SHELDON INSTRUMENTS

Getting Started

August 2006

Table of Contents

1.0 Getting Started

The following steps must be taken in order to ensure proper operation of all SI hardware and software:

1) Install hardware.
2) Install drivers.
3) Install FPGA loader.
4) Install application software.
    a) For turnkey application development using LabVIEW, install QuVIEW.
    b) For turnkey application development using Visual Basic, install QuBASE.
    c) For custom application development using CCS for the DSP and VS/Linux for the host, install SI development kit.

NOTE: *When installing from from a CD, be sure to remove the 'read only' attribute from all files after copying to your hard drive.*

1.1 Detailed Information on SI Hardware

Please refer to the following paths:

File Path	Comments
\si_support\Docs\SIDSP*	Detailed information about SI-DSP carrier cards.
\si_support\Docs\SIMOD*	Detailed information about SI-MOD analog I/O modules.
\si_support\Docs\SITerm*	Detailed information about SI Terminal cards and accessories.

2.0 Hardware and Driver Installation

All drivers for SI hardware are exclusively delivered on a CD or emailed upon request, all other SI software is available for download at our site.

Please refer to the following paths:

File Path	Comments
\sidev\binaries\drivers*	kernel mode drivers
\si_support\Docs\Drivers\readme-drivers.rtf	Notes on driver installation.

3.0 Loading FPGAs

Please refer to the following paths:

File Path	Comments
Turnkey development with QuX:	
\QuX\SIC30dsp*	All files pertinent for loading FPGA, default.
Custom code development:	
\sidev\binaries\fpgaloader*	All files pertinent for loading FPGA.
\si_support\Docs\SIDSP\FPGALoad\FPGALoad.rtf	Notes on FPGA Loader utility.

NOTE: *By default, the path of the FPGA Loader batch files is \SIC30DSP. Refer to documentation about alterations in the batch file.*

4.0 QuX Installation for Turnkey Application Development

4.1 QuVIEW Installation for LabVIEW

Please refer to the following paths:

File Path	Comments
\QuX\QuVIEW\LabVIEWx*	QuVIEW libraries to be installed for LabVIEWx.
\QuX\QuVIEW\Sheldon_LVx	QuVIEW examples for a particular version of LabVIEW.
\QuX\SIC30dsp*	All files pertinent for loading FPGA and QuX system files.
\si_support\Docs\QuX\QuVIEW.pdf	Notes on QuVIEW installation and function reference list.
\si_support\Docs\QuX*	Notes on QuX examples and tutorials.

4.2 QuBASE Installation for Visual Studio

Please refer to the following paths:

File Path	Comments
\QuX\QuBASE\QuBase_6\Binaries*	Standalone executables created with VB6.
\QuX\QuBASE\QuBase_6\QuBASE_Lib	QuBASE for VB6 libraries with source code.
\QuX\QuBASE\QuBase_6\demos*	QuBASE for VB6 examples with source code.
\QuX\QuBASE\QuBase_6\Tests*	QuBASE for VB6 simple tests and templates..
\QuX\QuBASE\QuBase_6\tutors*	QuBASE for VB6 tutors with source code.
\QuX\QuBASE\QuBase_net\QuBASE_Lib	QuBASE for VB.net libraries with source code.
\QuX\QuBASE\QuBase_net\demos*	QuBASE for VB.net examples with source code.
\QuX\QuBASE\QuBase_net\Tests*	QuBASE for VB.net simple tests and templates..
\QuX\QuBASE\QuBase_net\tutors*	QuBASE for VB.net tutors with source code.
\QuX\SIC30dsp*	All files pertinent for loading FPGA and QuX system files.
\si_support\Docs\QuX\QuVIEW.pdf	Reference on function list, same functionality as QuVIEW functions.
\si_support\Docs\QuX*	Notes on QuX examples and tutorials.

5.0 SI Development Kit Installation for Custom Application Development

Please refer to the following paths:

File Path	Comments
\sidev\binaries*	Completed executables and drivers for all SI hardware.
\sidev\siddk*	All host level source code for Windows and Linux.
\sidev\siddk\siddk_plx\docs	Documentation for kernel mode drivers. for Windows and Linux.
\sidev\sidsp*	All DSP level source code.
\si_support\Docs\sidev\sidev_dir_structure.txt	Detailed description of SI Development folder tree.

5.1 SI-DSP Utility Software Description

The SI-DSP software utilities can be found in binary form inside of the '\sidev\binaries\sihw_apps' subfolder. These software utilities are intended as a guide and foundation for your own custom development, and are organized as follows:

1) SISAMPLE.EXE
A basic yet comprehensive command line utility devised to integrate all SI-DSP functionality, most importantly all host-DSP communications. The menus are divided into a pair of sections, with the upper section pertaining to basic PCI bridge device functionality and the lower section pertaining to the DSP's functionality.

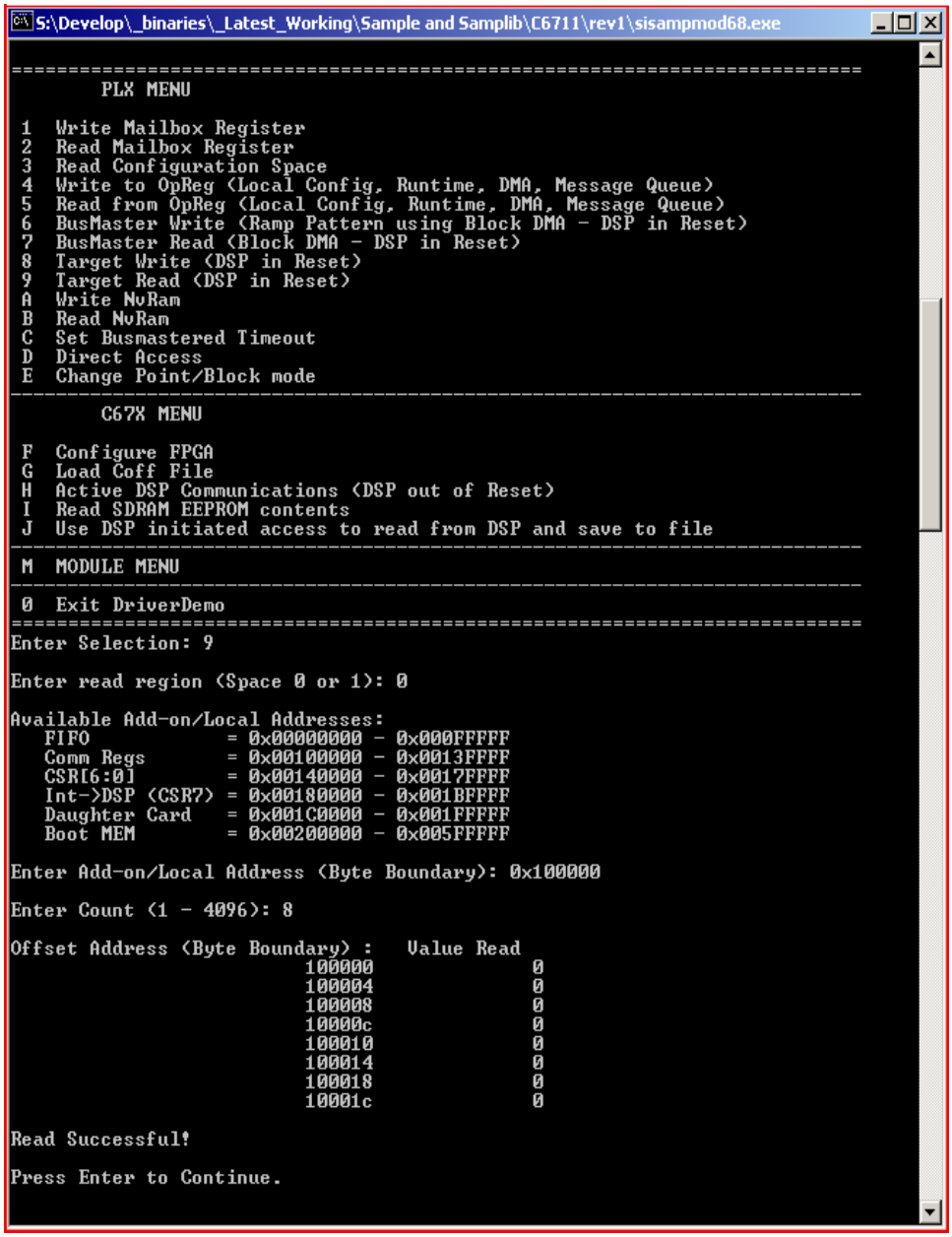
2) SISAMPLIB.DLL
The same basic and comprehensive utility but supplied in DLL form, with all functions callable as a shared library.

3) SISAMPMOD68.EXE
The same basic and comprehensive command line utility, but with an added third menu section supporting the SI-MOD68xx expansion I/O module.

In order to test the SI-DSP and SI-MOD hardware together, we will describe basic diagnostics using the more complete SISAMPMOD68.EXE utility. The exact same steps are applicable to the SISAMPLE.EXE utility if only the SI-DSP card is of interest.

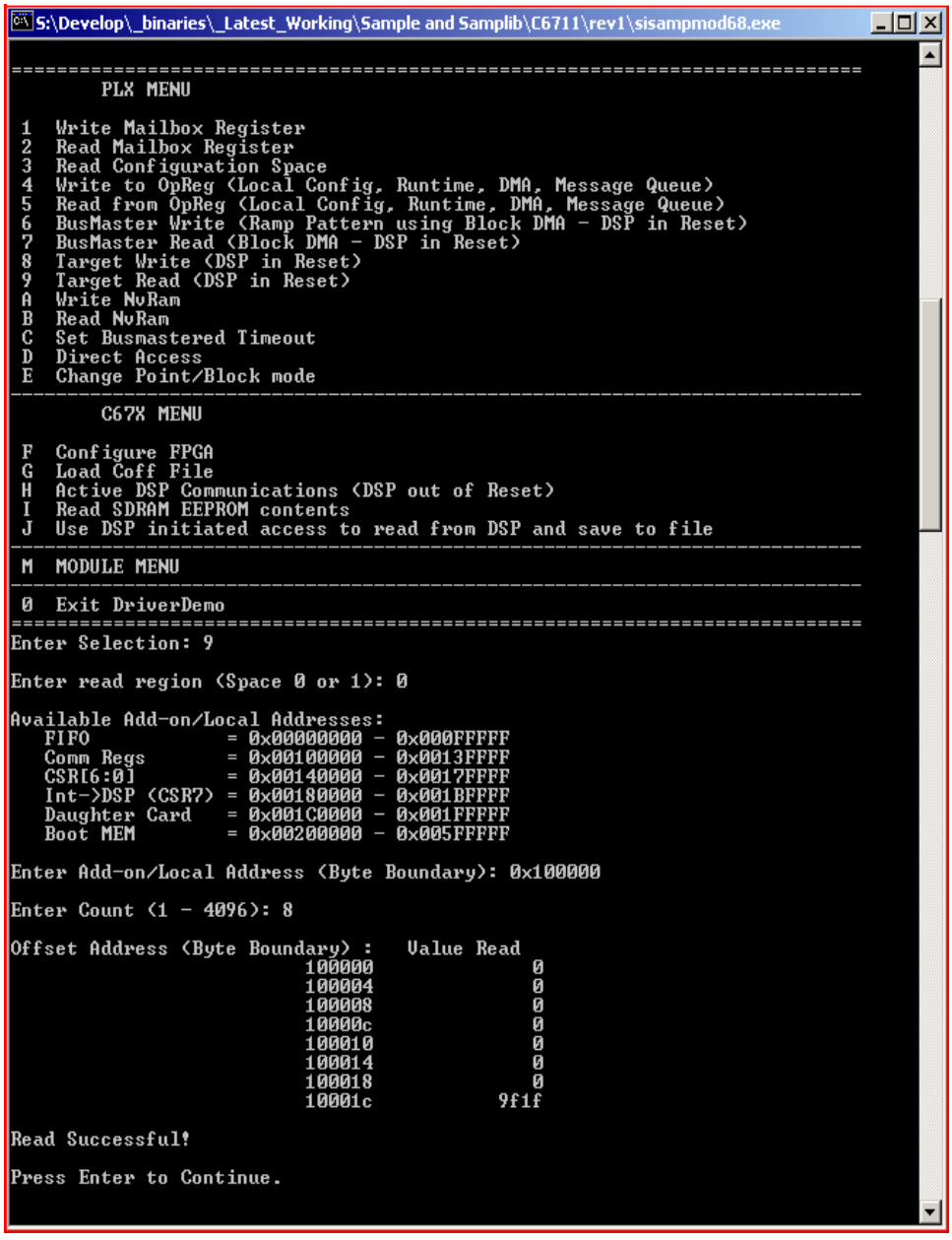
Basic target accesses from the host and DSP initialization using SISAMPLE.EXE or SISAMPMOD68.EXE:

1. After the FPGA loader has completed, invoke the SISAMPLE or SISAMPMOD68 utility.
2. Perform a target read of CommRegs by selecting menu option ”9”, Space “0”, address “100000”, and count of “8” as indicated in figure 1a. Note that if a second instance of a target read of these eight CommRegs is performed, the values remain static or zero which indicates that the DSP is inactive.



3. COFF load the DSP by selecting menu option “g” as indicated in figure 2. *Note that the entire correct path of the COFF file must be entered*. If the COFF load is successful, you should see a summary of the SDRAM parameters listed on the screen.

4. Repeat step 2 of performing a target read of 8 CommRegs. After performing a few more instances of a target read of the CommRegs, now notice that the eighth CommReg constantly changes, which indicates the DSP is actively running code and is inserting a heartbeat at this eighth CommReg location as shown in figure 1b. If the eighth CommReg value remains constant (normally zero immediately after an FPGALoad), it indicates that the DSP is inactive which may be due to one of the previous steps not being done correctly. It is recommended to repeat these first four steps until the heartbeat appears.



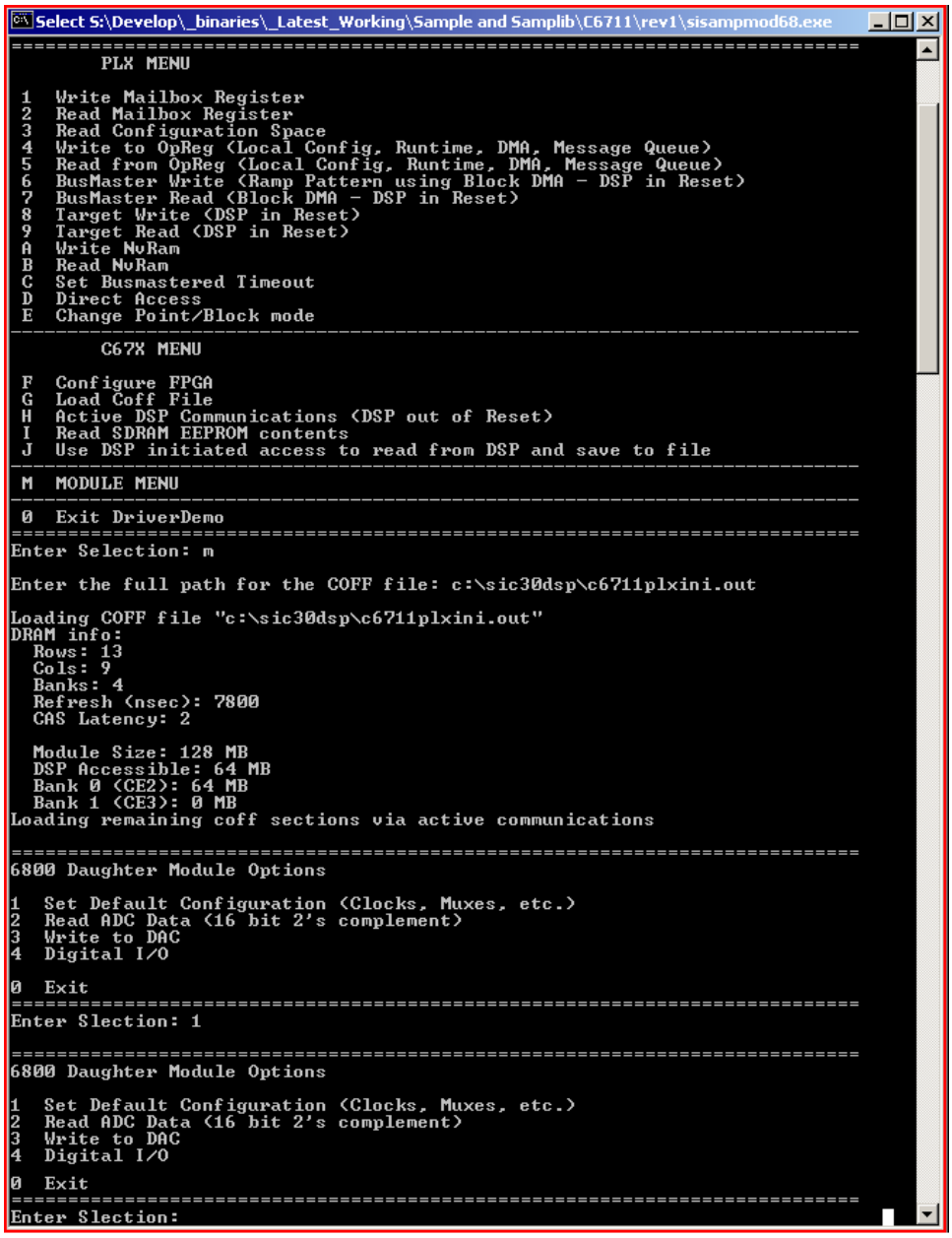
Now that the DSP has been verified to be properly activated, we can perform basic communications actively involving the DSP to perform the transfers with the host using the SISAMPLE.EXE or SISAMPMOD68.EXE:

5. Select menu option “h”, Data Trnasfers “0”, Mode “4”, Read from DSP “0”, DSP address “80300000”, and count of “8” as shown in figure 3. Just like in step 4, after performing a few more instances of an active DSP read of the CommRegs, now notice the heartbeat in the eighth CommReg.

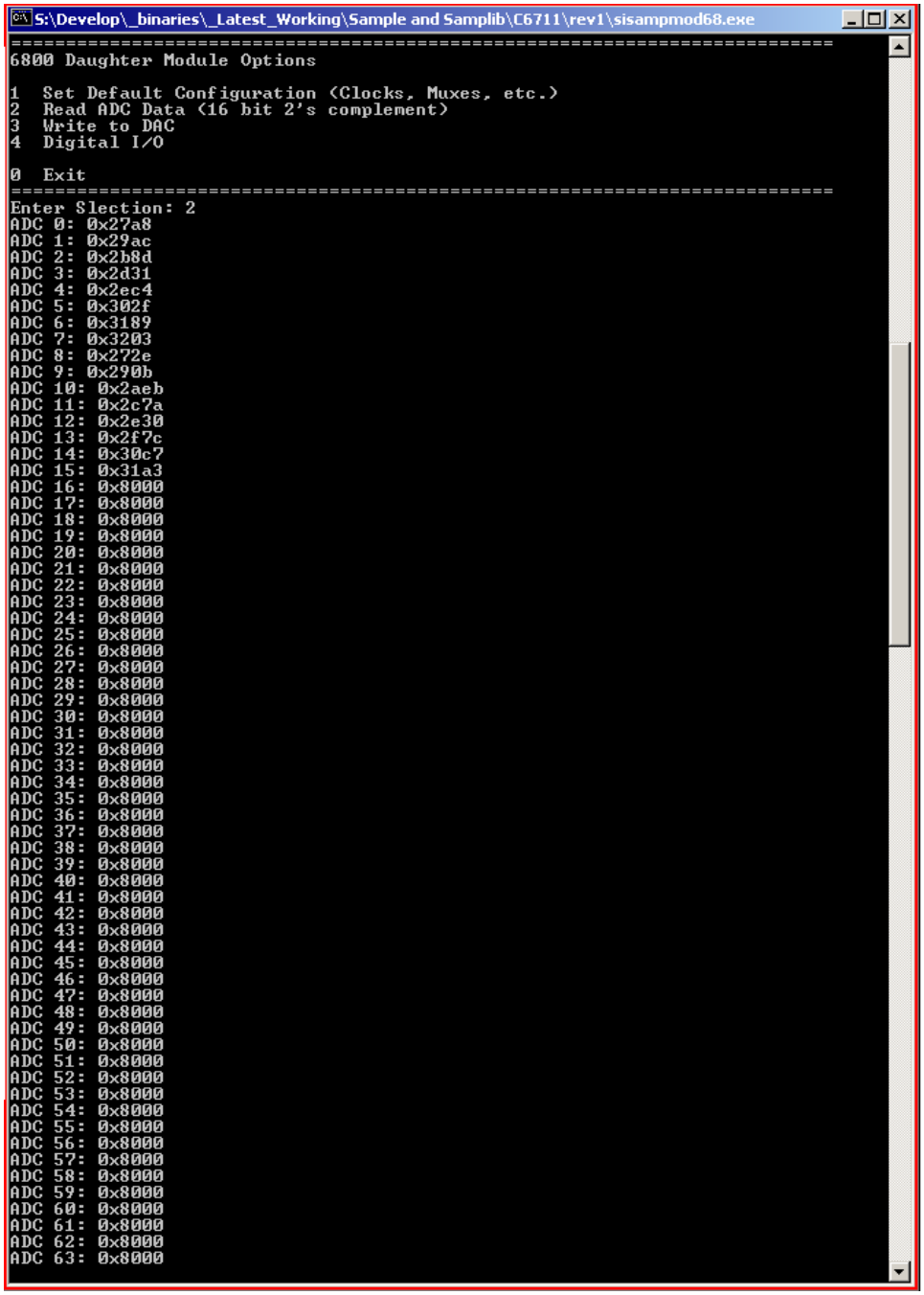
Only the SISAMPMOD68.EXE will have the third menu section present where basic diagnostics can be performed to check the SI-MOD expansion I/O module:

6. Select menu option “m” in order activate the entire third menu section for the SI-MOD as shown in figure 4. Note that you will be forced to perform another DSP COFF load in order to guarantee DSP activation.

7. Select menu option “1” in order to initialize the SI-MOD with the default configuration settings, as shown in figure 5.



8. Now that the SI-MOD has been initialized, you can read the ADC values expressed as signed integers for all available inputs by selecting menu option “2”, as in figure 6. Note that the DAC outputs may also be updated by selecting menu option “3” and verified with an oscilloscope.



5.2 Notes on Compiling Host Side Applications

All source code and related projects for creating host side applications reside inside of the \SIDEV\SIDDK subfolder. The project files for the various applications supporting all SI hardware reside inside of the '\sidev\siddk\siddk_plx\apps\sihw_plx\Rev1' subfolder, where each card type has its own subfolder. The source code and related project files for the kernel mode drivers reside inside of the '\sidev\siddk\siddk_plx\drivers' subfolder.

The Windows projects are targeted for use with Visual C version 6. Updating to later MS environments is a relatively straightforward process, and hence is left up to the user to create or update.

The Linux makefiles are targeted for Fedora 4 or later, and Linux kernel 2.6.x.

In general, three utility projects for generating binaries are available:

1) SISAMPLE.EXE, inside of the '\sidev\siddk\siddk_plx\apps\sihw_plx\Rev1\c6711pci\sisample' subfolder. A basic yet comprehensive command line utility devised to integrate all SI-DSP functionality, most importantly all host-DSP communications. The menus are divided into a pair of sections, with the upper section pertaining to basic PCI bridge device functionality and the lower section pertaining to the DSP's functionality.
2) SISAMPLIB.DLL, inside of the '\sidev\siddk\siddk_plx\apps\sihw_plx\Rev1\c6711pci\sisamplib' subfolder. The same basic and comprehensive utility but supplied in DLL form, with all functions callable as a shared library.
3) SISAMPMOD68.EXE, inside of the '\sidev\siddk\siddk_plx\apps\sihw_plx\Rev1\c6711pci\sisampmod\sisampmod68' subfolder. The same basic and comprehensive command line utility, but with an added third menu section supporting the SI-MOD68xx expansion I/O module.

5.3 Notes on Compiling DSP Side COFF Files

All source code and related projects for creating DSP side applications reside inside of the '\sidev\sidsp' subfolder.

The source code and related project files for the C33 based hardware reside inside of the '\sidev\sidsp\basiccom\plxc33' subfolder, and are supplied to be compiled using TI's DOS based tools or Code Composer Studio version 4.x.

The source code and related project files for the C67x based hardware reside inside of the '\sidev\sidsp\basiccom\plxc6711\Rev1' subfolder, and are supplied to be compiled using TI's Code Composer Studio version 3.1 or later.

NOTE: *By default, Code Composer Studio is set to compile projects in 'debug' mode. However, in order to avoid compilation errors when building C67x projects, the active configuration must be set to 'release' mode every time Code Composer is opened.*

In general, five projects for generating COFF files are available:

1) C6711PLXINI.OUT, inside of the '\sidev\sidsp\basiccom\plxc6711\Rev1\coff_x1' subfolder. A basic yet comprehensive COFF file that includes support for all host-DSP communications. The "x1" refers to the host downloading the entire COFF image from host memory to the DSP bootMEM in a single step while the DSP is inactive, and then activating the DSP.
2) COFF_x2.OUT, inside of the '\sidev\sidsp\basiccom\plxc6711\Rev1\coff_x2' subfolder. A basic yet comprehensive COFF file that includes support for all host-DSP communications. The "x2" refers to the host downloading only a portion of the COFF image from host memory to the DSP bootMEM while the DSP is inactive, and then downloading the remaining portion of the COFF file after the DSP is actively running communications.
3) COFFBIOS_x1.OUT, inside of the '\sidev\sidsp\basiccom\plxc6711\Rev1\coffbios_x1' subfolder. Same as the COFF file example inside of the 'coff_x1' subfolder, but instead uses DSP Bios library calls instead of straight C for DSP configuration.
4) COFFBIOS_x2.OUT, inside of the '\sidev\sidsp\basiccom\plxc6711\Rev1\coffbios_x2' subfolder. Same as the COFF file example inside of the 'coff_x2' subfolder, but instead uses DSP Bios library calls instead of straight C for DSP configuration.
5) COFFJTAG_x1.OUT, inside of the '\sidev\sidsp\basiccom\plxc6711\Rev1\coffjtag_x1' subfolder. Same as the COFF file example inside of the 'coff_x1' subfolder, but with copy sections removed in order allow COFF files to be downloaded to the DSP via the JTAG port.