPLD pads. Several distinct controls determine the direction of the signals on each port:

- The FPGA configuration and the Port Configuration Register (if implemented) determine whether the FPGA drives signals to the buffers or receives signals from them.
- The Transmit Select bit in the CPLD Port Options Register determines whether the buffers drive data out from the FPGA to the port connector or in from the connector to the FPGA.
- The jumper block for the port determine whether the PECL clock buffer is driving or accepting clocks on the PSTROBE lines.
- The TTL Clock Enable bit in the CPLD Port Options Register determines whether the buffer drives the TTL clock in or out.
- PIO controls in the CPLD Port Options Register determine whether the buffers drive either of the PIO lines.

The table in Figure 33 shows the valid combinations of controls for various port configurations

Port direction	Input	input	output	output	Comment
Strobe used	TTL	PECL	TTL	PECL	TTL always present
FPGA direction	in	in	out	out	
Transmit Select	0	0	1	1	See note
Jumpers	none	R/RM	none	TM	
TTL Enable	1	0	-	-	Port Option Reg
Clock Select	0	1	-	-	global

Figure 33: Port Direction Controls

Note: CPLD Version x.1 and Version 1.3 differ in how they determine the Transmit Select bit. Versions 0.1 and 1.1, which run on all hardware Revs, require the host to write the Transmit Select bit in the Port Options Register for output ports. Version 1.3, which runs only on Rev 3 hardware, uses a jumper to specify Transmit Select.

Problems occur when inconsistent controls are specified, or if ports are connected incorrectly:

- Transmit Select 0 and FPGA port direction out will cause the buffers and FPGA to drive the data lines against each other. The FPGA logic prevents this case.
- Transmit Select 1 and FPGA port direction in will cause neither to drive the data lines, letting them float.
- Connecting ports that both drive the data lines (Transmit Select 1), PECL Strokes (jumpers), or either PIO line (PIO bits) will cause potentially damaging contention.

1.16.4 Run States and Transitions

The halted state allows maintenance and configuration of the delay line:

- The FPGA microprogram is stopped, allowing access to the CPLD and flash memory from the VMEBus interface.
- The delay line memory is not refreshed, so delay line memory content is not accessible and is not preserved.
- Outputs to the FPDP buffer chips are tristated to avoid potentially damaging contention. Boards receiving data from these ports may see misleading levels.

The idle state allows VME access to delay line memory and provides stable output levels, without running the delay line function:

- The FPGA microprogram runs, preventing access to the CPLD and flash memory.
- Delay line memory is refreshed, and dump data is transferred
- Input data is ignored.

- Outputs to the FPDP output port buffer chips are asserted, specifying no valid data. Boards receiving data from these ports see valid levels, no valid data.

The running state adds FPDP data processing:

- As in idle state, CPLD and flash memory are inaccessible; refresh and dump proceed.
- Input data is received and stored in the delay line memory.
- Output data is produced for each output port after the delay specified for it.

When the Global Halt bit is 1 and the host writes a zero to it to leave the halted state, the delay line checks the CPLD registers that control the ports specified as output. If a port specified as output does not have its Transmit Enable bit set, enabling the port would cause the port's buffer chips to drive against the FPGA outputs, causing immediate damage. In this case, or if the checking logic is unable to verify the port status or fails in some other way, the delay line remains in the halted state. Both LEDs become red; the Status field identifies the fault.

If all output ports are properly configured, the Global Halt bit is reset and the outputs are enabled, establishing valid logic levels and driving data valid high (DVALID\ unasserted) to indicate absence of output data. The running state results if the corresponding bit is set in the write value.

All run state transitions are supported and safe, as described above. Each assignment to Register 0 sets LED 2 to reflect the run state.

1.16.5 Snapshots and Dumps

Relative snapshots allow the host to examine data received for a specified interval prior to initiating the snapshot. The host can specify the interval by:

- converting the time interval to memory length (seconds * FPDP frequency * 4 bytes * data valid duty cycle),
- rounding the result up to a 128-byte boundary and writing that value to the Snapshot Length/Address Register, and
- signaling the end of the interval by writing 0 to the Snapshot Initiate Register.

The delay line computes the starting address of a relative snapshot by subtracting the Snapshot Length/Address Register from the effective input pointer when the snapshot is initiated. Thus setting the Snapshot Length/Address Register to 0x1000000 will cause the snapshot dump to start at the sample block containing the sample 0x400000 (4,194,304) samples before the current input sample. The input pointer is captured at the first FPDP clock after the initiation logic is invoked. A relative snapshot will not start if a running FPDP clock is not selected by the clock configuration controls (CPLD Clock Select Register and FPDP Port Options Registers or PECL clock jumpers).

The VME dump FIFO starts filling immediately after initiation. The offset must exceed the input buffer size (512 bytes) to assure that current data is dumped.

Three registers are set when a relative snapshot is initiated:

- The Snapshot End Address Register (0x15) contains the byte address of the first sample read in after the initiate signal.
- The Snapshot Start Address Register (0x14) contains the byte address of the first sample in the snapshot.
- The Dump Offset Register (0x09) contains the byte offset of the first sample in the first sample block available to the host.

Figure 34 shows the relationship of the registers to the input pointer at the time a relative snapshot is initiated, showing the memory in dump buffer units.

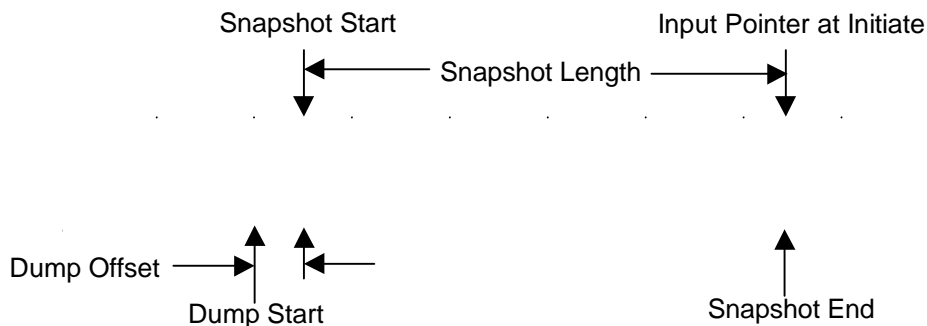


Figure 34: Relative Dump Details

Absolute snapshots allow the host to examine data in a specified part of delay line memory. The host writes the start address to the Snapshot Length/Address Register and writes an odd value to the Snapshot Initiate Register. The snapshot starts at any address specified, including zero. No FPDP clock input is needed for an absolute snapshot dump. The Snapshot Start Address Register, Snapshot End Address Register and Dump Offset Register are not affected by an absolute snapshot. The freeze option is not available for absolute snapshots.

1.17 Board Setup

Before inserting the V6SSDLF board into the backplane, the configuration controls on the board should be set for the application environment. The A16 base address switches should match the VME address allocated to the board from the host perspective, and the FPDP PECL Clock jumpers should match the FPDP connections to be made.

1.17.1 VME A16 Base Address Switches

The two rotary hex switches located at the top of the board next to the front panel provide the most significant 6 bits of the A16 address of the board. The right switch provides the highest four bits of the address; the high two bits of the left switch (nearer the front panel) are next two. Thus, a switch setting of "8" (1000) on the left switch and "9" (1001) on the right enables the board to respond to addresses in hex range 9800 (1001100000000000) to 9bff (1001101111111111).

1.17.2 FPDP Clock Jumpers

Each FPDP has a jumper block to configure the PECL clock signal paths and terminate some of the FPDP signals. The main function of the jumpers is to connect the clock source to the input of the PECL clock driver for the port, and to connect the output of the clock driver to the clock load. For an input port, the FPDP is the clock source and the board clock lines are the load. For an output port, the lines are the source and the port is the load. Additional jumpers provide termination for the PECL clock lines if needed, and termination for FPDP signals that are not otherwise terminated. Figure 35 identifies the signals on the pins.

Pins	Signal	Notes
1	transmit enable source	Rev 3 board only
2	port transmit enable signal to CPLD	Rev 3 board only
3, 4	PECL clock driver output	same as 11, 12
5, 6	FPDP PSTROBE and PSTROBE/	
7, 8	PECL clock driver input	
9, 10	board FPDP clock lines	
11, 12	PECL clock driver output (duplicate)	same as 3,4
13, 14	PECL clock terminating resistors	
15	termination enable source	
16	FPDP signal termination enable	

Figure 35: FPDP Clock Termination Signals

Figure 36 shows the jumper connections for standard port functions:

- RM: Receiver Master, input with line termination.
- R: Receiver, input without line termination.
- TM: Transmitter Master, output with line termination.

function	RM	R	TM
Transmit enable (see Note)			1-2
port PECL clock to driver	5-7, 6-8	5-7, 6-8	
clock driver to board lines	9-11, 10-12	9-11, 10-12	
board lines to clock driver			7-9, 8-10
driver to port PECL clock			3-5, 4-6
terminate port PECL clock	13-14		
terminate port TTL signals	15-16		15-16

Figure 36: Standard Jumper Configurations

The appearance of the jumper blocks is suggested by the diagrams in Figure 37 (the arrow identifies pin 1, pin 2 is to its right, pin 3 below, etc.):

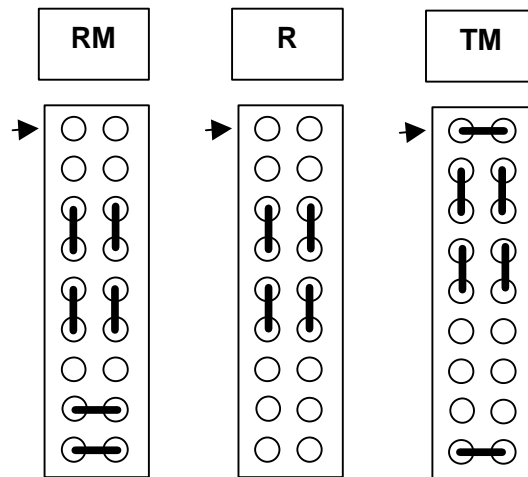


Figure 37: Jumper Block Diagrams

Pins 1 and 2 are not wired as show prior to Revision 3.0 of the hardware. The Transmit Master jumper between them will not cause any problem with prior hardware, but will have no effect.

Important Note: The PECL clock drivers will burn out very quickly if two ports jumpered as transmit masters are connected.

1.18 Operating Concepts

This section presents basic procedures for the V6SSDLF.

1.18.1 Installation and Initialization Sequence

When a board is installed, the following sequence is recommended:

- Determine A16 base address and set the onboard switches.
- Determine the FPDP I/O assignments and configure the PECL Clock jumpers.
- Insert the board and power it up. It will start up in the "halted" state (LED L2 becomes red).
- From the host, configure the board as required while it is halted:
 - Verify that the desired FPGA configuration is present by examining the version register and the version information in the flash memory configuration headers (locations 80000, A0000, C0000 *et seq.*).
 - If the desired FPGA configuration is not the default, reconfigure the FPGA using the FPGA Configuration Register in the CPLD.
 - If the desired FPGA configuration is not in flash, load it from the host as configuration 1 (at flash address a0000) or configuration 2 (at c0000). (The default FPGA configuration 0, at 80000, should be loaded only if confidence is very high that it will work properly. FPGA Configuration 0 must work well enough to support flash update and/or reconfiguration; otherwise, the board must be returned to CAC for restoration of a viable default FPGA configuration.)
 - Verify that the microcode for the FPGA configuration is present in flash at the specified location. If it is not, load it from the host.
 - Set the FPDP Port Options Registers in the CPLD to the proper values for the FPDP I/O assignments.
 - Set the FPDP Clock Select Register in the CPLD to identify the FPDP clock source.
- When configuration requirements have been satisfied, initialize the delay line from the host:
 - Set the desired initial delay, if any, in the Output Delay Registers.
 - Set the Memory Size Mask to correspond to the memory installed and to be used.
 - Reset the Halted bit and set the Run bit in the Command/Status Register (LED L2 becomes green unless a port configuration error is found).

If the settings of the FPDP Port Options Registers in the CPLD are not compatible with the configuration (Transmit bits not set in the registers for the Output Port and, if configured, the Tap Port), both LEDs will be red. (To avoid confusing indications, host applications should not set both LEDs red.)

The FPDP cables can be connected before power up if the FPGA configuration and the clock jumpers are known to be compatible and appropriate for the application environment. When initializing a new board, or a new FPGA configuration, it is advisable to examine the configuration information without I/O connections to avoid problems with unexpected port assignments.

1.18.2 Startup Sequence

Normal processing by the V6SSDLF when the delay line starts up (Run bit set) includes the following:

- All FPDP address registers are set to zero.
- Input processing is enabled, and the delay counter is initialized to the delay register value for the output port.
- Input data values are accepted while Data-Valid* is asserted on the input port and the input inhibit bit is not set.
- As each bufferful of data is received from the input port, it is queued for writing into the delay line memory. If the delay counter is non-zero, it is decremented.
- When the delay counter reaches zero, or if the delay register is zero at startup, output is enabled and data starting at address zero is fetched to the port FIFO buffers.
- Valid Data is asserted and buffer data put on the data lines of the output port as soon as it is available if the upstream Not-Ready and Suspend signals are not asserted.
- Data Valid is de-asserted and no new data values are provided if the Suspend signal is asserted. (As many as eight data items may be sent before the suspension is effective.)

1.18.3 Snapshot Sequences

This section presents three different snapshot initiation sequences, one each for absolute and relative snapshots on a single delay line, and one for synchronizing across delay lines. Registers not mentioned in a sequence are assumed to have their default values.

1.18.3.1 Single Delay Line, Absolute Snapshot

To take an absolute snapshot of a sample sequence in a single delay line, the host performs the following steps:

- Writes the starting address of the desired snapshot to the Snapshot Length/Address Register.

- Writes the value 1 (or any value with the LSB set) to the Snapshot Initiate Register to initiate the snapshot.
- Reads the snapshot data from the Dump Data Register into a host buffer.

1.18.3.2 Single Delay Line, Relative Snapshot

To take an input-relative snapshot of a sample sequence in a single delay line, the host performs the following steps:

- Writes the offset (aligned on 128-byte blocks) from the input data of the desired snapshot start to the Snapshot Length/Address Register.
- Writes zero to the Snapshot Control Register to make the snapshot local.
- Writes the value 0 (or any value with the LSB 0) to the Snapshot Initiate Register to signal the end of the snapshot interval.
- Reads the snapshot data from the Dump Data Register into a host buffer.
- If precise sample identification is desired, reads the Snapshot Offset Register and drops the number of initial bytes specified from the host data buffer.

1.18.3.3 Multiple Delay Lines

For a synchronized multi-board snapshot, the host performs these steps:

- Writes the snapshot size in bytes to the Snapshot Length/Address Register of each board.
- Writes the IRQ line and Freeze-on-Snap option flag into the Snapshot Control Register of each board to identify the reserved IRQ line used to start the synchronized snapshot, and whether to idle the delay line on snapshot initiation.
- Writes the value 0 (or any value with the LSB 0) to the Snapshot Initiate Register of any one of the boards.
- Reads the snapshot data for each board from its Data Register, dropping initial bytes as above. Each board will have its own offset, since the input pointers may not be synchronized.

The host is responsible for ensuring that any necessary data is read out of the Delay Line memory before the Delay Line overwrites it; three methods are available:

- Putting the Delay Line into idle state at snapshot initiation by setting the Freeze-on-Snap option in the Snapshot Control Register before initiation.
- Putting the Delay Line into idle state after initiation by writing zero to the Status/Command Register.
- Completing the snapshot dump within the wrap-around time of the Delay Line.

For a synchronized snap shot, a combination of methods may be useful, depending on the system structure, data rates, and application needs. The dump

can proceed if the delay line is running (L2 green) or idle (L2 yellow), but not if it is halted (L2 red).

There are two considerations if the delay line continues to run after snapshot initiation:

- The VME host can read valid data past the initiation point in the data (the snapshot end address), since the dump FIFO fetch logic does not terminate until the next initiation.
- The delay line will start to overwrite the snapshot data with new input data when the input pointer comes around to the snapshot start address.

The host can detect these conditions by monitoring:

- the Snapshot Start Address Register and Snapshot End Address Register, which are set when the snapshot is initiated, and
- the Input Address Register and Dump Address Register, which reflect the current state of the input and dump FIFOs.

Input data starts to overwrite the snapshot area when the Input Address Register equals the Snapshot Start Address Register. Snapshot dump data is lost when the Input Address Register equals the Dump Address Register.

1.19 Flash Memory Data Formats

This section outlines the data formats used in flash memory, particularly the user-readable identification fields. The flash memory consists of 16-bit words arranged in 31 segments of 32K words and 8 segments of 4K words. Segment formats differ according to their usage. The table below summarizes segment usage; start and length, in hex, refer to word addresses:

Start	Length	Usage
00000	8000	Unassigned
08000	8000	Unassigned
10000	8000	Unassigned
...		
78000	8000	Unassigned
80000	8000	Configuration 0 segment 0
88000	8000	Configuration 0 segment 1
90000	8000	Configuration 0 segment 2
98000	8000	Configuration 0 segment 3
A0000	8000	Configuration 1 segment 0
A8000	8000	Configuration 1 segment 1
B0000	8000	Configuration 1 segment 2
B8000	8000	Configuration 1 segment 3
C0000	8000	Configuration 2 segment 0
C8000	8000	Configuration 2 segment 1
D0000	8000	Configuration 2 segment 2
D8000	8000	Configuration 2 segment 3
E0000	8000	Unassigned
E8000	8000	Unassigned
F0000	8000	Unassigned
F8000	1000	Microcode
F9000	1000	Microcode
FA000	1000	Microcode
FB000	1000	Microcode
FC000	1000	Microcode
FD000	1000	Microcode
FE000	1000	Microcode
FF000	1000	Serial Number information

Figure 38: Flash Memory Segments

The following sections describe fields in these data formats that are of use in identifying and verifying the flash memory contents. Erasing a segment sets all bits to 1; only zero bits are altered when the host writes to the memory.

1.19.1 FPGA Configuration Segments

Each of the initial segments used for the FPGA Configuration data has a header of two to eight words to identify the FPGA configuration it defines. The headers start at flash locations 0x80000 (configuration 0), 0xa0000 (configuration 1), and 0xc0000 (configuration 2).

Figure 39 shows the information in the 16-bit words at the start of a configuration.

Offset	Contents	Format	Hex Example
0 - 1	Timestamp	Unix date	3CDB DBDB
2 - 3	Configuration ID	BCD: vvr ccuu	F1F0 03FA
4 - 5	Creation Date	BCD: yyyy mmdd	2002 0510
6 - 7	Creation Time	BCD: hhmm 0000	1040 0000
8 - 9	End of Header	all bits on	FFFF FFFF

Figure 39: FPGA Configuration Initial Segment Header

The Unix time stamp is automatically generated when the configuration is formatted for flash memory. Words 2 and 6 are BCD encodings designed to be readable in a hexadecimal dump. Words 2 and 3 contain the value found in the FPGA Version Register when the configuration is installed, 2 digits each for the version, revision, cycle, and microcode segment. Words 4 to 6 encode the configuration creation date and time; it is normally slightly earlier than the time encoded in the Unix time stamp. The example header data shown is for release FIFO.3, which was created on May 10, 2002 at 10:40 AM and loads its microcode from the segment at flash address FA000.

This format after the Unix date is a identification information, and is not strictly required in all configuration files. Following the header information are two words with all bits 1; the FPGA configuration specification starts with the next word and extends through the next three segments.

1.19.2 Microcode Segments

Each of the seven microcode segments may contain microcode for one or more FPGA configurations. Each FPGA configuration loads its microcode from a fixed segment, starting at offset 0; the low eight bits of the FPGA Version Register are the high eight bits of the microcode segment. Bit 23 of the Status/Command Register is set during initialization if the segment does not have valid load parameters in the header or if the microcode is missing or erroneous.

The initial few words of the segment contain header information. The first four words are used during initialization by current configurations; the remaining words are documentary information inserted by convention.

Figure 40 describes externally useful aspects of the microcode header:

Offset	Contents	Significance
0	Header Size	words, minimum 4, typically 8
1 - 3	Load Parameters	for micro-engine initialization
4	Flash Segment	high 8 address bits, to verify loading
5	Version ID	BCD microcode identifier
6 - 7	Timestamp	compilation date in Unix format

Figure 40: Microcode Segment Header

The Version ID, read as four hexadecimal digits, has the form "n0vv" and the significance shown in Figure 41.

Bits	Digits	Significance
15:12	n	number of ports supported, 2 or 3
11:8	0	none assigned
7:0	vv	version number

Figure 41: Microcode Version ID

The number of ports supported depends also on the FPGA configuration; three ports are supported only if the configuration supports a tap and n is 3. The version number typically identifies the fixed delay line configuration release that first uses the microcode release. The microcode itself starts after the header.

1.19.3 Serial Number Segment

The first four words of the segment starting at address 0xff000 contain serial number information, in the following format:

Offset	Contents
0	Baseboard Serial Number first 4 digits
1	Baseboard Serial Number last 4 digits
2	Mezzanine Serial Number first 4 digits
3	Mezzanine Serial Number last 4 digits

Figure 42: Serial Number Segment Format

All fields are BCD, i.e., each hex digit has a value from 0 to 9, representing the corresponding decimal digit. The Mezzanine Serial Number fields identify the

mezzanine shipped with the baseboard. If there is no mezzanine shipped with the baseboard, the Mezzanine Serial Number fields are 0. A hexadecimal dump of these four words will display the serials, for instance, "1150 0023 1160 0014" for baseboard #11500023 with mezzanine #11600014.

1.19.4 Reserved Segments

Reserved segments may be used by subsequent FPGA configurations.

V6SSDLF Software Support

The software support for the V6SSDLF is subject to change. Consult the release notes for the current software release for changes and additions.

Software for the V6SSDLF consists of:

- Host control, to initialize and start a delay line, monitor its behavior and performance, and examine the delay line state and contents.
- Utility programs, to update the configurable aspects of the V6SSDLF, determine the current configuration of the unit, and verify the configuration.

1.20 Host Control Libraries

Host control software for the V6SSDLF is designed to run either on SunOS or on VxWorks. Compiler switches for each environment conditionally compile the right code for the environment, using different makefiles.

The software consists of libraries of functions that control and monitor V6SSDL units, for use in application programs that provide the user interface and call the library functions.

1.20.1 dly_open()

Function: **DLY_DEV ***
 dly_open(devname)

Args: **char *devname**

Returns: **non-NULL** device successfully opened
 NULL the device cannot be opened (busy or not found), or
 mmap() failed to map the VME space into memory, or
 malloc() failed to allocate space for the structure, or
 error indicated by **errno** or I/O Error

Errors: **EBADF** **devname** is **NULL** or does not point to a valid delay
 line device name
 EINVAL device opened is not a delay line

The **dly_open** function opens a delay line device, creates a **DLY_DEV** structure for it, and maps its registers into host memory. The pointer returned is used in other library functions to identify the device. The pointer will be **NULL** if the device was not successfully opened. The delay line itself is not affected.

1.20.2 dly_close()

Function: **int**
 dly_close (dly)

Args: **DLY_DEV** ***dly**

Returns: 1 device successfully closed
 0 device not open, or cannot be closed, or
 system failed to return allocated memory, or
 error indicated by **errno** or I/O Error

Errors: **EBADF** **dly** is **NULL**

The **dly_close** function closes an open delay line device, returning allocated resources. The delay line itself is not affected.

1.20.3 dly_run(dly)

Function: **int**
 dly_run (dly)

Args: **DLY_DEV** ***dly**

Returns: 1 success, control operation complete
 0 error indicated by **errno** or I/O Error

Errors: **EINVAL** **dly** is **NULL**

The **dly_run** function puts the V6SSDL in run mode.

1.20.4 dly_stop()

Function: **int**
dly_stop (dly)

Args: **DLY_DEV *dly**

Returns: 1 success, control operation complete
0 error indicated by **errno** or I/O Error

Errors: **EINVAL dly is NULL**

The **dly_stop** function stops the V6SSDL delay line, leaving it in the idle state. Snapshot dumps are enabled in this state, but access to CPLD registers and flash memory is not.

1.20.5 dly_halt()

Function: **int**
dly_halt (dly)

Args: **DLY_DEV *dly**

Returns: 1 success, control operation complete
0 error indicated by **errno** or I/O Error

Errors: **EINVAL dly is NULL**

The **dly_halt** function halts the V6SSDL, putting it into "safe" mode. Snapshot dumps are not enabled in this state, but access to CPLD registers and flash memory is enabled.

1.20.6 dly_memsize()

Function: **int**
 dly_memsize (dly, mem)

Args: **DLY_DEV** ***dly**
 int **mem**

Returns: 1 success, control operation complete
 0 error indicated by **errno** or I/O Error

Errors: **EINVAL** **dly** is **NULL** or
 mem is not a valid size

The **dly_memsize** function sets the working size of the V6SSDL, which may be less than or equal to the actual size. The **mem** argument is the number of megabytes; values allowed are powers of two from 128 (128 MB) to 4096 (4 GB).

1.20.7 dly_clockin()

Function: **int**
 dly_clockin (dly, clockin)

Args: **DLY_DEV** ***dly**
 int **clockin**

Returns: 1 success, operation complete
 0 error indicated by **errno** or I/O Error

Errors: **EINVAL** **dly** is **NULLB** or
 clockin is not a legitimate value

The **dly_clockin** function sets the clock input mode to **V6SSDL_CLOCKIN_TTL** or **V6SSDL_CLOCKIN_PECL**.

1.20.8 dly_delay_bytes()

Function: **int**
 dly_delay_bytes(dly, delay)

Args: **DLY_DEV** ***dly**
 u_int **delay**

Returns: 1 success, control operation complete
 0 error indicated by **errno** or I/O Error

Errors: **EINVAL** **dly** is **NULL**

The **dly_delay_bytes** function sets the V6SSDL initial delay to the number of bytes specified in the **delay** argument.

1.20.9 dly_fpdpmode()

Function: **int**
 dly_fpdpmode(dly, fpdp_a, fpdp_b, fpdp_c)

Args: **DLY_DEV** ***dly** ptr to DLY_DEV type
 int **fpdp_a** mode for FPDP A
 int **fpdp_b** mode for FPDP B
 int **fpdp_c** mode for FPDP C

Returns: 1 success, control operation complete
 0 error indicated by **errno** or I/O Error

Errors: **EINVAL** **dly** is **NULL** or
 a mode argument is invalid

The **dly_fpdpmode** function sets the FPDP port modes and the CPLD port options. Each port mode may be one of the following values defined in **dlyutil.h**:

V6SSDL_FPDP_UNUSED,
V6SSDL_FPDP_IN,
V6SSDL_FPDP_OUT,
V6SSDL_FPDP_TAP

Only one FPDP may be set to each active mode

1.20.10 dly_fpga_config()

Function: **int**
dly_fpga_config (dly, config)

Args: **DLY_DEV** ***dly**
int **config**

Returns: 1 success, control operation complete
 0 error indicated by **errno** or I/O Error

Errors: **EINVAL** **dly** is **NULL** or
config is not 0, 1, or 2
ENODEV board is not accessible after reconfiguration

The **dly_fpga_config** function reconfigures the FPGA to the specified flash memory configuration and restores the A32 base address register. The starting flash address for each configuration is in Figure 43:

Config	Flash Address
0	80000
1	a0000
2	c0000

Figure 43: FPGA Configuration Addresses

In cases where the FPGA fails to reconfigure, or the configuration data is not valid, the device may fail to respond. For SunOS, this will cause a **SIGBUS** to the application. For VxWorks, the board is probed following reconfiguration. If it does not respond, **errno** is set to **ENODEV**.

If a selected configuration is not present in flash memory, the board continues to stream flash memory contents into the FPGA configuration port until it reaches either a valid configuration specification or the end of flash. If a valid configuration is not loaded, a "long" reset will restart the board with the safe configuration (number 0, at 0x80000, unless it was overwritten).

1.20.11 dly_ctrl()

Function: **int**
dly_ctrl (dly, cmd, val)

Args: **DLY_DEV** ***dly**
int **cmd**
int **val**

Returns: ≥ 0 control operation success (depends on **cmd**)
-1 error indicated by **errno** or I/O Error

Errors: **EINVAL** **dly** is **NULL**, or
cmd is invalid or not supported, or
ioctl error

The **dly_ctrl** function issues **ioctl()** calls to the host operating system. Figure 44 lists the **cmd** values, operations, and corresponding return values.

cmd	operation	return value (success)
DLY_INTENB	enable interrupt signals from board	0
DLY_INTDIS	disable interrupts signals from board	0
DLY_INTSTAT	get interrupt status from board	the interrupt status from the board
DLY_VMEIRQ	enable/disable VME interrupts from board	0
DLY_BOARDTYPE	get board type	DLY_V6SSDL_A16 or DLY_V6SSDL_A32
DLY_DRVVERSION	get device driver version	Device driver version
DLY_VIRTBRDADDR	get address of board	virtual address of the board
DLY_VMEBRDADDR	get physical VME address of board	physical VME address of the board
DLY_SEMINTMODE	enable/disable semaphore interrupt mode	0

Figure 44: VME ioctl() commands

The file **dlyioctl.h** contains definitions for the **cmd** and **val** values.

1.20.12 dly_fpga_read()

Function: **int**
dly_fpga_read (dly, reg, ptr)

Args:

DLY_DEV	*dly
int	reg
u_int	* ptr

Returns: 1 success, control operation complete
 0 error indicated by **errno** or I/O Error

Errors: **EINVAL** **dly** is **NULL**, or
reg out of valid range, or
ptr is **NULL**

The **dly_fpga_read** function reads an FPGA register, placing the value in the location designated by **ptr**. The register is specified by its number (word offset from the start of the board's register space). The file **dlyutil.h** contains symbolic definitions for the register numbers.

1.20.13 dly_fpga_write()

Function: **int**
dly_fpga_write (dly, reg, val)

Args:

DLY_DEV	*dly
int	reg
u_int	val

Returns: 1 success, control operation complete
 0 error indicated by **errno** or I/O Error

Errors: **EINVAL** **dly** is **NULL**, or
reg is out of valid range

The **dly_fpga_write** function writes the value provided into the designated FPGA register. The register is specified by its number (word offset from the start of the board's register space). The file **dlyutil.h** contains symbolic definitions for the register numbers.

1.20.14 dly_cpld_read()

Function: **int**
dly_cpld_read (dly, reg, ptr)

Args: **DLY_DEV *dly**
int reg
u_int *ptr

Returns: 1 success, control operation complete
0 error indicated by **errno** or I/O Error

Errors: **EINVAL** **dly** is **NULL**, or
reg is outside the valid range, or
ptr is **NULL**

The **dly_cpld_read** function reads a CPLD register, placing the value in the location designated by **ptr**. The register is specified by its number (word offset from the start of the CPLD register space). CPLD Registers have eight-bit register numbers and values; the value is written into the low eight bits of ***ptr**. The file **dlyutil.h** contains symbolic definitions for the CPLD register numbers.

1.20.15 dly_cpld_write()

Function: **int**
dly_cpld_write (dly, reg, val)

Args: **DLY_DEV *dly**
int reg
u_int val

Returns: 1 success, control operation complete
0 error indicated by **errno** or I/O Error

Errors: **EINVAL** **dly** is **NULL**, or
reg is outside the valid range

The **dly_cpld_write** function writes the value provided into the designated CPLD register. The register is specified by its number (word offset from the start of the CPLD register space). CPLD Registers have eight-bit register numbers and values; the value written is the low eight bits of **val**. The file **dlyutil.h** contains symbolic definitions for the CPLD register numbers.

1.20.16 dly_set_dump()

Function: **int**
dly_set_dump (dly, len, irq, frz)

Args: **DLY_DEV *dly**
u_int len
u_int irq
u_int frz

Returns: 1 successful dump setup
0 error indicated by **errno** or I/O Error

Errors: **EINVAL** **dly** is **NULL**, or
len is less than 128, or
irq is greater than 7, or
frz is not zero or one

The **dly_set_dump** function sets up a delay line board for a relative snapshot. The **len** argument is the number of samples before the current input pointer to start the snapshot dump when the snapshot is initiated. If **irq** is non-zero, it identifies one of the IRQ lines on the VMEBus backplane to use to synchronize the snapshot across multiple delay lines. If **frz** is one, the delay line will become idle (LED 2 yellow), and stop accepting data when the snapshot is initiated.

1.20.17 dly_start_dump()

Function: **int**
dly_start_dump (dly)

Args: **DLY_DEV *dly**

Returns: 1 successful dump start
0 error indicated by **errno** or I/O Error

Errors: **EINVAL** **dly** is **NULL**

The **dly_start_dump** function initiates a relative snapshot on the delay line board indicated by **dly**. If the board has been initialized with a non-zero dump IRQ (see the **dly_set_dump** function), the corresponding IRQ line will be used to synchronize a snapshot across all boards initialized to use the same IRQ. Otherwise, the snapshot will be limited to the indicated board.

1.20.18 dly_abs_dump()

Function: **int**
dly_abs_dump (dly, addr)

Args: **DLY_DEV *dly**
u_int addr

Returns: 1 successful dump start
 0 error indicated by **errno** or I/O Error

Errors: **EINVAL dly is NULL**

The **dly_abs_dump** function initiates a local absolute snapshot at byte address **addr** on the delay line board indicated by **dly**. The snapshot is initiated only on the indicated board, regardless of prior call to the **dly_set_dump** function.

1.20.19 dly_read_dump()

Function: **int**
dly_read_dump (dly, buffer, len)

Args: **DLY_DEV *dly**
u_int *buffer
u_int len

Returns: 1 successful dump read
 0 error indicated by **errno** or I/O Error

Errors: **EINVAL dly is NULL, or**
dly indicates a delay line without A32 access enabled
EBUSY the indicated device was not available
EIO the data transfer failed

The **dly_read_dump** function reads data from a delay line snapshot to the host. The **buffer** argument points to a buffer allocated by the application program to hold the data. The **len** argument is the number of 32-bit samples to read into the buffer.

1.21 Utility Programs

The Utility Programs use the host library functions to maintain configuration and related information on the V6SSDLF. Filename parameters allow full paths (starting with a slash), paths relative to the current directory (starting with "./"), or references to directories in the standard file hierarchy $\${CAC}/v6ssdl/*$ identified by the environment variable **CAC**.

1.21.1 dlyflash

The **dlyflash** program loads, compares, erases, dumps, or displays the contents of selected segments of flash memory, using a file as the flash image where needed. Command line parameters specify the function, the file, the flash address, and the output generated. Not all combinations of the parameters shown in Figure 45 can appear; **-c**, for example, implies values for **-a** and **-n**. The program will not alter the serial number segment. The standard directory for **flashfile** parameters is **flash**.

```
Unix:      dlyflash [options] flashfile dlydev
VxWorks:  dlyflash_cmd "[options] flashfile dlydev"
```

Operation specification:		
-l		load flash from file
-d		dump flash sectors to file
-e		erase flash sectors
-c		compare flash with file
-s		display (show) flash memory contents
Flash parameters:		
-a	addr	specify flash address (0xhhhhh)
-b		allow writing to backup FPGA configuration area
-C	num	specify FPGA config number (0, 1 or 2)
-n	num	specify number of segments (-d, -c, or -e)
-w	num	specify number of words to display (-s)
Miscellaneous options:		
-v		verbose output messages
-q		no output messages
-D		debug mode
Program parameters:		
flashfile		name of flash file for load, dump or compare
dlydev		name of V6SSDL device (e.g. dly0)

Figure 45: dlyflash Parameters

1.21.2 dlysetup

The **dlysetup** program interprets board control operations from a file, applying them to one or more boards. The program halts the board, allowing access to the CPLD registers, then interprets the `setup_file`. Command line parameters specify the operating mode of the program, the file of operations, and a list of board device names.

```
Unix:      dlysetup [options] setup_file [dly_devs]
VxWorks:  dlysetup_cmd "[options setup_file [dly_devs]]"
```

Mode options	
-v	verbose output messages
-q	no output messages ("quiet")
-D	debug mode (parse, do not execute)

Figure 46: dlysetup Options

The `setup_file` contains one operation per line, with text after a # character treated as comments. The operations are as follows:

Keyword	arguments	Operation
regr:	reg	read and display register value
regw:	reg = val	write val to register
	reg & val reg val	modify register with bitwise logical op
dlydev:	device	Open device
config:	N	switch to configuration N (0, 1, 2)
a32addr:	addr	Specify A32 device base addr
a32mask:	mask	Specify A32 device address mask
porta portb portc	unused input output tap	Set up port use
clkin:	ttl	Set up for TTL FPDP clock
	pecl	Set up for PECL FPDP clock
mem:	N	Set memory size to N GB (2, 4, 8)
delay;	N	Set output delay to N samples
run		Start delay line
say	string	Echo string to display

Figure 47: dlysetup Commands

The a32addr and a32mask operations are both necessary to set up the A32 Base Address Register; the **dlysetup** program initializes the register when it has found both.

Register identifiers can be numbers, either decimal or hexadecimal (starting with "0x"), or the symbols in Figure 48.

Symbol	Type	Number	Register Name
csr	Board	00	Command/Status
a32	Board	01	A32 Base Address
irq	Board	02	Interrupt
sslen	Board	03	Dump/Load Length
fpdp	Board	04	Port Configuration
outdelay	Board	05	Output Delay
tapdelay	Board	06	Tap Delay
chipsub	Board	07	Chip Substitution
led	Board	08	LED Control
dumpoff	Board	09	Dump Offset
crcstat	Board	0a	CRC Status
syndrome	Board	0b	CRC Syndrome
version	Board	0f	FPGA Version
debug	Board	0c	Debug control
snapctl	Board	0d	Snapshot Control
snapini	Board	0e	Snapshot Initiate
fifoin	Board	10	Input FIFO Address
fifoout	Board	11	Output FIFO Address
fifovme	Board	12	Dump FIFO Address
memsize	Board	13	Memory Size Mask
snapstart	Board	14	Snapshot Start Address
snapend	Board	15	Snapshot End Address
flashaddr	Board	20	Flash Memory Address
flashdata	Board	21	Flash Memory Data
cpldaddr	Board	22	CPLD Register Address
cplddata	Board	23	CPLD Register Data
fifodata	Board	40	Snapshot Data
fpgacfg	CPLD	0	FPGA Config Select
clockin	CPLD	1	FPDP Clock Select
fpdpa	CPLD	4	Port A Configuration
fpdpb	CPLD	5	Port B Configuration
fpdpc	CPLD	6	Port C Configuration

Figure 48: dlysetup Register Names