## Danville Signal Processing, Inc.

# dspstak<sup>™</sup> sx DSP Engine





Version 2.10

## Danville Signal Processing, Inc. dspstak™ sx User Manual

Copyright © 2004 – 2005 Danville Signal Processing, Inc. All rights reserved. Printed in the USA.

Under the copyright laws, this manual may not be reproduced in any form without prior written permission from Danville Signal Processing, Inc.

Danville Signal Processing, Inc. strives to deliver the best product to our customers. As part of this goal, we are constantly trying to improve our products. Danville Signal Processing, Inc., therefore, reserves the right to make changes to product specification or documentation without prior notice. Updated operating manuals and product specification sheets are available at our website for downloading.

This manual may contain errors, omissions or "typo's". Please send your comments, suggestions and corrections to:

Danville Signal Processing, Inc. 38570 100th Avenue Cannon Falls, MN 55009-5534

## **Trademark Notice**

dspstak is a trademark of Danville Signal Processing, Inc. VisualDSP++, EZ-Kit and SHARC are trademarks of Analog Devices, Inc. CyberClocks is a trademark of Cypress Semiconductor Corporation. Windows is a trademark of Microsoft Corporation.

## **Contact Information**

Danville Signal Processing, Inc. 38570 100th Avenue Cannon Falls, MN 55009

E-mail:	dsp@danvillesignal.com
Web Site:	http://www.danvillesignal.com
Voice:	507-263-5854
Fax:	877-230-5629

## **Table of Contents**

Overview	5
Introducing dspstak <sup>™</sup> dspstak <sup>™</sup> sx DSP Engine Introduction Product Revisions dspstak sx with ICE dspstak sx (Production versions)	
Hardware Description	9
Power Supply RS-232 Interface USB Interface Interconnect Port Programmable Clocks JTAG Emulation Port	
dspstak <sup>™</sup> sx Architecture	13
Hardware – DSP Core DAI Parallel Port SPI Port	13 13 15 16
Programming the dspstak <sup>™</sup> sx	
Programming Modes Boot & Clock Configuration Programming the Clock – Mode 7 Uploading Programs – Mode 6	18 19 19 21

Peripheral Microcontroller API	24
Command Summary System Commands Com Port Commands EE Memory Commands	24 25 29 31
USB & PI D Registers	
Addressing USB & PLD Registers PLD I/O Register & USB Status Data Format Memory Map USB Port Version Register PLD I/O & SPI SS Registers	
Software	42
Schematic	42
Mechanical Drawings	42
Product Revision	43
Interconnect Port PLD Registers Programmable Clocks	43 45 45
Higher Analog Current Capability Secondary Mounting Holes Jumpers ADSP-2136x Compatibility	

## Overview

DSP-based embedded applications often take the form of a digital signal processing engine coupled with a specialized data conversion and signal conditioning front end. The front end electronics and the DSP are almost always connected via high speed serial ports and the general purpose I/O ports of the DSP. In most cases, once the local memory and peripheral interfacing needs of the DSP are fulfilled, the DSP's data and address busses are no longer needed.

Standardized bus architectures such as PC/104, PCI and cPCI are all based on communicating via each board's data and address bus, while ignoring the needs of the most DSP / data conversion interfaces.

## Introducing dspstak<sup>™</sup>

Danville's dspstak modules are designed to simplify DSP-based embedded applications. Generally, each dspstak consists of two modules: a DSP Engine, and a signal conditioning/data conversion I/O Module. The Interconnect Port consists of SPORTs (high speed serial ports), SPI, general I/O, clocks and power connections.

DSP Engine modules generally consist of a DSP processor, memory, power supplies and standard digital I/O such as RS-232 and USB. We currently have products based on Analog Devices' SHARC<sup>®</sup> processors.

The I/O Modules may include signal conditioning electronics, A/D and/or D/A data converters, audio transceivers, unique connectors and a plethora of other special front end requirements. Since the I/O Module is separate from the DSP Engine Module, custom interfaces can be created quickly and inexpensively. Danville has a number of standard I/O Modules and often is willing to create a new one based on customer request. You can also create your own.

## dspstak<sup>™</sup> sx DSP Engine

This manual covers the dspstak sx DSP Engine and is complemented by the dspstak Family Users Manual. The family manual covers topics that all dspstaks have in common. You will want to review the family manual for mechanical dimensions, system configurations, basic connector specifications, etc.

The dspstak sx manual covers topics that are relevant to the specific implementation of the dspstak sx DSP Engine. You will also need the Analog Devices' ADSP-2126x SHARC DSP Core and Peripheral manuals. These manuals can be downloaded from the Analog Devices web site. Links to these manuals and other resources are found on our web site at http://www.danvillesignal.com/index.php?id=dspdev.

## **Intended Audience**

The dspstak sx is intended for DSP systems integrators, designers and programmers who may wish to integrate a dspstak into their products. This manual is primarily aimed at users who have a working knowledge of microcomputer technology and DSP related design. We assume that you are familiar with the Analog Devices ADSP-2126x or ADSP-21364 SHARC DSP.

### Introduction

The dspstak<sup>™</sup> sx is a high performance digital signal processing engine based on the Analog Devices' third generation family of SHARC DSPs. Members of this family include the ADSP-21261, ADSP-21262 and ADSP-21364. There are other pin compatible parts in this family as well that can be supported by the sx platform. The most popular member of this family, the ADSP-21262, supports 32 bit fixed point and 32/40 bit floating point formats with SIMD instructions running at up to 200 MHz. This translates to a peak computation rate of 1.2 GFLOPS. The ADSP-21262 also includes very flexible peripherals including six independent serial ports (SPORTs) and a SPI port. The dspstak sx Interconnect Port supports the complete DAI interface, SPI, general I/O, clocks and power connections.

There are two basic variations of the sx platform. The versions with "ICE" include a licensed version of Analog Devices EZ-Kit debugger. This allows a dspstak sx DSP Engine to function in a very similar manner to an Analog Devices EZ-Kit development board. You use the built in debugger to facilitate your software development and then use the standard production versions for your finished products. The debugger is fully supported by Visual DSP 4.0. You can use a free "KIT" license (with the same limitations as an EZ-Kit) or a full-featured Visual DSP license.

The dspstak sx also includes USB and RS-232 ports for easy interfacing to the outside world. It also has a JTAG interface to facilitate program development.

We manufacture several different standard configurations. We can also provide variations if you have a special requirement. For example, the ADSP-21266 is a variation of the ADSP-21262 that includes various audio decoder algorithms. We could provide a dspstak 21266sx (assuming you have the appropriate license from ADI).

Here are standard sx DSP Engine versions:

- dspstak 21262sx (ADSP-21262)
- dspstak 21262sx with ICE (ADSP-21262, ADI EZ-Kit debugger)
- dspstak 21364sx (ADSP-21364)
- dspstak 21364sx with ICE (ADSP-21364, ADI EZ-Kit debugger)

The dspstak sx includes the following items:

#### Hardware:

• dspstak sx board

#### **Documents & Programs (CD):**

- This Manual
- dspstak Family User Manual
- CAD footprints (Gerber & Protel 99SE formats)
- Free Terminal Program Tera Term Pro
- USB Device Driver

#### Firmware (CD):

- Flash Bootloader
- Sample Programs

We recommend that you have the following tools and documents:

- Analog Devices VisualDSP++<sup>™</sup>4.0 for SHARC®
- Danville JTAG adapter kit
- Analog Devices ADSP-2126x SHARC® DSP Core Manual
- Analog Devices ADSP-2126x SHARC® DSP Peripherals Manual
- Analog Devices ADSP-2136x SHARC Processor Hardware Reference Manual
- Analog Devices ADSP-2136x SHARC Processor Programming Reference Manual
- Analog Devices ADSP-21262 or ADSP-21364 Product Data Sheet
- Cypress Semiconductor CyberClocks™

Our website (www.danvillesignal.com) has downloads and links to other supplementary tools and documents.

## **Product Revisions**

The dspstak sx has gone through one major revision. The dspstak 21262sx is the only variation that was produced in the earlier version. You can identify this revision from the Danville P/N located in the lower left corner of the pcb assembly. For purposes of this manual, the revision starts with boards using pcb P/N MPCB.07266E.

In most cases, the changes will have a minimal impact on an existing system based on an earlier revision of the dspstak sx. The changes are summarized in the Revisions Appendix. Our software examples have undergone simple changes as well. You may need to add a few lines of code into an existing design to accommodate all the board revisions.

## dspstak sx with ICE

Although the dspstak sx with ICE is software and hardware compatible with the standard production versions of the sx platform, the actual construction is quite different. The dspstak sx with ICE is comprised of two boards, a dspblok sx module and a special dspstak baseboard.

The connectors are identical to the production dspstak sx with two exceptions. There is an additional USB connector that is used for the EZ-Kit style debugger that is located on the edge of the pcb assembly. The debugger uses a mini-B USB connector whereas the "User" USB connector is a larger "B" style USB connector. The production version includes a special 30 pin expansion connector that is not standard on other dspstak DSP Engine platforms. This connector is currently used to support an optional SRAM memory expansion board. This expansion connection is not available on the ICE version of the dspstak sx. All other features of the dspstak sx can be emulated with the dspstak sx with ICE.

The built in debugger is only capable of emulating the specific DSP that is on the dspstak sx with ICE board. For example, a dspstak 21364sx with ICE will not emulate a 21262sx. If you are using a Visual DSP "KIT" license, you must have the dspstak sx with ICE board's debugger connected to the development computer. You can create bootable loader files that can then be uploaded to a dspstak sx board without the need for Visual DSP or the dspstak sx with ICE.

In addition to developing code for dspstak sx production boards, the dspstak sx with ICE is also a convenient programmer or tester for the dspblok sx DSP Module. The dspblok sx is the signal processing engine of the dspstak sx with ICE but it is also sold independently for users we want to integrate a powerful DSP core into their designs.

## dspstak sx (Production versions)

The dspstak sx is intended for production targets. Since, it does not include the debugger, it is less expensive than the ICE version.

If you have an Analog Devices HP ICE or USB ICE, you might use this version for your code development as well since all dspstak DSP Engines include JTAG connections.

After your code is developed, you create a loader file from Visual DSP. This file can be uploaded to the dspstak sx via the USB or RS-232 ports using a simple terminal program. You do not need an ICE, the dspstak sx with ICE or Visual DSP to program the flash memory.

## **Hardware Description**

The dspstak sx includes all dspstak DSP Engine standard features. These features are discussed in a general way in the dspstak Family Users Manual and are as follows:

- Power Supply
- RS-232 Interface
- USB Interface
- Interconnect Port
- Programmable Clock
- JTAG Header

Each of these areas is discussed in more detail in this manual.

If you are creating your own I/O Modules, you may want to restrict your design to rely only on the standard features. This will insure that your I/O Modules are compatible with future dspstak DSP Engines. On the other hand, if you want to take advantage of the increased capabilities of the dspstak sx, you may want to disregard some of the restrictions. In general, you can assume that future dspstak DSP Engines based on the 3<sup>rd</sup> generation SHARC family will support the Interconnect Port in a manner similar to the dspstak sx.

## **Power Supply**

The dspstak sx uses a standard 2.1/5.5mm coaxial power jack to provide power to itself and also any I/O modules via the Interconnect Port. From the dspstak sx view, this supply may be either an AC supply or a DC supply with the center of the coaxial power plug connected as the positive pin. Adjacent to the coaxial jack, there are 3 pads spaced at 0.100 inch that may be used as an alternative connection to provide a bipolar DC supply to the board. These pads are labeled JH6 and can accommodate a .025 header or similar connector. This alternate connection method was not standard in earlier dspstak DSP Engines but is implemented in all current products.

The incoming voltage is half wave rectified to create new unregulated DC supplies called Va+ and Va-. An additional internal supply is also created to provide the input to switching regulators that are used for digital supplies. From these unregulated supplies, Va+5 (5.0 Volt Analog), Vd+5 (5 Volt Digital) and Vd+3.3 (3.3V Digital) are created and made available to the I/O Modules via the Interconnect Port. The DSP Core voltage, Vd+1.2 is also created via a switching power supply.

The dspstak sx DSP Engine does not use Va+, Va+5 or Va-. These supplies are available to I/O Modules. If the I/O Module does not need a negative supply, you can power the dspstak sx with only a positive DC power supply. The recommended input voltages are 9 VAC or 9 to 15VDC. Since switching regulators are used for the digital supplies, the input voltage is not critical.

In all cases, the DC supply or an unregulated DC supply created from an incoming AC supply should be less than 20VDC. This will protect both the DSP Engine and probably the components on the I/O Module.

One final thing to remember: The nominal output voltage of a wall adapter is usually much lower than its actual output voltage. We recommend that unregulated DC wall adapters should have an maximum nominal voltage of 12VDC. In most cases, a 9VDC/500mA wall adapter is a good choice.

## **RS-232 Interface**

The RS-232 interface is used to upload user application programs and to program the clock generator in the command mode. An application program may also use the RS-232 interface for its own purposes. The RS-232 port is provided by a dedicated microcontroller that also has other functions on the dspstak sx. These functions are all available via the SPI port of the DSP. The API for these functions is described in the Software Section of this manual. There are also software examples on calling these functions on the CD.

The RS-232 interface is configured as a DCE (Data Communications Equipment) device. It uses a DB-9F connector that is designed to mate without crossover connections to the DB-9M connector found on PC compatible computers. Connections are defined in the following table:

Signal	Direction	Pin	Notes
DCD		1	DCD, DTR & DSR are connected together
RD	DCE -> DTE	2	
TD	DTE -> DCE	3	
DTR		4	DCD, DTR & DSR are connected together
SIGNAL GND		5	
DSR		6	DCD, DTR & DSR are connected together
RTS	DTE -> DCE	7	Not currently supported
CTS	DCE -> DTE	8	Not currently supported
RI		9	NC

When the dspstak sx is programmed using the RS-232 interface, only RD & TD are used. DTR, DSR & DCD are simply connected together and ignored by the dspstak.

The dspstak sx supports standard bauds of 9600, 19.2k, 38.4k, 57.6k, 115.2k and 230.4k. The expected protocol is No parity, 8 data bits, 1 stop bit, (N:8:1).

## **USB Interface**

The USB interface is another way that programs may be uploaded to the dspstak. Unlike most Analog Devices EZ-Kit development boards, the USB interface is also available for user applications. A device driver is included with the dspstak sx to support Windows 2000 & Windows XP applications. The USB connection is made via a standard USB type B connector.

### **Interconnect Port**

The Interconnect Port is the only standard connection between the DSP Engine and I/O Modules. This port is described in detail in the dspstak Family Users Manual and the DAI section of this manual.

## **Programmable Clocks**

The dspstak sx has a very flexible clock configuration. A user reprogrammable clock generator provides the DSP clock and three I/O clocks that are available on the Interconnect Port. The three programmable clocks on the Interconnect Port allow I/O Modules to use convenient clocks for whatever devices that might be present. For example, an ADC might use an 18.432 MHz MCLK to sample at 96k or a 19.6608 MHz MCLK to sample at 102.4k. In addition to the on-board programmable clock, the DSP clock can also be provided externally by a dspstak I/O Module.

The dspstak sx uses a Cypress Semiconductor CY22393 flash programmable clock generator. This device has three PLLs which when combined with the reference oscillator can create four independent clock frequencies.

There are actually two sets of preprogrammed clocks. You can switch between these sets under software control by the DSP application program. For example, one set might be used to support 22.05k, 44.1k, 88.2k and related sampling rates while the alternate set is used for 24k, 48k, 96k sampling rates.

Cypress has a program called CyberClocks on their web site (<u>www.cypress.com</u>) that you can use to create a JEDEC file to reprogram the clock chip. There are support functions that allow this file to be uploaded via the RS-232 port. Since reprogramming certain clocks could make the RS-232 port inoperable, the dspstak DSP Engine filters these parameters from the JEDEC file to automatically protect the DSP Engine from unfortunate modifications. Details on programming the clock are in the Programming the dspstak sx section of this manual.

In many cases, the factory default settings are appropriate and no changes are ever necessary.

## JTAG Emulation Port

The dspstak sx has a right angle JTAG connector (JH3) mounted on the lower edge of the pcb assembly. This connector is assessable even when a dspstak I/O Module is positioned above the dspstak sx.

Danville JTAG connectors are smaller than the standard ADI recommended JTAG header.

The Danville JTAG connector is a male 16 pin, 2mm dual row header. Each pin directly corresponds to the same pins on the Analog Devices header. The remaining pins (15&16) are assigned to Vd+3.3. The extra power supply pins allow you to power a buffer in a system that has many JTAG devices. A JTAG adapter kit is available from Danville to convert the 2mm header to the 0.100 header used by the Analog Devices emulators.

The JTAG connections are pulled up or pulled down so that it is not necessary to add any shorting jumpers when an external emulator is not connected. *This is a change from earlier dspstak sx DSP Engines*.

## dspstak<sup>™</sup> sx Architecture

## Hardware – DSP Core

The heart of the dspstak sx is an Analog Devices ADSP-2126x or ADSP-2136x DSP. This manual does not discuss the internal workings of this DSP, the assembly language and other details specific to this DSP. You should refer to the fore mentioned ADI documents for this information. This SHARC has several important peripherals that we will expand on as they relate to the dspstak sx. They are as follows:

- Digital Audio Interface (DAI)
- Parallel Port
- SPI Interface

## DAI

The DAI is comprised of a group of peripherals including the SPORTs and a signal routing unit. There are 20 pins on the DSP that connect the DAI to the outside world. All 20 DAI pins are available on the Interconnect Port. If you are designing your own I/O module, you can freely assign the DAI pins based by your specific I/O requirements. If you want to interface in a Danville I/O module as well (a three board dspstak), contact us for DAI assignment recommendations.

**NOTE:** The DAI and all other dspstak I/O is intended for local interfacing. Do not drive long cables with these connections. **All I/O assumes 3.3V logic and are NOT 5V tolerant**!

Row	A&C	В	Name	Notes
1	*	*	AGND	Analog Ground
2	*	*	Va+	Unregulated Positive Analog
				Supply
3	*	*	Va-	Unregulated Negative Analog
				Supply
4	*	*	Va+5	Regulated Analog +5.0 Volt
				Supply
5	*	*	AGND	Analog Ground
6	*	*	Vd+5	Digital 5.0 Volt Supply
7	*	*	Vd+3.3	Digital 3.3 Volt Supply
8	*	*	DGND	Digital Ground – Main Return

Here are the DAI to Interconnect Port Mappings:

Row	A&C	В	Name	Notes
9	*		IO0/#SS1	3.3V Digital I/O or SPI SS
9		*	IO1/#SS2	3.3V Digital I/O or SPI SS
10	*		IO2/#SS3	3.3V Digital I/O or SPI SS
10		*	IO3/#SS4	3.3V Digital I/O or SPI SS
11	*		IO4/#SS5	3.3V Digital I/O or SPI SS
11		*	DAI 2	
12	*		DAI 3	
12		*	DAI 1	
13	*		#SPI_SS	Primary SPI Slave Select
13		*	SPICLK	SPI Clock – DSP Engine is Master
14	*		SPISO	SPI Serial Out (MISO)
14		*	SPISI	SPI Serial In (MOSI)
15	*		DAI 4	
15		*	#RESET	
16	*		DSPCLKIN	External DSP Clock Input
16		*	SYSCLK	Programmable clock
17	*		Vd+3.3	
17		*	GND	
18	*		DAI 5	
18		*	DAI 6	
19	*		GND	
19		*	MCLK0	Programmable Clock
20	*		DAI 7	
20		*	DAI 8	
21	*		DAI 9	
21		*	DAI 10	
22	*		GP0	General Purpose Output
22		*	GND	
23	*		DAI 11	
23		*	DAI 12	
24	*		GND	
24		*	Vd+3.3	
25	*		Vd+3.3	
25		*	GND	
26	*		DAI 13	
26		*	DAI 14	
27	*	<u> </u>	GND	
27		*	MCLK1	Programmable Clock

Row	A&C	В	Name	Notes
28	*		DAI 15	
28		*	DAI 16	
29	*		DAI 17	
29		*	DAI 18	
30	*		GP1	General Purpose Output
30		*	GND	
31	*		DAI 19	
31		*	DAI 20	
32	*		GND	
32		*	Vd+3.3	

\* IOO-4, SPI\_SSs, & GP0 & GP1 are controlled by the PLD Registers

Refer to the dspstak Family Users Manual for general Interconnect Port information. The USB & PLD Register section of this manual describes the software interface to those pins mapped by the PLD.

## **Parallel Port**

The Parallel Port is connected to two major components, the USB Port and a PLD that is used to create and manage much of the dspstak's I/O ports. In addition, the Parallel Port is brought out to a 30 pin 2mm female dual row header (JH2).

You can use JH2 to expand to your own Parallel Port devices provided you consider a few simple points. They are as follows:

- The port can be configured for either 8 bit or 16 bit operation but AD15 must select a low address (A23 or A15 = 0)
- A low profile board will fit between the dspstak sx and an I/O Module.
- A special 2mm male header can be used to connect from a dspstak I/O Module to JH2 but there is no guarantee that another dspstak DSP Engine will be compatible. Samtec is a supplier for this type of connector.

Note: JH2 does not exist on the dspstak sx with ICE.

#### JH2 Pinout

Pin	Name	Pin	Name	Pin	Name
1	GND	11	AD1	21	AD11
2	GND	12	AD0	22	AD10
3	Vd+3.3	13	#WR	23	AD9
4	Vd+3.3	14	#RD	24	AD8
5	AD7	15	#RESET	25	GND
6	AD6	16	ALE	26	CLKOUT
7	AD5	17	AD15	27	Vd+3.3
8	AD4	18	AD14	28	Vd+3.3
9	AD3	19	AD13	29	GND
10	AD2	20	AD12	30	GND

The two onboard components, the USB port and the PLD are both mapped into the upper address space of the Parallel Port. These devices assume that the interface is configured as a 16 bit data bus. The memory map and details on using these peripherals are covered in the USB & PLD Registers section of this manual.

### **SPI Port**

The SPI is used as a general purpose control bus on the dspstak sx. It is connected to two onboard components, a serial flash memory and the Peripheral Microcontroller. It is also available on the Interconnect Port to control data converters and provide general purpose I/O expansion. Like all other dspstak DSP Engines, the DSP is always the master.

The flash memory is selected via Flag 0 of the DSP. This device is used to bootload your application program into the internal memory of the DSP. Details on uploading programs to the flash memory are in the Programming the dspstak sx section of this manual. A special bootloader program resides in the lowest block of the flash and is unavailable for user applications.

The bootloader program erases the flash when new programs are uploaded. The only exception to this rule is the highest block of the flash. This block is completely ignored by the bootloader and may be used (with care) by your application. *Earlier dspstak sx DSP Engines erased the entire flash memory*.

**WARNING:** Be very careful of writing to the flash memory and never ERASE or WRITE the lowest block of the flash! The only way to recover from an inadvertent corruption of the bootloader program is to rewrite this space with an in-circuit emulator.

The Peripheral Microcontroller is a preprogrammed part that provides the RS-232 port, a watchdog timer, brown out protection and user EE memory. It communicates over the SPI port using Flag 2 as its slave select. Refer to the Peripheral Microcontroller API section of this manual for more information.

The SPI Port is also brought out to the Interconnect Port. Flag 3 is expanded via the PLD to provide six SPI slave select lines. Generally, the #SPI\_SS pin of the Interconnect Port is used to control a data converter on a dspstak I/O module. The additional 5 slave selects are shared with IOO – IO4, and are usually used for expanding the SPI interface for additional I/O. For example, you could use this interface to create a front panel on an instrument where switches or encoders were scanned and status indicators were controlled. You could control a small LCD display via this interface. Refer to the USB & PLD Register section of this manual for details.

Danville uses a 2x5 0.100 box header as a standard connector for SPI off-board expansion. It is described in the dspstak Family Users Manual. We suggest you add a small (33 ohm) series resistor at the driving end of the SPI SO line for any remote interfaces. Keep in mind that the SPI interface is not intended for long unbuffered runs. You may need to add buffers or run the SPI interface at slow Clock speeds for reliable operations.

## **Programming the dspstak<sup>™</sup> sx**

The dspstak sx can be programmed in a variety of ways. In most cases, programs are uploaded via either the USB port or RS-232 port into the on-board serial flash memory. You may also load and execute programs through the JTAG port by using an emulator.

## **Programming Modes**

The dspstak sx has eight different programming modes that are assigned via jumper on the Configuration Header (JH5). The purpose of the programming modes is to establish what the dspstak sx will do when it comes out of reset. Four of the programming modes are reserved for standard dspstak sx functions and the remaining four are available for user programs.

For example, you could use Mode 0 as a means to bootload one version of your application and use Mode 1 to bootload another. The dspstak sx uses Mode 7 as Clock Programming Mode and Mode 6 as Program Uploading Mode. Mode selection is under the control of the Peripheral Microcontroller. You can determine the mode through the Peripheral Microcontroller API.

Mode	Name	M2	M1	M0
0	User 0	On	On	On
1	User 1	On	On	Off
2	User 2	On	Off	On
3	User 3	On	Off	Off
4	Reserved	Off	On	On
5	Reserved	Off	On	Off
6	Program Uploading	Off	Off	On
7	Clock Programming	Off	Off	Off

Mode – Jumpers (JH5)

\* Jumper On = 0

## **Boot & Clock Configuration**

The DSP is configured to boot from serial flash (Boot Mode 01). This mode causes the SPI port to automatically load a DSP program from the flash memory after reset is deasserted.

The DSP is configured in hardware for a Core Clock /Clock In ratio of 3:1or 6:1 (ClkCfg Mode 00). In most cases, the DSP clock is 24.576 MHz. For example, with an ADSP-2126x DSP the clock will operate at an initial core clock of 73.728 MHz. The DSP can change the ratio via software commands. We generally set this ratio to 8:1 so that the DSP operates at 196.608 MHz, which is essentially the ADSP-21262 maximum core clock of 200 MHz. This is a little different on ADSP-2136x versions, since ClkCfg Mode 00 configures the clock to a 6:1 ratio.

The hardware settings will cause the Core Clock/Clk In ratio to be either 3:1 or 6:1 depending on the DSP. By preconfiguring the hardware for ClkCfg Mode 00, all pin compatible members of this SHARC family can be supported. We simply adjust the Core Clock/Clk In ratio in software.

## **Programming the Clock – Mode 7**

The dspstak sx uses a Cypress Semiconductor CY22393 programmable clock generator to provide clocks for the Peripheral Microcontroller, the DSP, and the Interconnect Port. The Interconnect Port clocks are SYSCLK, MCLK0 & MCLK1.

The CY22393 has three independent PLLs that allow you to generate clocks that are appropriate for your situation. The three programmable clocks on the Interconnect Port allow I/O Modules to use convenient clocks for whatever devices that might be present. For example, an ADC might use a 18.432 MHz MCLK to sample at 96k or a 19.6608 MHz MCLK to sample at 102.4k.

If you want to reprogram the CY22393, you should review the CY22393 data sheet. You will also need the Cypress program, CyberClocks<sup>™</sup> to generate a JEDEC file of clock parameters. The data sheet and the program are available on the Cypress web site and there are links on the Danville web site at http://www.danvillesignal.com/index.php?id=dspdev.

The default clock settings are programmed in the Peripheral Microcontroller via its RS-232 port. It is not dependent on the DSP. The Peripheral Microcontroller also gets its clock from the CY22393 so steps have been taken to protect this clock from inadvertent modification. The following table describes the CY22393 mapping to the dspstak sx.

There are two sets of clock frequencies that can be switched via a simple software call to the Peripheral Microcontroller. It is also possible to change the clock frequencies from the DSP via software commands. The software commands are volatile whereas the settings programmed via Mode 7 are stored in the flash.

#### **Clock Configuration – CY22393**

Clock	CY22393	Clk Set 1	Clk Set 0
		(Default)	
Peripheral Microcontroller	XBUF	18.432 MHz	18.432 MHz
DSP Clock (JH1 Pins 2&3)	CLKC	24.576 MHz	24.576 MHz
SYSCLK	CLKD	24.576 MHz	24.576 MHz
MCLK0	CLKA	22.5792 MHz	12.288 MHz
MCLK1	CLKB	12.288 MHz	12.288 MHz
Not Used	CLKE	Off	Off

It is also possible to provide an external clock to the DSP via the Interconnect Port. In this configuration, JH1 Pins 1&2 are shorted. It might be prudent to disable the DSP Clock (CLKB) in this situation.

Here are the basic steps to reprogramming the CY22393:

- Configure the dspstak sx to operate in Mode 7 (M2, M1, M0 Jumpers Off).
- Connect to an ASCII Terminal Program via RS-232. A 9 pin to 9 pin cable with no twists is appropriate on PC compatible computers. The Serial Configuration must be 19.2K, 8, 1, N. Handshaking lines are ignored. A free terminal program is provided on the distribution CD.
- Create a new JEDEC file using CyberClocks<sup>™</sup>. The file 21262.JED is included on the CD. This is the factory default and a good place to start.
- Power Up or Reset the dspstak sx. You should see a Clock> prompt. You may wish to type
  to see the available commands.
- Type **U** to upload a new JEDEC file. This will copy the file into the Working Registers (RAM). You can also use the **L** command to reload the factory default settings.
- If you are sure that you are ready to write, type the **W** command. You can verify your results against the Working Registers by typing **V**.
- Exit the Clock Mode with the **Q** command. This will vector you to Mode 6 where you can upload application programs.

## **Uploading Programs – Mode 6**

The dspstak sx uses a 4Mbit SPI serial flash memory device to bootload the DSP. Assuming that the boot mode selection is set for SPI Master Boot (see Configuration Jumpers), the DSP will boot from the SPI flash memory after reset is deasserted.

This boot capability is very useful in the DSP except that there is no native support in the DSP to program the flash memory. The dspstak sx (actually all dspstak DSP Engines) has a resident boot loader and flash writer program that addresses this problem.

The dspstak sx automatically boots a proprietary Danville program called boot262 that is located in the lowest block of the flash memory. The boot262 program in turn, boots your program uses the rest of the flash memory. You don't have to worry about reserving memory for boot262. In the process of loading your program, the boot262 program that resides in internal RAM is overwritten.

If the dspstak sx is connected to an ASCII terminal via either RS-232 or USB and it is operating in Mode 6, you have a five second window to interrupt the boot process before the application program is automatically loaded. When you interrupt the boot process, the dspstak enters Command Mode. Command Mode lets you upload programs to the flash memory. If the dspstak sx is operating in Modes 0 through Mode 3, your program will boot immediately as if the boot262 program didn't exist. It is still there and it loads your program, but it does so silently.

You write your DSP applications as normal and create a \*.ldr file (Boot Type = SPI Slave, Format = Binary, Width = 8). You don't really need to worry about where your program is stored in the flash memory, but we are going to tell you anyway. Your program starts at the beginning of the second block of the flash with an offset of 80H. The 80H space is used to store the name of your program, checksums, etc. When you upload a new program, all the memory in the flash is block erased with the exception of the first block where the boot262 program resides. It is possible to write the flash memory in the space above your application and the boot262 program after the application program has been uploaded, but you must be very careful that the boot262 program is never overwritten or erased. The only way to rewrite boot262 is with an in-circuit emulator.

It is very hard to damage the contents of the flash by accident, but it is possible. If you never write directly to the flash, you are unlikely to ever see a problem. Danville supplies sample code that manages the SPI bus where the flash memory resides. This code makes it even more difficult to write to the flash inadvertently.

Command mode is available via either the USB port or the RS-232 port. A USB device driver is included that emulates a COM port for Windows 2000 or Windows XP operating systems. Bootloader communications automatically switch between the RS-232 port and the USB port. If the USB port is active (enumerated), the USB port is used; otherwise, the RS232 port will be active. Here are the basic steps to uploading programs:

- Configure the dspstak sx to operate in Mode 6.
- Connect to an ASCII Terminal Program via RS-232 or USB. USB is faster.
  - If RS-232 is used, a 9 pin to 9 pin cable with no twists is appropriate on PC compatible computers. The default serial configuration is 19.2K, N, 8, 1. Handshaking lines are ignored. There is a command to change the baud rate to higher speeds. If you don't know the current baud rate, you can reset the baud via Mode 7 to 19.2K.
  - If USB is used, the baud settings are ignored and the interface will operate like a very fast COM port.
- Power Up or Reset the dspstak sx. You should see a **DSP**> prompt followed by moving **#** symbols. After about 5 seconds, the application program will boot unless you press a key to interrupt the boot process. This will bring a new **DSP**> prompt signifying Command Mode.
- Type ? to view a list of available commands.
- Type **U** to upload a new program file. Your program must be a \*.ldr (ADI loader file) with the following parameters:
  - Boot Type = SPI Slave
  - $\circ$  Format = Binary
  - $\circ$  Width = 8

You may also use the **UA** command to upload a similar file in ASCII format. This will take longer.

After you confirm the upload, the bootloader will erase all user flash memory space. At the "Ready:" prompt use your ASCII terminal program to send the .ldr file. Make sure you enable the "binary transfer" option.

In Tera Term Pro, choose "Send File..." from the "File" menu. Click the "Binary" checkbox, and then open your .ldr file.

As the file transfers, you will see one progress mark for every 32 bytes transferred. Data transfer must be continuous. If the host stops sending data for more than approximately 500ms, the bootloader will assume that the code upload is complete and terminate the transfer.

• After the program is uploaded, a checksum is calculated and you have the opportunity to enter a description of the program. The program description is an ASCII string that you can

use to identify your program. It does not impact the function of the DSP module in any way, but can be a helpful way to track program revisions. A typical program description might be "Filter version 1.00." The size of the program is also calculated.

• Type **Q** to quit Command Mode and the application program will boot.

There are a number of additional commands that are available in Command Mode. For example, you can read, write and erase EE memory form the DSP> prompt. This memory is also available to the DSP via the Peripheral Microcontroller API.

There is no option to read back the application program via the Command Mode. This is to help you keep your code a little more proprietary. We suggest that you use the checksum information as a check during production programming to insure that your program was loaded correctly.

## **Peripheral Microcontroller API**

The Peripheral Microcontroller communicates via the SPI bus using a 3 byte packet structure. Flag 1 serves as the SPI slave select line. The SPI bus is bi-directional. Commands are always initiated by the DSP. The Peripheral Microcontroller only responds. The Peripheral Microcontroller is not a fast device; as such you must delay transmission of each byte by 15us or more. This gives the Peripheral Microcontroller time to execute each command.

Most packets will have this form:

Cycle	DSP -> PM	PM-> DSP
Byte 1	Command	Status
Byte 2	Data	Echo Command
Byte 3	0x00	Response

### **Command Summary**

Name		Description
PM_Cmd_NOP	0x00	No Operation
PM_Cmd_COM_WR	0x01	Transmit a byte from the RS-232 port
PM_Cmd_COM_RD	0x02	Receive a byte from the RS-232 port
PM_Cmd_VERSION	0x03	Firmware version of the Peripheral Microcontroller
PM_Cmd_COM_BAUD	0x04	Set the UART baud rate
PM_Cmd_EE_ADDR	0x05	Set the address of the EE Memory
PM_Cmd_EE_WR	0x06	Write a byte to EE Memory
PM_Cmd_WD	0x07	Watchdog Timeout
PM_Cmd_COM_RESET	0x08	Reset the UART
PM_Cmd_EE_WREN	0x09	Enable Writes to EE Memory
PM_Cmd_RESET	0x0A	Reset the dspstak
PM_Cmd_MODE	0x0B	Get the dspstak mode setting
PM_Cmd_COM_RTS_CTS	0x0C	RS-232 handshaking
PM_Cmd_EE_RD	0x0D	Read a byte from EE Memory
PM_Cmd_CLOCK_BANK	0x0E	Selects one of two sets of predefined clock settings
PM_Cmd_RESYNC	0x0F	Used to resync the Peripheral Microcontroller
continued		

Name		Description
I2C Commands:		12C is used to control the Clock Generator
PM_Cmd_I2C_START	0x10	
PM_Cmd_I2C_STOP	0x11	
PM_Cmd_I2C_TX	0x12	
PM_Cmd_I2C_RX	0x13	
PM_Cmd_I2C_RXNA	0x14	
PM_Cmd_I2C_GETDATA	0x15	
Reserved	>0x16	

### **System Commands**

#### Status

Description: Peripheral Microcontroller Status Command: Any

Status is automatically updated each time a packet is transmitted by the DSP. It is reported in Byte 1 in the next packet. This means that the status reported in the very first packet may not be valid.

Status is only valid up to the point of the completion of the previous command. So, if the last command was 2 weeks ago, the status bits will be two weeks old. Unless you are doing commands regularly, it's best to always do a PM\_Cmd\_NOP before you read the status bits.

Status Bits:

- Bit 0: UART RX Data Available
- Bit 1: UART TX Space Available
- Bit 2: -- Reserved
- Bit 3: I2C Busy (Operation in progress, wait until complete before next command)
- Bit 4: UART Overrun (Reset using PM Cmd COM RESET)
- Bit 5: -- Reserved
- Bit 6: EE Busy (Write in progress, will ignore any write attempts)
- Bit 7: WDT Reset (The last reset was due to a watchdog timer timeout)

1 = TRUE, 0 = FALSE

#### PM\_Cmd\_NOP

Description:No OperationCommand:0x00Data:Don't CareResponse:0x00

The NOP command is a good way to get the status of the Peripheral Microcontroller without any other action.

#### PM\_Cmd\_VERSION

Description:Returns the firmware version of the Peripheral MicrocontrollerCommand:0x03Data:Don't CareResponse:0x00

The current version is 0x00, but future versions will increment this number. This allows you to determine what functions this particular Peripheral Microcontroller implements in the event that there are changes in a later version.

#### PM\_Cmd\_CLOCK\_BANK

Description:Select Clock BankCommand:0x0EData:See BelowResponse:Don't Care

The Cypress CY22393 can be programmed with alternate sets of clock assignments. This commands selects the active bank. Only Bit 0 is checked. You can use Mode 7 to load a clock configuration file.

Clock frequencies can also be modified by using I2C commands to read and write the CY22393. This is more flexible and more complicated.

Data: 0x0 Select Bank 0 0x1 Select Bank 1 (Default)

#### PM\_Cmd\_WD

Description:Set Watchdog Timeout PeriodCommand:0x07Data:See BelowResponse:Don't Care

If the watchdog timer is active, the application must ping the WD input of the Peripheral Microcontroller with a state change before the watchdog timer expires or the system will reboot. The watchdog timer is initially disabled.

Data:	0xA6	Disable Watchdog
	0x00	18 ms
	0x01	36 ms
	0x02	72 ms
	0x03	144 ms
	0x04	288 ms
	0x05	576 ms
	0x06	1.152 s
	0x07	2.304 s

With the exception of 0xA6, all data is masked to use only the lowest 3 bits, i.e. 0xF1 = 0x01.

The watchdog timer uses an independent RC oscillator with a nominal period of 18 ms. This period can vary from 7 ms to 33 ms, so you may want to assume a worst case situation of 7ms, to keep the watchdog timer from overflowing inadvertently.

#### PM\_Cmd\_RESET

Description:Reset the dspstak sxCommand:0x0AData:0xC4Response:Don't Care

This is a software reset that acts just like a hardware reset.

#### PM\_Cmd\_MODE

Description:Return Mode Jumper SettingsCommand:0x0BData:Don't CareResponse:See Below

Returns the mode setting selected by M2, M1, M0 in the lowest three bits of the response. Modes 0 through 3 are User Modes. You may want to use this information to cause your program to execute different applications depending on the Mode settings. For example, your product might be an encoder in Mode 0 and a decoder in Mode 1. You would only need to upload one program for both applications.

The upper 5 bits are 10100. These bits are used as an integrity check to make sure that the Peripheral Microcontroller is actually present.

Mode 0:	0xA0	(User Mode 0)
Mode 1:	0xA1	(User Mode 1)
Mode 2:	0xA2	(User Mode 2)
Mode 3:	0xA3	(User Mode 3)
Mode 4:	0xA4	(Reserved, Do not use)
Mode 5:	0xA5	(Reserved, Do not use)
Mode 6:	0xA6	(Upload Programming Mode)
Mode 7:	0xA7	(Clock Programming Mode)

#### PM\_Cmd\_RESYNC

Description:Resync the Peripheral MicrocontrollerCommand:0x0FData:0x0FByte 3 (DSP):0x0FResponse:0x00

This command is used in the event that packet order is somehow misaligned. After this command is executed, the Peripheral Controller will expect the next byte to be Byte 1.

### **Com Port Commands**

#### PM\_Cmd\_COM\_WR

Description:Write to the Com Port.Command:0x01Data:Byte to transmitResponse:0x00

This command puts a byte on the transmit FIFO. The length of the FIFO is 32 bytes The Peripheral Controller automatically sends out by bytes at the selected baud rate.

You should check the UART TX Space Available Status bit in the next packet. If the last byte was unsuccessfully added to the FIFO (the Transmit FIFO was full), the application can resend this byte.

#### PM\_Cmd\_COM\_RD

Description:	Read from the Com Port.
Command:	0x02
Data:	Don't Care
Response:	Next available byte from the Receive FIFO.

This command gets a byte from the receive FIFO. The length of the FIFO is 32 bytes.

If there is no pending data (the Receive FIFO was empty), the command is treated as a NOP, and therefore the Response will be 0x00. The UART RX Data Available Status bit can be used to validate the data. This would distinguish between a data "0x00" and a NOP response.

#### PM\_Cmd\_COM\_RESET

Description:Resets the Com PortCommand:0x08Data:Don't CareResponse:0x00

This command resets the UART, flushes the Transmit and Receive FIFOs and clears the UART Overrun Status bit.

#### PM\_Cmd\_COM\_BAUD

Description:Sets the Baud rateCommand:0x04Data:See BelowResponse:0x00

Baud =  $(1.152 * 10^{6}) / (DATA + 1)$ 

Typical Values:

230.4K	0x04
115.2K	0x09
38.4K	0x1F
19.2K	0x3D
9600	0x79

#### PM\_Cmd\_COM\_RTS\_CTS

Description:RS-232 HandshakingCommand:0x0CData:Bit 0 is CTSResponse:Bit 0 is RTS

Since the RS-232 Port is defined as a DCE, CTS is an output and RTS is an input. This command is used to handshake RS-232 lines. With an RS-485/RS-232 adapter, CTS may also be used as a RS-485 DE (Driver Enable).

### **EE Memory Commands**

#### PM\_Cmd\_EE\_ADDR

Description:Sets the EE Address PointerCommand:0x05Data:EE AddressResponse:0x00

The EE Address pointer is auto incremented after every valid EE\_WR or EE\_RD operation. The valid address range is 0x00 to 0x7F (128 bytes). The EE Data at Address 0x7F is reserved by the bootloader to hold the default baud rate, so don't overwrite this location.

#### PM\_Cmd\_EE\_WREN

Description:Enables Writes to the EE MemoryCommand:0x09Data:0x42 (Enable), All other values (Disable)Response:0x00

This command allows the EE memory to be written. You should disable writes after you are done writing the EE Memory. EE\_WREN is always disabled after a system reset.

#### PM\_Cmd\_EE\_WR

Description:Write EE at current EE AddressCommand:0x06Data:EE DataResponse:0x00

This command is only valid if the EE\_WREN has been enabled. EE Data is written to the current address pointed to by the EE Address Pointer. This address is set by the EE ADDR command and auto incremented after every EE WR or EE RD.

The EE Data at Address 0x7F is reserved by the bootloader to hold the default baud rate, so don't overwrite this location.

You cannot write the EE Memory faster than 1 byte every 10 ms. The EE Busy Status Bit will be 1 while a write is in progress. You can use this bit to time EE writes.

#### PM\_Cmd\_EE\_RD

Description:Read EE at current EE AddressCommand:0x0DData:Don't CareResponse:EE Data

EE Data is read from the current address pointed to by the EE Address Pointer. This address is set by the EE\_ADDR command and auto incremented after every EE\_WR or EE\_RD.

#### **I2C Commands**

PM	_Cmd_I2C	_START
PM	_Cmd_I2C	_STOP
PM	_Cmd_I2C	_TX
PM	_Cmd_I2C	C_RX
PM	_Cmd_I2C	_RXNA
PM	_Cmd_I2C	_GETDATA

Description:Collection of I2C Commands to Write/Read Clock GeneratorCommand:0x10 - 0x15Data:Byte to transmitResponse:Return Byte

I2C commands don't really function by themselves. Each command is part of a larger I2C operation. Here are some examples:

I2C Write Sequence:

I2C START
I2C TX – Device Address Write
I2C TX – Register Address
I2C TX – First Data Byte
I2C TX – Second Data Byte
I2C TX – Next Data Byte, etc.
I2C STOP

I2C Read Sequence:

I2C START
I2C TX – Device Address Write
I2C TX – Register Address
I2C Start
I2C TX – Device Address Read
I2C RX – Read Data over wires
I2C GETDATA – Return Data Byte that was read
I2C RX – Read Data over wires
I2C GETDATA – Return Data Byte that was read
I2C RX – Read Data over wires
I2C GETDATA – Return Data Byte that was read
I2C RX – Read Data over wires
I2C GETDATA – Return Data Byte that was read
I2C GETDATA – Return Data Byte that was read
I2C GETDATA – Return Data Byte that was read
I2C GETDATA – Return Data Byte that was read
I2C GETDATA – Return Data Byte from final read
I2C STOP

After every I2C operation, you need to poll the I2C Active Bit (Bit 3) of the status flags. This bit will be high whenever an I2C operation is in progress. You must wait until this bit goes low before starting the next operation.

## **USB & PLD Registers**

The dspstak sx uses a PLD for I/O expansion. It is connected to the DSP's Parallel Port. The USB port is also connected to the Parallel Port. It uses the PLD for part of its internal interface.

The Parallel Port on the DSP has 16 multiplexed address and data lines and three control lines, ALE, RD & WR. It can be configured as either an 8 bit bus with 24 address lines or a 16 bit bus with 16 address lines. ALE is used as an address latch enable to demultiplex the bus.

Another characteristic of the Parallel Port is that it always wants to do 32 bit transfers. It automatically packs and unpacks data so that from an external device point of view, an 8 bit transfer is always 4 cycles and a 16 bit transfer is always 2 cycles. If the Parallel Port is mapped to an external memory device such as a parallel flash memory or SRAM, this packing/unpacking feature is quite convenient, but it creates difficulties when the target is an I/O port or FIFO.

Both the PLD and the USB port are mapped to the upper half of the Parallel Port memory. This means that you are free to use JH2 for external Parallel Port expansion as long as your devices are mapped into the lower half of the memory. In other words, AD15 must be demultiplex as either A15 or A23 = 0. The PLD and USB require that the Parallel Port be configured as a 16 bit data bus, however external devices can use either the 8 bit or 16 bit configuration.

The PLD is used to create seven 5 bit wide registers, demultiplex the upper AD lines, and map the USB port to the Parallel Port. It also qualifies the RD & WR lines with AO, to deal with the "helpful" packing/unpacking feature. The RD and WR lines are only active when AO is low. This means that the upper 16 bits of the 32 bit word are always ignored.

## Addressing USB & PLD Registers

AD15 – AD11 (A15 – A11) are used for address decoding. Lower address bits are ignored.

## PLD I/O Register & USB Status Data Format

The registers are 5 bits wide where D15-D11 are the only valid bits. The Parallel Port's Transmit Buffer Register (TXPP) is used to transmit data and the Receive Buffer Register (RXPP) is used to receive data.

TXPP Register Format (Binary):	xxxx xxxx xxxx abcd exxx xxxx xxxx where x is Don't Care and abcde represents the register data.
RXPP Register Format (Binary):	xxxx xxxx xxxx abcd exxx xxxx xxxx where x is Don't Care and abcde represents the register data.

## **Memory Map**

Name	Addr	Dir	Bits Used	Description
USB_TX_DATA	0x8F00	W	D7-D0	USB Data Bus
USB_RX_DATA	0x8FFE	R	D7-D0	USB Data Bus
USB_STATUS	0x9000	R	D15-D12	USB Status
VERSION	0x9800	R	D15-D11	Version
Reserved	0xA000			Used in previous revision
Reserved	0xA800			
PLD_OUT0	0xB000	R/W	D15-D12	Digital Outputs
PLD_IO_DIRECTION	0xB800	R/W	D15-D11	1 = Output, 0 = Input
PLD_OUT1	0xC000	R/W	D15-D12	Digital Outputs
SPI/IO_SELECT	0xC800	R/W	D15-D11	1 = SPI, 0 = I/O
PLD_SPI_SELECT	0xD000	R/W	D15-D12	1 = SPI SS Active
PLD_SPI_SELECT	0xD800	R/W	D15-D12	1 = SPI SS Active
PLD_WD	0xE000	W	None	Watchdog Strobe
Reserved	0xE800			
User Space (JH2)	<0x8000			This space is available to external
				devices
User Space (JH2)	0xF000			This space is available to external
				devices

Danville recommends writing A10-A8 with 0 for future compatibility. The exception to this rule is USB\_RX\_DATA. This address is always 0x8FFE to protect the USB FIFO.

The PLD Write Registers will ignore all bits except D15-D11.

The PLD Read Registers will return invalid data in all bits except D15-D11

### **USB Port**

The USB port on the dspstak sx uses an FTDI FT245BM USB transceiver. This device is a USB 2.0 compliant, full speed part (12Mb/s). This part is very easy to use since in most cases it just looks like a COM Port to the USB Master. FTDI has royalty free device drivers on their web site that support a variety of operating systems. We suggest that you visit their web site (www.ftdichip.com) for additional information.

We supply Windows® drivers on our distribution CD. The driver files includes dspstak \*.inf files that are used for installation in Microsoft Windows® operating systems. If you are using another OS,

you may want to compare the ftdibus.inf and ftdiport.inf files to the dspstakbus.inf and dspstakport.inf files to configure your driver. Instructions on installing the Windows® drivers are on the CD.

The FT245BM has receive and transmit FIFOs that are read from or written to the DSP. There are three USB status bits: RXF, TXE and PWREN. PWREN is low when the USB port is enumerated (recognized by the USB Master). If RXF is low, you can receive one byte; if TXE is low, you can transmit one byte. There is no guarantee that the USB port can accept or deliver more than one byte based on the flags, therefore you must check the status before sending or receiving the next byte.

Reading and writing the Parallel Port is more complicated than a simple I/O or Memory R/W instruction. You will want to review the Analog Devices documentation and the sample code provided on our Distribution CD to see how it's done in detail.

#### **USB Status Bits**

Address:	0x9000	
PWREN:	D14	0 – USB Port is enumerated
TXE:	D13	0 – Transmit FIFO can accept a byte
RXF:	D12	0 – Receive Buffer has a byte ready

All other bits are Don't Care.

#### **Reading the USB Port**

- Read the USB Status Register. If RXF is low (and assuming PWREN is low), you may read the USB Data.
- Read the USB Data Register.

This seems pretty straightforward until you learn the "gotcha." The Parallel Port has a two word receive buffer that has to fill before the data can be read internally. This means that there will be three read cycles before you actually get the data. The USB may or may not have valid data available if it is clocked successive times since the RXE flag only guarantees one valid byte is available.

So here's the trick!

We address the Parallel Port to read from Address 0x8FFE. You may recall that only A15-A11 is decoded by the PLD. The address then auto increments to 0x9000 which points to the USB Status Register and does not strobe the USB Read Line. You can read the USB Status

Register successively with no side effects. This is a convenient way to check the status for the next byte in the USB queue.

This trick may be useful for interfacing high speed ADCs as well since many of the same issues may apply. Of course, if the FIFO status flags are deeper, then this won't be a problem.

#### Writing the USB Port

- Read the USB Status Register (0x9000). If TXE is low (and assuming PWREN is low), you may write the USB Data.
- Write the USB Data Register (Address 0x8F00).

There are no extra buffers in the transmit path, so the Receive "trick" is not needed.

### **Version Register**

#### VERSION

Description:Returns current version of the dspstak sxDestination:NoneAddress:0x9800

The version register is used for software compatibility with older and future revisions of the dspstak sx. This register was previously undefined in the earlier dspstak sx and will probably return 0x1F (D15-D11). The current version returns 0x0A (D15-D11).

You can read the version if you are supporting earlier versions of the dspstak and need to know if some features are not available.

### PLD I/O & SPI SS Registers

#### PLD\_OUT0

Description:	3.3V Digital Output	
Destination:	Interconnect Port	
Address:	0xB000	
IO3:	D15	1 – Output High, 0 – Output Low
IO2:	D14	1 – Output High, 0 – Output Low
IO1:	D13	1 – Output High, 0 – Output Low
IO0:	D12	1 – Output High, 0 – Output Low

All other bits are Don't Care.

Writing this register causes the IO3-IO0 lines on the Interconnect Port to be driven if the I/O pin is configured as an output (PLD\_IO\_DIRECTION register) and the pin is not configured as an SPI SS line (SPI\_SELECT\_SECONDARY register). If the I/O pin not configured as an output, the setting is ignored.

#### PLD\_IN0

Description: Destination: Address:	3.3V Digital Input Interconnect Port 0xB000	
IO3:	D15	1 – Input High, 0 – Input Low
IO2:	D14	1 – Input High, 0 – Input Low
IO1:	D13	1 – Input High, 0 – Input Low
IO0:	D12	1 – Input High, 0 – Input Low

All other bits are Don't Care.

Reading this register returns the state of the IO3-IO0 lines on the Interconnect Port.

#### PLD\_OUT1

Description:	3.3V Digital Output	
Destination:	Interconnect Port	
Address:	0xC000	
#RESET:	D15	1 – Output High, 0 – Output Low
GP1:	D14	1 – Output High, 0 – Output Low
GP0:	D13	1 – Output High, 0 – Output Low
IO4:	D12	1 – Output High, 0 – Output Low

All other bits are Don't Care.

Writing this register causes the corresponding pins on the Interconnect Port to be driven. IO4 will only be driven if it is configured as an output (PLD\_IO\_DIRECTION register) and it is not configured as an SPI SS line (PLD\_SPI\_IO\_MODE register). #RESET, GP1 & GP0 are always outputs.

#### PLD\_IN1

Description: Destination:	3.3V Digital Input Interconnect Port	
Address:	0xC000	
#RESET:	D15	1 – Input High, 0 – Input Low
GP1:	D14	1 – Input High, 0 – Input Low
GP0:	D13	1 – Input High, 0 – Input Low
IO4:	D12	1 – Input High, 0 – Input Low

All other bits are Don't Care.

Reading this register returns the state of the corresponding lines on the Interconnect Port.

#### I/O DIRECTION

I/O Di Interna	rection & SPI 3 al Mask	SS State
0xB80	0	
ION:	D15	1 – Output, 0 - Input
ION:	D14	1 – Output, 0 - Input
ION:	D13	1 – Output, 0 - Input
ION:	D12	1 – Output, 0 - Input
ION:	D11	1 – Output, 0 - Input
DE:	D15	1 – Active, 0 - Open Drain
DE:	D14	1 – Active, 0 - Open Drain
DE:	D13	1 – Active, 0 - Open Drain
DE:	D12	1 – Active, 0 - Open Drain
DE:	D11	1 – Active, 0 - Open Drain
	I/O Di Interna 0xB80 ION: ION: ION: ION: ION: DE: DE: DE: DE: DE:	I/O Direction & SPI : Internal Mask 0xB800 ION: D15 ION: D14 ION: D13 ION: D12 ION: D11 DE: D15 DE: D14 DE: D14 DE: D13 DE: D12 DE: D12 DE: D11

All other bits are Don't Care.

If the pin is defined as I/O, then the direction is defined. If the pin is defined as an SPI SS, then the pin may be open drain or driven.

#### PLD\_SPI\_IO\_MODE

Description:	3.3V Digital Output	
Destination:	Internal Mask	
Address:	0xC800	
SS5/IO4:	D15	1 – SPI, 0 – I/O
SS4/IO3:	D14	1 – SPI, 0 – I/O
SS3/IO2:	D13	1 – SPI, 0 – I/O
SS2/IO1:	D12	1 – SPI, 0 – I/O
SS1/IO0:	D11	1 – SPI, 0 – I/O

All other bits are Don't Care.

This register is determines whether a pin is used for SPI SS or General Purpose I/O. The SS will be high (or open drain) if Flag 3 is inactive or the corresponding bit in PLD\_SPI\_SELECT\_AUX\_SS is 0.

#### PLD\_SPI\_SELECT\_SS

Description:3.3V Digital OutputDestination:Internal MaskAddress:0xD000

SPI SS (SS0): D12 1 – SS Active

All other bits are Don't Care.

This register is used for SPI slave select expansion. If the mask bit is set, the pin will follow the state of FLAG3 on the DSP. In most cases, only one bit should be set in PLD\_SPI\_SELECT\_SS and PLD\_SPI\_SELECT\_AUX\_SS, since contention will occur if more than one device drives the MISO line.

#### PLD\_SPI\_SELECT\_AUX\_SS

Description: Destination: Address:	3.3V Digital Output Internal Mask 0xD800	
SS5:	D15	1 – SS Active
SS4:	D14	1 – SS Active
SS3:	D13	1 – SS Active
SS2:	D12	1 – SS Active
SS1:	D11	1 – SS Active

All other bits are Don't Care.

This register is used for SPI slave select expansion. If the mask bit is set, the pin will follow the state of FLAG3 on the DSP. In most cases, only one bit should be set in PLD\_SPI\_SELECT\_SS and PLD\_SPI\_SELECT\_AUX\_SS, since contention will occur if more than one device drives the MISO line.

#### PLD\_WD

Description: 3.3V Digital Output Destination: Peripheral Microcontroller Address: 0xE000

All data bits are Don't Care.

A write to this register causes the Watchdog input on the Peripheral Microcontroller to toggle. If the Watchdog Timer is enabled, you must write to this register before the Watchdog Timer can overflow or the system will reset.

## Software

The Distribution CD includes software examples that demonstrate many of the features of the dspstak sx. You may use these examples as take off points to start your own applications.

Our software library is constantly expanding. This code and other resources are available to our customers through our web site. Send an email to support@danvillesignal.com to gain access to the customer section of our web site.

## Schematic

The Distribution CD includes a schematic diagram of the dspstak sx in files sa21262.pdf and sb21262.pdf.

## **Mechanical Drawings**

Mechanical Drawings are included in the dspstak Family Users Manual. If you any need additional information on board mounting, clearances, etc, send an email to support@danvillesignal.com.

If you are designing your own I/O Module or want to mount the dspstak sx in an unusual configuration, you may want to check with us for suggestions, design review, etc. We also have PCB templates available to help you get started on an I/O Module layout. This will help insure that the connectors and holes on your board actually line up.

## **Product Revision**

The dspstak sx is in its second major revision. You can identify the revision by the pcb part number on the lower left corner of the board. The second major revision starts with PCB P/N MPCB.07266E. Earlier revisions are labeled MPCB.07266 through MPCB.07266D, and perform largely the same. The following sections outline the significant changes and the impact on existing systems.

### **Interconnect Port**

1. The pins on the Interconnect Port have been renamed to conform with the third generation SHARC family of DSPs. The original mapping imposed more rigid rules since older SHARCs such as the ADSP-21161 and other architectures were substantially less flexible.

Many of these changes are discussed in the dspstak Family User Manual (Version 2).

Impact: None.

2. DAIs 1-4 are no longer shared with auxiliary SPI Slave Select Lines. The auxiliary SPI Slave selects are now shared with general purpose I/O. This frees up the DAI for more complex I/O interfacing requirements.

Impact: Most early Danville dspstak I/O Modules used SPI\_SS (#SSO), Pin 13A&C. This assignment is unchanged. If auxiliary SPI SS were used, the slave selects can either be controlled manually via the DAI or they should be reassigned to new pin assignments.

3. The SPI port is now buffered from the DSP for increased drive capability.

Impact: None.

4. Digital I/O IO0 – IO4 replaced LED0 – LED4. These lines also can be defined as SPI Slave Selects #SS1-#SS5. Outputs can be either active or open drain.

Impact: The earlier revision allowed these lines to be Outputs only. In most cases, nothing needs to be changed.

5. Reset0# and Reset1# are renamed GP0 and GP1 to reflect actual usage. They are digital outputs only.

Impact: None.

Row	A&C	В	Version 2	Original	Notes - Changes
9	*		IO0/#SS1	LED0	Were originally Output Only
9		*	IO1/#SS2	LED1	
10	*		IO2/#SS3	LED2	
10		*	IO3/#SS4	LED3	
11	*		IO4/#SS5	LED4	
11		*	DAI 2	DAI 2 / SS6#	No longer SPI Slave Selects
12	*		DAI 3	DAI 3 / SS7#	
12		*	DAI 1	DAI 1 / SS8#	
13	*		#SPI SS	#SPI SS	
13		*	SPICLK	SPICLK	Buffered
14	*		SPISO	SPISO	Buffered
14		*	SPISI	SPISI	
15	*		DAI 4	DAI 4 / IRQ	
15		*	#RESET	#RESET	
16	*		DSPCLKIN	SYSCLKIN	New Name
16		*	SYSCLK	SYSCLKOUT	New Name
18	*		DAI 5	DAI 5	
18		*	DAI 6	DAI 6	
20	*		DAI 7	DAI 7	
20		*	DAI 8	DAI 8	
21	*		DAI 9	DAI 9	
21		*	DAI 10	DAI 10	
22	*		GP0	RESET0#	New Name
23	*		DAI 11	DAI 11	
23		*	DAI 12	DAI 12	
26	*		DAI 13	DAI 13	
26		*	DAI 14	DAI 14	
28	*		DAI 15	DAI 15	
28		*	DAI 16	DAI 16	
29	*		DAI 17	DAI 17	
29		*	DAI 18	DAI 18	
30	*		GP1	RESET1#	New Name
31	*		DAI 19	DAI 19	
31		*	DAI 20	DAI 20	

Here is a comparison of the changes to the Interconnect Port Mapping:

## **PLD Registers**

1. The PLD has doubled in size to accommodate more registers with greater complexity. Address A11 is now used as part of mapping.

Impact: The earlier revision supported digital outputs only. Assuming that an earlier design used the I/O for digital outputs you should set the PLD\_IO\_DIRECTION REGISTER bits to 1 and the PLD\_IO\_MODE bits to 0. These registers will be ignored by the old board and will correctly configure the newer board. If you are using auxiliary SS lines in the original design, you can configure the DAI to manually control the older SS lines or reassign the I/O to use the newer assignments. In general, the newer registers do not exist in the original design. This assumes that the original design always sets A11 to 0.

2. System Reset now reinitializes the PLD.

Impact: None.

3. The pins on the Interconnect Port have been renamed to conform with the third generation SHARC family of DSPs. The original mapping imposed more rigid rules since older SHARCs such as the ADSP-21161 and other architectures were substantially less flexible.

Impact: The earlier revision allowed these lines to be Outputs only. In most cases, nothing needs to be changed.

### **Programmable Clocks**

1. The default clock settings have changed from the original design. There are now two alternate sets of clock frequencies that can be changed via a simple software control. It is also possible to write the clock from the DSP using I2C commands (via new PM command).

Impact: You can change the default clock settings via Mode 7 in either version. Please note that the JEDEC files are different for each version.

## **Higher Analog Current Capability**

1. The rectifiers used for the unregulated analog supplies have been increased from 1A to 3A to support additional I/O cards. The Va+5 supply has not changed however so boards that require substantial 5V analog power should incorporate on-board regulators. This is generally important when multiple I/O boards may be stacked.

Impact: None, it is possible to retrofit 3A rectifiers on the earlier boards.

### **Secondary Mounting Holes**

1. There were two extra mounting holes located on the right side of the PCB (not the corner holes). These holes no longer exist on the new revision.

Impact: No workaround.

### Jumpers

1. The Boot Mode and Clock Mode jumpers have been eliminated. Boot Mode was superfluous since the boot mode was always SPI Master. The Clock Mode was eliminated since it can be easily programmed after booting and also guarantees that the bootloader will be running at a known core clock. Different members of the 2126x and 2136x family use different settings for Clock Mode as well. For this reason, Clock Mode is always set to Clock\_In x 3. We suggest that the Core Clock should be adjusted in the first few lines of the application program. It most cases, you will configure the Core Clock to be Clock In x 8. The example code illustrates this.

Impact: Add Clock Setting Instructions at the beginning of your application program.

2. The JTAG pins now pull high or low so that jumpers are no longer necessary when an ICE is not connected.

Impact: None.

## ADSP-2136x Compatibility

1. The new pcb layout now supports the ADSP-2136x DSP. The dspstak 21364sx is available for applications requiring greater speed (333MHz), more memory (3Mbit) or additional peripherals (S/PDIF, SRC, Aux SPI).

## **Product Warranty**

Danville Signal Processing, Inc. products carry the following warranty:

Danville Signal Processing products are warranted against defects in materials and workmanship. If Danville Signal Processing receives notice of such defects during the warranty period, Danville Signal Processing shall, at its option, either repair or replace hardware products, which prove to be defective.

Danville Signal Processing software and firmware products, which are designated by Danville Signal Processing for use with our hardware products, are warranted not to fail to execute their programming instructions due to defects in materials and workmanship. If Danville Signal Processing receives notice of such defects during the warranty period, Danville Signal Processing shall, at its option, either repair or replace software media or firmware, which do not execute their programming instructions due to such defects. Danville Signal Processing does not warrant that operation of the software, firmware, or hardware shall be uninterrupted or error free.

The warranty period for each product is one year from date of installation.

#### **Limitation of Warranty:**

The forgoing warranty shall not apply to defects resulting from:

- Improper or inadequate maintenance by the Buyer;
- Buyer-supplied software or interfacing;
- Unauthorized modification or misuse;
- Operation outside the environmental specification of the product;
- Improper site preparation and maintenance.

#### **Exclusive Remedies:**

The remedies provided herein are the Buyer's sole and exclusive remedies. In no event shall Danville Signal Processing, Inc. be liable for direct, indirect, special, incidental or consequential damages (including loss of profits) whether based on contract, tort, or any other legal theory.