

SHELDON INSTRUMENTS

Getting Started

August 2006

Table of Contents

1.0 Getting Started.....	3
1.1 Detailed Information on SI Hardware.....	3
2.0 Hardware and Driver Installation.....	4
3.0 Loading FPGAs.....	5
4.0 QuX Installation for Turnkey Application Development.....	6
4.1 QuVIEW Installation for LabVIEW.....	6
4.2 QuBASE Installation for Visual Studio.....	6
5.0 SI Development Kit Installation for Custom Application Development.....	7
5.1 SI-DSP Utility Software Description.....	8
5.2 Notes on Compiling Host Side Applications.....	16
5.3 Notes on Compiling DSP Side COFF Files.....	17

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\fploader*	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.

```
S:\Develop\_binaries\_Latest_Working\Sample and Samplib\C6711\rev1\sisampmod68.exe

=====
          PLX MENU
=====
1 Write Mailbox Register
2 Read Mailbox Register
3 Read Configuration Space
4 Write to OpReg (Local Config, Runtime, DMA, Message Queue)
5 Read from OpReg (Local Config, Runtime, DMA, Message Queue)
6 BusMaster Write (Ramp Pattern using Block DMA - DSP in Reset)
7 BusMaster Read (Block DMA - DSP in Reset)
8 Target Write (DSP in Reset)
9 Target Read (DSP in Reset)
A Write NuRam
B Read NuRam
C Set Busmastered Timeout
D Direct Access
E Change Point/Block mode

=====
          C67X MENU
=====
F Configure FPGA
G Load Coff File
H Active DSP Communications (DSP out of Reset)
I Read SDRAM EEPROM contents
J Use DSP initiated access to read from DSP and save to file

=====
          M MODULE MENU
=====
0 Exit DriverDemo
=====
Enter Selection: 9

Enter read region (Space 0 or 1): 0

Available Add-on/Local Addresses:
FIFO      = 0x00000000 - 0x0000FFFF
Comm Regs  = 0x00100000 - 0x0013FFFF
CSR16:01   = 0x00140000 - 0x0017FFFF
Int->DSP (CSR7) = 0x00180000 - 0x001BFFFF
Daughter Card = 0x001C0000 - 0x001FFFFF
Boot MEM   = 0x00200000 - 0x005FFFFF

Enter Add-on/Local Address (Byte Boundary): 0x100000
Enter Count (1 - 4096): 8

Offset Address (Byte Boundary) :      Value Read
100000                          0
100004                          0
100008                          0
10000c                          0
100010                          0
100014                          0
100018                          0
10001c                          0

Read Successful!
Press Enter to Continue.
```

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.

```
S:\Develop\_binaries\_Latest_Working\Sample and Samplib\C6711\rev1\sisampmod68.exe

SI-DDK PLX Sample Application 4.0
Sheldon Instruments, Inc. 2002
Supporting the PLX 90xx PCI chip
Using Device Number 0

NOTE: Remember to use option F to load FPGA
before continuing with the rest of the functions
Press Enter to Continue.

=====
          PLX MENU
=====
1 Write Mailbox Register
2 Read Mailbox Register
3 Read Configuration Space
4 Write to OpReg (Local Config, Runtime, DMA, Message Queue)
5 Read from OpReg (Local Config, Runtime, DMA, Message Queue)
6 BusMaster Write (Ramp Pattern using Block DMA - DSP in Reset)
7 BusMaster Read (Block DMA - DSP in Reset)
8 Target Write (DSP in Reset)
9 Target Read (DSP in Reset)
A Write NuRam
B Read NuRam
C Set Busmastered Timeout
D Direct Access
E Change Point/Block mode

-----
          C67X MENU
-----
F Configure FPGA
G Load Coff File
H Active DSP Communications (DSP out of Reset)
I Read SDRAM EEPROM contents
J Use DSP initiated access to read from DSP and save to file

-----
          M  MODULE MENU
-----
0 Exit DriverDemo
=====
Enter Selection: g

Enter the full path for the COFF file: c:\sic30dsp\c6711plxini.out
Loading COFF file "c:\sic30dsp\c6711plxini.out"
DRAM info:
  Rows: 13
  Cols: 9
  Banks: 4
  Refresh (nsec): 7800
  CAS Latency: 2

  Module Size: 128 MB
  DSP Accessible: 64 MB
  Bank 0 (CE2): 64 MB
  Bank 1 (CE3): 0 MB
Loading remaining coff sections via active communications
Press Enter to Continue.
```

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 Load), 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.

```
S:\Develop\binaries\Latest_Working\Sample and Samplib\C6711\rev1\sisampmod68.exe

=====
          PLX MENU
=====
1 Write Mailbox Register
2 Read Mailbox Register
3 Read Configuration Space
4 Write to OpReg (Local Config, Runtime, DMA, Message Queue)
5 Read from OpReg (Local Config, Runtime, DMA, Message Queue)
6 BusMaster Write (Ramp Pattern using Block DMA - DSP in Reset)
7 BusMaster Read (Block DMA - DSP in Reset)
8 Target Write (DSP in Reset)
9 Target Read (DSP in Reset)
A Write NuRam
B Read NuRam
C Set Busmastered Timeout
D Direct Access
E Change Point/Block mode
=====

          C67X MENU
=====
F Configure FPGA
G Load Coff File
H Active DSP Communications (DSP out of Reset)
I Read SDRAM EEPROM contents
J Use DSP initiated access to read from DSP and save to file
=====

M MODULE MENU
=====
0 Exit DriverDemo
=====
Enter Selection: 9

Enter read region (Space 0 or 1): 0

Available Add-on/Local Addresses:
FIFO           = 0x00000000 - 0x0000FFFF
Comm Regs      = 0x00100000 - 0x0013FFFF
CSR16:01       = 0x00140000 - 0x0017FFFF
Int->DSP (CSR7) = 0x00180000 - 0x001BFFFF
Daughter Card  = 0x001C0000 - 0x001FFFFF
Boot MEM       = 0x00200000 - 0x005FFFFF

Enter Add-on/Local Address (Byte Boundary): 0x100000

Enter Count (1 - 4096): 8

Offset Address (Byte Boundary) :   Value Read
100000                          0
100004                          0
100008                          0
10000c                          0
100010                          0
100014                          0
100018                          0
10001c                          9f1f

Read Successful!
Press Enter to Continue.
```

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 Trnsfers “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.

```

S:\Develop\_binaries\_Latest_Working\Sample and Samplib\C6711\rev1\sisampmod68.exe
=====
PLX MENU
1 Write Mailbox Register
2 Read Mailbox Register
3 Read Configuration Space
4 Write to OpReg <Local Config, Runtime, DMA, Message Queue>
5 Read from OpReg <Local Config, Runtime, DMA, Message Queue>
6 BusMaster Write <Ramp Pattern using Block DMA - DSP in Reset>
7 BusMaster Read <Block DMA - DSP in Reset>
8 Target Write <DSP in Reset>
9 Target Read <DSP in Reset>
A Write NoRam
B Read NoRam
C Set Busmastered Timeout
D Direct Access
E Change Point/Block mode
=====
C67X MENU
F Configure FPGA
G Load Coff File
H Active DSP Communications <DSP out of Reset>
I Read SDRAM EEPROM contents
J Use DSP initiated access to read from DSP and save to file
=====
M MODULE MENU
=====
0 Exit DriverDemo
=====
Enter Selection: h
=====
0 = Data Transfers
1 = Messaging
0
Following modes are available:
0 = Add-on: DSP Async IO - Host Int.
1 = Add-on: DSP Async DMA - Host Int.
2 = Target: DSP Async IO FIFO
3 = Block DMA : DSP DMA FIFO
4 = Target: DSP Sync IO CommReg
5 = Target: FPGA Direct Access SDRAM
6 = Block DMA: FPGA Direct Access SDRAM

4
0 = Read From DSP
1 = Write To DSP
0
Available DSP Addresses:
Comm Regs = 0x80300000 - 0x8033FFFF
PLX OpRegs = 0x80340000 - 0x8037FFFF
Daughter Card = 0x80380000 - 0x803BFFFF
Boot MEM <CE1> = 0x90000000 - 0x9FFFFFFF
SDRAM0 <CE2> = 0xA0000000 - 0xA7FFFFFF
SDRAM1 <CE3> = 0xB0000000 - 0xB7FFFFFF
Enter DSP Address: 0x80300000
count:8
0x0
0x8
0x80300000
0x80300014
0x0
0x0
0x2
0x28def
Press Enter to Continue.

```

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.

```
S:\Develop\_binaries\_Latest_Working\Sample and Samplib\C6711\rev1\sisampmod68.exe

=====
PLX MENU
1 Write Mailbox Register
2 Read Mailbox Register
3 Read Configuration Space
4 Write to OpReg (Local Config, Runtime, DMA, Message Queue)
5 Read from OpReg (Local Config, Runtime, DMA, Message Queue)
6 BusMaster Write (Ramp Pattern using Block DMA - DSP in Reset)
7 BusMaster Read (Block DMA - DSP in Reset)
8 Target Write (DSP in Reset)
9 Target Read (DSP in Reset)
A Write NuRam
B Read NuRam
C Set Busmastered Timeout
D Direct Access
E Change Point/Block mode
=====

C67X MENU
F Configure FPGA
G Load Coff File
H Active DSP Communications (DSP out of Reset)
I Read SDRAM EEPROM contents
J Use DSP initiated access to read from DSP and save to file
=====

M MODULE MENU
=====
0 Exit DriverDemo
=====
Enter Selection: m

Enter the full path for the COFF file: c:\sic30dsp\c6711plxini.out

Loading COFF file "c:\sic30dsp\c6711plxini.out"
DRAM info:
  Rows: 13
  Cols: 9
  Banks: 4
  Refresh (nsec): 7800
  CAS Latency: 2

  Module Size: 128 MB
  DSP Accessible: 64 MB
  Bank 0 (CE2): 64 MB
  Bank 1 (CE3): 0 MB
Loading remaining coff sections via active communications
=====

6800 Daughter Module Options
1 Set Default Configuration (Clocks, Muxes, etc.)
2 Read ADC Data (16 bit 2's complement)
3 Write to DAC
4 Digital I/O

0 Exit
=====
Enter Selection:
```

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

```
Select S:\Develop\_binaries\_Latest_Working\Sample and Samplib\C6711\rev1\sisampmod68.exe

=====
PLX MENU
=====
1 Write Mailbox Register
2 Read Mailbox Register
3 Read Configuration Space
4 Write to OpReg <Local Config, Runtime, DMA, Message Queue>
5 Read from OpReg <Local Config, Runtime, DMA, Message Queue>
6 BusMaster Write <Ramp Pattern using Block DMA - DSP in Reset>
7 BusMaster Read <Block DMA - DSP in Reset>
8 Target Write <DSP in Reset>
9 Target Read <DSP in Reset>
A Write NoRam
B Read NoRam
C Set Busmastered Timeout
D Direct Access
E Change Point/Block mode

=====
C67X MENU
=====
F Configure FPGA
G Load Coff File
H Active DSP Communications <DSP out of Reset>
I Read SDRAM EEPROM contents
J Use DSP initiated access to read from DSP and save to file

=====
M MODULE MENU
=====
0 Exit DriverDemo
=====
Enter Selection: m

Enter the full path for the COFF file: c:\sic30dsp\c6711plxini.out
Loading COFF file "c:\sic30dsp\c6711plxini.out"
DRAM info:
  Rows: 13
  Cols: 9
  Banks: 4
  Refresh (nsec): 7800
  CAS Latency: 2

  Module Size: 128 MB
  DSP Accessible: 64 MB
  Bank 0 <CE2>: 64 MB
  Bank 1 <CE3>: 0 MB
Loading remaining coff sections via active communications

=====
6800 Daughter Module Options
=====
1 Set Default Configuration <Clocks, Muxes, etc.>
2 Read ADC Data <16 bit 2's complement>
3 Write to DAC
4 Digital I/O
0 Exit
=====
Enter Selection: 1

=====
6800 Daughter Module Options
=====
1 Set Default Configuration <Clocks, Muxes, etc.>
2 Read ADC Data <16 bit 2's complement>
3 Write to DAC
4 Digital I/O
0 Exit
=====
Enter Selection:
```

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.

```
S:\Develop\binaries\Latest_Working\Sample and Samplib\C6711\rev1\sisampmod68.exe
=====
6800 Daughter Module Options
=====
1 Set Default Configuration (Clocks, Muxes, etc.)
2 Read ADC Data (16 bit 2's complement)
3 Write to DAC
4 Digital I/O

0 Exit
=====
Enter Selection: 2
ADC 0: 0x27a8
ADC 1: 0x29ac
ADC 2: 0x2b8d
ADC 3: 0x2d31
ADC 4: 0x2ec4
ADC 5: 0x302f
ADC 6: 0x3189
ADC 7: 0x3203
ADC 8: 0x272e
ADC 9: 0x290b
ADC 10: 0x2aeb
ADC 11: 0x2c7a
ADC 12: 0x2e30
ADC 13: 0x2f7c
ADC 14: 0x30c7
ADC 15: 0x31a3
ADC 16: 0x8000
ADC 17: 0x8000
ADC 18: 0x8000
ADC 19: 0x8000
ADC 20: 0x8000
ADC 21: 0x8000
ADC 22: 0x8000
ADC 23: 0x8000
ADC 24: 0x8000
ADC 25: 0x8000
ADC 26: 0x8000
ADC 27: 0x8000
ADC 28: 0x8000
ADC 29: 0x8000
ADC 30: 0x8000
ADC 31: 0x8000
ADC 32: 0x8000
ADC 33: 0x8000
ADC 34: 0x8000
ADC 35: 0x8000
ADC 36: 0x8000
ADC 37: 0x8000
ADC 38: 0x8000
ADC 39: 0x8000
ADC 40: 0x8000
ADC 41: 0x8000
ADC 42: 0x8000
ADC 43: 0x8000
ADC 44: 0x8000
ADC 45: 0x8000
ADC 46: 0x8000
ADC 47: 0x8000
ADC 48: 0x8000
ADC 49: 0x8000
ADC 50: 0x8000
ADC 51: 0x8000
ADC 52: 0x8000
ADC 53: 0x8000
ADC 54: 0x8000
ADC 55: 0x8000
ADC 56: 0x8000
ADC 57: 0x8000
ADC 58: 0x8000
ADC 59: 0x8000
ADC 60: 0x8000
ADC 61: 0x8000
ADC 62: 0x8000
ADC 63: 0x8000
```

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.