Engineer-to-Engineer Note EE-268

ANALOG DEVICES

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

Programming Asynchronous Sample Rate Converters on ADSP-2136x SHARC® Processors

Contributed by Aseem Vasudev Prabhugaonkar

Rev 1 – April 25, 2005

Introduction

This EE-Note explains the fundamentals of sample rate conversion (SRC) and demonstrates how to program the sample rate converters of ADSP-2136x SHARC® processors. The code examples provided with this application note also demonstrate the Signal Routing Unit (SRU) programming required to interface the sample rate converter with the serial ports (SPORTs) of ADSP-21364 processors. A real-time code example supplied with this document demonstrates a simple talk-through with various sample rate ratios.

About SRCs

Sample rate converters (SRCs) are used to convert data from one clock source at a particular sample rate to another clock source at the same sample rate or at a different sample rate (Figure 1). There are two types of sample rate converters: synchronous sample rate converters and asynchronous sample rate converters. For synchronous SRCs, the output devices interfaced with the SRCs are “slaves” to the SRCs. For asynchronous SRCs, the output devices interfaced with the SRCs are “masters” to the SRCs. Most software implementations are based on synchronous SRCs and provide only small integer ratios of sample rate conversion. The SRC blocks implemented in ADSP-21364 processors are asynchronous sample rate converters.

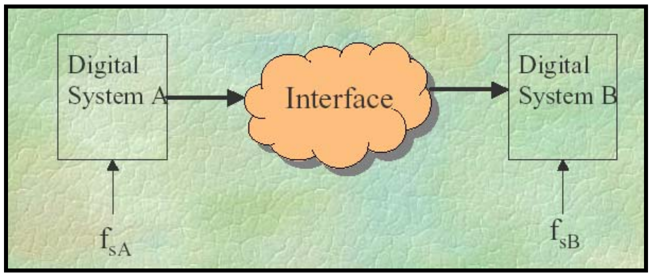


Figure 1. SRC as an Interface Between Two Devices with Different Sample Rates

The main application of SRCs is to change sample rate and to clean up the audio data from jittery clock sources like S/PDIF (Sony Philips Digital Interface) receivers.

SRC Fundamentals

A simple method of changing from one sample rate to another sample rate is to use a D/A converter in combination with a “brick wall” filter. Combining a D/A converter with a filter produces an output signal in the analog domain. The obtained analog signal is then converted back to digital data using an A/D converter that runs at a different sample rate. Figure 2 depicts this simple analog method of sample rate conversion.

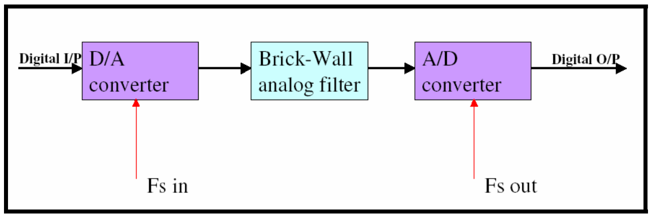


Figure 2. Analog Method of Sample Rate Conversion

For all-digital sample rate converters, the general conversion principles still remain the same. The analog filter is replaced by a combination of a digital interpolation filter with a very high output sample rate and a zero-order hold (ZOH) to convert the discrete-time output of a digital filter to a continuous time signal. Figure 3 depicts a digital method of sample rate conversion.

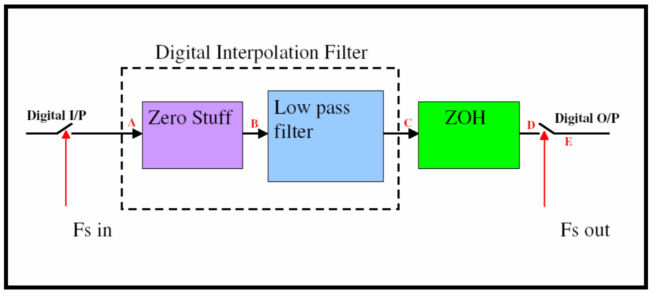


Figure 3. Digital Method of Sample Rate Conversion

Figure 4 through Figure 7 presents a time domain view of various signals in an all-digital SRC. Zero stuffing followed by a digital interpolation filter produces a stream of output samples (C) that have much finer time grid than the original input samples (A). These interpolated values are later fed to the ZOH and then sampled asynchronously at an output sampling frequency of `Fs out`.

The output samples always contain a degree of error because the output-sampling switch does not close at a time that exactly corresponds to a point on the fine time grid of the interpolated outputs.

This error can be made arbitrarily small by using a very large interpolation ratio.

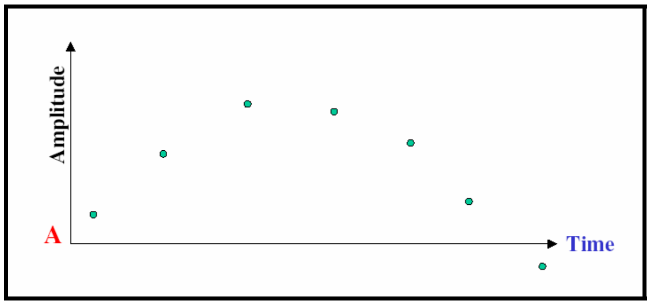


Figure 4. Input Samples at Fs in

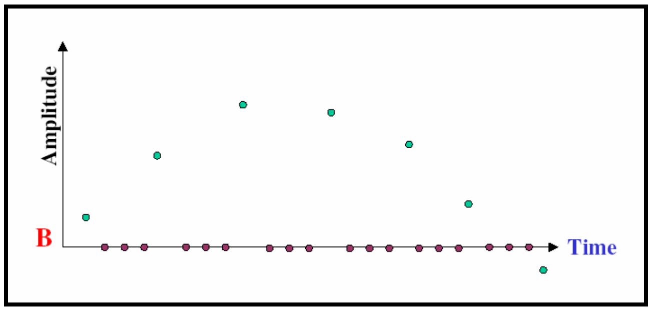


Figure 5. After Zero Stuffing

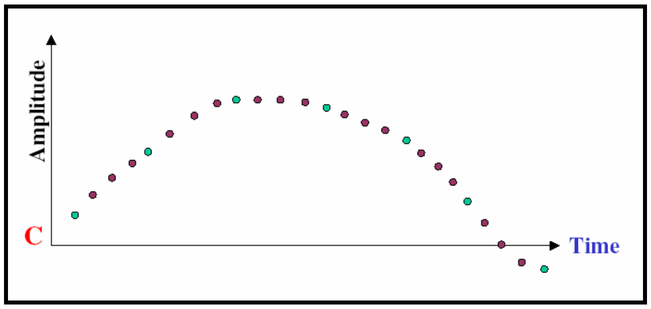


Figure 6. After Interpolation

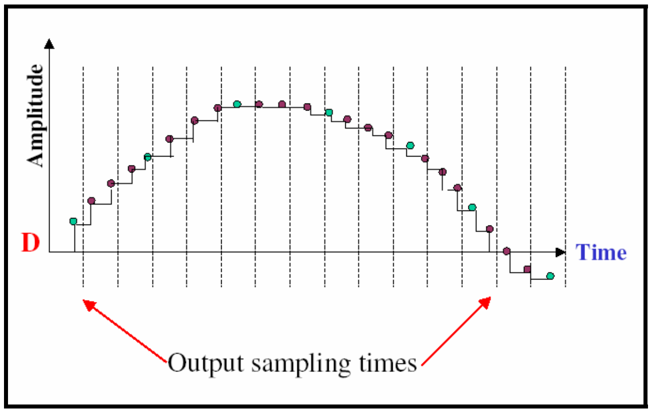


Figure 7. Zero-Order Hold (ZOH) and Output Sampling at Fs out

SRC Implementation on ADSP-21364 SHARC Processors

The SRC implemented on ADSP-2136x processors contains four SRC blocks (SRC0–3) and is the same core found in Analog Devices AD1896 192 kHz stereo asynchronous sample rate converters.

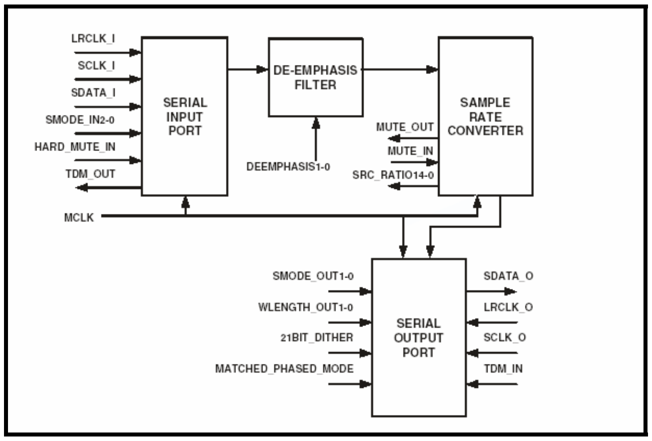


Figure 8. Block Diagram of ADSP-21364 SRC Block

The SRC block consists of a serial input port, de-emphasis filter, sample rate converter (SRC), and a serial output port. The serial input port supports left-justified, I2S, time division multiplexing (TDM), and right-justified 24-, 20-, 18-, and 16-bit serial formats. The serial input format is set by the `SMODE_IN[2:0]` bits in the SRC Control register. In the following examples, the serial format selected is I2S. The `SAMPLE_RATE_RATIO[14:11]` bits indicate the sample rate ratio. The `SAMPLE_RATE_RATIO [14:11]` bits are the integer part of the sample rate ratio, and the `SAMPLE_RATE_RATIO[10:0]` bits are the fractional part. For details on SRC registers, refer to the *ADSP-2136x SHARC Processor Hardware Reference* [1].

Refer to the AD1896 data sheet [4] for information about various SRC timing parameters.

SRC Example 1

This example setup (Figure 9) consists of SPORT0, SPORT1, and the SRC0 blocks of ADSP-21364. SPORT0 interfaces with the serial input port of SRC0 and transfers data to input port of SRC0. The SRC in the ADSP-21364 is a slave, and SPORT0 provides input frame sync `LRCLK_I` and `SCLK_I` to the input port of SRC0. SPORT1 interfaces to the serial output port of SRC0. `LRCLK_O` and `SCLK_O` are provided by SPORT1, which is configured to receive data from SRC0. SPORT0 and SPORT1 are configured for different frame sync frequencies and hence different sampling frequencies.

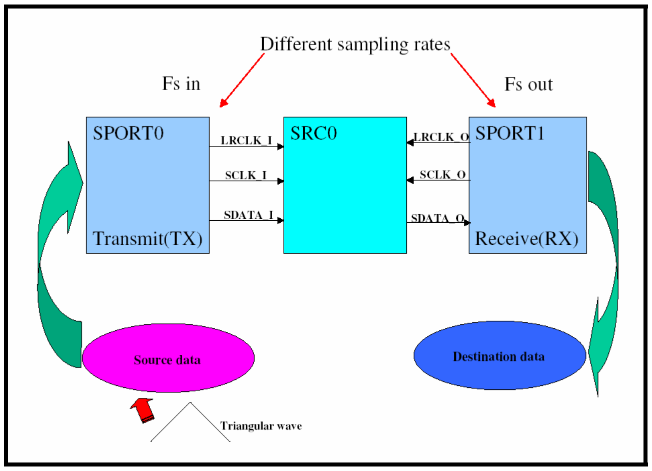


Figure 9. Block Diagram of SRC Example 1

The data values that correspond to a triangular wave are initialized in the processor's internal memory. This data is sent to SRC0 at a sampling frequency of `Fs in`. SPORT1 drives the `Fs out` at a different frequency than that of `Fs in` and collects converted data from SRC0. The data received by SPORT1 is transferred into the processor's internal memory in the interrupt service routine (ISR) of SPORT1. Use the plotting capabilities of VisualDSP++® to compare the received (destination) data against the source data. See the Appendix for screen shots.

SRC Example 2

SRC example 2 consists of SRC0, SPORT0, SPORT1, and SRC1. This example runs a talk-through by passing audio data through SRC0 and SRC1 in real time. Audio data from the AD1835 codec is sent to SRC0. The codec drives the `LRCLK_I` and `SCLK_I` of SRC0. SPORT0 is configured to receive data from SRC0 at a different sampling rate than that of the codec. SPORT1 is configured to transmit data to SRC1 at the same rate that SPORT0 receives data from SRC0. The data received by SPORT0 is transferred to the SPORT1 transmit buffer in the receive ISR of SPORT0. IRQ1 ISR performs a "softmute" of SRC1.

The codec receives data from SRC1. `LRCLK_O` and `SCLK_O` of SRC1 are provided by the codec. Figure 10 shows the block diagram of SRC example 2.

SRC examples 1 and 2 are tested on the ADSP-21364 EZ-KIT Lite® evaluation system board.

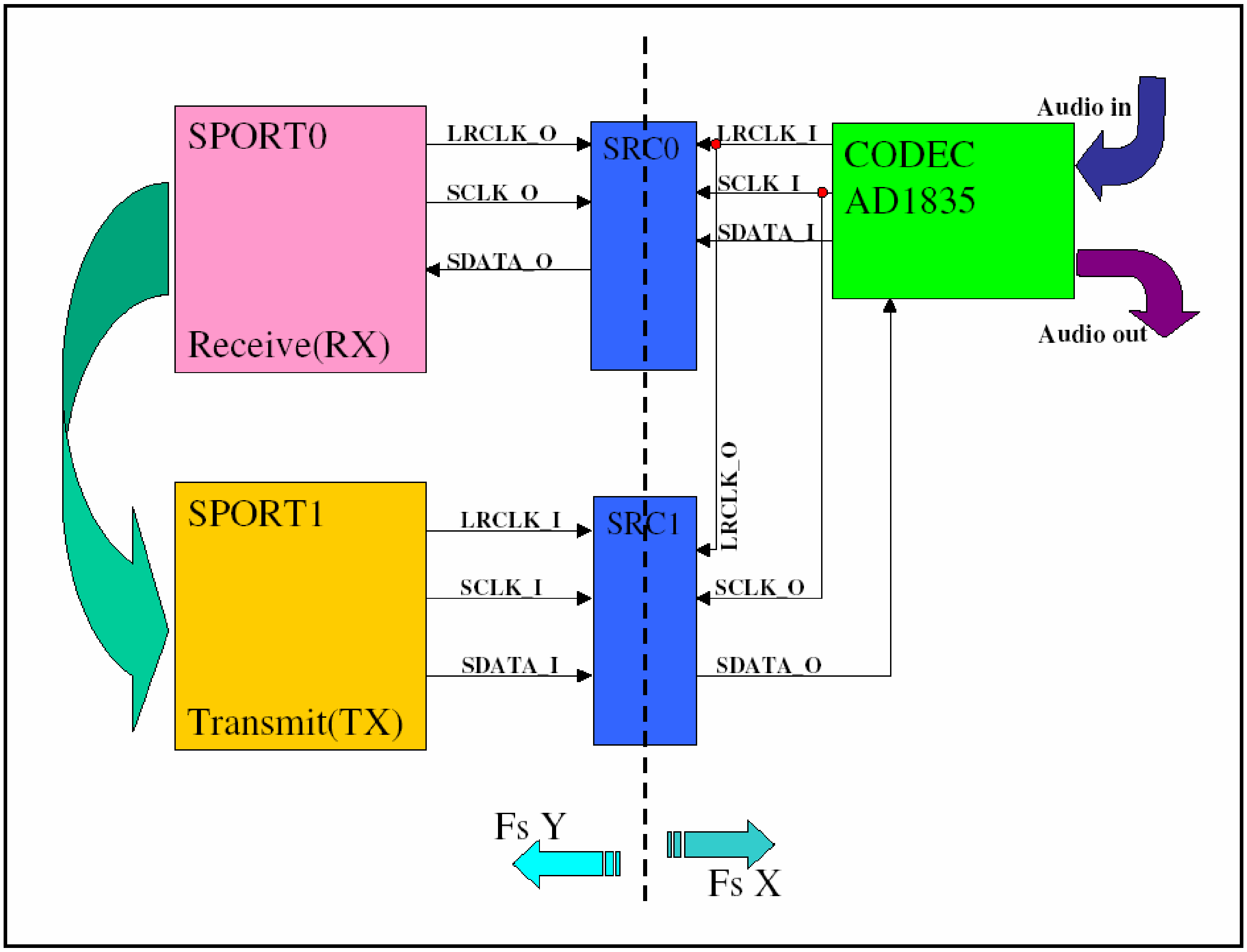
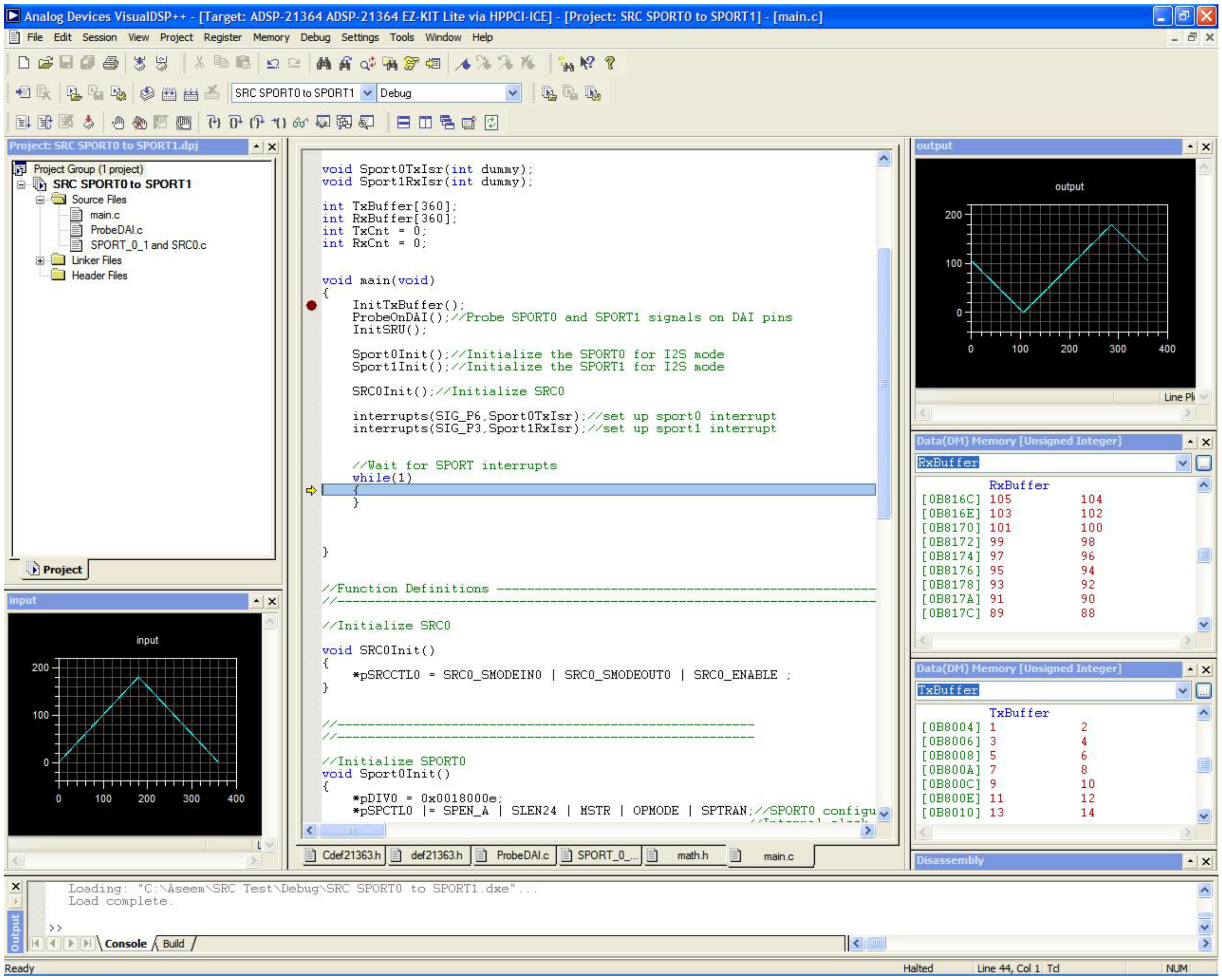


Figure 10. Block Diagram of SRC Example 2

Appendix


Analog Devices VisualDSP++ - [Target: ADSP-21364 ADSP-21364 EZ-KIT Lite via HPPCI-ICE] - [Project: SRC SPORT0 to SPORT1] - [main.c]
File Edit Session View Project Register Memory Debug Settings Tools Window Help
SRC SPORT0 to SPORT1
Debug
Project: SRC SPORT0 to SPORT1.dpj
Project Group (1 project)
SRC SPORT0 to SPORT1
Source Files
main.c
ProbeDAI.c
SPORT_0_1 and SRC0.c
Linker Files
Header Files
Project
input
output
Line Pl
Cdef21363.h
def21363.h
ProbeDAI.c
SPORT_0_...
math.h
main.c
Disassembly
Output
Console
Build
Ready
Halted
Line 44, Col 1
Tcl
NUM

```
void Sport0TxIsr(int dummy);
void Sport1RxIsr(int dummy);

int TxBuffer[360];
int RxBuffer[360];
int TxCnt = 0;
int RxCnt = 0;


void main(void)
{
    InitTxBuffer();
    ProbeOnDAI();//Probe SPORT0 and SPORT1 signals on DAI pins
    InitSRU();

    Sport0Init();//Initialize the SPORT0 for I2S mode
    Sport1Init();//Initialize the SPORT1 for I2S mode

    SRC0Init();//Initialize SRC0

    interrupts(SIG_P6,Sport0TxIsr);//set up sport0 interrupt
    interrupts(SIG_P3,Sport1RxIsr);//set up sport1 interrupt


    //Wait for SPORT interrupts
    while(1)
    {
    }



}


//Function Definitions -------------------------------------------------------
//----------------------------------------------------------------------------

//Initialize SRC0

void SRC0Init()
{
    *pSRCCTL0 = SRC0_SMODEIN0 | SRC0_SMODEOUT0 | SRC0_ENABLE ;
}


//----------------------------------------------------------
//----------------------------------------------------------

//Initialize SPORT0
void Sport0Init()
{
    *pDIV0 = 0x0018000e;
    *pSPCTL0 |= SPEN_A | SLEN24 | MSTR | OPMODE | SPTRAN;//SPORT0 configu
```

Data(DM) Memory [Unsigned Integer]

RxBuffer

```
         RxBuffer
[0B816C] 105         104
[0B816E] 103         102
[0B8170] 101         100
[0B8172] 99          98
[0B8174] 97          96
[0B8176] 95          94
[0B8178] 93          92
[0B817A] 91          90
[0B817C] 89          88
```

Data(DM) Memory [Unsigned Integer]

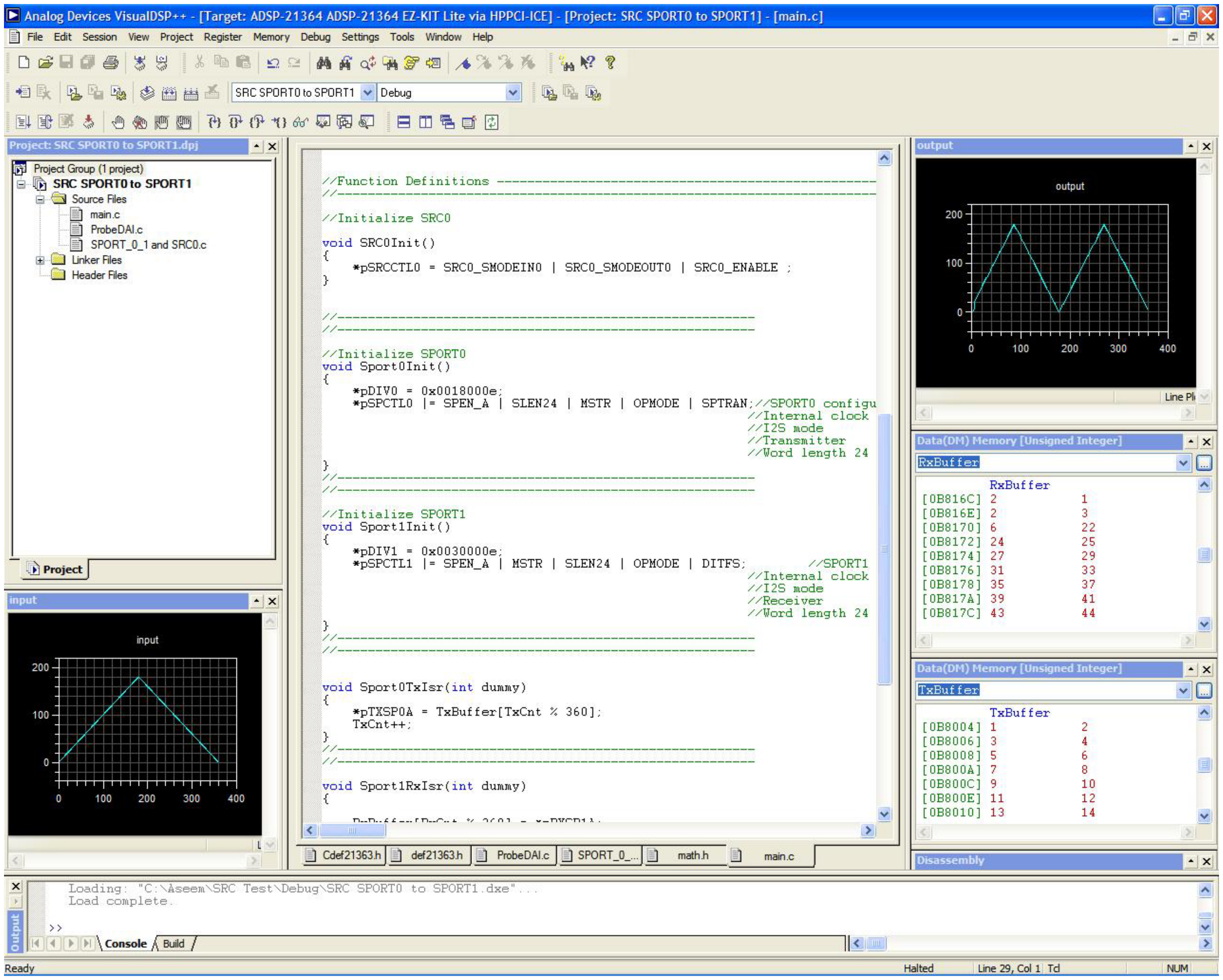
TxBuffer

```
         TxBuffer
[0B8004] 1           2
[0B8006] 3           4
[0B8008] 5           6
[0B800A] 7           8
[0B800C] 9           10
[0B800E] 11          12
[0B8010] 13          14
```

```
Loading: "C:\Aseem\SRC Test\Debug\SRC SPORT0 to SPORT1.dxe"...
Load complete.

>>
```

Figure 11. VisualDSP++ Screen Shot for Sample Rate Ratio of 1:1 (Fs in = Fs out)


Analog Devices VisualDSP++ - [Target: ADSP-21364 ADSP-21364 EZ-KIT Lite via HPPCI-ICE] - [Project: SRC SPORT0 to SPORT1] - [main.c]
File Edit Session View Project Register Memory Debug Settings Tools Window Help
SRC SPORT0 to SPORT1
Debug
Project: SRC SPORT0 to SPORT1.dpj
Project Group (1 project)
SRC SPORT0 to SPORT1
Source Files
main.c
ProbeDAI.c
SPORT_0_1 and SRC0.c
Linker Files
Header Files
Project
input
output
Line Pl
Cdef21363.h
def21363.h
ProbeDAI.c
SPORT_0_...
math.h
main.c
Disassembly
Console
Build
Ready
Halted
Line 29, Col 1
Tcl
NUM

```
//Function Definitions -----------------------------------------------------
//--------------------------------------------------------------------------

//Initialize SRC0

void SRC0Init()
{
    *pSRCCTL0 = SRC0_SMODEIN0 | SRC0_SMODEOUT0 | SRC0_ENABLE ;
}


//------------------------------------------------------------
//------------------------------------------------------------

//Initialize SPORT0
void Sport0Init()
{
    *pDIV0 = 0x0018000e;
    *pSPCTL0 |= SPEN_A | SLEN24 | MSTR | OPMODE | SPTRAN;//SPORT0 configu
                                                       //Internal clock
                                                       //I2S mode
                                                       //Transmitter
                                                       //Word length 24

}
//------------------------------------------------------------
//------------------------------------------------------------

//Initialize SPORT1
void Sport1Init()
{
    *pDIV1 = 0x0030000e;
    *pSPCTL1 |= SPEN_A | MSTR | SLEN24 | OPMODE | DITFS;        //SPORT1
                                                       //Internal clock
                                                       //I2S mode
                                                       //Receiver
                                                       //Word length 24

}
//------------------------------------------------------------
//------------------------------------------------------------


void Sport0TxIsr(int dummy)
{
    *pTXSP0A = TxBuffer[TxCnt % 360];
    TxCnt++;
}
//------------------------------------------------------------
//------------------------------------------------------------

void Sport1RxIsr(int dummy)
{
```

Data(DM) Memory [Unsigned Integer]

```
RxBuffer
          RxBuffer
[0B816C] 2          1
[0B816E] 2          3
[0B8170] 6          22
[0B8172] 24         25
[0B8174] 27         29
[0B8176] 31         33
[0B8178] 35         37
[0B817A] 39         41
[0B817C] 43         44
```

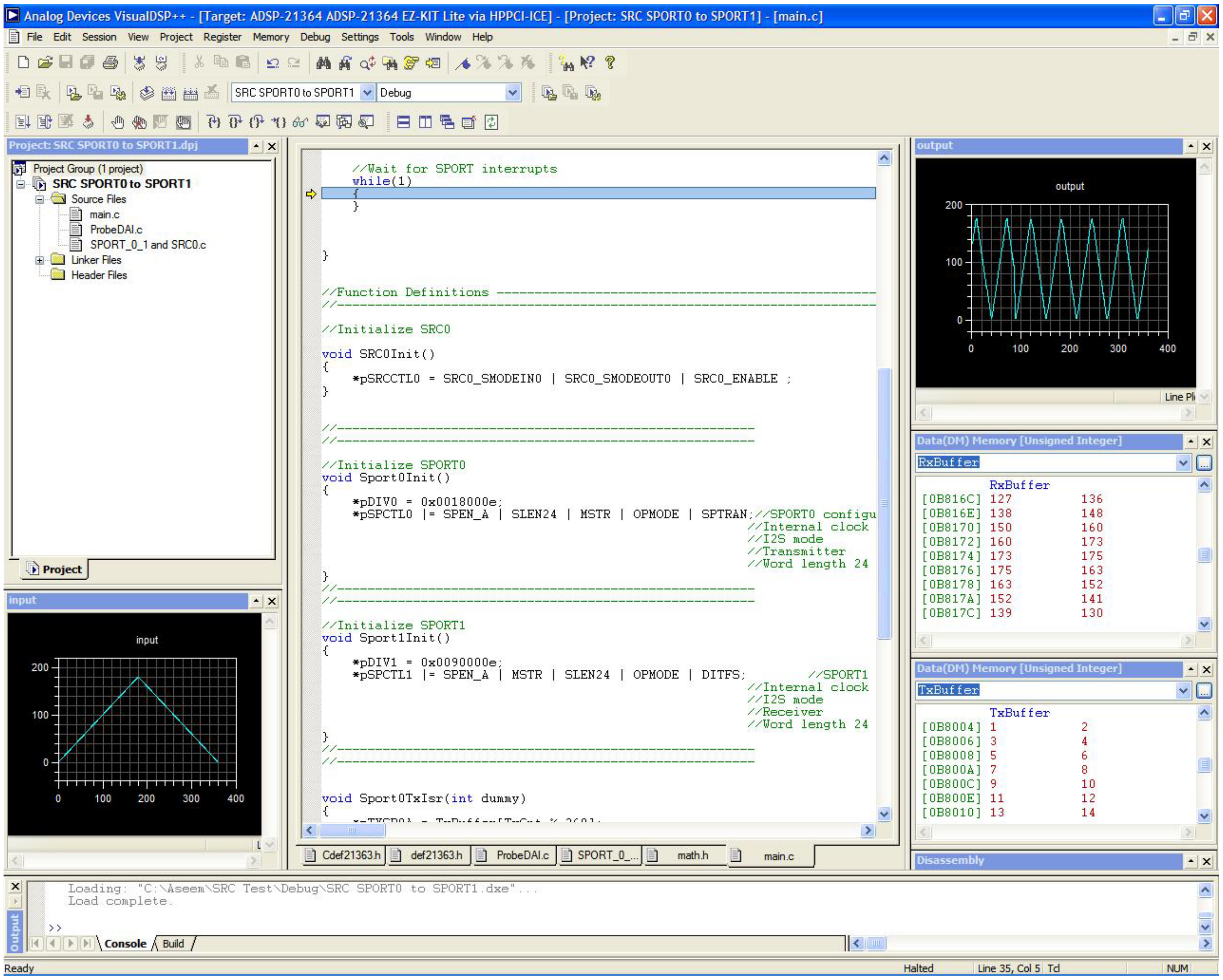
Data(DM) Memory [Unsigned Integer]

```
TxBuffer
          TxBuffer
[0B8004] 1          2
[0B8006] 3          4
[0B8008] 5          6
[0B800A] 7          8
[0B800C] 9          10
[0B800E] 11         12
[0B8010] 13         14
```

```
Loading: "C:\Aseem\SRC Test\Debug\SRC SPORT0 to SPORT1.dxe"...
Load complete.

>>
```

*Figure 12. VisualDSP++ Screen Shot for Sample Rate Ratio of 0.5 (Fs in = 2*Fs out)*


Analog Devices VisualDSP++ - [Target: ADSP-21364 ADSP-21364 EZ-KIT Lite via HPPCI-ICE] - [Project: SRC SPORT0 to SPORT1] - [main.c]
File Edit Session View Project Register Memory Debug Settings Tools Window Help
SRC SPORT0 to SPORT1
Debug
Project: SRC SPORT0 to SPORT1.dpj
Project Group (1 project)
SRC SPORT0 to SPORT1
Source Files
main.c
ProbeDAI.c
SPORT_0_1 and SRC0.c
Linker Files
Header Files
Project
input
output
Line Pl
Data(DM) Memory [Unsigned Integer]
Disassembly
Cdef21363.h
def21363.h
ProbeDAI.c
SPORT_0_...
math.h
main.c
Console
Build
Output
Ready
Halted
Line 35, Col 5
Tcl
NUM

```
    //Wait for SPORT interrupts
    while(1)
    {
    }



}


//Function Definitions ------------------------------------------------------
//---------------------------------------------------------------------------

//Initialize SRC0

void SRC0Init()
{
    *pSRCCTL0 = SRC0_SMODEIN0 | SRC0_SMODEOUT0 | SRC0_ENABLE ;
}


//-----------------------------------------------------------
//-----------------------------------------------------------

//Initialize SPORT0
void Sport0Init()
{
    *pDIV0 = 0x0018000e;
    *pSPCTL0 |= SPEN_A | SLEN24 | MSTR | OPMODE | SPTRAN;//SPORT0 configu
                                                        //Internal clock
                                                        //I2S mode
                                                        //Transmitter
                                                        //Word length 24
}
//-----------------------------------------------------------
//-----------------------------------------------------------

//Initialize SPORT1
void Sport1Init()
{
    *pDIV1 = 0x0090000e;
    *pSPCTL1 |= SPEN_A | MSTR | SLEN24 | OPMODE | DITFS;        //SPORT1
                                                        //Internal clock
                                                        //I2S mode
                                                        //Receiver
                                                        //Word length 24
}
//-----------------------------------------------------------
//-----------------------------------------------------------


void Sport0TxIsr(int dummy)
{
```

```
RxBuffer
          RxBuffer
[0B816C] 127          136
[0B816E] 138          148
[0B8170] 150          160
[0B8172] 160          173
[0B8174] 173          175
[0B8176] 175          163
[0B8178] 163          152
[0B817A] 152          141
[0B817C] 139          130
```

```
TxBuffer
          TxBuffer
[0B8004] 1            2
[0B8006] 3            4
[0B8008] 5            6
[0B800A] 7            8
[0B800C] 9            10
[0B800E] 11           12
[0B8010] 13           14
```

```
Loading: "C:\Aseem\SRC Test\Debug\SRC SPORT0 to SPORT1.dxe"...
Load complete.

>>
```

Figure 13. VisualDSP++ Screen Shot for Sample Rate Ratio of Fs in = 6 Fs out*

Code for SRC example 1 and SRC example 2 are provided in the accompanying `.ZIP` file.

References

[1] *ADSP-2136x SHARC Processor Hardware Reference.* Revision 0.3, February 2005. Analog Devices, Inc.

[2] *ADSP-21364 EZ-KIT Lite Evaluation System Manual.* Analog Devices, Inc.

[3] *A Stereo Asynchronous Digital Sample-Rate Converter for Digital Audio.* Robert Adams and Tom Kwan. IEEE Journal of solid state circuits. Vol. 29, No. 4, April 1994

[4] *AD1896 192 kHz Stereo Asynchronous Sample Rate Converter* Data Sheet. Rev. 0, Analog Devices, Inc.

Document History

Revision	**Description**
Rev 1 – April 25, 2005 *by Aseem Vasudev Prabhugaonkar*	Initial release