SHELDON INSTRUMENTS

SI-C6xDSP USER'S GUIDE
September 2016

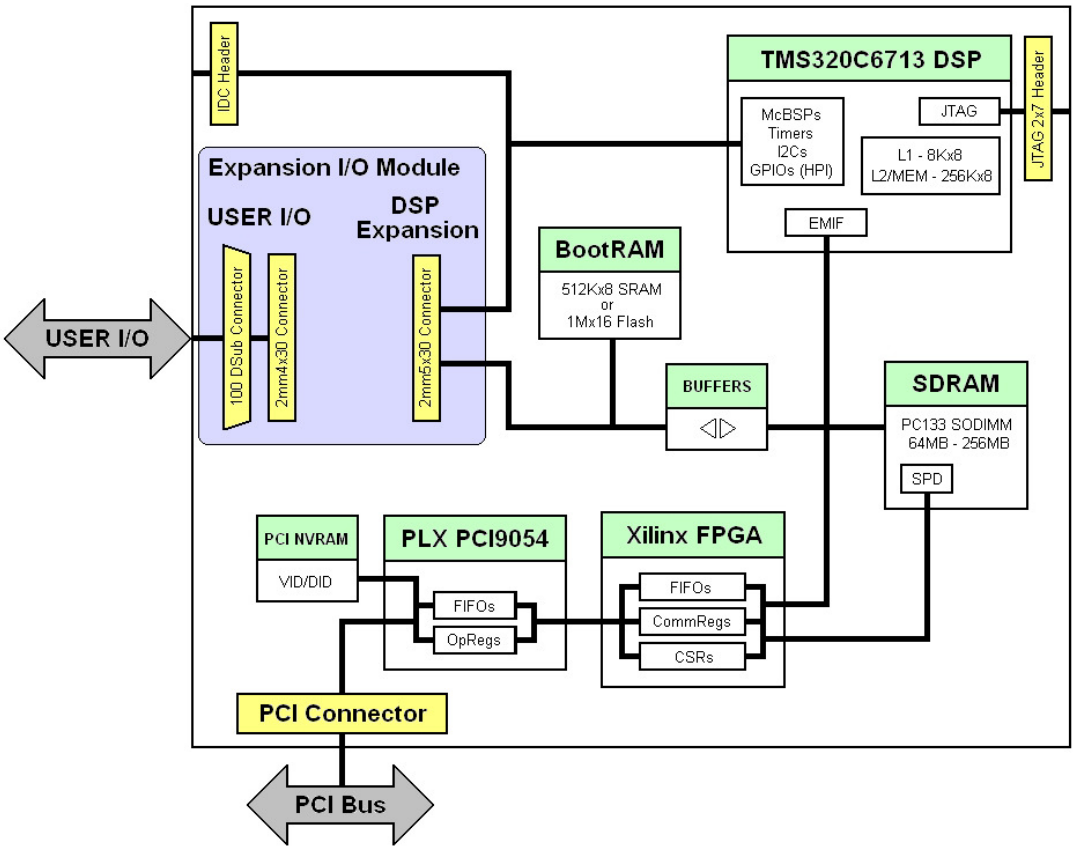
Table of Contents

1.0 System Overview

The SI-C671xDSP family from Sheldon Instruments is a powerful Digital Signal Processor (DSP) card for your PC equipped with a 32 bit PCI bus. It is based on Texas Instruments' 300/225Mhz TMS320C6713, 32 bit DSPs, and can transform your PC into an ultra high performance development system and DSP accelerator. A full line of software development tools are available from Sheldon Instruments and TI, which include compilers, assemblers, linkers, and debuggers.

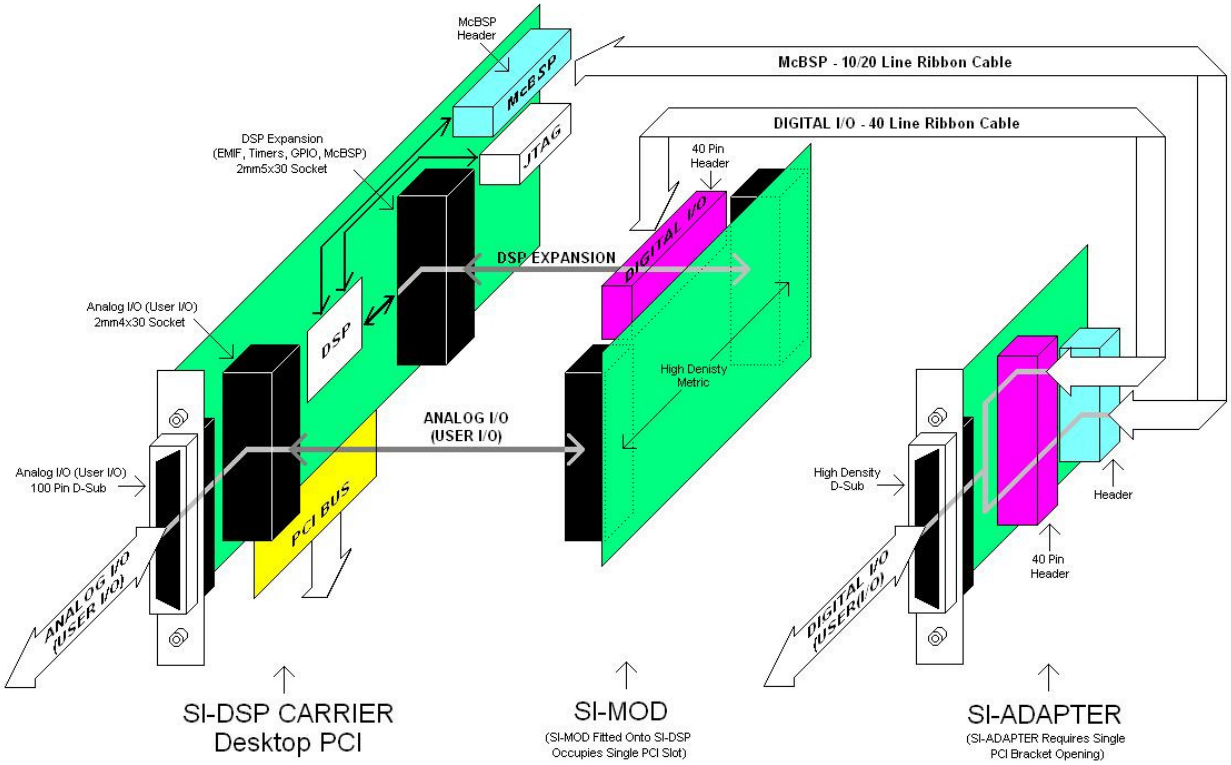
Key features include:

- 1800 MFLOP peak performance with C6713, 1200 MFLOPs with C6711, 32 bit floating/fixed point precision.
- Up to 256MB SDRAM, using conventional PC133 SDRAM DIMM/SO-DIMM format.
- Various flavors of PCI form factors: standard PCI, PC104Plus, PMC, and CompactPCI.
- Full bi-directional PCI initiated bus mastering, with 132MB/sec peak transfer rate.
- Memory mapped host communications port.
- Software development tools from TI and Sheldon Instruments, including QuVIEW and QuBASE.
- Windows and Linux 32/64 bit drivers and sample application support.
- Expansion connectors for prototyping, analog & digital I/O daughtercards.
- JTAG port for in system development and debugging.



2.0 Hardware Support

The SI-C6xDSP includes expansion connectors allowing for custom designs, or for attaching 'off the shelf' multifunction I/O modules from Sheldon Instruments. Sheldon Instruments offers several daughter modules for multichannel analog and digital I/O, including 4 to 64 channels of 16 bit ADCs and DACs.



3.0 Software Support

The SI-C6xDSP is available with extensive development tools from Sheldon Instruments and TI.

For quick turnkey development, Sheldon Instruments offers QuVIEW and QuBASE, which are a set of DSP-resident libraries for real time performance that greatly accelerate data acquisition, signal processing, and control applications. QuVIEW is a real time accelerator for LabVIEW, and QuBASE a real time accelerator for Visual Basic. A full range of examples and tutors are provided to demonstrate their ease of use and breadth of functionality and capabilities. QuBASE runs under Win98/2000/XP, while QuVIEW also runs under Linux.

When purchased as a DSP evaluation board, Sheldon Instruments includes projects and related source for the DSP side that is fully compatible with TI's development environment, along with projects and related source code for Windows or Linux 32/64 drivers and applications. The software illustrates full communication modes between the host and DSP, along with COFF file loader utilities.

4.0 System Initialization

4.1 Host System Resources

The PCI Base Addresses (BADDR) are automatically assigned by the host at boot time, with the requested resources as outlined in the following table:

PCI BADDRn	Range (Bytes)		Resource		Host BADDRn+Offset (Byte Boundary)	DSP CEn	DSP Base Address (Byte Boundary)
0	256		PLX OpRegs	Memory Mapped	0x00-0xFF	0	0x8034,0000-0x8037,FFFF
1				I/O Mapped (unused)			
2	8M	2M	FPGA: 2x 32 Deep FIFOs		0x00,0000-0x0F,FFFF		0x8020,0000-0x802F,FFFF
			FPGA: x64 CommRegs		0x10,0000-0x13,FFFF		0x8030,0000-0x8033,FFFF
			FPGA: x12 CSRs		CSR11: 0x10,0100-0x10,40FF; CSR[0:10]: 0x14,0000-0x1B,FFFF		0x8030,0100-0x8030,40FF; 0x803C,0000-0x803C,FFFF
			Expansion Module (64Kx32)		0x1C,0000-0x1F,FFFF		0x8038,0000-0x803B,FFFF
		6M	BootMEM (512Kx8 SRAM or 1Mx16 NOR Flash)		0x20,0000-0x7F,FFFF	1	0x9000,0000-0x9FFF,FFFF
3	128K		n/a		n/a	n/a	n/a
n/a	64K-2M		Add-on Init Buffer (DSP Side Only)		n/a	0	0x8000,0000-0x801F,FFFF

4.2 Loading Onboard Logic Inside FPGA

When the DSP board is first powered, no onboard logic is valid. Therefore, the FGPA that contains all onboard logic must be loaded in one of two ways:
1) Software from host PC (FPGA's SPROM is absent).
For most delivered systems, the FPGA contents are bitloaded from the host PC by invoking a batch file when the SPROM is absent. The batch file is named SIDSP_FPGALOAD.BAT which calls the SISAMPLE.EXE utility and the corresponding FPGA bitfile. The batch file is to be run only once, typically at bootup of the PC if placed inside the Windows 'Startup' folder.
2) Hardware, with FPGA's SPROM.
For standalone systems an onboard SPROM is present, contents are automatically loaded into the FPGA at power up.

***NOTE**:*
1) Without loading the FPGA, only internal registers inside of the PLX PCI bridge device will be accessible. All accesses to other external memories and registers will not be valid and will cause hard system lockups. It is imperative that the FPGA be loaded for proper system operation.
2) Before the batch file can be invoked, be sure the driver of a particular OS (Windows or Linux) is already loaded. Please refer to the Driver installation documentation for more details on loading drivers. By default, all SI-DSP/SI-MOD cards are shipped without the SPROMs; however, they may be optionally included at an added cost.

Under Windows, it is recommended that a batch file of the form " SIDSP_FPGALOAD.BAT ", be inserted into the \..\STARTUP subdirectory. Alternatively, a shortcut may be created to invoke the batch file from anywhere in Windows.

Under Linux, it is recommended that a script file of the form "FPGALOAD", be inserted into the

user's root or home directory.

***NOTE**: Please refer to the FPGALoad documentation for more details.*

4.3 DSP COFF File Downloads

In order for the DSP to be active, the contents of a COFF file (Common Object File Format) must be downloaded to the bootMEM memory from the host. The COFF file is a binary that contains the actual code to be run on the DSP, and is generated using TI's Code Composer Studio development environment. When first powered and after the onboard FPGA logic has been loaded, the DSP is defaulted to be in an inactive or in a 'Reset' state. However, before any COFF file is downloaded, the host will place the DSP into this inactive state for deterministic behavior. Once disabled, the COFF file contents must be written to the bootMEM with the host PC using target mode accesses.

The following sequence of events must occur for a successful COFF load:

1. The host disables the DSP by asserting the Reset line or writing a "0" value to CSR0-Bit0, the RESET Control bit.
2. While maintaining the DSP in the reset state, the host downloads the COFF file contents to the bootMEM. An optional readback of the bootMEM contents may be performed for verification.
3. The host activates the DSP by removing the reset or by writing a "1" to CSR0-Bit0, the RESET Control bit.
4. The DSP starts execution of its internal bootloader code, which in turn invokes the bootMEM.

NOTE:
1) A COFF file must be first downloaded from the host PC, subsequently with the DSP actively running in order for the JTAG emulator to operate correctly.
2) The DSP boot mode is configured to run from an external boot memory device, and NOT from the DSP's HPI port. Please refer to section 8.2 for more details on the type of boot memories available.

4.4 Supplied Files for Software Development and System Verification

Host Side Files

The files that run on the host side are supplied in the following paths:

File Path	Comments
\sidev\binaries\32bit\apps\	All host binaries for FPGA and COFF loading
\sidev\siddk*	Source code for all host applications
\sidev\siddk\apps\	Host application project files

DSP Side Files

The utility COFF file that runs on the DSP side are supplied in the following paths:

File Path	Comments
\sidev\binaries\32bit\apps\c6711plxini.out	DSP utility COFF file
\sidev\sidsp\basiccom\plxc6711\rev1	DSP utility COFF file C source code and project files.

5.0 Hardware Description

The SI-C6xDSP board consists of a TMS320C671x DSP, a system clocking circuit, BootMEM (Asynchronous SRAM or optional Flash), Xilinx FPGA for glue logic and commonly shared resources between the host and the DSP (FIFOs, memory, control and status registers), an expansion card interface, and host communications via the PCI bus using PLX's PCI9054.

5.1 System Clocking

The clock for the system is generated from a circuit composed of an external 15Mhz crystal, an EEPROM based PLL device, and a clock buffer distribution device.

The external crystal is used as the clock source to the PLL, which in turn is used to generate the various required base frequencies used throughout the board. A clock distribution device is used as a buffer and stabilizer in order to minimize loading on those frequencies that are used by multiple devices.

Two different values are generated by the PLL: 1) 37.5Mhz fed into the C6x's internal PLL for generating its own 225Mhz-300Mhz core CPU clock, the PLX bridge device, the Xilinx FPGA, as well as the expansion card interface, 2) 75Mhz fed into the C6x's external memory interface or EMIF bus, the SDRAM, as well as the Xilinx or Altera FPGA.

5.2 DSP Memory Interface - EMIF Port

The C6x's EMIF bus runs Synchronously with the SDRAM module, and Asynchronously with all other peripherals including the FPGA ((FIFOs, CommRegs, control and status registers), the bootMEM, the PCI9054's Opregs and host resident memory (only accessible during Add-on Initiated Bus Master transfers), and the expansion daughter card interface. The DSP's EMIF port is subdivided into four (4) pages, with a range of up to 128MB for each. Please refer to the DSP's mapping section in this document.

A summary of each DSP EMIF page is mapped out as follows:

EMIF Region	Operation	Resources
PAGE0:CE0	Asynchronous	FPGA (FIFO, CSRs, CommRegs), PLX Opregs, Expansion Module
PAGE1:CE1	Asynchronous	BootMEM (512Kx8 SRAM or 1Mx16 NOR Flash)
PAGE2:CE2	Synchronous	Up to 128MB SDRAM
PAGE3:CE3	Synchronous	Up to 128MB SDRAM

Synchronous EMIF Bus Operation. The many timing parameters for synchronous EMIF bus operation reside inside of the SDRAM module's serial EEPROM, which varies between different SDRAM module manufacturers. Conventional 144 pin sockets are used to accommodate standard, 3.3V non-buffered PC133 SO-DIMM modules found on laptops. Module sizes of 64MB, 128MB, 256MB, and 512MB are supported, making it a very cost effective solution for the most demanding and memory intensive applications. Two of the four DSP EMIF pages (CE2 and CE3 lines) are reserved for SDRAM memory access. Please refer to the DSP's mapping section in this document.

NOTE:
1) Being only 32 bits wide, the DSP's EMIF bus maps one half (single rank modules) or one quarter (dual rank modules) of the SDRAM module.
2) Only SO-DIMM modules whose SDRAM ICs' parameters fall within the C67x's supported range will be fully operational. The C67x's internal SDRAM controller supports the following parameters:

i) Columns: 8, 9, 10.

ii) Rows: 11, 12, 13.

iii) Memory Device Banks: 2, 4. Not to be confused with the SODIMM's "rank", which is either 1 or 2.

iv) Cache Latency: 2, 3.

Asynchronous EMIF Bus Operation. The timing for asynchronous EMIF bus operation is fixed and unique for each of the peripherals. The bootMEM is defaulted to the maximum value of wait states at power up, the FPGA registers (including the FIFO and CommRegs) operate at two wait states, the PXI9054 Opregs and host expansion memory (accessible during Add-on initiated Bus Master accesses only) operate at a minimum of four (4) wait states and more if the PCI9054 is busy, while the expansion card operates at four (4) wait states.

5.3 DSP and PCI Interface

The hardware interface between the host PCI bus and the DSP is implemented with circuitry that combines PLX's PCI9054 device and an FPGA for control logic, and shared resources (FIFOs, memory, control and status registers).

The PLX's PCI9054 IC is set to operate in "J" mode, with the address and data buses multiplexed on the add-on side. This allows resources to be saved on the FPGA, and also completely isolates the DSP's EMIF bus from the PCI9054. When power is first applied, the PCI bridge device's own serial EEPROM device, which is easily identified as the socketed 8 pin DIP labeled as an 93LCxxB, supplies it with default values for initialization which are critical proper PCI bus configuration. Please refer to PLX's PCI9054 documentation for more details.

The FPGA acts as a bridge between the DSP and the PCI9054; it contains the communication registers (CommRegs), control/status registers (CSRs), and a pair of FIFO buffers (one for each direction). The FPGA also arbitrates the appropriate timing between the C6x's EMIF bus, the boot SRAM/Flash, the expansion card interface, and the PCI9054's local bus. By default, the FPGA contents are downloaded via the host PC using an FPGALoader utility. An FPGA SPROM device may be optionally installed.

A combination of hardware and software handshaking takes place in order to support a myriad of data transfer schemes, where the host side can be selected to operate in one mode and the DSP in another.

From the host PCI side, three (3) modes are available:
1) Target/Slave Mode. The host performs accesses over the PCI bus as a master, with the card acting as a target or slave device. The host performs transfers using programmed I/O accesses.
2) Bus Master Mode. The host temporarily relinquishes control of the PCI bus over to the

PCI9054's internal DMA engine, where data accessed is mapped directly into application space using a scatter gather technique.
3) Add-on Initiated Bus Master Mode. The host temporarily relinquishes control of the PCI bus over to the DSP, which resides on the PCI9054's Local or Add-on side. The DSP accesses a separate, contiguous block of host memory as its own peripheral.

From the DSP side, two (2) modes are available:
1) Programmed I/O. The DSP controls the transfers using programmed I/O.
2) DMA. The DSP's internal DMA engine performs the transfers by breaking the entire block into subsections or frames, with transfers within a frame being bursted by a predetermined count. Each frame is triggered by an external DSP interrupt routed to the DMA engine to signal a new frame transfer, thereby allowing the DSP's processor core access over the EMIF bus so as to avoid instruction drops in the case that the overall block size is very large.

Host target/slave mode accesses are performed when the DSP is either disabled/passive (Reset asserted) or enabled/active (Reset deasserted). While the DSP is disabled, the host uses target/slave mode transfers to load an initialization COFF file to the card's boot SRAM/Flash memory, as well as to access expansion daughter modules. After DSP activation (Reset deasserted), any combination of active data transfer modes can be used on either side.

For most applications, the most efficient method to implement large data block transfers is to use PCI bus mastering, as it requires minimal host intervention. The first bus master method involves using the PCI9054's DMA engine in block mode, in conjunction with the DSP using its own DMA engine. The second bus master transfer method allows the DSP to act as the PCI bus master, where it actually has direct access to the host computer's main memory!

NOTE*:*
1) The C6x's internal memory, registers, and SDRAM cannot be accessed directly by the PCI9054. Please refer to the C6x and PCI9054 data sheets and the SICommModes documentation for details on how to access the internal memory, registers, and SDRAM.
2) The boot SRAM memory can only be accessed by the host while the DSP is disabled (Reset asserted).
3) The PCI9054 bridge device and the FPGA each have their own configuration memory. By default, only the PCI9054's serial SPROM is placed, while the FPGA's SPROM is absent.

6.0 Communication Modes

The SI-C6xDSP card supports multiple, active, bi-directional communication modes between its DSP/PCI9054 and the host computer, which can be separated into the following:

1) Messaging. Either the DSP or the host computer may initiate an action by alerting the other processor of any task to be performed, especially useful for non deterministic or asynchronous events such as conditional transfers or alarm conditions.
2) Data Transfers. Transferring data may be considered a deterministic event since the action is always initiated by the host computer. From the perspective of the host computer, the SI-C6xDSP is either a slave or the bus master. When the SI-C6xDSP card is the bus master, the PCI9054 transfers data using its own DMA engine in conjunction with the DSP which may be invoked to use either programmed IO or its own DMA engine.

NOTE*:*
1) All communication modes are active, which require the DSP to be actively running code in order to interpret the command phase and subsequently allow for the transaction to take place. Passive communication modes are not supported for the C6x based DSP cards.
2) Please refer to the SICommModes.rtf document for more details.

6.1 Messaging

In messaging, there is always an initiator and a recipient. The following sequence of events are necessary for a message to be exchanged:

1) ***The initiator writes command and message parameters to the recipient.*** The command and message parameters are passed to a common, shared memory region. This region is grouped as a set of communication registers defined in the next section.
2) ***The initiator sends an interrupt to alert the recipient that a message exchange is to take place*****.** Once the initiator has finished passing down the command and message parameters, it will poll the recipient through a flag in the communication registers and check the status of the recipient. Polling on the part of the initiator avoids it from sending another message before the current action has been completed.
3) ***The recipient performs the user defined action contained in the message.***
4) ***The initiator and recipient synchronize when the message exchange has completed.*** The recipient will flag the polling initiator when it has completed the action. Please consult manuals and source code of a specific board for further details.

6.2 Data Transfers

The following sequence of events is necessary for a data transfer to complete:

All active communication transfers between the host and the DSP consist of two phases:
1) The Command Phase. Details of the specific type of transfer to take place are loaded to the Communication Registers (CommRegs), which reside inside of the FPGA.
2) The Data Transfer Phase. The data in question is actually transferred, using either the CommRegs or a FIFO as a temporary depository.

To clarify the nomenclature, all read and write accesses are referred from the host/PCI bus' point of view. Data writes occur when the data is transferred from the host to the DSP external memory space, while data reads occur when the data is transferred from the DSP memory space to the host.

The effective active communication modes for the latest *REV1* of the SI-C6xDSP's support software are summarized below (please check the older REV0 modes further down):

1) [AI-BM][IO][Async, BB, Int.]

- Accesses to the host are performed using the DSP as the bus master, with the DSP performing accesses using programmed I/O.
- The CommRegs are only used for the command phase, while the data is directly transferred to host memory mapped in the DSP's memory space.
- Synchronization between the DSP and the host memory is implemented with glue logic which effectively performs hardware wait states when one or the other is not accessible. The DSP's READY line is deasserted thus holding the EMIF port, and no external DSP interrupts are used.
- The DSP alerts the host that a transaction is complete by sending it an interrupt through one of the PLX's doorbell registers.

2) [AI-BM][DMA][Async, BB, Int.]

- Accesses to the host are performed using the DSP as the bus master, with the DSP performing accesses using its DMA engine in Asynchronous mode.
- The CommRegs are only used for the command phase, while the data is directly transferred to host memory mapped in the DSP's memory space.
- Synchronization between the DSP and the host memory is implemented with glue logic which effectively performs hardware wait states when one or the other is not accessible. The DSP's READY line is deasserted thus holding the EMIF port, and no external DSP interrupts are used.
- The DSP alerts the host that a transaction is complete by sending it an interrupt through one of the PLX's doorbell registers.

3) [Target][IO][BP-FIFO].

- Both the host and the DSP perform accesses using programmed I/O
- The CommRegs only used for the command phase, while the FIFO is used for the data transfer phase to allow for continuous data flow, thus avoiding slow polling of the Flag register.
- Synchronization between the DSP and the FIFO is implemented with glue logic which effectively performs hardware wait states when one or the other is not accessible. The DSP's READY line is deasserted thus holding the EMIF port, and no external DSP interrupts are used.

4) [BM][DMA][Async, BP-FIFO]

- Accesses to the host are performed using the PCI9054's DMA engine as the bus master, while the DSP performs accesses using its DMA engine in Asynchronous Burst mode.
- The CommRegs are only used for the command phase, while the FIFO is used for the data transfer phase.

• Synchronization between the DSP and the FIFO is implemented with glue logic which effectively performs hardware wait states when one or the other is not accessible. The DSP's READY line is deasserted thus holding the EMIF port, and no external DSP interrupts are used.

5) [Target][IO][Sync-Flg, BP-DReg].

• Both the host and the DSP perform accesses using programmed I/O.

• The CommRegs are used for both the command and data transfer phases. This mode is also referred to as "hostpolling" because the CommRegs serve as a temporary depository for each data point transferred

• Synchronization between the host and the DSP is performed with both polling the Flag register.

NOTE*: The following two communication modes for directly accessing SDRAM without DSP intervention is a feature that only exists on Rev4 boards populated with a larger XC2S100E FPGA.*

6) [Target][DAM SDRAM].

• The host performs direct accesses to the SDRAM using programmed I/O, where the onboard logic mimics the functionality of the DSP's HPI port, thereby not requiring any communications overhead by the DSP. The DSP must be actively running code because its internal SDRAM controller is required for SDRAM refreshes.

• This is a special transfer mode, where a pair of special CSRs described below is used by the host to access the SDRAM.

• Synchronization between the DSP and the FIFO is implemented with glue logic which effectively places the DSP's EMIF bus on hold until the transfer is complete. Note that the DSP's EHOLD and EHOLDA lines are used instead of its READY line, and no external DSP interrupts are used.

7) [BM][DAM SDRAM].

• The host performs direct accesses to the SDRAM using the PCI9054's DMA engine as the bus master, where the onboard logic mimics the functionality of the DSP's HPI port, thereby not requiring any communications overhead by the DSP. The DSP must be actively running code because its internal SDRAM controller is required for SDRAM refreshes.

• This is a special transfer mode, where a pair of special CSRs described below is used by the host to access the SDRAM.

• Synchronization between the DSP and the FIFO is implemented with glue logic which effectively places the DSP's EMIF bus on hold until the transfer is complete. Note that the DSP's EHOLD and EHOLDA lines are used instead of its READY line, and no external DSP interrupts are used.

NOTE*: Please refer to the SICommModes.rtf document for more details.*

The effective <u>active</u> communication modes for the older *REV0* of the SI-C6xDSP's support software are summarized below:

1) [AI-BM][IO][Async, BB, Int.]

• Accesses to the host are performed using the DSP as the bus master, with the DSP performing accesses using programmed I/O.
• The CommRegs are only used for the command phase, while the data is directly transferred to host memory mapped in the DSP's memory space.
• Synchronization between the DSP and the host memory is implemented with glue logic which effectively performs hardware wait states when one or the other is not accessible. The DSP's READY line is deasserted thus holding the EMIF port, and no external DSP interrupts are used.
• The DSP alerts the host that a transaction is complete by sending it an interrupt through one of the PLX's doorbell registers.

2) [AI-BM][DMA][Async, BB, Int.]

• Accesses to the host are performed using the DSP as the bus master, with the DSP performing accesses using its DMA engine in Asynchronous mode.
• The CommRegs are only used for the command phase, while the data is directly transferred to host memory mapped in the DSP's memory space.
• Synchronization between the DSP and the host memory is implemented with glue logic which effectively performs hardware wait states when one or the other is not accessible. The DSP's READY line is deasserted thus holding the EMIF port, and no external DSP interrupts are used.
• The DSP alerts the host that a transaction is complete by sending it an interrupt through one of the PLX's doorbell registers.

3) [Target][IO][Async, BP-FIFO].

• Both the host and the DSP perform accesses using programmed I/O
• The CommRegs only used for the command phase, while the FIFO is used for the data transfer phase to allow for continuous data flow, thus avoiding slow polling of the Flag register.
• Synchronization between the DSP and the FIFO is implemented with glue logic which effectively performs hardware wait states when one or the other is not accessible. The DSP's READY line is deasserted thus holding the EMIF port, and no external DSP interrupts are used.

4) [Target][DMA][Async, BP-FIFO]

• The host performs accesses using programmed I/O, while the DSP performs accesses using its DMA engine in Asynchronous mode.
• The CommRegs are only used for the command phase, while the FIFO is used for the data transfer phase.
• Synchronization between the DSP and the FIFO is implemented with glue logic which effectively performs hardware wait states when one or the other is not accessible. The DSP's READY line is deasserted thus holding the EMIF port, and no external DSP interrupts are used.

5) [BM][IO][Async, BP-FIFO]

• Accesses to the host are performed using the PCI9054's DMA engine as the bus master, while the DSP performs accesses using programmed I/O.
• The CommRegs are only used for the command phase, while the FIFO is used for the data

transfer phase.

• Synchronization between the DSP and the FIFO is implemented with glue logic which effectively performs hardware wait states when one or the other is not accessible. The DSP's READY line is deasserted thus holding the EMIF port, and no external DSP interrupts are used.

6) [BM][DMA][Async, BP-FIFO]

• Accesses to the host are performed using the PCI9054's DMA engine as the bus master, while the DSP performs accesses using its DMA engine in Asynchronous mode.

• The CommRegs are only used for the command phase, while the FIFO is used for the data transfer phase.

• Synchronization between the DSP and the FIFO is implemented with glue logic which effectively performs hardware wait states when one or the other is not accessible. The DSP's READY line is deasserted thus holding the EMIF port, and no external DSP interrupts are used.

7) [Target][IO][Sync-Flg, BP-DReg].

• Both the host and the DSP perform accesses using programmed I/O.

• The CommRegs are used for both the command and data transfer phases. This mode is also referred to as "hostpolling" because the CommRegs serve as a temporary depository for each data point transferred

• Synchronization between the host and the DSP is performed with both polling the Flag register.

8) [Target][DMA][Sync, BP-FIFO]

• The host performs accesses using programmed I/O, while the DSP performs accesses using its DMA engine in Synchronous mode.

• The CommRegs are only used for the command phase, while the FIFO is used for the data transfer phase.

• Synchronization between the DSP and the FIFO is implemented with interrupts which alert the DSP's DMA engine when the FIFO is accessible, no wait states are necessary. In effect, the DSP's READY line is always asserted without ever holding the EMIF port, and external DSP interrupts are used. More details about the DSP's external interrupt configuration is covered in sections below.

9) [BM][IO][Sync, BP-DReg]

• Accesses to the host are performed using the PCI9054's DMA engine as the bus master, while the DSP performs accesses using programmed I/O.

• The CommRegs are used for both the command and data transfer phases.

• Synchronization between the host and the DSP is performed with both polling the Flag register.

10) [BM][DMA][Sync, BP-FIFO]

• Accesses to the host are performed using the PCI9054's DMA engine as the bus master, while the DSP performs accesses using its DMA engine in Synchronous mode.

• The CommRegs are only used for the command phase, while the FIFO is used for the data transfer phase.

• Synchronization between the DSP and the FIFO is implemented with interrupts which alert the DSP's DMA engine when the FIFO is accessible, no wait states are necessary. In effect, the DSP's READY line is always asserted without ever holding the EMIF port, and external DSP interrupts are used. More details about the DSP's external interrupt configuration is covered in sections below.

7.0 Host Address Mapping Via The PCI9054.

This section describes the common resources used by the host and the DSP's. The DSP's space is directly mapped into the PCI9054's BADDR2 region (Space 0), with a total depth of 8M bytes, or 2Mx32 words:

Range (Bytes)	Resource		Width (Bits)	Host BADDR2+Offset (Byte Boundary)	DSP CEn (External Space)	DSP Base Address (Byte Boundary)
2M	FPGA: 2x 32 Deep FIFOs		32	0x00,0000-0x0F,FFFF	0	0x8020,0000-0x802F,FFFF
	FPGA: x64 Comm Regs			0x10,0000-0x13,FFFF		0x8030,0000-0x8033,FFFF
	FPGA: x12 CSRs			CSR11: 0x10,0100-0x10,40FF; CSR[0:10]: 0x14,0000-0x1B,FFFF		0x8030,0100-0x8030,40FF; 0x803C,0000-0x803C,FFFF
	Expansion Module (64Kx32)			0x1C,0000-0x1F,FFFF		0x8038,0000-0x803B,FFFF
6M	BootMEM	SRAM (512Kx8)	8	0x20,0000-0x7F,FFFF	1	0x9000,0000-0x9FFF,FFFF
		NOR Flash (1Mx16)	16			

7.1 PCI9054 Opreg Mapping

The PLX Operation Registers are accessed by both the host and the DSP, and are mirrored across the entire 256KB (64KDWord) address range of 0x8034,0000 - 0x8037,FFFF on the DSP. Please refer to the PLX documentation for more details.

PCI BADDRn	Range (Bytes)	Resource		Host BADDRn+Offset (Byte Boundary)	DSP CEn	DSP Base Address (Byte Boundary)
0	256	PLX OpRegs	Memory Mapped	0x00-0xFF	0	0x8034,0000-0x8034,00FF (mirrored across 0x8034,0000-0x8037,FFFF)
1			I/O Mapped (unused)			

7.2 FIFO (FPGA) Address Mapping

There are two 32 deep FIFOs, one for each direction that are accessible by both the host and the DSP. The FIFO is mirrored across the entire 1MB (256KDWord) address range of 0x0-0x0F,FFFF on the host, and 0x8020,0000-0x802F,FFFF on the DSP.

Resource	Host BADDR2+Offset (Byte Boundary)	DSP Base Address (Byte Boundary)
FPGA: 2x 32 Deep FIFO	0x00,0000-0x0F,FFFF	0x8020,0000 (mirrored across 0x8020,0000-0x802F,FFFF)

7.3 Communication Registers (FPGA) Address Mapping

The CommRegs[15:0] are accessed by both the host and the DSP and used for general communications, while CommRegs[63:16] are Scratch Pad CommRegs intended for debugging and by default only accessible by the Host, but may be accessible by the DSP after a specific Key value is written in to the CommReg Key register. The 64 CommRegs are mirrored across the entire 256KB (64KDWord) address range of 0x10,0000 - 0x13,FFFF on the host, and 0x8030,0000-0x8033,FFFF on the DSP.

Register	Name	Host BADDR2+Offset (Byte Boundary)		DSP Address (Byte Boundary)	
CommReg0	CommMode/Control	0x10,0000	0x10,0000-0x10,007F (mirrored across 0x10,0000-0x13,FFFF)	0x8030,0000	0x8030,0000-0x8030,00FF (mirrored across 0x8030,0000-0x8033,FFFF)
CommReg1	Count (DWords)	0x10,0004		0x8030,0004	
CommReg2	DSP Source Address	0x10,0008		0x8030,0008	
CommReg3	DSP Destination Address	0x10,000C		0x8030,000C	
CommReg4	Comm/Status Flag	0x10,0010		0x8030,0010	
CommReg5	Data	0x10,0014		0x8030,0014	
CommReg6	Heartbeat for DMA, host comm, etc. (Optional)	0x10,0018		0x8030,0018	
CommReg7	Main Heartbeat	0x10,001C		0x8030,001C	
CommReg8	Host->DSP Flag/Status (Host->DSP Interrupt)	0x10,0020		0x8030,0020	
CommReg9	Host->DSP Message (user defined)	0x10,0024		0x8030,0024	
CommReg10	DSP->Host Flag/Status (DSP->Host Interrupt via PLX Doorbell)	0x10,0028		0x8030,0028	
CommReg11	DSP->Host Message (user defined)	0x10,002C		0x8030,002C	
CommReg12	Secondary Heartbeat (Optional)	0x10,0030		0x8030,0030	
CommReg[13:15]	user defined	0x10,0034-0x10,003C		0x8030,0034-0x8030,003C	
CommReg[16:63]	user defined/Scratch Pad Region (DSP can only access with by writing a fixed Key value)	0x10,0040-0x10,00FF		0x8030,0040-0x8030,00FF	

CommReg0: *Communication Mode/Control Register.*
Written by the host processor, defines the type of transfer to take place.

CommReg1*: Count Register in DWords.*
Written by the host processor, defines the size or number of 32 bit data values to transfer.

CommReg2*: Source Address Register.*
Written by the host processor, defines the source address within the DSP's memory range, of the data to be transferred.

CommReg3*: Destination Address Register.*
Written by the host processor, defines the destination address within the DSP's memory range, of the data to be transferred.

CommReg4*: Data Transfer Flag/Status Register.*
Accessed by both the host and the DSP, serves as a status indicator for synchronizing both the host and DSP processors. Typically, the host must clear or write a "0" value before starting the data phase.

CommReg5*: Data Register.*
Optional register used only for "Host Polling" transfer modes. Accessed by both the host and the DSP, and serves as an intermediate depository for the current data value being transferred.

NOTE*: Only valid for those transfers requiring both the host and the DSP to poll one another. Otherwise, the data is transferred through a FIFO buffer that is present between the DSP and the PCI bridge device.*

CommReg6*: First Heartbeat Register.*
Updated exclusively by the DSP during interrupts, useful as a debugging tool and simply serves as an indicator to the host that the DSP is actively running code inside of service routines.

CommReg7*: Second Heartbeat Register.*
Updated exclusively by the DSP, useful as a debugging tool and simply serves as an indicator to the host that the DSP is actively running code.

CommReg8*: Host->DSP Message Flag/Status Register.*
Accessed by both the host and the DSP, serves as a status indicator for synchronizing both processors when the Host 'initiator' sends a message to the DSP 'recipient'. The Host will write a '0x1' value to indicate that a message has been sent, and will expect the DSP 'recipient' to clear or write a '0x0' value to indicate that it has completed the message processing.

NOTE: *The Host 'initiator' also sends an interrupt to the DSP 'recipient' in order to alert it of the pending message.*

CommReg9*: Host->DSP Message Register.*
A user defined value written by the Host when the Host 'initiator' is to pass a user defined message to the DSP 'recipient'.

CommReg10*: DSP->Host Message Flag/Status Register.*
Accessed by both the host and the DSP, serves as a status indicator for synchronizing both processors when the DSP 'initiator' sends a message to the Host 'recipient'. The DSP will write a '0x1' value to indicate that a message has been sent, and will expect the Host 'recipient' to clear or write a '0x0' value to indicate that it has completed the message processing.

NOTE: *The DSP 'initiator' also simultaneously updates the PLX's Doorbell register defined below so as to cause an interrupt to the Host 'recipient'. The driver automatically and immediately clears the Doorbell register after it detects the interrupt.*

CommReg11*: DSP->Host Message Register.*
A user defined value written by the DSP when the DSP 'initiator' is to pass a user defined message to the Host 'recipient'.

CommReg12*: Third Heartbeat Register.*
Updated exclusively by the DSP, useful as a debugging tool and simply serves as an indicator to the host that the DSP is actively running code during a dual phase COFF load operation.
NOTE: *Used during dual phase COFF load operations when the COFF file is too large to fit into the bootMEM.*

***CommReg[13:15]**: User Defined Registers.*
These are the 3 unlocked dual port registers, which are not reserved. They are open for general usage as Host<->DSP mailbox registers, completely user definable.

***CommReg[16:63]**: Locked User Defined Registers – Scratch Pad Registers.*
These are the 32 remaining dual port registers, which are not reserved and by default only accessible by the host where the DSP is locked out. They are used as a scratch pad and intended for DSP code debugging but open for general usage as Host<->DSP mailbox registers, completely user definable. However, to enable DSP accesses to the scratch pad, a "key" value may be written into the 'CommReg Key' CSR by either host or DSP in order enable DSP accesses. Likewise, any other value written into the CommReg Key will lock out DSP accesses.

***NOTE**:*
1)The PLX's Doorbell register is used as an additional communication register as it facilitates synchronizing DSP->Host initiated events such as 'Add-on Initiated' data transfers and DSP->Host messages. Please refer to the SICommModes.rtf document for more details.
2) Only the least significant address bits (from the DSP's EMIF bus) to the CommRegs are decoded, and hence the mirroring.

Interrupts are used in conjunction with the Communication Registers, where an 'initiator' interrupts a 'recipient' in order to alert it of a pending transaction.

Register	Name	Host BADDR+Offset (Byte Boundary)	DSP Address (Byte Boundary)	Interrupt Direction
CSR1	General Control - GC	BADDR2+0x18,0000	n/a	Host->DSP
PLX Doorbell	L2PDBELL	BADDR0+0x64 (PLX OpReg)	0x8034,00E4	DSP->Host

***Control/Status Register 1 (CSR1)**: Host->DSP Interrupt.*
When the host performs a target write to this location, the onboard control logic generates and interrupt to the DSP, which alerts the DSP that either a data transfer or a message is to be serviced.

***PLX Doorbell**: L2PDBELL for DSP->Host Interrupt.*
If the DSP needs to alert the host of a pending transaction, it must alert it by causing a PCI interrupt by accessing the PLX's Local-to-PCI L2PDBELL doorbell register. The value written by the DSP will vary depending if the DSP has completed a data transfer (Add-on Init mode) or if a new message to the Host is pending.

7.4 Control/Status Registers (FPGA) Address Mapping

All Control/Status Registers are accessible by the host, and are used to control overall board functionality as well as interrupt control to the DSP. Only CSR0, CSR3, CSR8, CSR9 and CSR10 are accessible by the DSP.

On the host, the CSRs are mirrored across the entire 512KB (128KDWord) address range of 0x14,000-0x1B,FFFF while on the DSP they are mirrored across the entire 64KB (16KDWords) address range of 0x803C,0000-0x803C,FFFF.

Register	Name	Host BADDR2+Offset (Byte Boundary)	DSP Address (Byte Boundary)	Accessibility
CSR0	General Control - GC	0x14,0000 (mirrored across 0x14,0000-0x14,3FFFF)	0x803C,0000 (mirrored across 0x803C,0000-0x803C,3FFF)	Host & DSP
CSR1	Interrupt Routing Mask - IRM	0x16,0000 (mirrored across 0x16,0000-0x16,3FFFF)	n/a	Host only
CSR2	PCI Timeout – PCITO (Works in Conjunction with CSR10)	0x17,0000 (mirrored across 0x17,0000-0x17,3FFFF)	n/a	Host only
CSR3	FPGA Revision/Date	0x17,8000 (mirrored across 0x17,8000-0x17,BFFFF)	0x803C,4000 (mirrored across 0x803C,4000-0x803C,7FFF)	Host & DSP
CSR4	DSP DMA Burst Transfer Counter	0x17,C000 (mirrored across 0x17,C000-0x17,CFFFF)	n/a	Host only
CSR5	DAM SDRAM Address (100e only)	0x17,D000 (mirrored across 0x17,D000-0x17,DFFFF)	n/a	Host only
CSR6	DAM SDRAM Count (100e only)	0x17,E000 (mirrored across 0x17,E000-0x17,EFFFF)	n/a	Host only
CSR7	Host Generated Interrupt Control - HGIC	0x18,0000 (mirrored across 0x18,0000-0x1B,FFFF)	n/a	Host only
CSR8	SDRAM Control - SDCTL	n/a	0x803C,8000 (mirrored across 0x803C,8000-0x803C,BFFF)	DSP only
CSR9	DAM SDRAM Extensions - SDEXT	n/a	0x803C,C000 (mirrored across 0x803C,C000-0x803C,FFFF)	DSP only
CSR10	PCI Timeout Status – PCITOSTAT (Works in Conjunction with CSR2) (Later FPGA release only)	0x14,8000 (mirrored across 0x14,8000-0x14,8FFFF)	n/a	Host only
CSR11	CommRegs Key (DSP Lock) – KEY (Later FPGA release only)	0x10,0100 (immediately after CommReg63)	0x803C,0100	Host & DSP

***CSR0**: General Control - GC Register.*

General purpose board level control register, where the host has full access to the entire register, while the DSP can only read the upper three bytes.

***NOTE:** DSP writes to CSR0 is only decoded for resetting the FPGA clocks and does NOT alter the contents of CSR0.*

Register		**Description**	**Host Mapping**	**DSP Mapping**
CSR0-General Control (GC)		General purpose board level control register, where the host has full access to the entire register, while the DSP can only read the upper three bytes	0x14,0000 (mirrored across 0x14,0000-0x14,3FFFF)	0x803C,0000 (mirrored across 0x803C,0000-0x803C,3FFF)
Bit Position	**Name**	**Description**	**Initial Value**	**Accessibility**
0	DSP Reset	Controls the active state of the DSP, typically applicable during the COFF file loading stage. 0 = DSP is disabled. 1 = DSP is enabled to actively run code.	0	Host R/W only
1	SPD_SDA_CTRL	SPD Data Line Control or Drive. Activates the data line on the SDRAM module's I2C configuration EEPROM or SPD. 0 = Disable and Tristate the SPD data line. 1 = Activate the SPD data line.	0	Host R/W only
2	SPD_SDA	SPD I2C data. Physically connected to SODIMM pin 141.	HiZ	Host R/W only
3	SPD_SCL	SPD I2C clock pin. Physically connected to SODIMM pin 142.	0	Host R/W only
4	DSP_WR_DMAINT	DSP-write-to-FIFO interrupt generation enable. Controls the generation of external interrupts to the DSP's DMA engine. External interrupts are necessary when the DSP's DMA engine is configured to operate in Synchronous mode, where a single interrupt is generated for every FIFO access, thus allowing the DSP to operate at full speed without holding the EMIF port through its READY line. 0 = Disable DSP DMA Interrupts. Do not generate interrupts to the DSP when the FIFO is accessible for DSP writes. 1 = Enable DSP DMA Interrupt. Automatically generate interrupts to the DSP when the FIFO is accessible for DSP writes during DMA transfers.	0	Host R/W DSP Read Only
5	DSP_RD_DMAINT	DSP-read-from-FIFO interrupt generation enable. Controls the generation of external interrupts to the DSP's DMA engine. External interrupts are necessary when the DSP's DMA engine is configured to operate in Synchronous mode, where a single interrupt is generated for every FIFO access, thus allowing the DSP to operate at full speed without holding the EMIF port through its READY line. 0 = Disable DSP DMA Interrupts. Do not generate interrupts to the DSP when the FIFO is accessible for DSP reads. 1 = Enable DSP DMA Interrupt. Automatically generate interrupts to the DSP when the FIFO is accessible for DSP reads during DMA transfers.	0	Host R/W DSP Read Only
7:6	Reserved	Unused, R/W capable.	0	Host R/W only
8	BOOTMEM_SEL	Boot Device Selector. Hardwired pin used to differentiate a byte wide SRAM or a word wide Flash boot memory. 0 = 512kx8 SRAM (default) 1 = 1Mx16 Flash	0	Host R/W DSP Read Only

9	FLASH_CTRL	Flash 'OEn' Signal Control. Controls the Flash's 'OEn' or low asserted 'Output Enable' input signal. 0 = 'OEn' is '!AWEn' (inverse of Flash's low asserted Write Enable line). 1 = 'OEn' is set to a HI or 'b1' value in order to allow a Flash command to occur.	0	Host R/W DSP Read Only
11:10	Reserved	Unused, R/W capable.	0	Host R/W DSP Read Only
13:12	DSP_ERRORSTAT	Debugging - DSP Lock Status or Error Bits. Read Only bits that indicate the status of the DSP's EMIF clock or if DSP is erroneously accessing the PCI Bus. 00 = No error condition. x1 = Error condition: Loss of DSP EMIF Clock. 1x = Error condition: Errant DSP Access Across PCI Bus. Error Condition: The FPGA will put the DSP in reset, reset clocks and relock the system PLLs; must be cleared by host writing '0b00'.	0	Host R/W DSP Read Only
23:14	Reserved	Unused, R/W capable.	0	Host R/W DSP Read Only
25:24	DSP-PCI_FIFOSTAT	DSP->PCI FIFO Status Lines. Read Only bits that indicate the occupancy of the DSP->PCI FIFO in quarter increments. 00 = FIFO is between empty (can be empty or full) and 1/4 full. 01 = FIFO is between 1/4 full and 1/2 full. 10 = FIFO is between 1/2 full and 3/4 full. 11 = FIFO is between 3/4 full and full (is not full).	0	Host Read Only DSP Read Only
26	DSP-PCI_FIFOEMPTY	DSP->PCI FIFO Empty Status Line. Read only bit that indicates if the DSP->PCI FIFO is empty or not. 0 = FIFO is not empty. 1 = FIFO is empty.	0	Host Read Only DSP Read Only
27	DSP-PCI_FIFOFULL	DSP->PCI FIFO Full Status Line. Read only bit that indicates if the DSP->PCI FIFO is full or not. 0 = FIFO is not full. 1 = FIFO is full.	0	Host Read Only DSP Read Only
29:28	PCI-DSP_FIFOSTAT	PCI->DSP FIFO Status Lines. Read Only bits that indicate the occupancy of the PCI->DSP FIFO in quarter increments. 00 = FIFO is between empty (can be empty or full) and 1/4 full. 01 = FIFO is between 1/4 full and 1/2 full. 10 = FIFO is between 1/2 full and 3/4 full. 11 = FIFO is between 3/4 full and full (is not full).		Host Read Only DSP Read Only
30	PCI-DSP_FIFOEMPTY	PCI->DSP FIFO Empty Status Line. Read only bit that indicates if the PCI->DSP FIFO is empty or not. 0 = FIFO is not empty. 1 = FIFO is empty.	0	Host Read Only DSP Read Only
31	PCI-DSP_FIFOFULL	PCI->DSP FIFO Full Status Line. Read only bit that indicates if the PCI->DSP FIFO is full or not. 0 = FIFO is not full. 1 = FIFO is full.	0	Host Read Only DSP Read Only

***CSR1**: Interrupt Routing Mask Register or IRM Register.*

Board level control register only accessible by the host, used to route the source of the external DSP interrupts. Works in conjunction with CSR7, the Host Generated Interrupt Control register (HGIC) described below.

Register		**Description**	**Host Mapping**	**DSP Mapping**	**Default Initial Value**
CSR1 – Interrupt Routing Mask (IRM)		Route the source of the external DSP interrupts.	0x16,0000 (mirrored across 0x16,0000-0x16,3FFFF)	n/a	0x8844,0221
DSP External INT	**Bit Position**	**Interrupt Source**			**Default Initial Value**
4 (Highest Priority)	0	HostInt[0] (enabled but normally unused, unless for debugging)			1
	1	HostInt[1]			0
	2	HostInt[2]			0
	3	HostInt[3]			0
	4	Expansion Module[1]			0
	5	Expansion Module[0] (enabled and used by Expansion module)			1
	6	DSP DMA FIFO			0
	7	General Purpose			0
5	8	HostInt[0]			0
	9	HostInt[1] (enabled and used to alert DSP of a pending transaction)			1
	10	HostInt[2]			0
	11	HostInt[3]			0
	12	Expansion Module[1]			0
	13	Expansion Module[0]			0
	14	DSP DMA FIFO			0
	15	General Purpose			0
6	16	HostInt[0]			0
	17	HostInt[1]			0
	18	HostInt[2] (enabled but unused)			1
	19	HostInt[3]			0
	20	Expansion Module[1]			0
	21	Expansion Module[0]			0
	22	DSP DMA FIFO (enabled and used to trigger DSP DMA engine)			1
	23	General Purpose			0
7 (Lowest Priority)	24	HostInt[0]			0
	25	HostInt[1]			0
	26	HostInt[2]			0
	27	HostInt[3] (enabled but unused)			1
	28	Expansion Module[1]			0
	29	Expansion Module[0]			0
	30	DSP DMA FIFO			0
	31	General Purpose (enabled but unused)			1

All CSR1 mask bits are enabled with a HI or logic '1' assertion.
0 = Disable Interrupt Source.
1 = Enable Interrupt Source.

When one or more mask bits are enabled, the corresponding interrupt source or events will be "ORed" together in order to cause an interrupt to the DSP.

Several examples are listed below:

CSR1 = 0x08040201. Only the host can generate INT4 to INT7.

CSR1 = 0x00400220. This value is what is used by default since the host cannot inadvertently cause spurious interrupts.

EXTINT7 = Not used
EXTINT6 = DSP DMA FIFO Access interrupt
EXTINT5 = Host generated interrupt (HostInt[1])
EXTINT4 = DaughterInt[0] interrupt

CSR1 = 0x88440221. This value is what you would normally use during development since the host can emulate any interrupt.

EXTINT7 = General Purpose OR HostInt[3] (Normally unused)
EXTINT6 = DSP DMA FIFO Access interrupt OR HostInt[2]
EXTINT5 = Host generated interrupt (HostInt[1])
EXTINT4 = DaughterInt[0] interrupt OR HostInt[0]

***CSR2 & CSR10**: PCI Transfer Timeout and Timeout Status.*

CSR2 is the timer control register that sets a time limit for when a PCI access must occur. If an access does not occur within the timeout period, a null transfer is performed to allow the host driver to release the bus and avoid a host lockup. Both are accessible by the host only.

Register	**Description**	**Host Mapping**	**DSP Mapping**	**Default Initial Value**
CSR2 – PCI Timeout (PCITO)	Timer Period implemented as a 'divide by N' counter, with a 37.5Mhz reference clock. Clears the PLX Ready signal if no activity is sensed during a valid PCI transfer. The value is calculated as: Timout = (PCITO)/(37.5Mhz)	0x17,0000 (mirrored across 0x17,0000-0x17,3FFFF)	n/a	0x8000,0000
Bit Position	**Description**			
31:0	Timeout Period In Counts			

CSR10 simply sets bit 23 when a timeout limit set by CSR2 has occurred.

***NOTE**: CSR10 & CSR11 only implemented on FPGA bitfiles dated after 2013.*

Register	**Description**	**Host Mapping**	**DSP Mapping**	**Default Initial Value**
CSR10 – PCI Timeout Status (PCITOSTAT)	For debugging purposes, sets a bit when the PCI bus activity period has been exceeded by the value set in CSR2.	0x14,8000 (mirrored across 0x14,8000-0x14,8FFFF)	n/a	0x0000,0000
Bit Position	**Description**			
31:24	n/a			
23	PCITO Status. 0 = CSR2 limit NOT exceeded – valid PCI bus transfer. 1 = CSR2 limit exceeded – INvalid PCI bus transfer.			
22:0	n/a			

***CSR3**: FPGA Revision/Date Code.*

Date code, read only register that can be used for version control. Accessible by both the Host and the DSP.

Register	Description	Host Mapping	DSP Mapping
CSR3 – FPGA Date (DATE)	Hex encoded date code indicating the FPGA bitfile compilation date. The date value is formatted as: [2 digit year][2 digit month][2 digit day]	0x17,8000 (mirrored across 0x17,8000-0x17,BFFFF)	0x803C,4000 (mirrored across 0x803C,4000-0x803C,7FFF)
Bit Position	**Description**		
31:0	Date listed as YYMMDD.		

***CSR4**: DMA Burst Transfer Counter.*

DSP's DMA burst transfer count register, which sets a limit for the number of DMA burst transfers to be performed by the DSP's internal EDMA engine before an external interrupt is generated by the FPGA to the DSP. The external interrupt to the DSP is used to trigger each DMA burst transfer between the DSP and the FIFO (eventually to the host), where the default value is 16 or half of the FIFO total depth. Accessible by the host only.

Register	Description	Host Mapping	DSP Mapping	Default Initial Value
CSR4 – DSP DMA Burst Transfer Count (DMABURST)	Sets a limit for the number of DMA burst transfers to be performed by the DSP's internal EDMA engine before an external interrupt is generated by the FPGA to the DSP.	0x17,C000 (mirrored across 0x17,C000-0x17,CFFFF)	n/a	0x10
Bit Position	**Description**			
31:0	DSP DMA Burst Transfer count limit			

When the minimum value of '1' is entered, an interrupt will be generated to the DSP for every DMA transfer between it and the FIFO; despite being very slow, one interrupt per word is very secure and minimizes the bottlenecks imposed on the EMIF bus. On the other hand, the maximum value entered will not cause an interrupt to the DSP until the transfer of the entire block is completed thereby rendering the maximum transfer speed between the DSP and the host; however, there is a risk that the EMIF bus can be easily bottlenecked which can cause DSP execution stalls. As a result, the default value of '16' is the most efficient value which allows for decent data transfer speeds without excessive risk of EMIF bus bottlenecks, and hence it is strongly recommended to not be altered.

***CSR5**: Dual Access SDRAM Address Register.*

For some applications, especially those where a very small number of transfers are required, it is more efficient to perform a direct access to the external SDRAM without any communications overhead required by a normal active DSP communications transfer. When a direct access is performed however, the DSP's EMIF bus is held. Because this type of transfer is direct and occurs without DSP intervention, the host requires an SDRAM address register to reside inside of the FPGA. These types of transfers mimic the HPI port for direct external access to the DSP's SDRAM. Accessible by the host only.

***NOTE**: Dual Access SDRAM capability only exists on Rev4 boards populated with a larger XC2S100E FPGA.*

Register	**Description**	**Host Mapping**	**DSP Mapping**
CSR5 – DAM SDRAM Address (SDRAMADDR)	Dual Access SDRAM Address register, when host is to access the DSP's SDRAM without active communications. ***NOTE**: Dual Access SDRAM capability only exists on Rev4 boards populated with a larger XC2S100E FPGA.*	0x17,D000 (mirrored across 0x17,D000-0x17,DFFFF)	n/a
Bit Position	**Description**		
31:0	DSP SDRAM Address		

***CSR6**: Dual Access SDRAM Count Register.*

As mentioned above, some applications, especially those where a very small number of transfers are required, it is more efficient to perform a direct access to the external SDRAM without any communications overhead required by a normal active DSP communications transfer. Because this type of transfer is direct and occurs without DSP intervention, the host requires an SDRAM count register to reside inside of the FPGA. Accessible by the host only.

***NOTE**: Dual Access SDRAM capability only exists on Rev4 boards populated with a larger XC2S100E FPGA.*

Register	**Description**	**Host Mapping**	**DSP Mapping**
CSR6 – DAM SDRAM Count (SDRAMCNT)	Dual Access SDRAM Count register, when host is to access the DSP's SDRAM without active communications. **NOTE**: Dual Access SDRAM capability only exists on Rev4 boards populated with a larger XC2S100E FPGA.	0x17,E000 (mirrored across 0x17,E000-0x17,EFFFF)	n/a
Bit Position	**Description**		
19:0	Transfer count in DWords. A total of 1Mx32 values can be transfers in a single instance.		
20	SDRAM Read Start. 0 = Idle. 1 = Activate host read of SDRAM.		
21	SDRAM Write Start. 0 = Idle. 1 = Activate host write of SDRAM.		
31:22	n/a		

CSR7: *Host Generated Interrupt Control Register.*

Board level control register only accessible by the host, used to generate an interrupt to the DSP in order to start either a message or a data transfer. No values are latched, but rather a pulse is generated by the host access to the register. The specific interrupt routed to the DSP is defined by CSR1 described above.

Register	**Description**	**Host Mapping**	**DSP Mapping**
CSR7 – Host Generated Interrupt Control (HGIC)	Host writes to a bit defined by the IRM (CSR) register will generate an external interrupt to the DSP, and a write to bit 4 clears the FIFOs.	0x18,0000 (mirrored across 0x18,0000-0x1B,FFFF)	n/a
Bit Position	**Description**	**Direction**	
3:0	HostInt[n] Event, where n = 0 to 3, which corresponds to the bit position. If bit position 0 is written with a value of '1' by the host, then a HostInt[0] event will be generated to the DSP. 0 = No HostInt[n] event. 1 = HostInt[n] event routed to DSP, causing an external Interrupt as defined by the IRM CSR1 register.	Host Write only	
4	FIFO Reset 0 = FIFO active, pointer incremented. 1 = Activate host read of SDRAM.	Host Write only	
31:5	n/a	n/a	

Several examples for CSR1 values are listed below:

With *CSR1 = 0x08040201*, we have:
CSR7 = 0x00000001: HostInt[0] will cause a DSP EXTINT4
CSR7 = 0x00000002: HostInt[1] will cause a DSP EXTINT5
CSR7 = 0x00000004: HostInt[2] will cause a DSP EXTINT6
CSR7 = 0x00000008: HostInt[3] will cause a DSP EXTINT7
CSR7 = 0x0000000F: HostInt(3:0) will cause all DSP EXTINT[7:4]

With *CSR1 = 0x04020108*, we have:
CSR7 = 0x00000001: HostInt[0] will cause a DSP EXTINT5
CSR7 = 0x00000002: HostInt[1] will cause a DSP EXTINT6
CSR7 = 0x00000004: HostInt[2] will cause a DSP EXTINT7
CSR7 = 0x00000008: HostInt[3] will cause a DSP EXTINT4
CSR7 = 0x0000000F: HostInt(3:0) will cause all DSP EXTINT[7:4]

With *CSR1 = 0x04020101*, we have:
CSR7 = 0x00000001: HostInt[0] will cause a DSP EXTINT4 and EXTINT5
CSR7 = 0x00000002: HostInt[1] will cause a DSP EXTINT6
CSR7 = 0x00000004: HostInt[2] will cause a DSP EXTINT7
CSR7 = 0x00000008: No DSP interrupts.
CSR7 = 0x0000000F: HostInt(3:0) will cause all DSP EXTINT[7:4]

With *CSR1 = 0x00000000*, we have:
No interrupts are possible to the DSP.

CSR2 = 0x00000010 will reset the FIFO.

With *CSR1 = 0x88440221*, (the default value as indicated in CSR1), we have:

CSR7 = 0x00000001: HostInt[0] will cause a DSP EXTINT4 (shared with interrupt sourced from Expansion Module).

CSR7 = 0x00000002: HostInt[1] will cause a DSP EXTINT5 (used to alert DSP of a pending transaction).

CSR7 = 0x00000004: HostInt[2] will cause a DSP EXTINT6 (shared with interrupt generated by the FPGA's DMA transfer counter).

CSR7 = 0x00000008: HostInt[3] will cause a DSP EXTINT7 (normally unused).

CSR7 = 0x0000000F: HostInt(3:0) will cause all DSP EXTINT[7:4]

***NOTE**:*
1) Host Write to CSR7:0x80000 with 0x00000010 will Rest FIFO.
2) Please note that only the least significant address bits are decoded, and hence the mirroring.

***CSR8**: DSP SDRAM Control Register.*
SDRAM Control register, sets the SDRAM's pair of bank select bits, as well as add support for configuring the SDRAM for dual access by the host when the DSP is active. Accessible by the DSP only.

Register	**Description**	**Host Mapping**	**DSP Mapping**
CSR8 – SDRAM Control (SDCTL)	Sets the SDRAM Module bank lines, and configuration bits used by the FPGA's internal SDRAM controller.	n/a	0x803C,8000 (mirrored across 0x803C,8000-0x803C,BFFF)
Bit Position	**Description**		
11:0	SDRAM Control bits used by FPGA SDRAM Controller.		
13:12	SDRAM Bank Select bits, which set the multiplexer that routes the proper DSP address bits to accommodate the rows of the memory ICs on the SDRAM module. 00 = 11 rows. 01 = 12 rows. 1X = 13 rows.		
15:14	SDRAM Control bits used by FPGA SDRAM Controller.		
31:16	n/a		

***CSR9:** DSP SDRAM Extension Register.*

Extended SDRAM control for allowing the host access to the SDRAM when the DSP is active. Accessible by the DSP only.

Register	**Description**	**Host Mapping**	**DSP Mapping**
CSR9 – SDRAM Extensions (SDEXT)	Extended SDRAM control used by the FPGA's internal SDRAM controller.	n/a	0x803C,C000 (mirrored across 0x803C,C000-0x803C,FFFF)
Bit Position	**Description**		
31:0	Extended SDRAM Control bits used by FPGA SDRAM Controller.		

***CSR11**: CommRegs Key/DSP Lockout.*

By default, the CommRegs[16:63] serve as a scratch pad region by blocking DSP accesses. However, to enable DSP accesses to the scratch pad, a "key" value may be written into the 'CommRegs Key' CSR by either host or DSP in order enable DSP accesses. Likewise, any other value written into the CommRegs Key will lock out DSP accesses. Mainly intended for DSP code debugging.

***NOTE**:*

1) *Used in conjunction with CSR0[13:12] DSP Error Status bits.*
2) *CSR10 & CSR11 only implemented on FPGA bitfiles dated after 2013.*

Register	**Description**	**Host Mapping**	**DSP Mapping**
CSR11 – CommReg KEY	A random value highly unlikely to be duplicated by an erroneous DSP access.	0x10,0100	0x8030,0100
Bit Position	**Description**		
31:0	0x83E70B13: Key Value allows DSP accesses to Scratch Pad or CommRegs 16:63		

7.5 Expansion Card Address Mapping

The Expansion space is accessible by both the host and the DSP, and occupies the entire 256KB (64KDWord) address range, and is NOT mirrored.

Resource	Host BADDR2+Offset (Byte Boundary)	DSP Base Address (Byte Boundary)
Expansion Module (64Kx32)	0x1C,0000-0x1F,FFFF	0x8038,0000-0x803B,FFFF

7.6 Boot Memory Address Mapping

The DSP's boot memory may be configured with either a 512Kx8 SRAM device, or a 1Mx16 Flash device. The BootMEM space is mirrored across the entire 6M (1.5MDWord) address range. Please refer to section 8.2 for more details.

BootMEM Type	Physical Device Size	Actual Host Range (Bytes)	Byte Lane Used (0-3 possible)	Host BADDR2+Offset (Byte Boundary)
SRAM	512Kx8	2M of 6M	0 (bits 7:0)	0x20,0000-0x3F,FFFF (unused range 0x40,0000-0x7F,FFFF)
			n/a (bits 15:8)	
			n/a (bits 23:16)	
			n/a (bits 31:24)	
NOR Flash	1Mx16	4M of 6M	0 (bits 7:0)	0x20,0000-0x5F,FFFF (unused range 0x60,0000-0x7F,FFFF)
			1 (bits 15:8)	
			n/a (bits 23:16)	
			n/a (bits 31:24)	

bootMEM Mapping with 512Kx8 SRAM
Mapped as 512Kx32, with upper 24 bits ignored.

bootMEM Mapping with 1Mx16 Flash
Mapped as 1Mx32 with upper 16 bits ignored.

The DSP's boot memory, as seen from the host, is broken down into sections organized as follows:

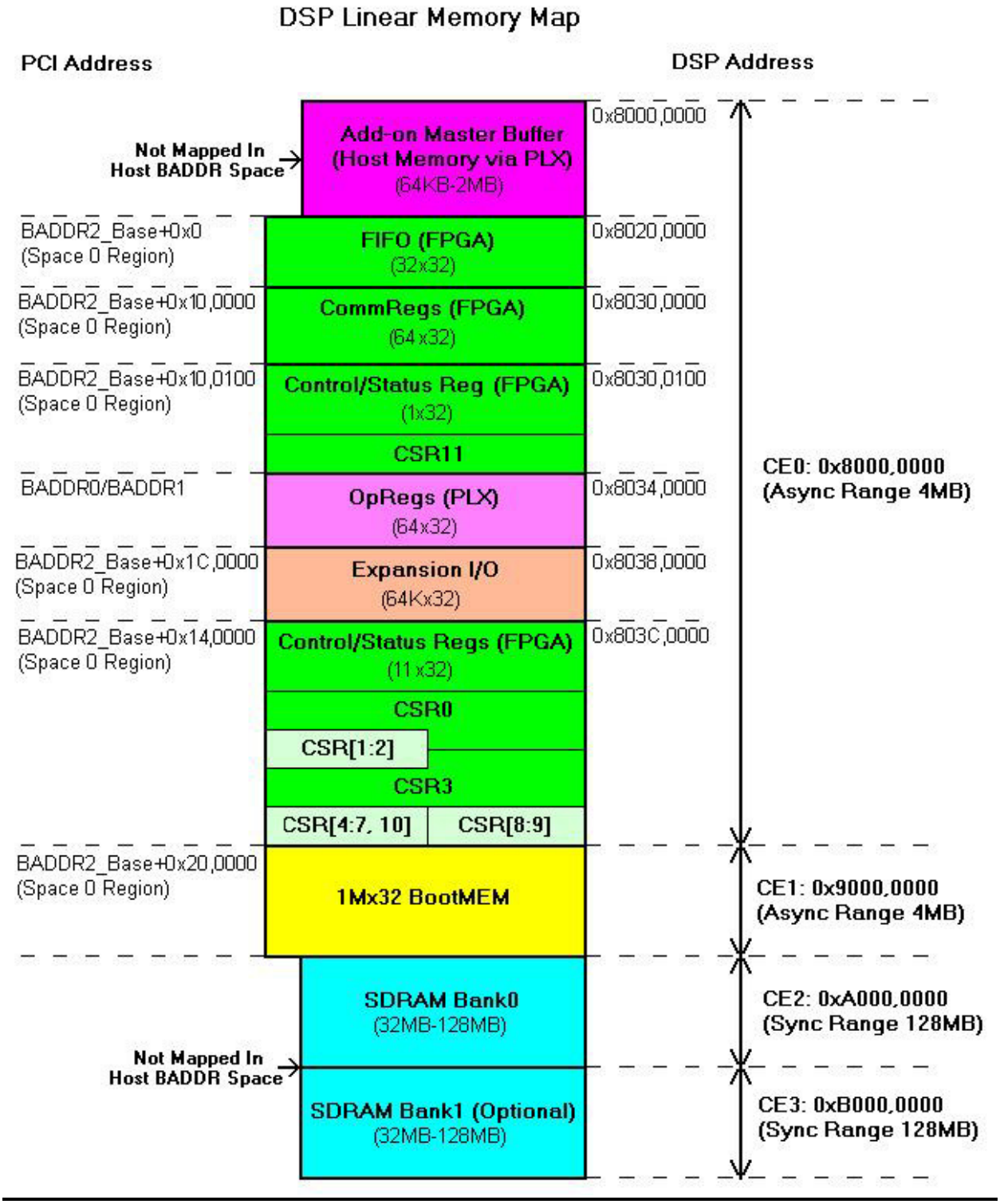
Boot Load Extender & PLL (x2) Registers: 0x200,000 - 0x200,FFF (4K Bytes)

SDRAM SPD Contents: 0x201,000 - 0x201,7FF (2K Bytes)

C671x SDRAM (x3): 0x201,800 - 0x203,FFF (10K Bytes)

COFF Data and User Space: 0x204,000 - Boot Device's Upper Limit (508K Bytes to 4M Bytes)

NOTE:
1) The base address for the boot memory is 0x9000,0000 as seen from the DSP; while the base is 0x200,000 as seen from the host with the added exception that both memories are always mapped as 32 bit devices, with the upper data bits ignored.
2) None of the boot memory range is mirrored on the host side.


DSP Linear Memory Map
PCI Address
DSP Address
Not Mapped In
Host BADDR Space
Add-on Master Buffer
(Host Memory via PLX)
(64KB-2MB)
0x8000,0000
BADDR2_Base+0x0
(Space 0 Region)
FIFO (FPGA)
(32x32)
0x8020,0000
BADDR2_Base+0x10,0000
(Space 0 Region)
CommRegs (FPGA)
(64 x32)
0x8030,0000
BADDR2_Base+0x10,0100
(Space 0 Region)
Control/Status Reg (FPGA)
(1x32)
0x8030,0100
CSR11
BADDR0/BADDR1
OpRegs (PLX)
(64x32)
0x8034,0000
CE0: 0x8000,0000
(Async Range 4MB)
BADDR2_Base+0x1C,0000
(Space 0 Region)
Expansion I/O
(64Kx32)
0x8038,0000
BADDR2_Base+0x14,0000
(Space 0 Region)
Control/Status Regs (FPGA)
(11 x32)
0x803C,0000
CSR0
CSR[1:2]
CSR3
CSR[4:7, 10]
CSR[8:9]
BADDR2_Base+0x20,0000
(Space 0 Region)
1Mx32 BootMEM
CE1: 0x9000,0000
(Async Range 4MB)
SDRAM Bank0
(32MB-128MB)
CE2: 0xA000,0000
(Sync Range 128MB)
Not Mapped In
Host BADDR Space
SDRAM Bank1 (Optional)
(32MB-128MB)
CE3: 0xB000,0000
(Sync Range 128MB)

8.0 DSP Address Mapping

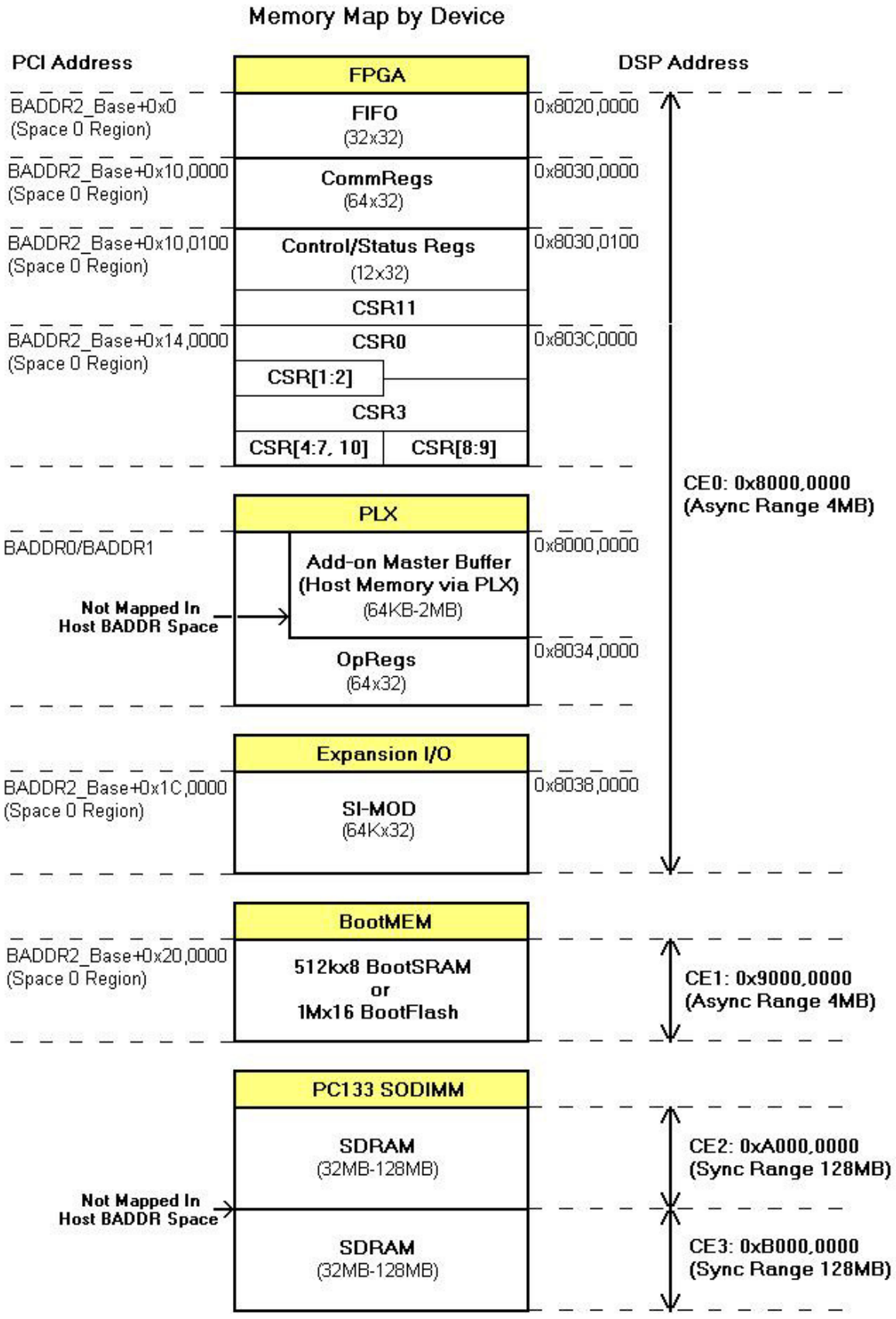
This section describes memory mapping as seen from the C6x DSP. The DSP's EMIF space is partitioned into four separate pages, with varying depths per page. Each page is externally identified in hardware with the CE[3:0] signals. Please consult the C6x's documentation for more details.

Used Physical Range	DSP CEn (External Space)	Resource		DSP Base Address (Byte Boundary)	
64KB-2MB	0	Add-on Init Buffer (DSP side only)		0x8000,0000-0x801F,FFFF	
1		FPGA: 2x 32 Deep FIFO		0x8020,0000 (mirrored across 0x8020,0000-0x802F,FFFF)	
64		FPGA: x64 CommRegs		0x8030,0000-0x8030,00FF (mirrored across 0x8030,0000-0x8033,FFFF)	
64		PLX OpRegs		0x8034,0000-0x8034,00FF (mirrored across 0x8034,0000-0x8037,FFFF)	
64Kx32		Expansion Module		0x8038,0000-0x803B,FFFF	
5		FPGA: x5 CSRs	CSR11 (CKEY)	0x8030,0100 (mirrored across 0x8030,0100-0x8030,40FF)	0x8030,0100-0x8030,0103 (mirrored across 0x8030,0100-0x8030,40FF), 0x803C,0000-0x803C,C000 (mirrored across 0x803C,0000-0x803C,FFFF)
			CSR0 (GC)	0x803C,0000 (mirrored across 0x803C,0000-0x803C,3FFF)	
			CSR3 (DATE)	0x803C,4000 (mirrored across 0x803C,4000-0x803C,7FFF)	
			CSR8 (SDCTL)	0x803C,8000 (mirrored across 0x803C,8000-0x803C,BFFF)	
			CSR9 (SDEXT)	0x803C,C000 (mirrored across 0x803C,C000-0x803C,FFFF)	
512Kx32 (SRAMx8) 1Mx32 (NOR Flash x16)	1	BootMEM		0x9000,0000-0x9FFF,FFFF	
128MB	2	SDRAM		0xA000,0000-0xAFFF,FFFF	
128MB	3	SDRAM		0xB000,0000-0xBFFF,FFFF	

All CE0 regions are mirrored after the maximum address value. Addresses are continuously decoded, however the state machine transitions only when the WR or RD lines are active thereby giving extra setup time and increasing the logic's decoding reliability.

The DSP's CE[3:0] lines must be configured in software through the EMIF control registers, with the following parameters in order to function properly:

DSP CEn	EMIF Parameters
0	Asynchronous memory (FPGA, PLX OpRegs, Expansion) Setup = 1 (or 0 if it works) Strobe = 3 Hold = 0
1	Asynchronous memory (bootMEM) Setup = 1 (or 0 if it works) Strobe = 3 Hold = 0
2	SDRAM Region 0 (determined by SPD on SO-DIMM module)
3	SDRAM Region 1 (determined by SPD on SO-DIMM module)


Memory Map by Device
PCI Address
DSP Address
FPGA
BADDR2_Base+0x0
(Space 0 Region)
FIFO
(32x32)
0x8020,0000
BADDR2_Base+0x10,0000
(Space 0 Region)
CommRegs
(64x32)
0x8030,0000
BADDR2_Base+0x10,0100
(Space 0 Region)
Control/Status Regs
(12x32)
0x8030,0100
CSR11
BADDR2_Base+0x14,0000
(Space 0 Region)
CSR0
0x803C,0000
CSR[1:2]
CSR3
CSR[4:7, 10]
CSR[8:9]
CE0: 0x8000,0000
(Async Range 4MB)
PLX
BADDR0/BADDR1
Add-on Master Buffer
(Host Memory via PLX)
(64KB-2MB)
0x8000,0000
Not Mapped In
Host BADDR Space
OpRegs
(64x32)
0x8034,0000
Expansion I/O
BADDR2_Base+0x1C,0000
(Space 0 Region)
SI-MOD
(64Kx32)
0x8038,0000
BootMEM
BADDR2_Base+0x20,0000
(Space 0 Region)
512kx8 BootSRAM
or
1Mx16 BootFlash
CE1: 0x9000,0000
(Async Range 4MB)
PC133 SODIMM
SDRAM
(32MB-128MB)
CE2: 0xA000,0000
(Sync Range 128MB)
Not Mapped In
Host BADDR Space
SDRAM
(32MB-128MB)
CE3: 0xB000,0000
(Sync Range 128MB)

8.1 DSP's CE0 Page Mapping

There are six (6) subsections as seen by the DSP in the first page denoted as CE0:

Used Physical Range	Resource		DSP Base Address (Byte Boundary)	
64KB-2MB	Add-on Init Buffer (DSP side only)		0x8000,0000-0x801F,FFFF	
4B (1 DWord)	FPGA: 2x 32 Deep FIFO		0x8020,0000 (mirrored across 0x8020,0000-0x802F,FFFF)	
256B (64 DWords)	FPGA: x64 CommRegs		0x8030,0000-0x8030,00FF (mirrored across 0x8030,0000-0x8033,FFFF)	
20B (5 DWords)	FPGA: x5 CSRs	CSR11 (COMKEY)	0x8030,0100 (mirrored across 0x8030,0000-0x8030,40FF)	0x8030,0100-0x8030,0103 (mirrored across 0x8030,0100-0x8030,40FF);
		CSR0 (GC)	0x803C,0000 (mirrored across 0x803C,0000-0x803C,3FFF)	0x803C,0000-0x803C,C000 (mirrored across 0x803C,0000-0x803C,FFFF)
		CSR3 (DATE)	0x803C,4000 (mirrored across 0x803C,4000-0x803C,7FFF)	
		CSR8 (SDCTL)	0x803C,8000 (mirrored across 0x803C,8000-0x803C,BFFF)	
		CSR9 (SDEXT)	0x803C,C000 (mirrored across 0x803C,C000-0x803C,FFFF)	

8.1.1 Host Memory/Add-on Initiated Mapping From The DSP

When using the Add-on Initiated mode of communications, the DSP becomes the PCI bus master and also has direct access to memory that is physically located inside of the host computer. The default size of host buffer set aside for use by the DSP is set to 64KB, but may be increased to a maximum of 2MB.

The size of this host buffer is determined by a register value that resides inside of the PCI bridge device's configuration EEPROM, which is only read once at boot time by the Operating System. Therefore, if altered the system must be rebooted in order for the new size to take effect. Please consult the PLX's PCI9054 manual for more details.

NOTE:
1) The size of the DSP buffer that physically resides inside of host memory is determined by the DMRR register inside of the EEPROM, byte boundary offset 0x30. By default, this value is set to 0xFFFF0000 to yield a host buffer size of 64KB.
2) PCI Opreg and EEPROM must be changed so that DMLBAM register is set to decode on

0x00000000 for the C671x based boards. Note that this register is set for decoding on 0xFF000000 on the C33 based DSP cards. Please refer to PLX's PCI9054 documentation for more information.

8.1.2 PCI9054 Opreg Mapping From The DSP

Please refer to section 7.1 for more details.

8.1.3 FIFO (FPGA) Address Mapping From The DSP

Please refer to section 7.2 for more details.

8.1.4 Communication Registers (FPGA) Address Mapping From The DSP

Please refer to section 7.3 for more details.

8.1.5 Control/Status Registers (FPGA) Address Mapping From The DSP

Only CSR0, CSR3, CSR8, CSR9, and CSR11 are accessible by the DSP, while all are visible by the host. Please refer to section 7.4 for more details.

8.1.6 Expansion Card Address Mapping From The DSP

Please refer to section 7.5 for more details.

8.2 DSP's CE1 Page Mapping: Boot Memory

The SI-DSP's boot memory (bootMEM) is COFFloaded with the contents of a DSP binary or COFF file from the host PC using the SISAMPLE.EXE command line utility. The bootMEM space is mirrored across the entire 6M (1.5MDWord) address range.

The bootMEM may consist of either of the following:

1) 512Kx8 volatile SRAM
Normally, the SI-DSP is shipped with the bootMEM consisting of a 512Kx8 volatile SRAM, which requires the card to operate as a PCI agent device that resides inside a host PC.

2) 1Mx16 nonvolatile Flash
The bootMEM may optionally consist of a 1Mx16 nonvolatile Flash, which allows for the card to operate as either a PCI agent card inside the PC or as a standalone card outside of the PC.

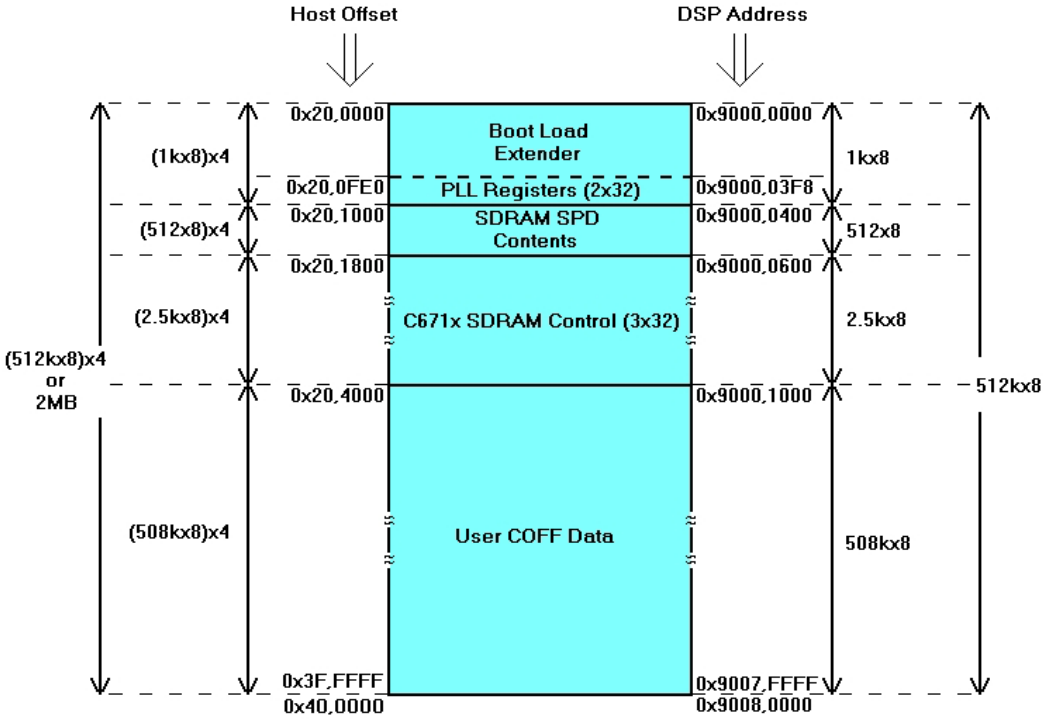
Note: In order for the SI-DSP to operate outside the PC, the boot Flash must be initially loaded with a COFF file.

NOTE:

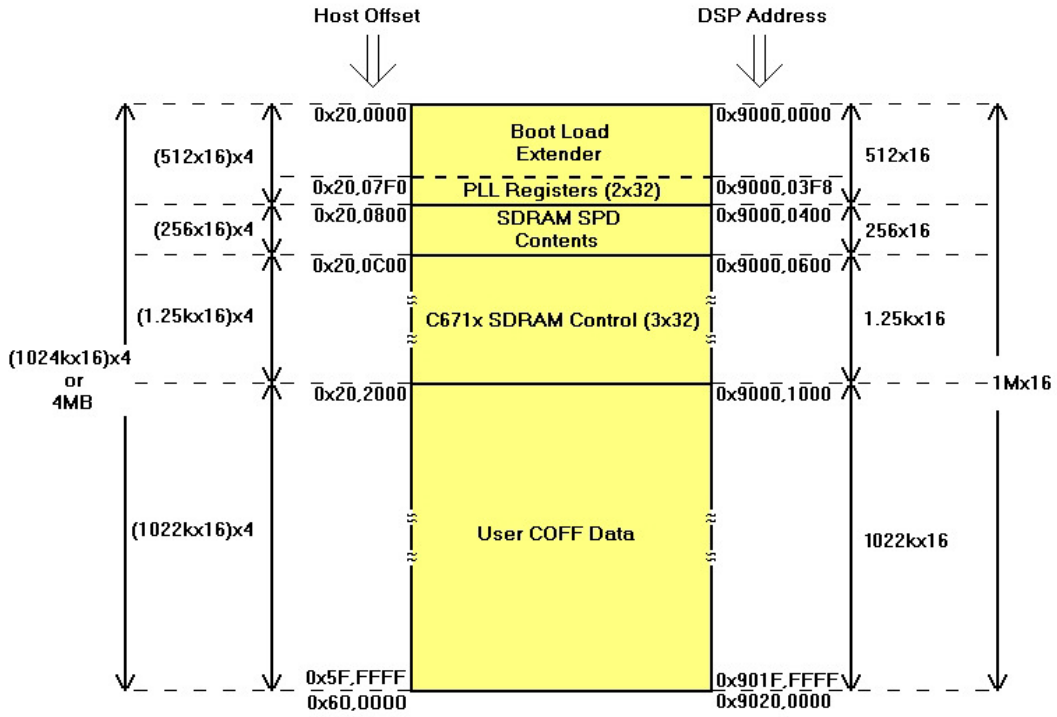
1) *The C67x's CE1 region must be configured for linear Asynchronous devices since the external boot memory can only be a ROM, Flash or SRAM device. This limits the total depth of the EMIF bus to a linear addressing range of 1M, comprised of 20 address bits.*
2) *The base address for the boot memory is 0x9000,0000 as seen from the DSP; while the base is 0x200,000 as seen from the host with the added exception that both memories are always mapped as 32 bit devices, with the upper data bits ignored.*

BootMEM Type	Actual Range (Bytes)	Byte Lane Used (0-3 possible)	DSP Address	Host BADDR2+Offset (Byte Boundary)
512Kx8 SRAM	2M of 6M	0 (1through 3 unused)	0x9000,0000-0x9007,FFFF (unused range 0x9008,0000-0x9FFF,FFFF)	0x20,0000-0x3F,FFFF (unused range 0x40,0000-0x7F,FFFF)
1Mx16 NOR Flash	4M of 6M	0 and 1 (2 and 3 unused)	0x9000,0000-0x900F,FFFF (unused range 0x9010,0000-0x9FFF,FFFF)	0x20,0000-0x3F,FFFF (unused range 0x40,0000-0x7F,FFFF)

512kx8 BootSRAM Mapping



1Mx16 BootFLASH Mapping



8.2.1 Boot Memory Organization

The DSP's boot memory, as seen from the *DSP*, is broken down into sections organized as follows:

Boot Load Extender &PLL (x2) Registers: 0x9000,0000 - 0x9000,03FF (1K Bytes)
NOTE:
1) The boot load extender resides inside of the COFF file.
2) As outlined in the TI documentation, the DSP's internal boot loader is limited to reading only 1KBytes when first activated or pulled out of Reset. From studying the supplied source code, please note the presence of a pair of copy sections that behave as Boot Load extenders, which must also be present in a single DSP COFF file, if only a single COFF file is to be loaded to the DSP. The Boot Load Extender section inside of the supplied DSP utility COFF file amends this 1KByte limitation such that COFF files up to 512KBytes may be inserted when the bootMEM is configured with the default 512KBytes SRAM, or up to 2MBytes when the optional Flash is used for the bootMEM. IF larger COFF files are necessary, it can be done in a second instance of a COFF file load using active DSP communications.
3) The two PLL register values are derived from reading the board's DID register, which in turn resides inside of the PCI configuration NVRAM and is used by the host to identify the DSP's core clock speed. They are placed at the bottom of the 1Kbyte extender section so that they are automatically included inside of the DSP's internal SRAM since the bootMEM is not accessible by the DSP when it is configuring its PLL.

SDRAM SPD Contents: 0x9000,0400 - 0x9000,05FF (512 Bytes)
NOTE:
1) The SDRAM SPD contents reside on the SDRAM module.
2) Because the SDRAM's SPD contents are only accessible by the host via the FPGA, a mirror image of these contents are placed into the boot memory so the DSP may also have access to its contents after it is activated.

C671x SDRAM Control (x3): 0x9000,0600 - 0x9000,0FFF (2.56K Bytes)
NOTE:
1) The host places these registers into the boot memory during the COFF load, before the DSP is taken out of reset so it may begin its boot process.
2) The three C671x SDRAM Control register values are derived from the SDRAM's SPD and preformatted by the host in order to simplify the DSP's boot process.

COFF Data and User Space: 0x9000,1000 - Boot Device's Upper Limit (508K Bytes to 4M Bytes)
NOTE: The COFF data and user space resides inside of the COFF file.

The DSP's boot memory, as seen from the *host*, is broken down into sections organized as follows:

Boot Load Extender & PLL (x2) Registers: 0x200,000 - 0x200,FFF (4K Bytes)

SDRAM SPD Contents: 0x201,000 - 0x201,7FF (2K Bytes)

C671x SDRAM (x3): 0x201,800 - 0x203,FFF (10K Bytes)

COFF Data and User Space: 0x204,000 - Boot Device's Upper Limit (508K Bytes to 4M Bytes)

NOTE:
1) The base address for the boot memory is 0x9000,0000 as seen from the DSP; while the base is 0x200,000 as seen from the host with the added exception that both memories are always mapped as 32 bit devices, with the upper data bits ignored.
2) The boot memory is only directly accessible from the host when the DSP is inactive or in Reset. Once the DSP is activated (Reset removed), the boot memory contents can only be accessed from the host using one of the various active communication modes described in another section.

8.3 DSP's CE2 & CE3 Page Mapping

The last pair of pages are reserved for the SDRAM module, where each page has up to 128MB available each, for a maximum of 256MB. The SDRAM module is a standard PC133, 144 pin SODIMM module.

Used Physical Range	DSP CEn (External Space)	Resource	DSP Base Address (Byte Boundary)
128MB	2	SDRAM	0xA000,0000-0xAFFF,FFFF
128MB	3	SDRAM	0xB000,0000-0xBFFF,FFFF

PAGE2:CE2. This page operates Synchronously, maps one half (single rank modules) or one quarter (dual bank modules) of the SDRAM module.

PAGE3:CE3. This page operates Synchronously, maps one quarter (dual bank modules only) of the SDRAM module.

Synchronous EMIF Bus Operation. The many timing parameters for synchronous EMIF bus operation reside inside of the SDRAM module's serial EEPROM, which varies between different SDRAM module manufacturers. Conventional 144 pin sockets are used to accommodate standard, 3.3V non-buffered PC133 SO-DIMM modules found on laptops. Module sizes of 64MB, 128MB, 256MB, and 512MB are supported, making it a very cost effective solution for the most demanding and memory intensive applications.

NOTE: *Only SO-DIMM modules whose DRAM ICs' parameters fall within the C67x's supported range will be fully operational. The C67x's internal DRAM controller supports the following parameters:*
1) Columns: 8, 9, 10.
2) Rows: 11, 12, 13.
3) Memory Device Banks: 2, 4. Not to be confused with the SODIMM's "bank", which is either 1 or 2.
4) Cache Latency: 2, 3.

9.0 DSP Reset Operation

In order to disable the DSP, it must be placed in a 'Reset' state by the host. While in reset, all of the DSP address and data lines are placed in a high impedance state. Once the DSP is activated by removing the Reset, it will begin by fetching the Reset Vector by accessing the boot memory (bootMEM). Please refer to the C67x data sheet for further details. The RESET function is controlled by the host via a target mode access to the CSR0-Bit0.

CSR0*: General Control - GC Register.*
Address Range (Bytes): 0x00140000 - 0x0015FFFF
Range Size (Bytes): 128k
Range Size (DWords): 32k
Physical Depth: 1 DWord mirrored across entire range.

CSR0 Bits[5:0]:
Bit0: DSP Reset Control
0 = DSP is disabled.
1 = DSP is activated to run code.
Bit[5:1]: Please refer to above section for more details.

NOTE*: The RESET VECTOR and the RESET control bit are completely separate and not to be confused. The RESET VECTOR contains the pointer to the first instruction of executable DSP code, while the RESET control bit is only accessible by the host so as to impose a reset condition on the DSP.*

10.0 Code Composer Studio

For custom code development on the DSP, TI's Code Composer Studio version 4.x and up are recommended. Sheldon Instruments supplies a complete utility and related source code that integrates all of the card's communication modes with the host PC, and is intended to be used as a template from which to build the foundation of your own custom application.

The COFF file that runs on the DSP side are supplied in the following paths:

File Path	Comments
\sidev\binaries\32bit\apps\c6711plxini.out	DSP utility COFF file
\sidev\sidsp\basiccom\plxc6711\rev1	DSP utility COFF file C source code and project files.

NOTE:
1) When first invoking Code Composer Studio, make sure that the C67xx chipset is the target device.
2) The Chip Support Library (CSL) is a part of CCS.
3) SI-DSP cards do not supply any specific Board Support Library (BSL).

11.0 JTAG Debugging

Step 1: Configure the JTAG Device

1a) JTAG Emulator Software.

Follow the instructions provided by JTAG vendor for installing the drivers and configuring the proper CCS setup for the JTAG emulator.

1b) Attach the JTAG 14 pin connector with System Power Turned OFF.

Before attaching the JTAG's 14 pin connector, be sure there is NO power to either the host PC or to the SI-DSP card. Once attached, host PC and SI-DSP power may be turned on.

***NOTE**: Please refer to the FPGALoad documentation for more details.*

Step 2: Configure the SI-DSP Card.

2a) Load SI-DSP Onboard Logic with FPGA Bitloader Utility.

The onboard logic inside the FPGA must be bitloaded for proper system operation. Since no external FPGA SPROM is usually supplied, the FPGA must be bitloaded; however, if an external FPGA SPROM is present and if the SI-DSP card is in standalone operation, then the FPGA is automatically loaded and hence no utility is required.

NOTE:

Please refer to the FPGALoad documentation for more details.

2b) Activate the DSP Emuation Circuitry by COFFLoading.

In order to perform JTAG debugging, the DSP's internal emulation circuitry must be activated by deasserting its 'Reset' line, which is automatically achieved by COFFloading. If the SI-DSP card resides inside the host PC, COFFloading will automatically perform this function and properly activate the DSP. However, if the SI-DSP card is in standalone operation, the onboard logic will automatically perform this function after a 10 second delay and the DSP will automatically COFFload from the Flash.

Step 3: Open CCS.

Open CCS and select the target created in the step above.

Step 4: Open the CCS Project.

Project->Open: \sidev\..\sidsp\basiccom\plxc6711\Rev1\coff_x1

Step 5: Set Active Configuration to Debug.

Project->Configurations->Debug->Set Active

In order to build a DSP binary to be loaded via the JTAG port, set the active configuration to 'debug' mode, which will insert debugging codes for use by the JTAG emulator. Contrarily, note that by setting the active configuration to 'Release' mode, the smaller COFF files that are built cannot be loaded via the JTAG port as these binaries are intended to be COFFloaded via the host PC or from the Flash withOUT JTAG support.

Step 6: Build CCS Project.

Project->Build Clean

Project->Build

Step 7: Connect the JTAG (the physical connector should be connected before powering on the board).
Debug->Connect

Step 8: Load the DSP Binary.
File->Load Program->c6711plxini.out

Step 9: Start Using JTAG Debugger Controls.
You can now use the various debugger controls: Run, Step (along with its variations such as Halt, Add to Watch Window, ...and so forth.

Step 10: Verification.
10a) Verification for SI-DSP Inside Host PC.
If the SI-DSP card resides inside the host PC, in order to verify success you can read the heart beat after running the default (Release) coff file at Commreg[7] (host address 0x10_001C, or DSP address 0x8030_001C). After you connect the JTAG (software connection), load and run the 'Debug' COFF file and the heartbeat shall be located at Commreg[15] (host address 0x10003C, or DSP address 0x8030_003C).

10b) Verification for SI-DSP In Standalone Operation.
After you connect the JTAG (software connection), load and run the 'Debug' coff file, and the heartbeat shall be located at Commreg[15] (DSP address 0x8030_003C).

NOTE:
1) Make sure that Code Composer Studio has been properly configured for your JTAG debugger and the C67XX chipset.
2) When compiling Release and Debug versions of COFF files, where Debug versions are intended to be loaded via the JTAG port, it is recommended to have both the 'Simulator' as well as the corresponding 'Emulator' entries added to the System Configuration tool during CCS setup. Therefore, both Release and Debug versions of COFF files are compiled with CCS invoked exclusively via the 'Simulator' entry of System Configuration, especially since compiling with the 'Emulator' entry may render unpredictable results. However, JTAG loading of the Debug version should be carried out with CCS invoked exclusively via the 'Emulator' entry of the System Configuration.

12.0 SI-C6xDSP Custom Hardware Interface

The SI-C6xDSP includes expansion connectors allowing for custom designs, or for attaching 'off the shelf' multifunction I/O modules from Sheldon Instruments. Sheldon Instruments offers several daughter modules for multichannel analog and digital I/O, including 4 to 64 channels of 16 bit ADCs and DACs.

12.1 DSP Expansion Bus Connector for All PCI Form Factors: 2mm 5x30 Header to C6x EMIF Bus

A 2mm connector organized as 5x30 is used to interface the custom daughtercard to the C6x's EMIF bus. 64Kx32 words are decoded for use by the custom daughtercard, which is mapped into the C6x's EMIF bus, byte boundary address space ranging from 0x80380000 to 0x803BFFFF. Please refer to the TMS320C6x User's Manual for further details.

The following signals follow the timing outlined in the DSP's User Manual, and are a buffered version of the DSP signals. All signals are buffered through a 74FCT2245 device, while the clocks are buffered with an IDT2308-2/CY2308-2 zero delay clock buffer device.

X_CLK[1:0]: *37.5Mhz Expansion Clock signals.*
Input signals to expansion card, generated by DSP card. X_CLK0 is the primary clock source, while X_CLK1 is the secondary clock source, both are derived from the DSP's 75Mhz *ECLK*.
R/Wn: *EAWEn, DSP EMIF Bus Write Enable signal.*
Input signal to expansion card, generated by DSP card. Routed from DSP's EMIF bus *EAWEn* line. When driven LO, the DSP is performing a write cycle, otherwise a read cycle is assumed.
D[31:0]: *ED[31:0],Data Bus.*
Bidirectional data bus. Routed from DSP's EMIF bus *ED[31:0]* bus.
A[15:0]: *EA[17:2], Address Bus.*
Input bus to expansion card, generated by DSP card. Routed from DSP's EMIF bus *EA[17:2]* bus. The expansion card is designed to be addressed within a 32 bit boundary, while the DSP has an 8 bit boundary; therefore, the DSP's two LSBs are ignored and not routed to the expansion card connector.

The following signals still follow the timing outlined in the User Manual, but are regenerated by the FPGA which includes an extra layer of decoding:

X_INT[1:0]: *Expansion Card Interrupts to DSP.*
Output signals generated by expansion card. Interrupt signals routed to one of the DSP's external interrupt lines labeled as *EXTINT[7:4]*; with the primary X_INT0 line defaulted to use the C6x's *EXTINT4*, while the secondary X_INT1 line is disabled. When driven LO, the expansion card is generating an interrupt to the DSP. Please refer to the 'Interrupt Mask CSR' for more information.
X_INTACK: *Interrupt Acknowledge from DSP to Expansion Card.*
Not supported on C6x cards, only defined for compatibility with other DSP cards.
X_CS[1:0]n: *Expansion Card Select signal.*
Input signal to expansion card, generated by DSP card. Equivalent to DSP's EMIF bus *CE0* line, with additional logical ANDing with expansion card's decoded address lines. When driven LO,

the DSP is performing an access to the expansion card. X_CS0n corresponds to the primary daughter module and is always present, while X_CS1n corresponds to the optional, stackable daughter module.

X_RDYn: *EARDYn, Expansion Card Ready to DSP.*
Output signal generated by expansion card. Routed to DSP's EMIF bus *EARDYn* line. When driven HI, hardware wait states are inserted into the DSP's EMIF bus access cycle.

NOTE*: Signals are 3.3V, and thus NOT 5V tolerant.*

DSP Carrier Pinout:

D.30. **GND**	C.30. **GND**	B.30. **GND**	A.30. **GND**	AUX.30. **GND**
D.29. **+3.3V**	C.29. **+3.3V**	B.29. **+3.3V**	A.29. DSPSPa2(CLKX0)	AUX.29. DSPSPa1(DX0)
D.28. **+5V**	C.28. **+1.8V**	B.28. **+5V**	A.28. DSPSPa0(FSX0)	AUX.28. DSPSPa5(CLKR0)
D.27. **+12V/+15V**	C.27. RSV2	B.27. **+12V/+15V**	A.27. DSPSPa4(DR0)	AUX.27. DSPSPa3(FSR0)
D.26. **-12V/-15V**	C.26. **-5V**	B.26. **-12V/-15V**	A.26. DSPSPa6(CLKS0)	AUX.26. **GND**
D.25. **GND**	C.25. **GND**	B.25. **GND**	A.25. TIMEROUTa1	AUX.25. TIMERINa0
D.24. FPGA_DATA	C.24. FPGA_INIT	B.24. A15	A.24. TIMEROUTb1	AUX.24. TIMERINb0
D.23. FPGA_CLK	C.23. FPGA_PGRMn	B.23. A14	A.23. **GND**	AUX.23. **GND**
D.22. FPGA_DONE	C.22. **GND**	B.22. A13	A.22. -	AUX.22. -
D.21. X_CLK1	C.21. X_CLK0	B.21. A12	A.21. **GND**	AUX.21. **GND**
D.20. D15	C.20. **GND**	B.20. D31	A.20. DSPSPb2(CLKX1)	AUX.20. DSPSPb1(DX1)
D.19. D14	C.19. RSV1	B.19. D30	A.19. DSPSPb0(FSX1)	AUX.19. DSPSPb5(CLKR1)
D.18. D13	C.18. X_INT1	B.18. D29	A.18. DSPSPb4(DR1)	AUX.18. DSPSPb3(FSR1)
D.17. D12	C.17. X_INT0	B.17. D28	A.17. DSPSPb6(CLKS1)	AUX.17. **GND**
D.16. **GND**	C.16. **GND**	B.16. **GND**	A.16. HP4	AUX.16. HP7
D.15. D11	C.15. A11	B.15. D27	A.15. HP8	AUX.15 HP9
D.14. D10	C.14. A10	B.14. D26	A.14. HP10	AUX.14. HP11
D.13. D9	C.13. A9	B.13. D25	A.13. HP12	AUX.13. HP13
D.12. D8	C.12. A8	B.12. D24	A.12. HP14	AUX.12. HP15
D.11. D7	C.11. A7	B.11. D23	A.11. HP22 (HINTn)	AUX.11. **GND**
D.10. D6	C.10. A6	B.10. D22	A.10. HP20 (HASn)	AUX.10. HP25 (HCNTL1)
D.9. D5	C.9. A5	B.9. D21	A.9. HP2	AUX.9. HP16 (HRDYn)
D.8. D4	C.8. A4	B.8. D20	A.8. HP21 (HR/Wn)	AUX.8. HP23 (HWIL)
D.7. D3	C.7. A3	B.7. D19	A.7. HP6	AUX.7. **GND**
D.6. D2	C.6. A2	B.6. D18	A.6. HP19 (HCSn)	AUX.6. HP24 (HCNTL0)
D.5. D1	C.5. A1	B.5. D17	A.5. HP0	AUX.5. HP5
D.4. D0	C.4. A0	B.4. D16	A.4. HP3	AUX.4. HP17 (HDS2n)
D.3. R/Wn	C.3. X_CS0n	B.3. RSV0	A.3. NC/SDAa	AUX.3. HP18 (HDS1n)
D.2. X_RDYn	C.2. **+5V**	B.2. **+3.3V**	A.2. NC/SCLa	AUX.2. HP1
D.1. **GND**	C.1. **GND**	B.1. **GND**	A.1. **GND**	AUX.1. **GND**

Color Index:

BLUE: DSPSPa
BLACK: GPIO
RED: ASPb
GREEN: DSPSPb
PURPLE: DSPTIMERa:b

***NOTE**: Signals are 3.3V, and NOT 5V tolerant.*

12.2 User I/O Connector for PCI/cPCI Cards: 100 Pin D-Sub to 2mm 4x30 Header

A 100 pin half pitch (0.050"), Series III DSUB connector is used to interface external user defined signals to a single 2mm, 4x30 pin connector (120 contacts total). This connector is linked to the custom expansion card, and no signals from the DSP are routed. Below is the connection diagram.

***NOTE**:*

a) Signals designated as "USERI/On" correspond with pin "n" of the 100 pin DSub connector.

b) The USERI/O connectors are ABSENT on PC104Plus form factor cards. PC104Plus cards use a separate adapter for accessing USERI/O signals.

User I/O Pinout as seen from the 2MM 4x30 pin connector:

D.30. USERI/O99	C.30. USERI/O100	B.30. USERI/O49	A.30. USERI/O50
D.29. USERI/O97	C.29. USERI/O98	B.29. USERI/O47	A.29. USERI/O48
D.28. USERI/O95	C.28. USERI/O96	B.29. USERI/O45	A.28. USERI/O46
D.27. USERI/O93	C.27. USERI/O94	B.29. USERI/O43	A.27. USERI/O44
D.26. USERI/O91	C.26. USERI/O92	B.29. USERI/O41	A.26. USERI/O42
D.25. -	C.25. -	B.25. -	A.25. -
D.24. USERI/O89	C.24. USERI/O90	B.24. USERI/O39	A.24. USERI/O40
D.23. USERI/O87	C.23. USERI/O88	B.23. USERI/O37	A.23. USERI/O38
D.22. USERI/O85	C.22. USERI/O86	B.22. USERI/O35	A.22. USERI/O36
D.21. USERI/O83	C.21. USERI/O84	B.21. USERI/O33	A.21. USERI/O34
D.20. USERI/O81	C.20. USERI/O82	B.20. USERI/O31	A.20. USERI/O32
D.19. -	C.19. -	B.19. -	A.19. -
D.18. USERI/O79	C.18. USERI/O80	B.18. USERI/O29	A.18. USERI/O30
D.17. USERI/O77	C.17. USERI/O78	B.17. USERI/O27	A.17. USERI/O28
D.16. USERI/O75	C.16. USERI/O76	B.16. USERI/O25	A.16. USERI/O26
D.15. USERI/O73	C.15. USERI/O74	B.15. USERI/O23	A.15. USERI/O24
D.14. USERI/O71	C.14. USERI/O72	B.14. USERI/O21	A.14. USERI/O22
D.13. -	C.13. -	B.13. -	A.13. -
D.12. USERI/O69	C.12. USERI/O70	B.12. USERI/O19	A.12. USERI/O20
D.11. USERI/O67	C.11. USERI/O68	B.11. USERI/O17	A.11. USERI/O18
D.10. USERI/O65	C.10. USERI/O66	B.10. USERI/O15	A.10. USERI/O16
D.9. USERI/O63	C.9. USERI/O64	B.9. USERI/O13	A.9. USERI/O14
D.8. USERI/O61	C.8. USERI/O62	B.8. USERI/O11	A.8. USERI/O12
D.7. -	C.7. -	B.7. -	A.7. -
D.6. USERI/O59	C.6. USERI/O60	B.6. USERI/O9	A.6. USERI/O10
D.5. USERI/O57	C.5. USERI/O58	B.5. USERI/O7	A.5. USERI/O8
D.4. USERI/O55	C.4. USERI/O56	B.4. USERI/O5	A.4. USERI/O6
D.3. USERI/O53	C.3. USERI/O54	B.3. USERI/O3	A.3. USERI/O4
D.2. USERI/O51	C.2. USERI/O52	B.2. USERI/O1	A.2. USERI/O2
D.1. -	C.1. -	B.1. -	A.1. -

User I/O Pinout as seen from the 100 pin DSub connector:

USERI/O100. C30	USERI/O50. A30
USERI/O99. D30	USERI/O49. B30
USERI/O98. C29	USERI/O48. A29
USERI/O97. D29	USERI/O47. B29
USERI/O96. C28	USERI/O46. A28
USERI/O95. D28	USERI/O45. B28
USERI/O94. C27	USERI/O44. A27
USERI/O93. D27	USERI/O43. B27
USERI/O92. C26	USERI/O42. A26
USERI/O91. D26	USERI/O41. B26
USERI/O90. C24	USERI/O40. A24
USERI/O89. D24	USERI/O39. B24
USERI/O88. C23	USERI/O38.A23
USERI/O87. D23	USERI/O37. B23
USERI/O86. C22	USERI/O36. A22
USERI/O85. D22	USERI/O35. B22
USERI/O84. C21	USERI/O34. A21
USERI/O83. D21	USERI/O33. B21
USERI/O82. C20	USERI/O32. A20
USERI/O81. D20	USERI/O31. B20
USERI/O80. C18	USERI/O30. A18
USERI/O79. D18	USERI/O29. B18
USERI/O78. C17	USERI/O28. A17
USERI/O77. D17	USERI/O27. B17
USERI/O76. C16	USERI/O26. A16
USERI/O75. D16	USERI/O25. B16
USERI/O74. C15	USERI/O24. A15
USERI/O73. D15	USERI/O23. B15
USERI/O72. C14	USERI/O22. A14
USERI/O71. D14	USERI/O21. B14
USERI/O70. C12	USERI/O20. A12
USERI/O69. D12	USERI/O19. B12
USERI/O68. C11	USERI/O18. A11
USERI/O67. D11	USERI/O17. B11
USERI/O66. C10	USERI/O16. A10
USERI/O65. D10	USERI/O15. B10
USERI/O64. C9	USERI/O14. A9
USERI/O63. D9	USERI/O13. B9
USERI/O62. C8	USERI/O12. A8
USERI/O61. D8	USERI/O11. B8
USERI/O60. C6	USERI/O10. A6
USERI/O59. D6	USERI/O9. B6
USERI/O58. C5	USERI/O8. A5
USERI/O57. D5	USERI/O7. B5
USERI/O56. C4	USERI/O6. A4
USERI/O55. D4	USERI/O5. B4
USERI/O54. C3	USERI/O4. A3
USERI/O53. D3	USERI/O3. B3
USERI/O52. C2	USERI/O2. A2
USERI/O51. D2	USERI/O1. B2

NOTE: The manufacturer part numbers for the 100 pin, half pitch DSUB socket on the DSP card are as follows:

a) Amp - 787169-9
b) Amp - 787170-9
c) Amp - 787362-9
d) Honda - PCS-XE100LFD-HS

12.3 User I/O Connector for PMC Cards: 68 Pin D-Sub to 2mm 4x30 Header

A 68 pin half pitch (0.050"), Series III DSUB connector (SCSI style) is used to interface external user defined signals to a single 2mm, 4x30 pin connector (120 contacts total). This connector is linked to the custom expansion card, and no signals from the DSP are routed. Below is the connection diagram.

***NOTE**: Signals designated as "USERI/On" correspond with pin "n" of the 68 pin DSub connector.*

User I/O Pinout as seen from the 68 pin DSub connector:

USERI/O84. D2	USERI/O34. B2
USERI/O83. C2	USERI/O33. A2
USERI/O82. D3	USERI/O32. B3
USERI/O81. C3	USERI/O31. A3
USERI/O80. D4	USERI/O30. B4
USERI/O79. C4	USERI/O29. A4
USERI/O78. D5	USERI/O28. B5
USERI/O77. C5	USERI/O27. A5
USERI/O76. D6	USERI/O26. B6
USERI/O75. C6	USERI/O25. A6
USERI/O74. D8	USERI/O24. B8
USERI/O73. C8	USERI/O23. A8
USERI/O72. D9	USERI/O22. B9
USERI/O71. C9	USERI/O21. A9
USERI/O70. D10	USERI/O20. B10
USERI/O69. C10	USERI/O19. A10
USERI/O68. D11	USERI/O18. B11
USERI/O67. C11	USERI/O17. A11
USERI/O66. D12	USERI/O16. B12
USERI/O65. D17	USERI/O15. B17
USERI/O64. C17	USERI/O14. A17
USERI/O63. C22	USERI/O13. A22
USERI/O62. D23	USERI/O12. B23
USERI/O61. C23	USERI/O11. A23
USERI/O60. D24	USERI/O10. B24
USERI/O59. C24	USERI/O9. A24
USERI/O58. D26	USERI/O8. B26
USERI/O57. C26	USERI/O7. A26
USERI/O56. D27	USERI/O6. B27
USERI/O55. C27	USERI/O5. A27
USERI/O54. D28	USERI/O4. B28
USERI/O53. C28	USERI/O3. A28
USERI/O52. D29	USERI/O2. B29
USERI/O51. C30	USERI/O1. -

NOTE:The manufacturer part numbers for the 68 pin, half pitch DSUB socket (SCSI style) on the DSP card are as follows:

a) Amp - 787169-7
b) Amp - 787170-7
c) Amp - 787362-7

12.4 DSP Serial/Peripheral Port Connector for PCI Cards

A 2x5 pin (10 contacts), 0.1" pitch connector is used to interface to the C6x's first serial port:

10. **GND**	9. **GND**
8. DSPSPa2(CLKX0)	7. DSPSPa1(DX0)
6. DSPSPa0(FSX0)	5. DSPSPa5(CLKR0)
4. DSPSPa4(DR0)	3. DSPSPa3(FSR0)
2. DSPSPa6(CLKS0)	1. **GND**

A 2x25 pin (50 contacts), 0.1" pitch connector is used to interface to the C6x's second serial port, both timers, as well as the HPI/GPIO port:

50. TIMEROUTa1	49. TIMERINa0
48. TIMEROUTb1	47. TIMERINb0
46. **GND**	45. **GND**
44. -	43. -
42. **GND**	41. **GND**
40. DSPSPb2(CLKX1)	39. DSPSPb1(DX1)
38. DSPSPb0(FSX1)	37. DSPSPb5(CLKR1)
36. DSPSPb4(DR1)	35. DSPSPb3(FSR1)
34. DSPSPb6(CLKS1)	33. **GND**
32. HP4	31. HP7
30. HP8	29. HP9
28. HP10	27. HP11
26. HP12	25. HP13
24. HP14	23. HP15
22. HP22 (HINTn)	21. **GND**
20. HP20 (HASn)	19. HP25 (HCNTL1)
18. HP2	17. HP16 (HRDYn)
16. HP21 (HR/Wn)	15. HP23 (HWIL)
14. HP6	13. **GND**
12. HP19 (HCSn)	11. HP24 (HCNTL0)
10. HP0	9. HP5
8. HP3	7. HP17 (HDS2n)
6. NC/SDAa	5. HP18 (HDS1n)
4. NC/SCLa	3. HP1
2. **GND**	1. **GND**

Color Index:
BLUE: DSPSPa
BLACK: GPIO
RED: ASPb
GREEN: DSPSPb
PURPLE: DSPTIMERa:b

***NOTE**: Signals are 3.3V, and NOT 5V tolerant.*

12.5 DSP JTAG Port

A 2x7 pin arrangement (13 contacts, with pin 6 omitted to serve as the polarizing pin), 0.1" pitch connector is used to interface to the C671x's JTAG port. Pin 1 is distinguished with a square pad, with pinout as follows:

14. DSP_EMU1	13. DSP_EMU0
12. **GND**	11. TCK
10. **GND**	9. TCK_RTN
8. **GND**	7. TDO
6. No Connect	5. **+3Vdc**
4. **GND**	3. TDI
2. TRESET	1. TMS

***NOTE**: Signals are 3.3V, and NOT 5V tolerant.*

12.6 Jumpers for PCI Cards

VPP_EN: VPP Enable, for Flash programming enable.
LO= pins 2-3, disables Flash programming.
HI= pins 1-2, enables Flash programming (default). Places 3V onto the VPP pin of the Flash device.

This jumper is located at the bottom of the SDRAM connector.

OP_SEL: Operation Select.
SLAVE (LO)= pins 2-3, slave mode (default). The PCI bus bridge device is activated which allows access the DSP and vice versa. Therefore this is the mode to run if Flash is to be programmed while the card is inside of the host computer. The DSP's RESET line operates normally where it starts up asserted or '0', and then requires the host computer to deassert it as done in normal operation after COFF loading. In the case that the flash is already programmed, the host may simply deassert this line for DSP code execution to begin without the need to COFF load again.
F-STANALONE (HI)= pins 1-2, stand alone mode. Flash is expected to be fully functional, access to the PCI bus is blocked off. The DSP's RESET line automatically deasserts 10 seconds after power is applied, which will force DSP code execution.

This jumper is located at the top of the expansion connector.

MEMSEL: MS[1:0]: SDRAM row size selector jumpers.
00 (0x0) = 11 Row lines on SDRAM memory devices.
01 (0x1)= 12 Row lines on SDRAM memory devices.
1x (0x2-0x3)= 13 Row lines on SDRAM memory devices (default).

These jumpers are located at the bottom of the SDRAM connector.

REG_SRC: Selects the voltage source for all onboard regulators, including the 3.3Vout, 1.8Vout, and 1.2Vout regulators.
5V= pins 1-2, PCI 5Vdc source for all onboard regulators. Useful if the PCI bus does NOT supply 3.3V.
3V= pins 2-3, 3.3Vdc source for the 1.8Vout regulator (default). The 3.3Vdc itself must itself be extrapolated from the 3V_SRC jumper as described below.

3V_SRC: Selects the source of the 3.3Vdc used by the onboard 3.3Vout and 1.8Vout regulators.
PCI3V= pins 1-2, PCI 3.3Vdc source for the 1.8Vout and 1.2Vout regulators (default). Used if the PCI bus does supply 3.3V.
REG3V= pins 2-3, 3.3Vdc source from the 3.3Vout regulator. Useful if the PCI bus does NOT supply 3.3V.

Truth table for the voltage sources, REG_SRC and 3V_SRC, can be summarized as follows:

REG_SRC/3V_SRC
5V/REG3V = All onboard regulators use the PCI 5Vdc line for their source. Useful if the PCI bus does NOT supply 3.3Vdc.

3V/PCI3V = Only the 1.8Vout and 1.2Vout regulators use the PCI 3.3Vdc line for their source (default). Used if the PCI bus does supply 3.3Vdc.

12.7 Jumpers for PC104Plus Cards

VPP_EN: VPP Enable, for Flash programming enable.
LO= pins 2-3, disables Flash programming.
HI= pins 1-2, enables Flash programming (default). Places 3V onto the VPP pin of the Flash device.

This jumper is located to the right of the PCI connector.

OP_SEL: Operation select.
LO= pins 2-3, slave mode (default). The PCI bus bridge device is activated which allows access the DSP and vice versa. Therefore this is the mode to run if Flash is to be programmed while the card is inside of the host computer. The DSP's RESET line operates normally where it starts up asserted or '0', and then requires the host computer to deassert it as done in normal operation after COFF loading. In the case that the flash is already programmed, the host may simply deassert this line for DSP code execution to begin without the need to COFF load again.
HI= pins 1-2, stand alone mode. Flash is expected to be fully functional, access to the PCI bus is blocked off. The DSP's RESET line automatically deasserts 10 seconds after power is applied, which will force DSP code execution.

This jumper is located to the left of the PCI connector.

BASE: S[1:0]: PC104 Plus base address selector jumpers.
00 (0x0) = Base 0x0, for first PCI card in the stack (default).
01 (0x1)= Base 0x1, for second PCI card in the stack.
10 (0x2)= Base 0x2, for third PCI card in the stack.
11 (0x3)= Base 0x3, for fourth PCI card in the stack.

These jumpers are located in between the PCI connector and the expansion connector, on the corner.

MEMSEL: MS[1:0]: SDRAM row size selector jumpers.
00 (0x0) = 11 Row lines on SDRAM memory devices.
01 (0x1)= 12 Row lines on SDRAM memory devices.
1x (0x2-0x3)= 13 Row lines on SDRAM memory devices (default).

These jumpers are located to the right of the expansion connector, on the corner.

13.0 Technical Specifications

Processor for SI-C671xDSP:

- 300/225Mhz TMS320C6713.
- 16 DSP DMA channels, 2 PCI DMA channels.

Memory for SI-C671xDSP:

- SDRAM program and data memory:

1) Standard 3.3V, non-buffered PC133 SDRAM SO-DIMM format.

2) Sizes: 64MB, 128MB, 256MB and 512MB, with x16 organizations. Only half of capacity is used.

3) PC133 SDRAM module clocked from E Clock running at 75Mhz, maximum delay of memory ICs mounted on module not to exceed 7.5 nsec.

4) Single bank SO-DIMMs: supports *half* of memory ICs mounted on single bank modules, with this capacity only mapped on CE2 (DSP memory region starting at 0xA0000000). The DSP's CE3 region is left empty (DSP memory region starting at 0xB000000).

5) Dual bank SO-DIMMs: supports *half* of memory ICs mounted on dual bank modules, with quarter of module capacity mapped on CE2 (DSP memory region starting at 0xA0000000), and the second quarter mapped on CE3 (DSP memory region starting at 0xB000000). Remaining capacity is ignored.

6) Host accessible while DSP is active, with multiple communication options.

- Boot Memory:

1) 512kx8 SRAM or 1Mx16 Flash.

2) Configured as Dual Access memory: Accessible by host (only while DSP is inactive/reset) for downloading COFF files. Accessed by DSP during its boot loading process.

3) Mapped on CE1 (DSP memory region starting at 0x90000000).

NOTE: *SDRAM modules whose parameters fall within the C67x's supported range will be fully operational. The C67x's internal DRAM controller supports the following parameters:*
1) Columns: 8, 9, 10.
2) Rows: 11, 12, 13.
3) Memory Device Banks: 2, 4. Not to be confused with the module's "bank."
4) Cache Latency: 2, 3.

Interface to Host:

- Up to eight 32 bit, bi-directional communications modes between TMS320C6x and the PCI9054:

1) Host target/slave access mode, combined with DSP I/O or DSP's DMA engine.

2) Block Mode DMA Bus Master mode, using the PCI9054 as the PCI bus master, combined with DSP I/O or DSP's DMA engine.

3) PCI Initiated/Local Master with the DSP as the PCI bus master, with DSP I/O or DSP's DMA engine.

4) PCI Initiated/Local Master with DSP DMA, using the DSP's DMA as the PCI bus master.

- The PCI9054's internal registers are tied onto the C6x's EMIF bus, mapped on CE0 (DSP memory region starting at 0x80000000).
- Four C6x routable interrupts:

1) Up to three C6x interrupts can be routed for basic communication and DMA

synchronization.

2) Up to two C6x interrupts can be routed to expansion connectors.

NOTE: *For PC104p only, the "ISA" or PC104 connector is passive and is only present to allow for stacking of legacy PC104 cards.*

Peripheral Expansion on SI-C671xDSP:

- One external 100 pin half pitch DSUB connector, and two 2mm socket connectors:

a) DSP Expansion: First external 2mm pitch, 5x30 (150 contacts) socket connector for interfacing the expansion board to the DSP's bus, or linking to all of the DSP's peripheral ports (McBSP, McASP, HPI, Timers).

b) User I/O: Second external 2mm pitch, 4x30 (120 contacts) socket connector for interfacing external user defined signals to custom daughter board. Linked only to externally accessible 100 pin half pitch DSUB connector.

c) External 100 pin, half pitch (0.050"), Series III DSUB connector for interfacing external user defined signals to User I/O connector. AMP part 787169-9, 787170-9, or 787362-9; Thomas & Betts part HFR100RA29CS1.

- DSP Expansion 2mm connector decodes 64Kx32 words, mapped into the DSP's EMIF bus, which contains the following signals:

a) Address: A[15-0].

b) Data: D[31:0].

c) Control: X_R/Wn, X_CSn, X_INT[1:0] (software routable to DSP's EXTINT[7:4]), X_RDY, X_CLK[1:0].

d) Peripheral Port: McBSP[1:0], HPI/GPIO, TINP[1:0], TOUT[1:0].

e) Host +3.3Vdc, +5Vdc, -5Vdc, +12Vdc, -12Vdc & GND.

- One 50 pin header (2x25): McBSP1, TINP[1:0], TOUT[1:0].
- One 10 pin header (2x5): McBSP0.
- One 12 pin header for buffered JTAG port.

Software:

- Win98/2000/XP and Linux driver support.
- Extensive QuVIEW DSP-resident libraries for LabVIEW, including examples for real time acquisition, signal processing, and control.
- Extensive QuBASE DSP-resident libraries for Visual Basic, including examples for real time acquisition, signal processing, and control.
- Sample code for COFF loaders, PC <-> DSP communications source code and SI-DDK.
- Compatible with separately purchased TI debuggers, C/C++ compilers, DSP Bios, assemblers and linkers.

Physical Dimensions & Electrical Requirements:

- SI-C6xDSP-PCI: 3/4 size PCI-bus card measuring 9"(L) x 4.9"(H) or 23cm(L) x 12.5cm(H); 0.31lbs or 140 grams.
- SI-C6xDSP-cPCI/PXI: 3U size CompactPCI/PXI bus card measuring 160mm(L) x 100mm(H); 0.31lbs or 140 grams.
- SI-C6xDSP-PC104p: PC104 size card measuring 3.5"(L) x 3.775"(H); 0.18lbs or 85 grams.
- SI-C6xDSP-PMC: PMC size card measuring 5.86"(L) x 2.9"(H) or 149cm(L) x 74cm(H);

0.18lbs or 85 grams.

- Supply Voltages: 3.3V for all circuitry, and 5V for expansion bus buffers; 3V expansion buffers may be placed on special request. +/-12V supplies passed on to expansion connector and not used by onboard circuitry.
- 4.5 watts (3.3V @ 1.5A) typical with 128MB SDRAM.