Engineer-to-Engineer Note EE-243

ANALOG DEVICES

Technical notes on using Analog Devices DSPs, processors and development tools
Contact our technical support at processor.support@analog.com and dsptools.support@analog.com
Or visit our on-line resources http://www.analog.com/ee-notes and http://www.analog.com/processors

Using the Expert DAI for ADSP-2126x and ADSP-2136x SHARC® Processors

Contributed by Jagadeesh Rayala and Srinivas K

Rev 2 – June 1, 2005

Introduction

This EE-Note explains how you use the Expert DAI plug-in in VisualDSP++® 4.0 to configure the signal routing unit (SRU) in ADSP-2126x and ADSP-2136x SHARC® processors. The Expert DAI plug-in simplifies the task of generating the C and/or assembly code that is used to program the SRU.

Digital Audio Interface and SRU

The digital audio interface (DAI) in ADSP-2126x and ADSP-2136x SHARC processor derivatives comprises a group of peripherals and the SRU. The peripheral's inputs and the outputs do *not* connect to the 20 external DAI pins (`DAI_P20-1`) directly. Instead, the SRU establishes these connections, based on a set of configuration registers. This feature allows you to interconnect the peripherals to suit a wide variety of systems. It also allows ADSP-2126x and ADSP-2136x processors to include an arbitrary number and variety of peripherals while retaining high levels of compatibility without increasing pin count.

The SRU contains six groups of registers (named A through F). Each group routes a unique set of signals with a specific purpose. For example, group A routes clock signals, group B routes frame sync signals, and group C routes serial data signals. Together, the SRU's six groups include all of the DAI peripherals' inputs and outputs, additional signals from the core, and all the connections to the DAI pins. For additional information on the DAI and SRU, refer to the processor's *Peripherals Manual* [1] and [2].

Programming the SRU requires an in-depth understanding of the SRU registers, bit field positions corresponding to different destination signals in all the registers, the number of bits allocated for each bit field in all the registers, and the values that correspond to different source signals in all of the registers. VisualDSP++ includes a macro for programming the SRU registers. This macro requires that you have background knowledge about all of the SRU signals and registers. The Expert DAI plug-in, however, provides an easy method of generating the code necessary to configure the SRU registers. The Expert DAI tool's graphical user interface allows you to generate the code without having to worry about the internal details.

Note that in addition to SRU and DAI, ADSP-21367 and ADSP-21368 SHARC processors have a secondary signal routing unit (SRU2) and 14 pins, which are available on the digital peripheral interface (DPI). SRU2 is used to map different peripheral signals to the DPI pins. The Expert DAI plug-in can also be used to generate the C and/or assembly code that is used to program the SRU2.

Installing Expert DAI

To install the Expert DAI plug-in in the VisualDSP++ environment:

1. Extract the file `AdvancedExpertDAI.dll` from the associated ZIP file (`EE243v02.zip`) and place it in the VisualDSP++ `System` directory. If VisualDSP++ 4.0 is installed on your C drive, copy the attached file into the following directory:

   ```
   C:\Program Files\Analog Devices\VisualDSP 4.0\System
   ```

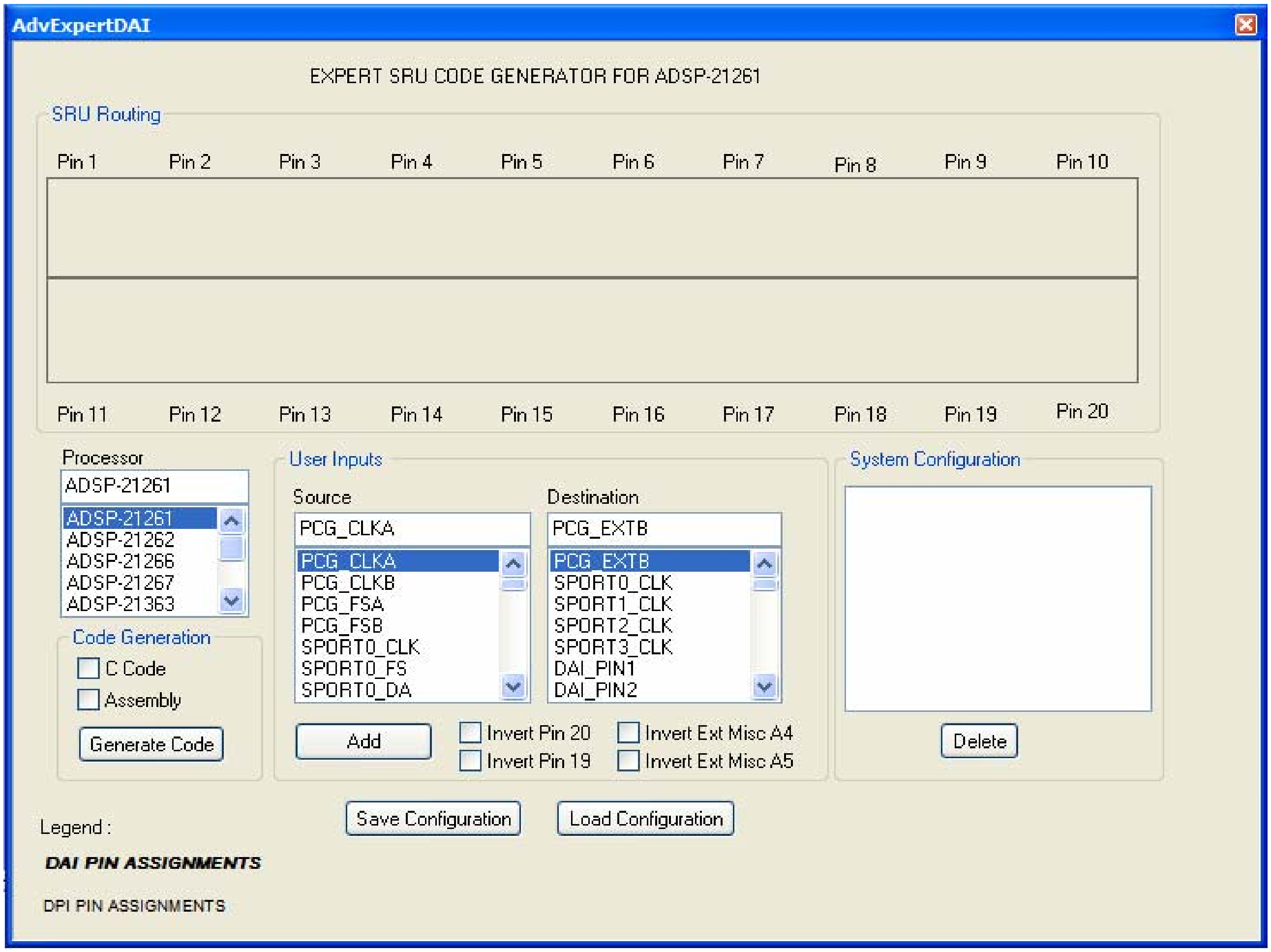
2. Register the `AdvancedExpertDAI.dll` file by typing the following command line:

   ```
   regsvr32.exe AdvancedExpertDAI.dll
   ```

   Note: Run `regsvr32.exe` from the `..\System` directory, not from the root directory

3. Extract the file `SRU.h` from the associated ZIP file and place it in the VisualDSP++ `Include` directory. If VisualDSP++ 4.0 is installed on your C drive, copy the attached file into the following directory:

   ```
   C:\Program Files\Analog Devices\VisualDSP 4.0\213xx\include
   ```



AdvExpertDAI

EXPERT SRU CODE GENERATOR FOR ADSP-21261

Processor: ADSP-21261

Source: PCG_CLKA

Destination: PCG_EXTB

Figure 1. Expert DAI Window

The Expert DAI tool now appears in the Settings => Preferences => Plugins menu. You can access Expert DAI from the Tools => Plugins menu. Figure 1 shows the default state of the Expert DAI window.

Using Expert DAI

To generate code:

1. In Processor, select the processor for which you want to generate code.
2. Under User Inputs, configure connections between the source and destination signals by selecting the signals from the respective Source and Destination list boxes and clicking Add.

   When you select a signal in the Source box, the signals that appear in the Destination box are updated with valid destinations that correspond to the selected source signal. If you try to route two source signals to the same destination signal, the Expert DAI tool will signal an error. Similarly, the tool will generate an error if you select a DAI pin as a source of one connection and try to select the same DAI pin as a destination in a subsequent connection (and vice versa).

   As you add connections, the System Configuration box and the SRU routing diagram update appropriately.
3. Select the appropriate check boxes to invert various signals (logic level of DAI pin 19, DAI pin 20, Ext Misc A4, and Ext Misc A5).
4. Select the appropriate check boxes and click Generate Code to generate the C and/or assembly code.

If the processor type is changed after you have added the connections, data in the SRU routing diagram and in the System Configuration is cleared automatically. At the same time, the signals that appear in the Source and Destination boxes are refreshed and updated for the selected processor.

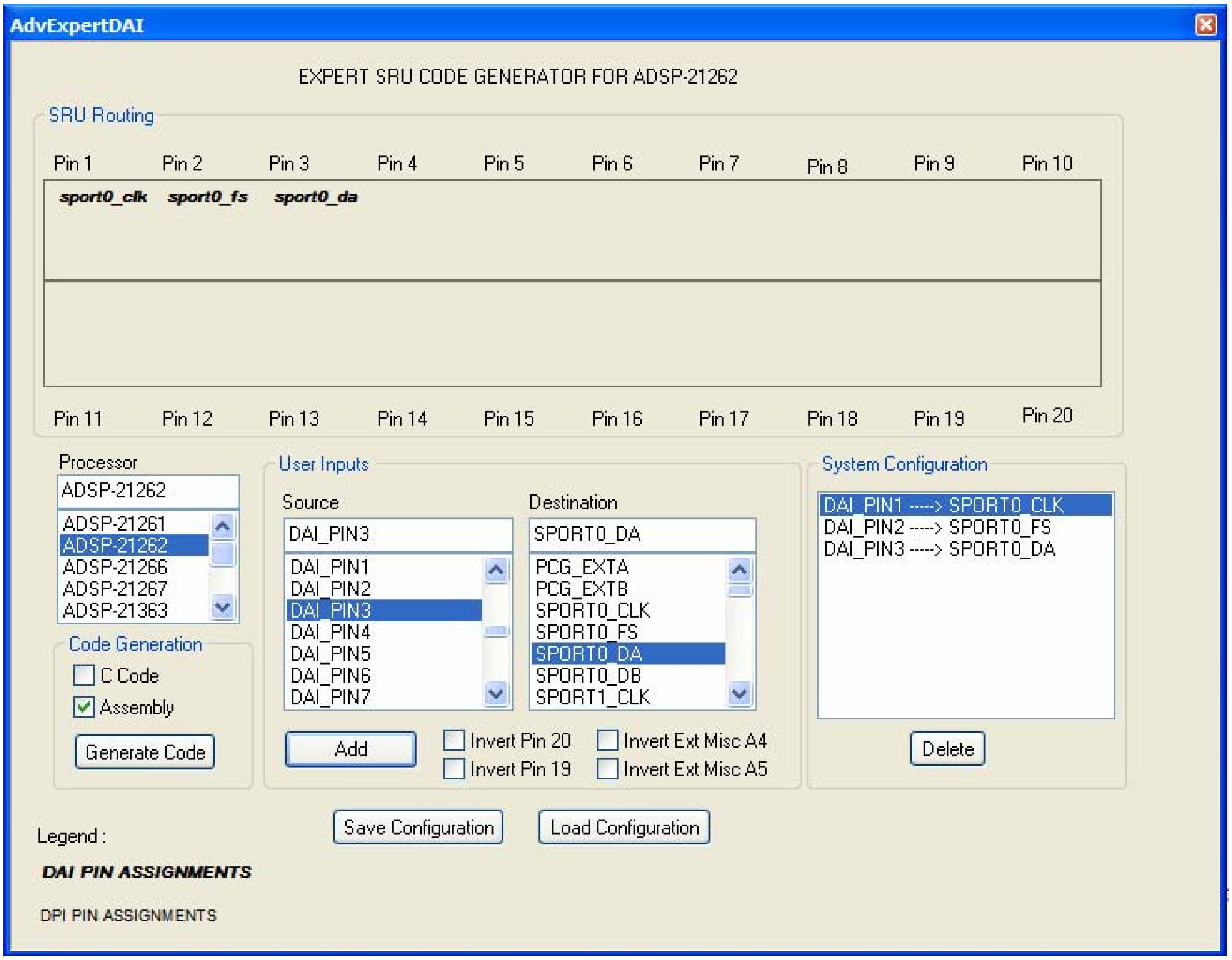
To remove a connection, select the connection in the System Configuration box and click Delete. The SRU routing diagram refreshes automatically, and the System Configuration box is updated.

The Expert DAI plug-in also provides these features:

- Save Configuration: Saves the information about the selected processor , the added connections, and the state of each check box. Clicking Save Configuration stores the information in an output file with a .CFG extension.
- Load Configuration: Loads a saved configuration (.CFG file). Clicking Load Configuration prompts you to select a .CFG file. After selecting the .CFG file, the Expert DAI window refreshes, presenting the contents in the .CFG file. At this point, you can add/delete/modify the connections per your new design and then create the code to configure the SRU.

Code Generation

This section use examples to describe the code-generation process. Figure 2 demonstrates the generation of assembly code for configuring the SRU in ADSP-21262 processors.



SRU Routing: Pin 1 *sport0_clk*, Pin 2 *sport0_fs*, Pin 3 *sport0_da*

Processor: ADSP-21262

Code Generation: ☐ C Code ☑ Assembly

Source: DAI_PIN3

Destination: SPORT0_DA

☐ Invert Pin 20 ☐ Invert Ext Misc A4
☐ Invert Pin 19 ☐ Invert Ext Misc A5

System Configuration:

```
DAI_PIN1 -----> SPORT0_CLK
DAI_PIN2 -----> SPORT0_FS
DAI_PIN3 -----> SPORT0_DA
```

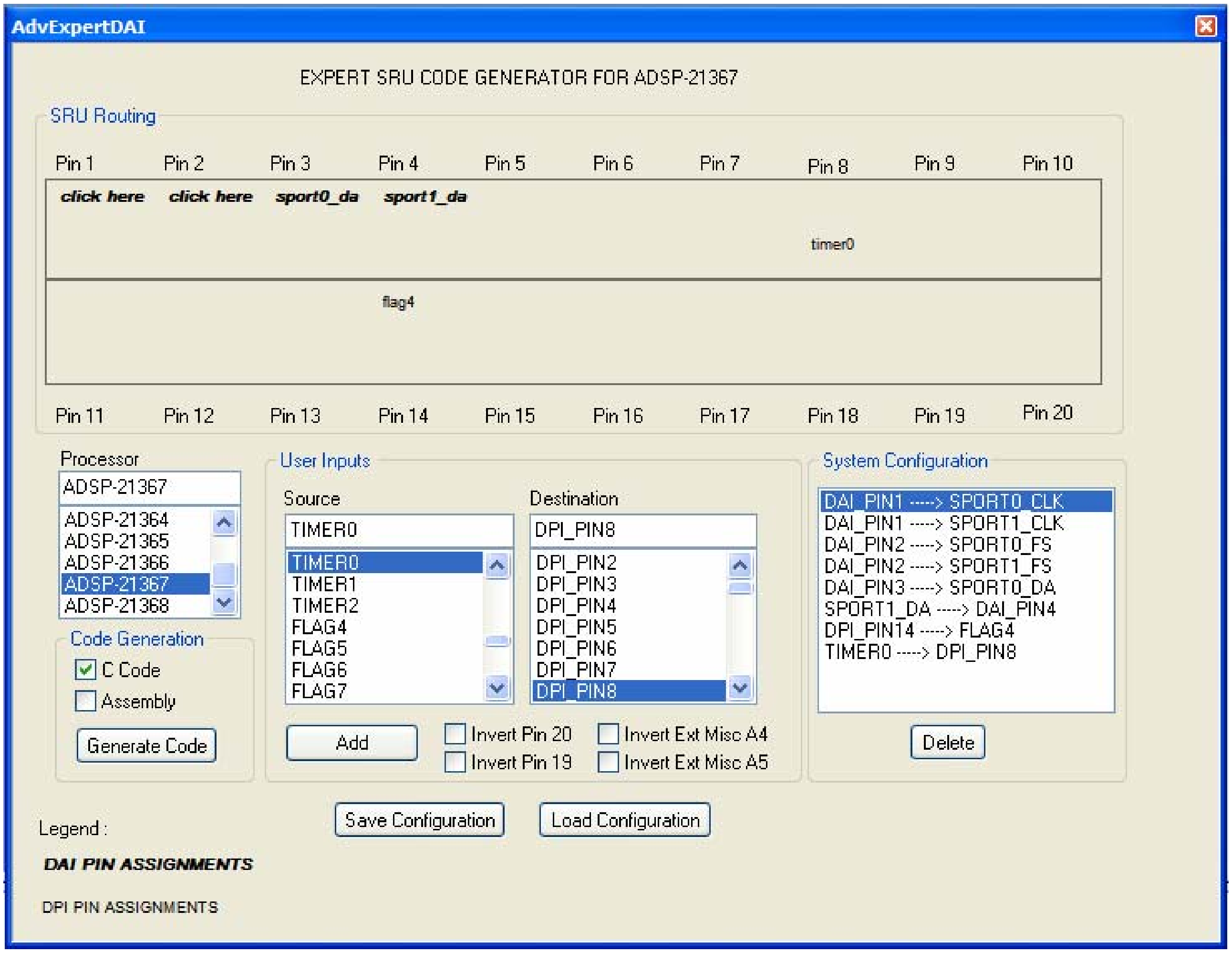
Figure 2. Generating Assembly Code for Configuring the SRU in ADSP-21262 Processors

Code is generated to perform the following connections:

- DAI pin 1 (`DAI_PIN1`) is configured as an input to provide the clock for SPORT0.
- DAI pin 2 (`DAI_PIN2`) is configured as an input to provide the frame sync for SPORT0.
- DAI pin 3 (`DAI_PIN3`) is configured as an input to provide the data for SPORT0 DA.

The SRU routing diagram is also updated to reflect the added connections. The assembly code generated for this configuration is shown in Listing 1 of the Appendix. Figure 3 demonstrates the generation of C code for configuring the SRU and SRU2 in ADSP-21367 processors. Code is generated to perform the following connections:

- DAI pin 1 (`DAI_PIN1`) is configured as an input to provide the clock for SPORT0 and SPORT1.
- DAI pin 2 (`DAI_PIN2`) is configured as an input to provide the frame sync for SPORT0 and SPORT1.
- DAI pin 3 (`DAI_PIN3`) is configured as an input to provide the data for SPORT0 DA.
- SPORT1 DA provides the data for DAI pin 4 (`DAI_PIN4`), which is configured as an output.
- DPI pin 14 (`DPI_PIN14)` is configured as an input to provide the signal for FLAG 4.
- TIMER 0 provides the signal for DPI pin 8 (`DPI_PIN8`), which is configured as an output.



Processor: ADSP-21367

ADSP-21364
ADSP-21365
ADSP-21366
ADSP-21367
ADSP-21368

☑ C Code
□ Assembly

Source: TIMER0

TIMER0
TIMER1
TIMER2
FLAG4
FLAG5
FLAG6
FLAG7

Destination: DPI_PIN8

DPI_PIN2
DPI_PIN3
DPI_PIN4
DPI_PIN5
DPI_PIN6
DPI_PIN7
DPI_PIN8

□ Invert Pin 20
□ Invert Pin 19
□ Invert Ext Misc A4
□ Invert Ext Misc A5

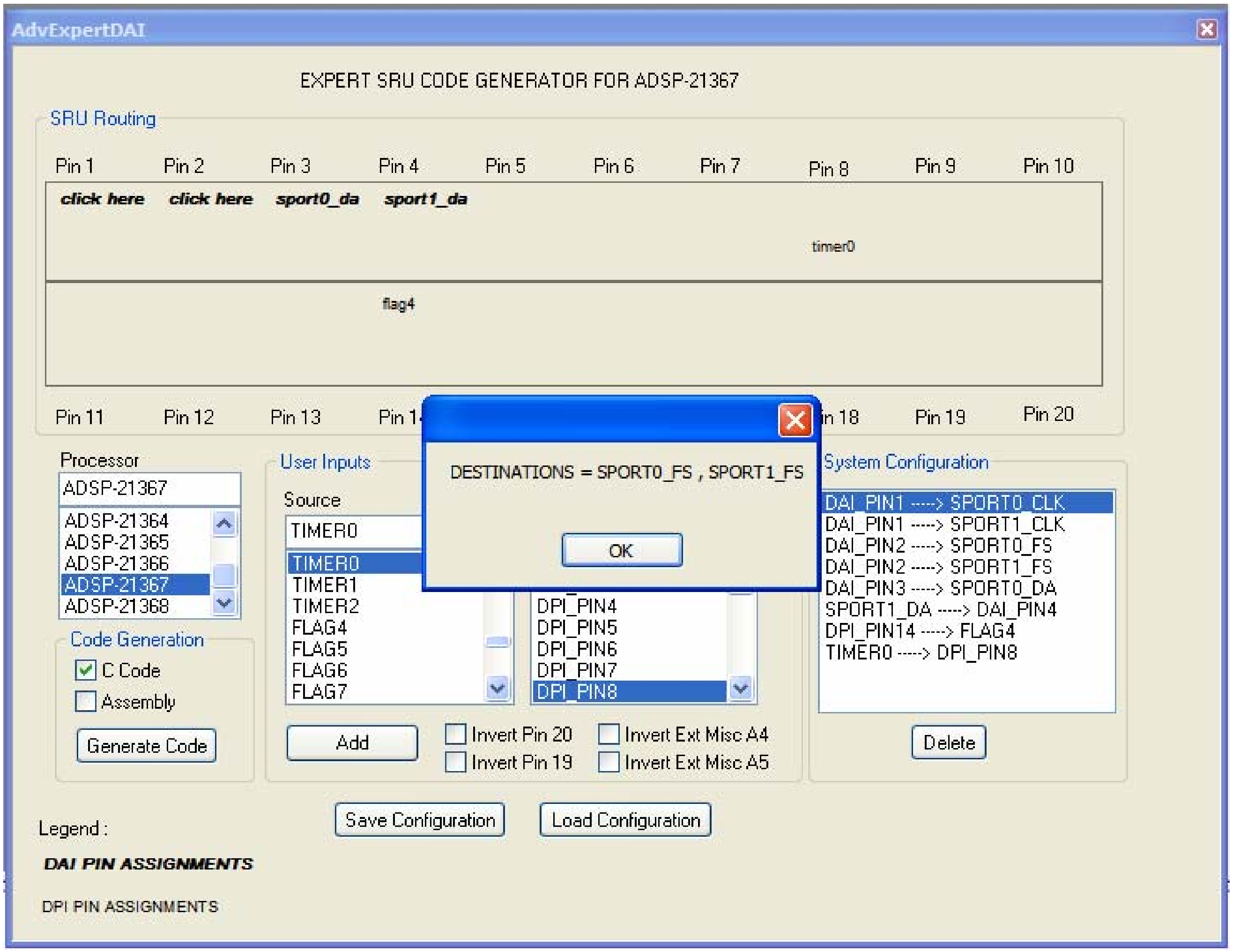
System Configuration:

```
DAI_PIN1 -----> SPORT0_CLK
DAI_PIN1 -----> SPORT1_CLK
DAI_PIN2 -----> SPORT0_FS
DAI_PIN2 -----> SPORT1_FS
DAI_PIN3 -----> SPORT0_DA
SPORT1_DA -----> DAI_PIN4
DPI_PIN14 -----> FLAG4
TIMER0 -----> DPI_PIN8
```

Figure 3. Generating C Code for Configuring the SRU and SRU2 in ADSP-21367 Processors

The SRU routing diagram is also updated to reflect the connections. The C code generated for this configuration is shown in Listing 2 of the Appendix.

Unlike the example in Figure 2, the DAI pins configured in Figure 3 demonstrate the signals for multiple destination signals. DAI pin 1 provides the clock for SPORT0 and SPORT1. Similarly, DAI pin 2 provides the frame sync for SPORT0 and SPORT1. When a single DAI pin feeds multiple destination signals, the destination signals connected to the DAI pin do *not* appear in the SRU routing diagram. You will be prompted to click for any DAI pin that provides the signal to multiple destination signals. After clicking a particular DAI pin, a small window appears (Figure 4), displaying all of the destination signals connected to that DAI pin. For example, clicking on DAI pin 2 causes the following window to pop up, identifying the destination signals as `SPORT0_FS` and `SPORT1_FS`.



DESTINATIONS = SPORT0_FS , SPORT1_FS

Processor: ADSP-21367

ADSP-21364
ADSP-21365
ADSP-21366
ADSP-21367
ADSP-21368

☑ C Code
□ Assembly

Source: TIMER0

TIMER0
TIMER1
TIMER2
FLAG4
FLAG5
FLAG6
FLAG7

DPI_PIN4
DPI_PIN5
DPI_PIN6
DPI_PIN7
DPI_PIN8

□ Invert Pin 20
□ Invert Pin 19
□ Invert Ext Misc A4
□ Invert Ext Misc A5

```
DAI_PIN1 -----> SPORT0_CLK
DAI_PIN1 -----> SPORT1_CLK
DAI_PIN2 -----> SPORT0_FS
DAI_PIN2 -----> SPORT1_FS
DAI_PIN3 -----> SPORT0_DA
SPORT1_DA -----> DAI_PIN4
DPI_PIN14 -----> FLAG4
TIMER0 -----> DPI_PIN8
```

Figure 4. Checking the (Destination) Signals Routed to DAI Pin 2

Once the C/assembly code is generated, you must add this file to your project. For C code, the `main` function must call the function `InitSRU()`. For assembly code, the `main` program must call the subroutine `InitSRU`.

Appendix

configdai.asm

```
#include <sru.h>
#include <def21262.h>

// This function will setup the SRU Registers
.section /pm seg_pmco ;
.global InitSRU ;
InitSRU :

    // Enable pull-up resistors on unused DAI pins
    r0 = 0xffff8;
    dm(DAI_PIN_PULLUP) = r0;

    //Generating Code for connecting : DAI_PIN1 to SPORT0_CLK
    SRU (LOW, PBEN01_I);
    SRU (DAI_PB01_O, SPORT0_CLK_I);

    //Generating Code for connecting : DAI_PIN2 to SPORT0_FS
    SRU (LOW, PBEN02_I);
    SRU (DAI_PB02_O, SPORT0_FS_I);

    //Generating Code for connecting : DAI_PIN3 to SPORT0_DA
    SRU (LOW, PBEN03_I);
    SRU (DAI_PB03_O, SPORT0_DA_I);

    // Return back from the subroutine
    rts;
```

Listing 1. configdai.asm

configdai.c

```
#include <sru.h>
#include <def21367.h>

// This function will setup the SRU Registers
void InitSRU()
{
    // Enable pull-up resistors on unused DAI pins
    * (volatile int *)DAI_PIN_PULLUP = 0xffff0;

    // Enable pull-up resistors on unused DPI pins
    * (volatile int *)DPI_PIN_PULLUP = 0x1f7f;

    //Generating Code for connecting : SPORT1_DA to DAI_PIN4
    SRU (HIGH, PBEN04_I);
    SRU (SPORT1_DA_O, DAI_PB04_I);

    //Generating Code for connecting : DAI_PIN1 to SPORT0_CLK
    SRU (LOW, PBEN01_I);
    SRU (DAI_PB01_O, SPORT0_CLK_I);
    //Generating Code for connecting : DAI_PIN1 to SPORT1_CLK
    SRU (LOW, PBEN01_I);
    SRU (DAI_PB01_O, SPORT1_CLK_I);

    //Generating Code for connecting : DAI_PIN2 to SPORT0_FS
    SRU (LOW, PBEN02_I);
    SRU (DAI_PB02_O, SPORT0_FS_I);

    //Generating Code for connecting : DAI_PIN2 to SPORT1_FS
    SRU (LOW, PBEN02_I);
    SRU (DAI_PB02_O, SPORT1_FS_I);

    //Generating Code for connecting : DAI_PIN3 to SPORT0_DA
    SRU (LOW, PBEN03_I);
    SRU (DAI_PB03_O, SPORT0_DA_I);

    //Generating Code for connecting : DPI_PIN14 to FLAG4
    SRU (LOW, DPI_PBEN14_I);
    SRU (DPI_PB14_O, FLAG4_I);

    //Generating Code for connecting : TIMER0 to DPI_PIN8
    SRU (HIGH, DPI_PBEN08_I);
    SRU (TIMER0_O, DPI_PB08_I);
}
```

Listing 2. configdai.c

References

[1] *ADSP-2126x SHARC DSP Peripherals Manual*. Rev 2.0, January 2004. Analog Devices, Inc.

[2] *ADSP-21365 SHARC Processor Peripherals Manual*. Rev 0.1, May 2004. Analog Devices, Inc.

[3] *Configuring the Signal Routing Unit of ADSP-2126x SHARC DSPs (EE-232)*. Rev 1. February 2004. Analog Devices, Inc.

Document History

Revision	**Description**
Rev 2 – June 01, 2005 *by R. Jagadeesh*	Updated the application note and the ExpertDAI plugin to include support for the complete series of ADSP-2136x processors
Rev 1 – October 11, 2004 *by R. Jagadeesh*	Initial Release