SHELDON INSTRUMENTS

Getting Started

March 2013

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) Load FPGA bitfile (using SISAMPLE utility).
4) Install application software.
    a) For custom application development using CCS for the DSP and VS/Linux for the host, install SI development kit.
    b) For turnkey application development using LabVIEW, install QuVIEW.
    c) For turnkey application development using Visual Basic, install QuBASE.

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\SI\32bit\drivers*	32 bit Windows WDF drivers
\sidev\binaries\SI\64bit\drivers*	signed 64 bit Windows WDF drivers
\si_support\Docs\Drivers\readme-drivers.rtf	Notes on driver installation.

3.0 Loading FPGA bitfiles

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\SI\32bit\apps*	All files pertinent for loading FPGA bitfiles under 32 bit Windows.
\sidev\\binaries\SI\64bit\apps*	All files pertinent for loading FPGA bitfiles under 64 bit Windows.
\si_support\Docs\SIDSP\FPGALoad\FPGALoad.rtf	Notes on SISAMPLE's FPGA Bitloader functionality.

NOTE: *By default, it is assumed that the path of the FPGA bitloader 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 32/64 bit executables and drivers for all SI hardware.
\sidev\siddk*	All host projects and related source code for Windows and Linux.
\sidev\siddk\docs*	Documentation for kernel mode drivers. for Windows and Linux.
\sidev\sidsp*	All DSP/CCS projects and related 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 32 bit software utilities can be found in binary form inside of the '\sidev\binaries\SI\32bit\apps ‘ subfolder. These software utilities are intended as a guide and foundation for your own custom development, and are organized as follows:

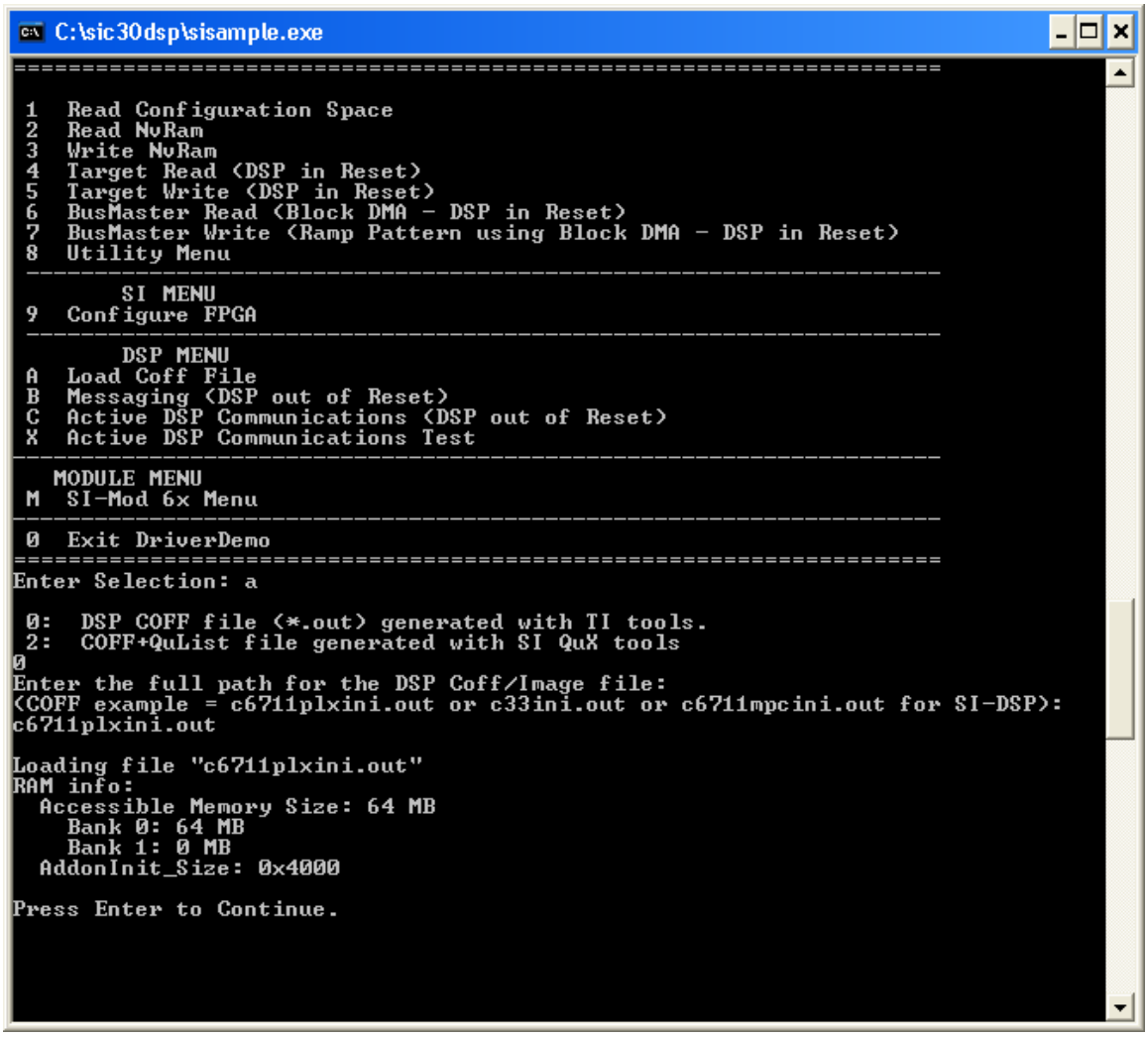
1) SISAMPLE.EXE
A basic yet comprehensive command line utility devised to integrate all SI-DSP functionality, most importantly all host-DSP communications, as well as the SI-MOD expansion I?O modules. The menus are divided into three sections, with the upper section pertaining to basic PCI bridge device functionality, the middle section pertaining to the DSP's functionality, and the bottom pertaining to the SI-MOD’s functionality.

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

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

1. After the FPGA bitloading has completed, invoke the SISAMPLE utility.
2. Perform a target read of CommRegs by selecting menu option ”4”, option “1” for Space “0”, count of “8”, and address “100000” 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 “a” as indicated in figure 2. If the COFF load is successful, you should see a summary of the SDRAM parameters listed on the screen.



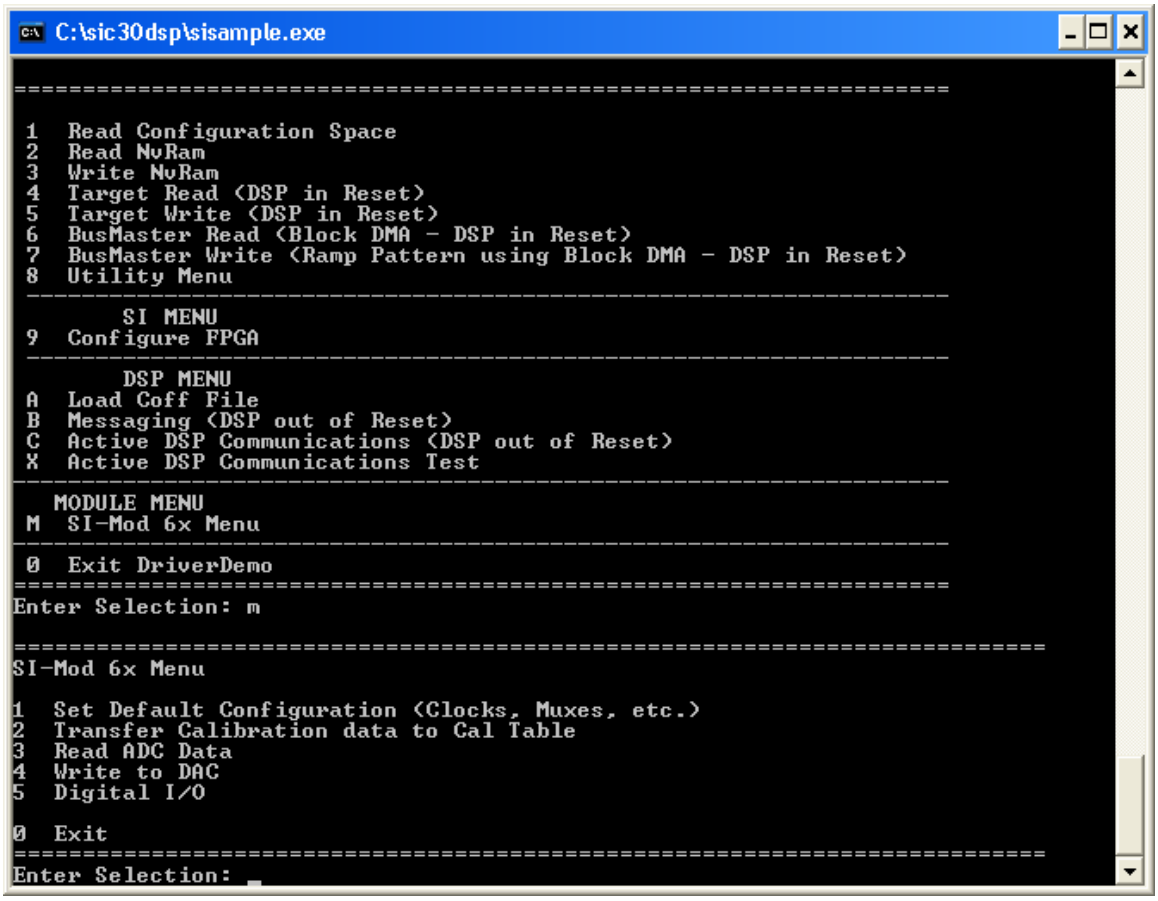
4. Repeat step 2 of performing a target read of 8 CommRegs. After performing a few more instances of a target read of the CommRegs, now notice that the eighth CommReg constantly changes, which indicates the DSP is actively running code and is inserting a heartbeat at this eighth CommReg location as shown in figure 1b. If the eighth CommReg value remains constant (normally zero immediately after an FPGA bitload), 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 running, we can perform basic communications actively involving the DSP to perform the transfers with the host using SISAMPLE.EXE:

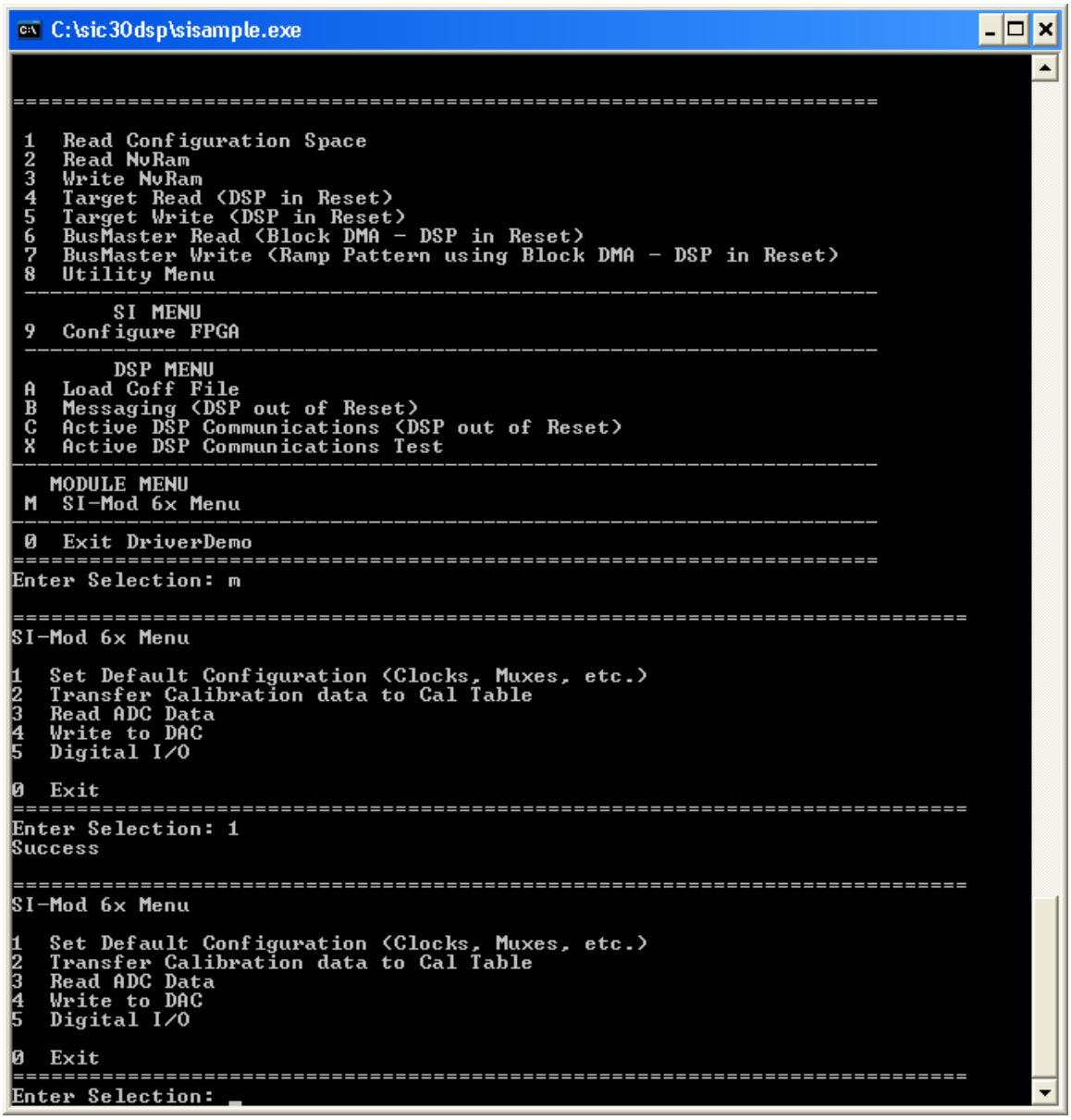
5. Select menu option “c”, Read from DSP “0”, Mode “4”, DSP address “80300000”, and count of “8” as shown in figure 3. Just like in step 4, after performing a few more instances of an active DSP read of the CommRegs, now notice the heartbeat in the eighth CommReg.

If the SI-MOD expansion I/O module is present, we can invoke the third menu section to further perform basic diagnostics:

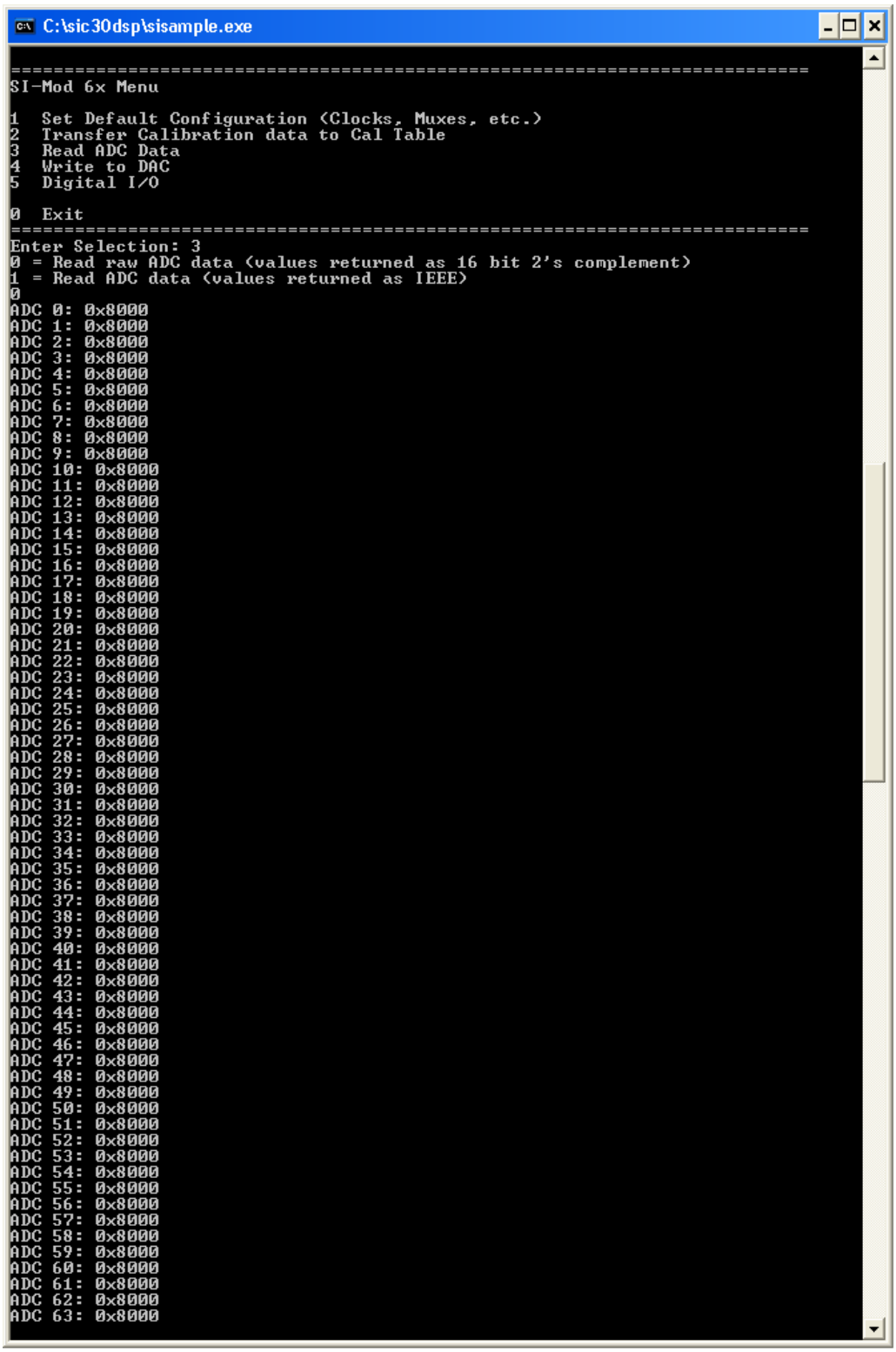
6. Select menu option “m” in order activate the entire third menu section for the SI-MOD as shown in figure 4. Note that you will be forced to perform DSP COFF load in order to guarantee DSP activation if it has not been previously done.



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 “3” and “0”, as in figure 6. Note that the DAC outputs may also be updated by selecting menu option “4” and verified with an oscilloscope.



5.2 Notes on Compiling Host Side Applications

All projects and related source code 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\apps*' subfolder, where each card type has its own subfolder. The projects and related source code for the kernel mode drivers reside inside of the '\sidev\siddk\siddk_plx\drivers' subfolder.

The Windows projects are targeted for use with Visual Studio 2010 or later. 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 in these paths:

\sidev\siddk\apps\sisample
The Windows and Linux project files for the SISAMPLE command line utility supporting all SI hardware

\sidev\siddk\apps\sisamplib
The Windows and Linux project files for the SISAMPLIB DLL utility supporting all SI hardware

\sidev\siddk\apps\dll_loader
The Windows and Linux project files that demonstrate the method for calling the SISAMPLIB.DLL.

5.3 Notes on Compiling DSP Side COFF Files

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

\sidev\sidsp\basiccom\plxc6711\Rev1\coff_x1
Basic project incorporating all DSP-Host communication modes. 'x1' refers to a single stage COFF loading process.

\sidev\sidsp\basiccom\plxc6711\Rev1\coffbios_x1
Functionally equivalent to 'coff_x1' project, where 'bios' refers to the use of DSP/Bios which replaces the ISR and memory directives in the *.cmd file.

\sidev\sidsp\basiccom\plxc6711\Rev1\coffbios_x2
Functionally equivalent to 'coffbios_x1' project, where 'x2refers to a two stage COFF loading process in the case the COFF file size exceeds 500KBx8 in size.

\sidev\sidsp\basiccom\plxc6711\Rev1\coffmod6x_x1
Functionally equivalent to 'coff_x1' project, with added funtionality for the DSP to exclusively configure the SI-MOD expansion cards.

***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.*