DSP

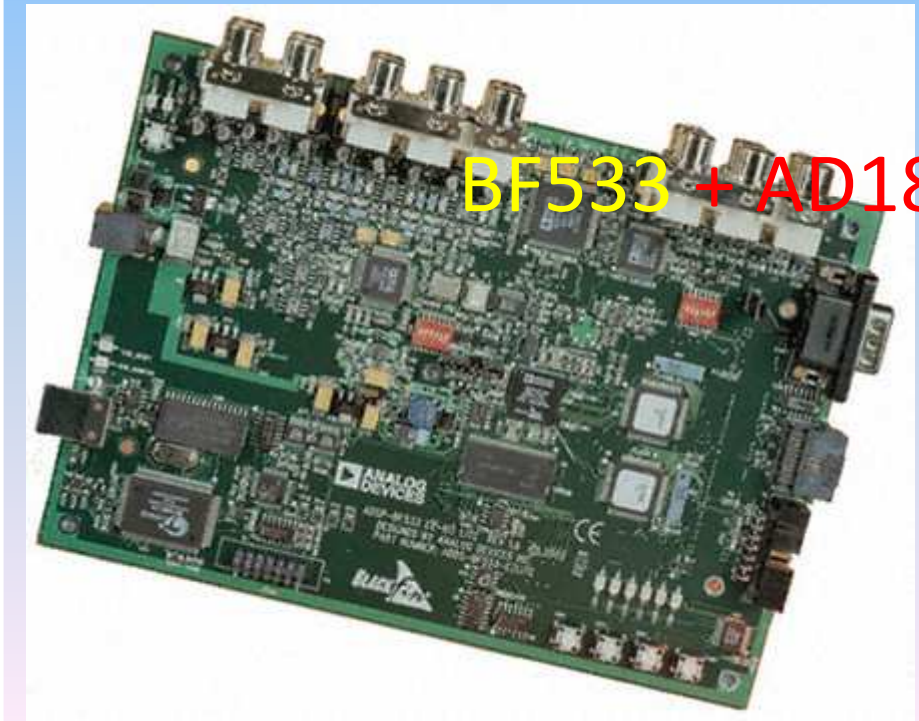
Lab. 2 – Audio signals with the Blackfin BF533 DSP

Stefan Ataman

2013 – 2014

Blackfin development

using the BF533 EZ-KIT Lite

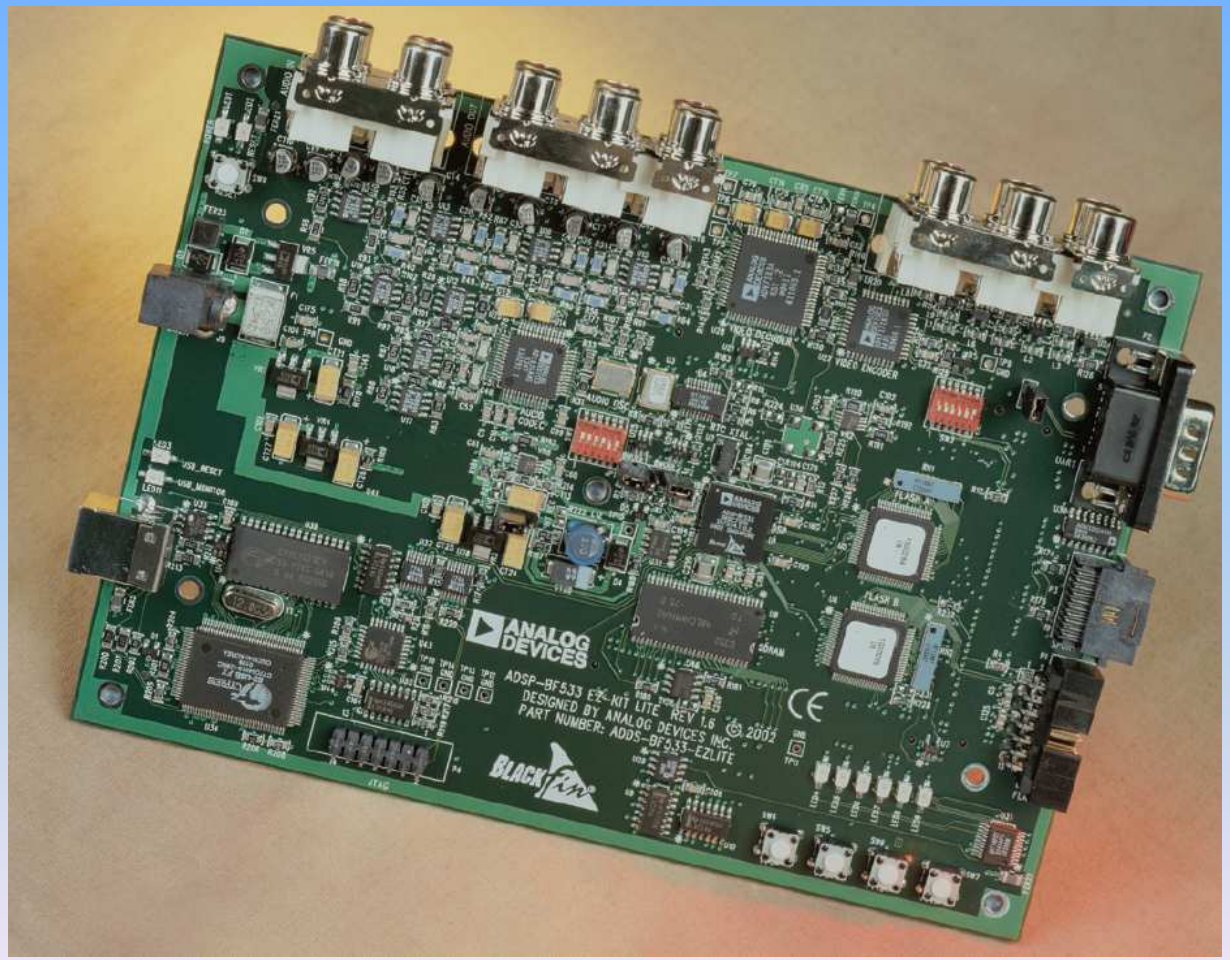


BF533 + AD1836 audio codec

In this lab we want to:

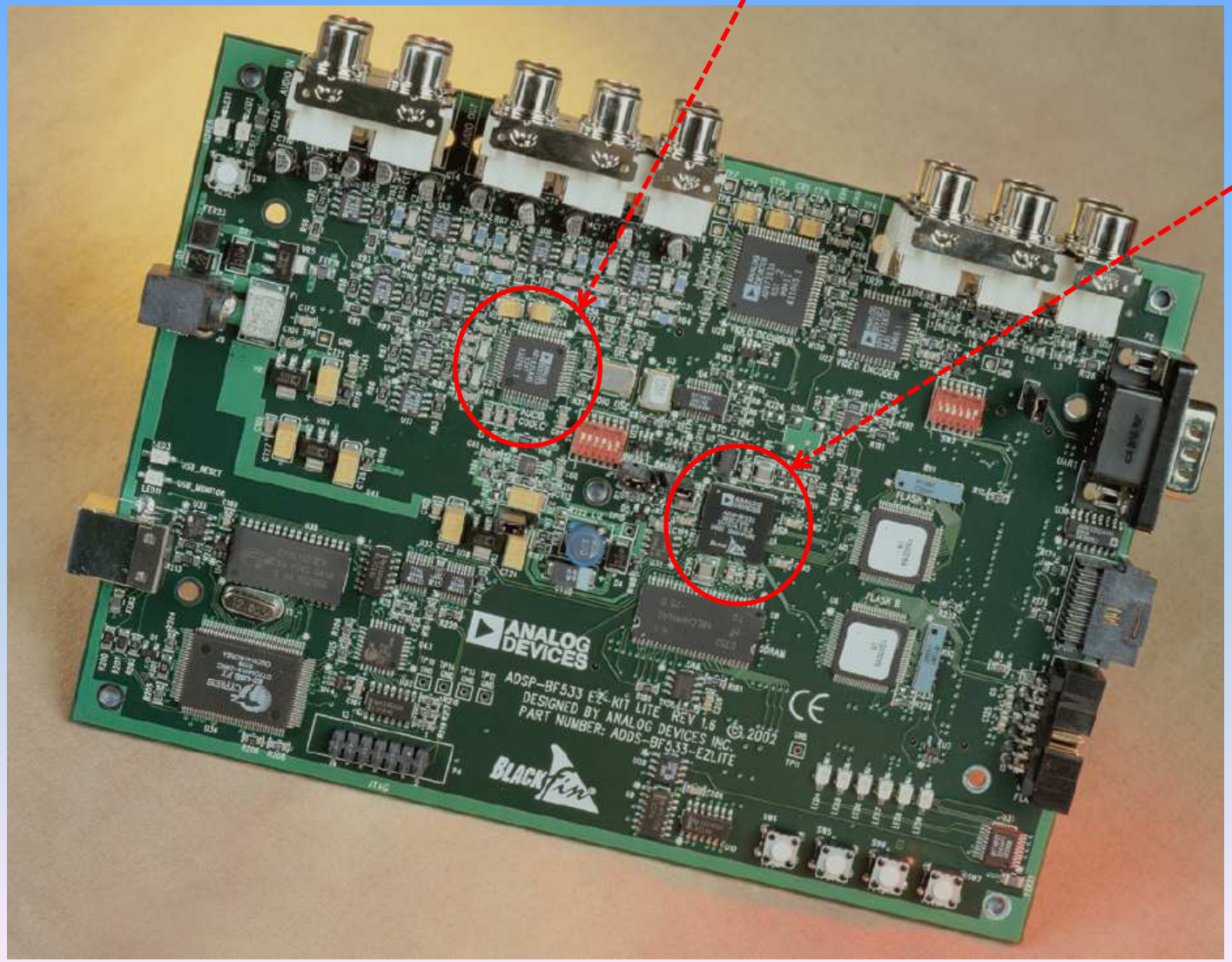
- 1) understand the hardware needed for audio signal processing
- 2) open a talk-through project in C
- 3) play around with the channels

Hardware first



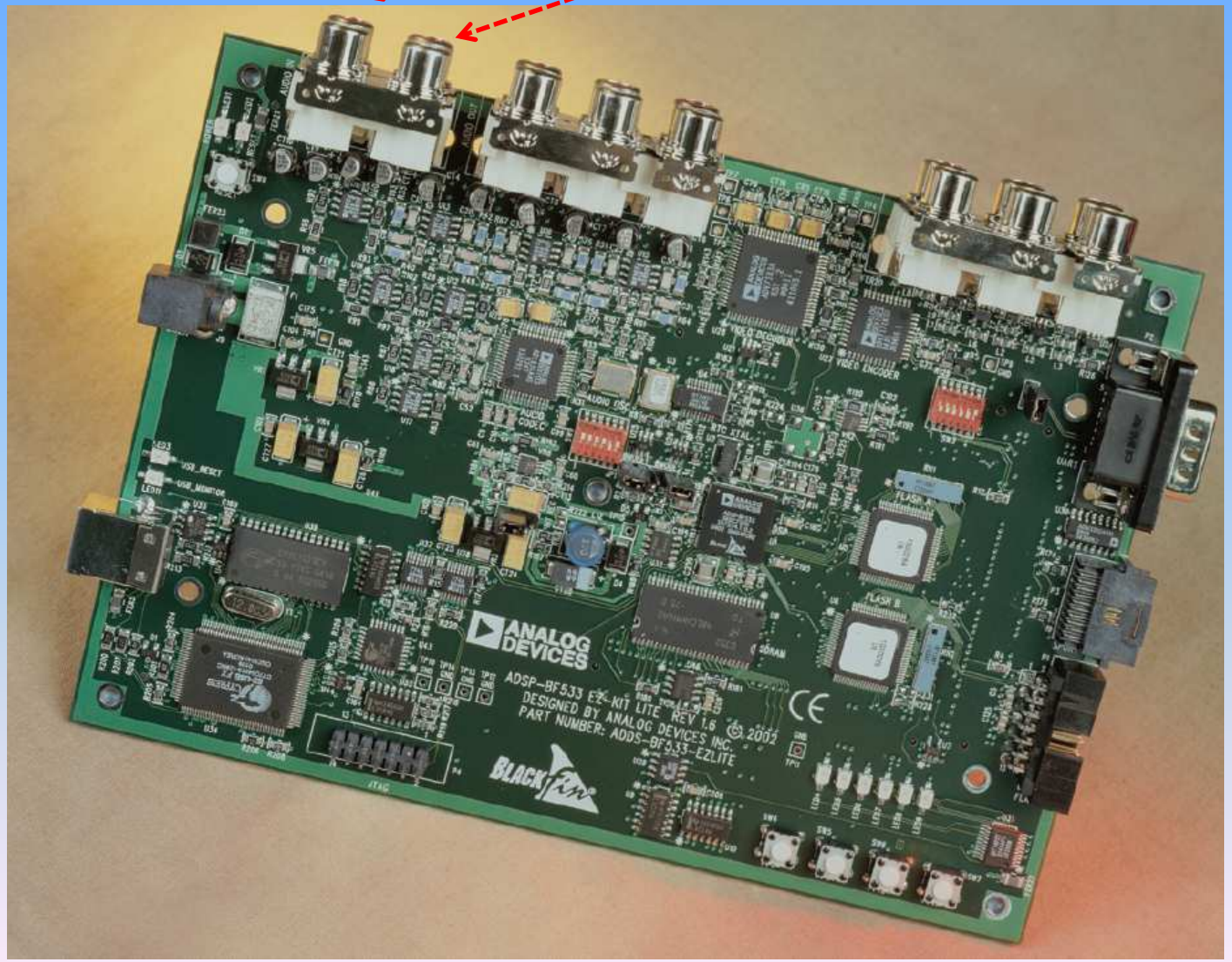
Identify the hardware:

- the Blackfin BF533 DSP
- the AD1836 audio codec
- the Channel0 and Channel1 stereo inputs
- the Channel0 and Channel1 stereo outputs


the AD1836 audio codec
the BF533 DSP
ANALOG DEVICES
ADSP-BF533 EZ-KIT LITE REV 1.6 © 2002
DESIGNED BY ANALOG DEVICES INC.
PART NUMBER: ADDS-BF533-EZLITE
BLACKfin

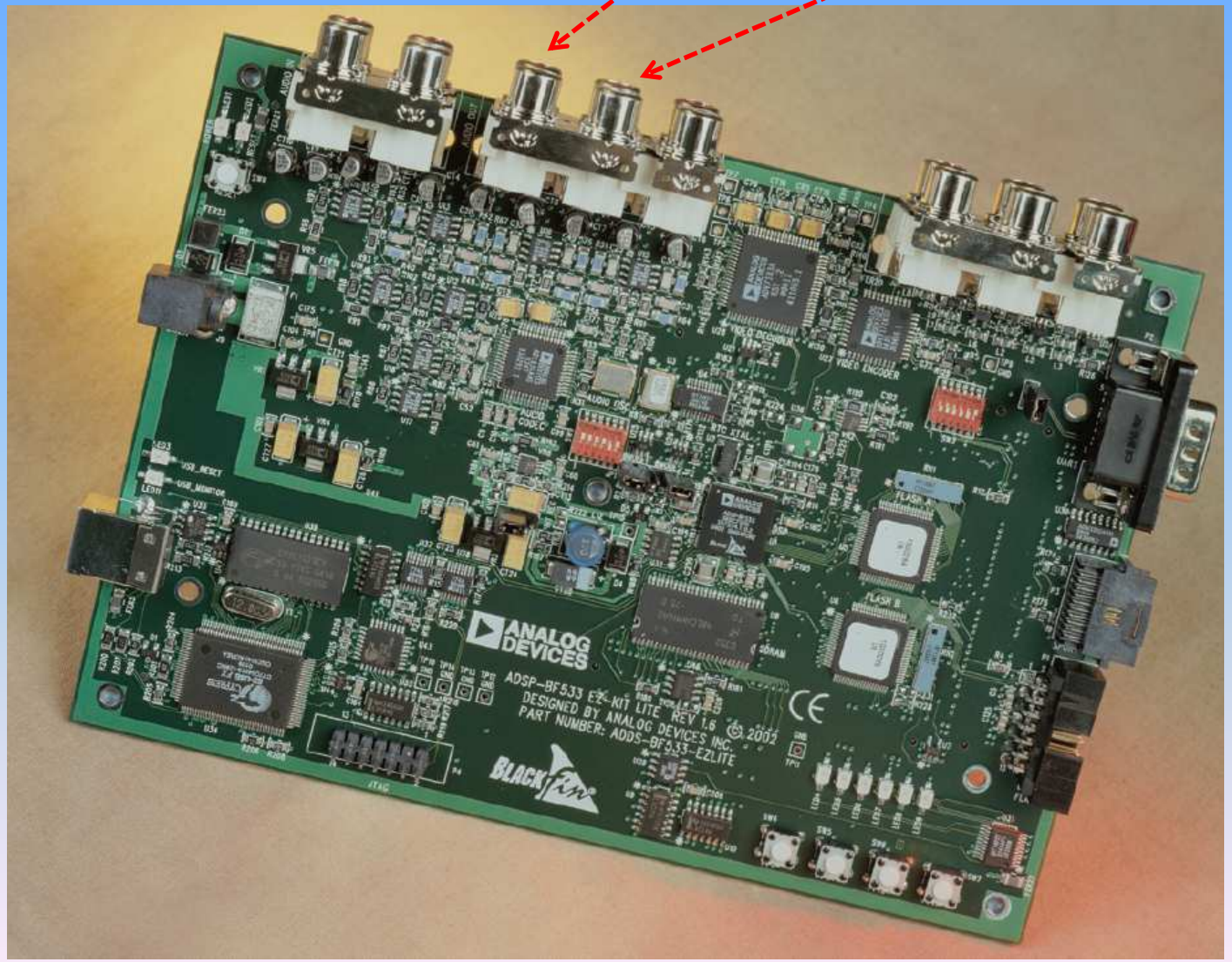
the Channel1 audio input

the Channel0 audio input



the Channel0 audio output

the Channel1 audio output



Audio Interface

- 3 channel of stereo audio output
- 2 channel of multi-channel of audio input is provided by the AD1836 audio codec
- Input sample rate of 96 kHz

- SPORT0 interface of the processor links with the stereo audio data input and output pins of the AD1836 codec.

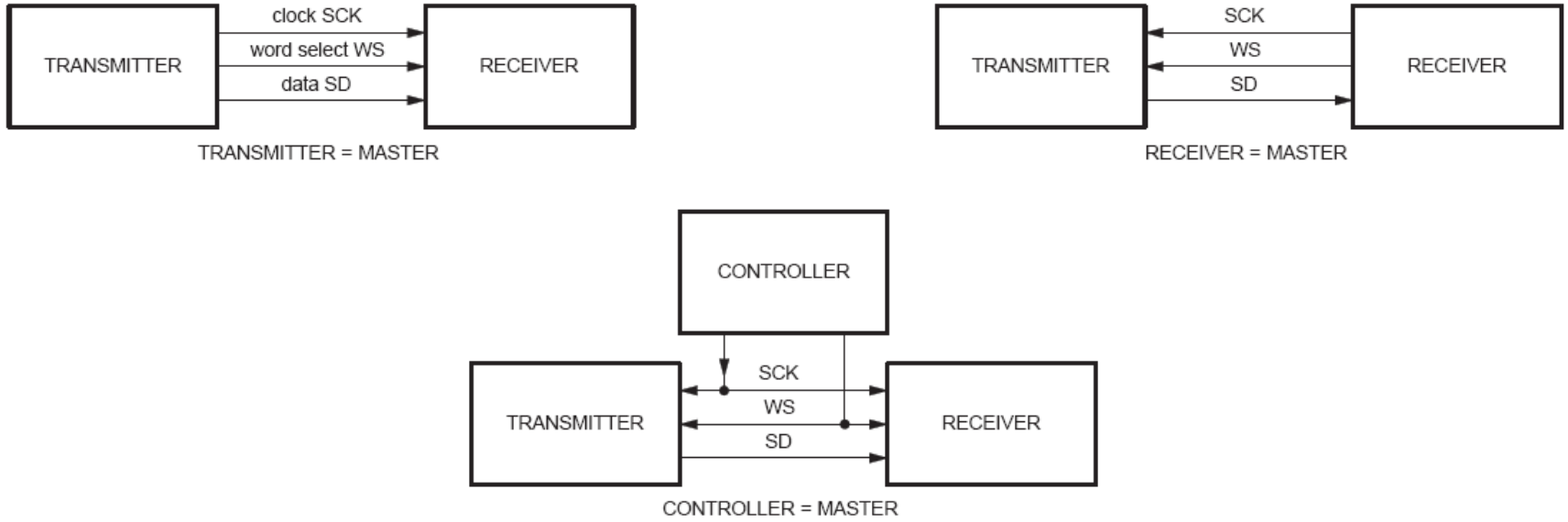
Audio Interface

- Processor is capable of transferring data to the audio codec in:
  - TDM – Time division multiplexed or
    - Operates at a maximum of 48 kHz sample rate but allows simultaneous use of all input and output channels
  - TWI – Twin Wire Interface mode
    - The TWI mode allows the codec to operate at a 96 kHz sample rate but limits the output channels to two

The I^2S bus

- I2S, also known as Inter-IC Sound, Integrated Interchip Sound, or IIS, is an electrical serial bus interface standard used for connecting digital audio devices together. It is most commonly used to carry PCM information between the CD transport and the DAC in a CD player. The I2S bus separates clock and data signals, resulting in a very low jitter connection.

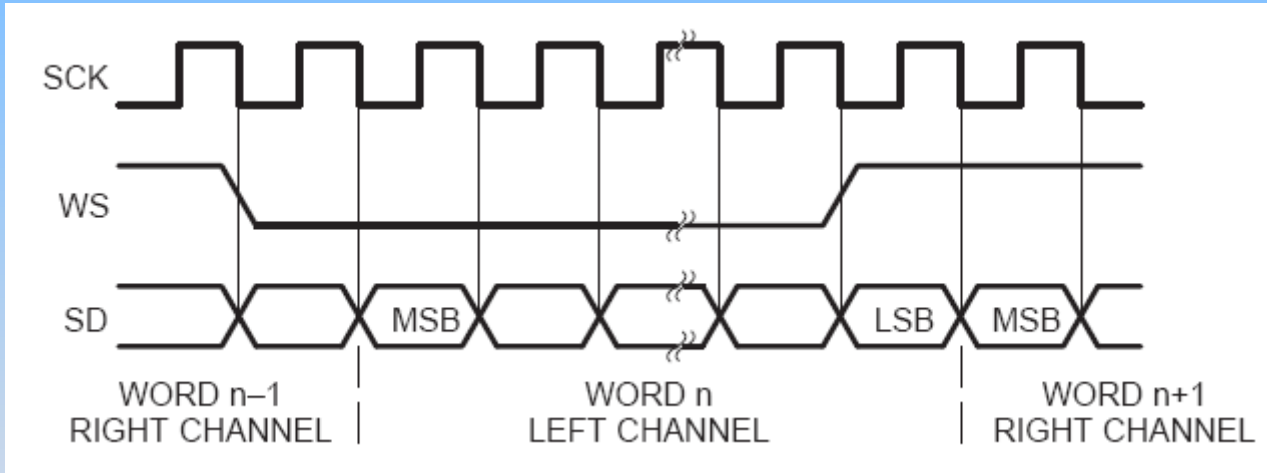
The I²S bus


TRANSMITTER
clock SCK
word select WS
data SD
RECEIVER
TRANSMITTER = MASTER
TRANSMITTER
SCK
WS
SD
RECEIVER
RECEIVER = MASTER
CONTROLLER
TRANSMITTER
SCK
WS
SD
RECEIVER
CONTROLLER = MASTER

The I^2S bus

- I^2S data is sent from MSB to LSB starting on the second bit clock cycle after the word select clock transition. Transmitting MSB first allows both the Transmitting and Receiving devices to not care what the audio precision of the remote device is. If the Transmitter is sending 32 bits per channel to a device with only 24 bits of internal precision, the Receiver may simply ignore the extra bits of precision by not storing the bits past the 24th bit.

The I²S bus


SCK
WS
SD
MSB
LSB
MSB
WORD n–1
RIGHT CHANNEL
WORD n
LEFT CHANNEL
WORD n+1
RIGHT CHANNEL

Open the talk-through C project in Visual DSP++ IDE

C Talk-through I^2S

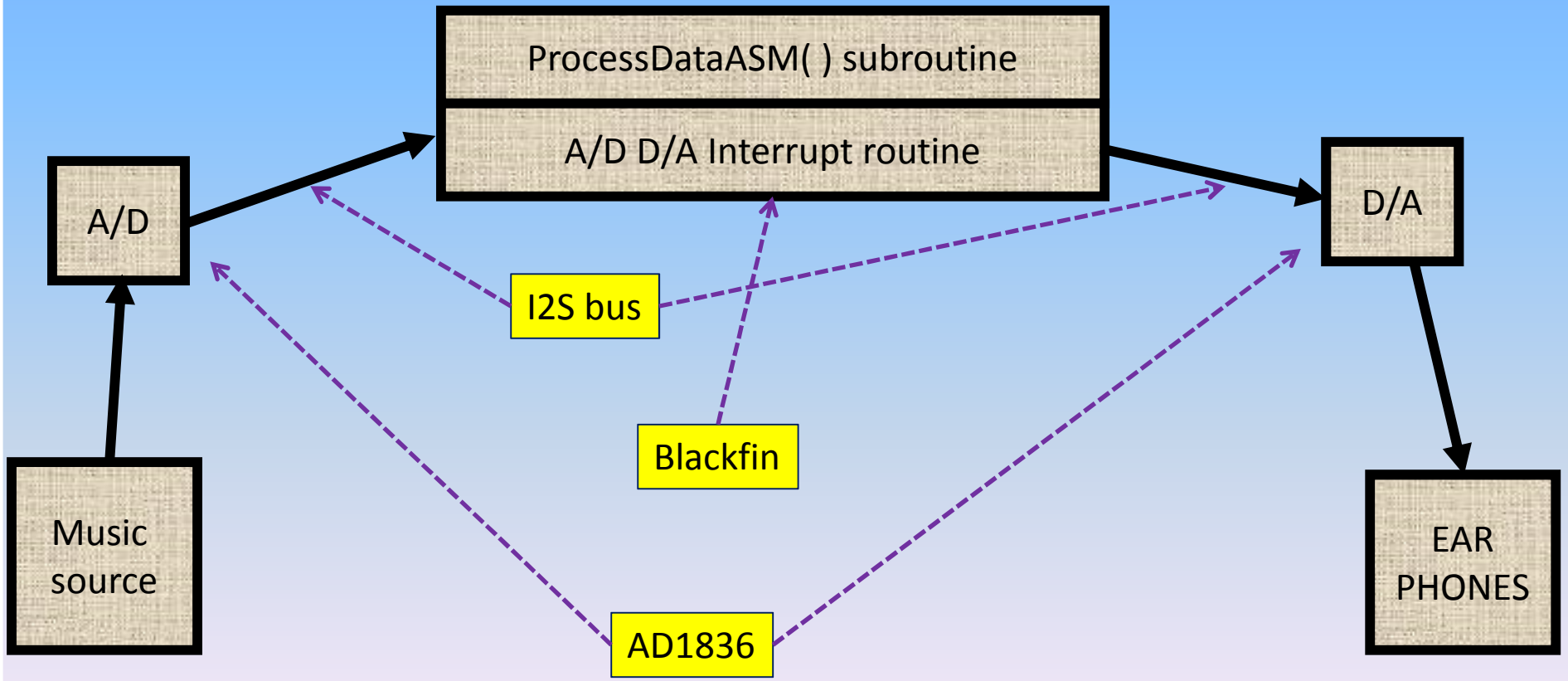
Open a Project

- From the File menu, choose Open and then Project
  - VisualDSP++ displays the Open Project dialog box.
- From campus download the archive :
  - Audio_codec_talkthrough_C.zip
- Unpack it to your own folder, *e.g.*
  - d:\your_name_GrXX\Audio_talkthrough

The project C_talkthrough_I2S

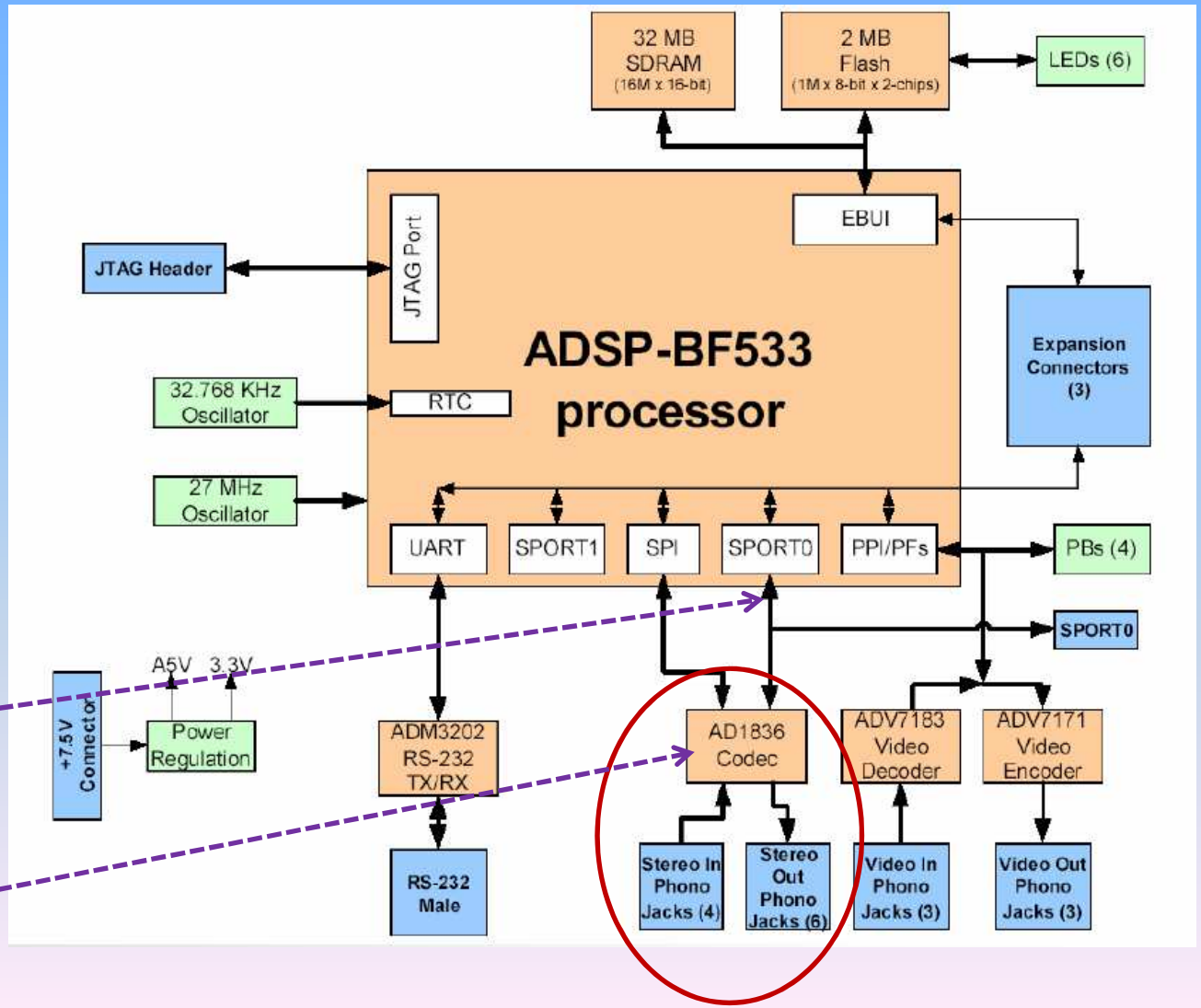
code structure

Data processing path



ADSP-BF533 EZ-KIT Lite

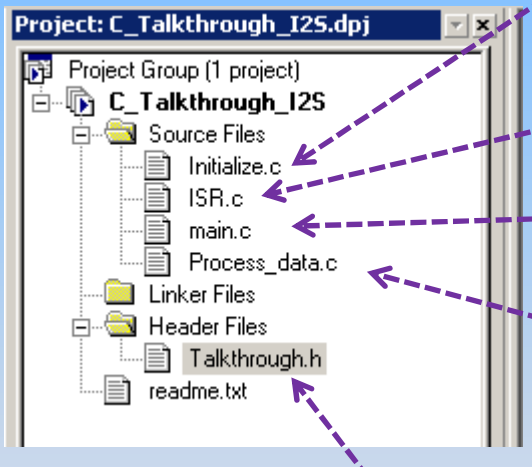
Real-time audio processing



I2S bus

AD1836

The project C_talkthrough_I2S



initializations for the processor, codec and stuff

the interrupt service routines

the main file

the data processing module

the header file with parameters and function prototypes

The main.c file

main.c

```
                    DAC_VOLUME_5     | 0x000,
                    ADC_CONTROL_1    | 0x000,
                    ADC_CONTROL_2    | 0x000,
                    ADC_CONTROL_3    | 0x000

};
// SPORT0 DMA transmit buffer
volatile int iTxBuffer1[4];
// SPORT0 DMA receive buffer
volatile int iRxBuffer1[4];


//-----------------------------------------------------------------------------//
// Function:    main                                                           //
//                                                                             //
// Description: After calling a few initalization routines, main() just        //
//              waits in a loop forever.  The code to process the incoming     //
//              data can be placed in the function Process_Data() in the       //
//              file "Process_Data.c".                                         //
//-----------------------------------------------------------------------------//
void main(void)
{

    sysreg_write(reg_SYSCFG, 0x32);     //Initialize System Configuration Register
    Init_EBIU();
    Init_Flash();
    Init1836();
    Init_Sport0();
    Init_DMA();
    Init_Interrupts();
    Enable_DMA_Sport0();

    while(1);
}
```

the main function

The main function

```
void main(void)
{
   sysreg_write(reg_SYSCFG, 0x32);
   Init_EBIU();
   Init_Flash();
   Init1836();
   Init_Sport0();
   Init_DMA();
   Init_Interrupts();
   Enable_DMA_Sport0();
       while(1);
}
```

Init System Configuration Register which controls the configuration of the processor

Init of External Bus Interface Unit

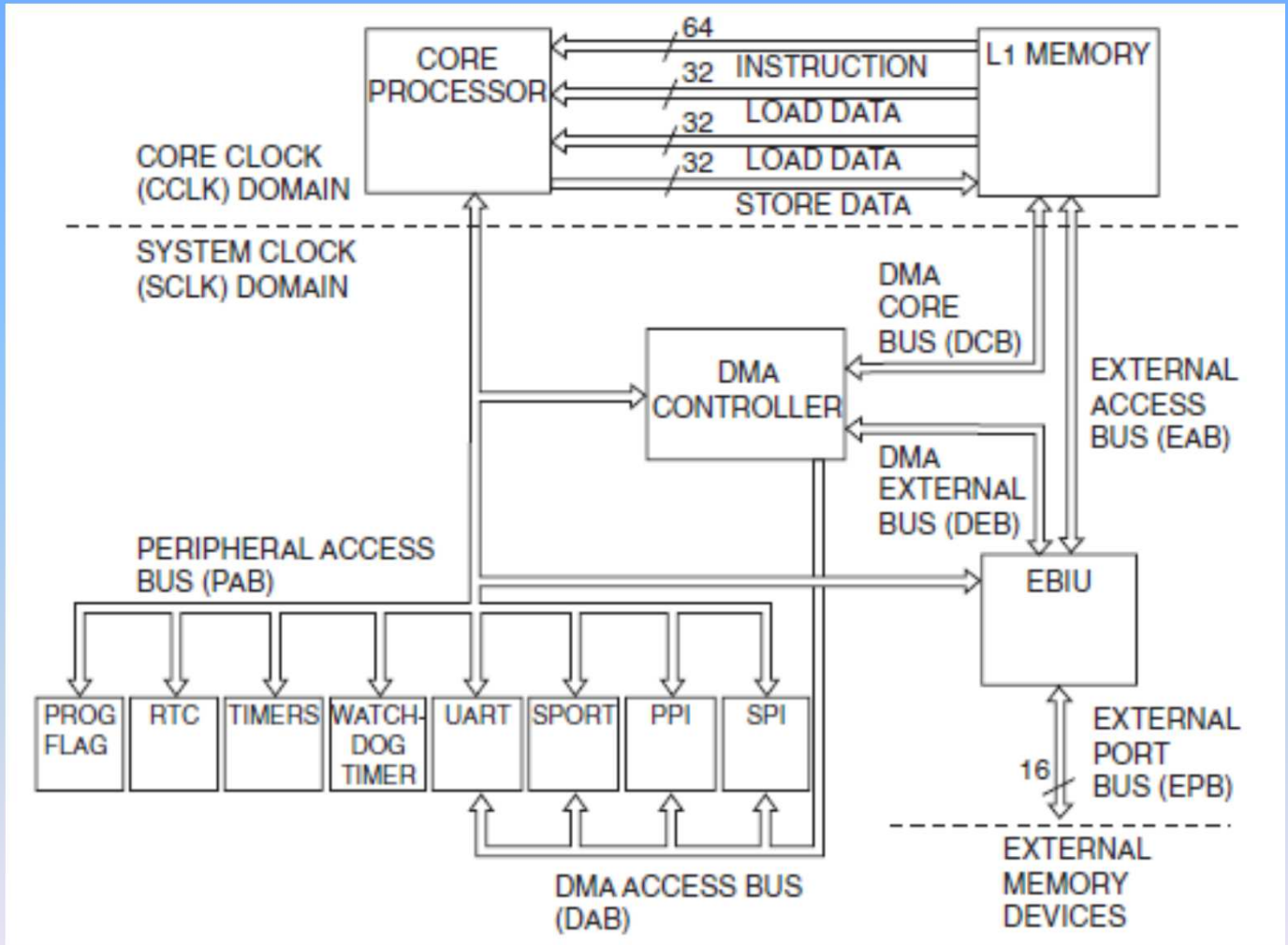
initialize the AD1836 audio codec

initialize the serial port (SPORT0)

initialize the interrupts

infinite wait loop

The main function



Blackfin BF533 processor memory architecture

The Process_Data.c file

Process_data.c

```
#include "Talkthrough.h"

//-----------------------------------------------------------------------------//
// Function:    Process_Data()                                                 //
//                                                                             //
// Description: This function is called from inside the SPORT0 ISR every       //
//              time a complete audio frame has been received. The new         //
//              input samples can be found in the variables iChannel0LeftIn,//
//              iChannel0RightIn, iChannel1LeftIn and iChannel1RightIn         //
//              respectively. The processed data should be stored in           //
//              iChannel0LeftOut, iChannel0RightOut, iChannel1LeftOut,         //
//              iChannel1RightOut, iChannel2LeftOut and iChannel2RightOut      //
//              respectively.                                                  //
//-----------------------------------------------------------------------------//
void Process_Data(void)
{
    /*
    iChannel0LeftOut = iChannel0LeftIn;
    iChannel0RightOut = iChannel0RightIn;
    iChannel1LeftOut = iChannel1LeftIn;
    iChannel1RightOut = iChannel1RightIn;
    */

}
```

all data processing happens inside this function

The Process_Data.c file

- all data processing is done here
- input samples from Channel0:

- `iChannel0LeftIn (int) ->` the input samples from Channel0, Left

- `iChannel0RightIn (int) ->` the input samples from Channel0, Right

The Process_Data.c file

- input samples from Channel1:

- `iChannel1LeftIn (int) ->` the input samples from Channel1, Left

- `iChannel1RightIn (int) ->` the input samples from Channel1, Right

The Process_Data.c file

- output samples from Channel0:

- **`iChannel0LeftOut (int) ->`** the output samples from Channel0, Left

- **`iChannel0RightOut (int) ->`** the output samples from Channel0, Right

The Process_Data.c file

- output samples from Channel1:

- **`iChannel1LeftOut (int) ->`** the output samples from Channel1, Left

- **`iChannel1RightOut (int) ->`** the output samples from Channel1, Right

Comment out the original lines

Process_data.c *

```
#include "Talkthrough.h"

//-------------------------------------------------------------------------//
// Function:    Process_Data()                                             //
//                                                                         //
// Description: This function is called from inside the SPORT0 ISR every   //
//              time a complete audio frame has been received. The new     //
//              input samples can be found in the variables iChannel0LeftIn,//
//              iChannel0RightIn, iChannel1LeftIn and iChannel1RightIn     //
//              respectively. The processed data should be stored in       //
//              iChannel0LeftOut, iChannel0RightOut, iChannel1LeftOut,     //
//              iChannel1RightOut, iChannel2LeftOut and iChannel2RightOut  //
//              respectively.                                              //
//-------------------------------------------------------------------------//

void Process_Data(void)
{
    /*
    iChannel0LeftOut = iChannel0LeftIn;
    iChannel0RightOut = iChannel0RightIn;
    iChannel1LeftOut = iChannel1LeftIn;
    iChannel1RightOut = iChannel1RightIn;
    */
}
```

We have now:

Process_data.c

```
#include "Talkthrough.h"

//-------------------------------------------------------------------------//
// Function:    Process_Data()                                             //
//                                                                         //
// Description: This function is called from inside the SPORT0 ISR every   //
//              time a complete audio frame has been received. The new     //
//              input samples can be found in the variables iChannel0LeftIn,//
//              iChannel0RightIn, iChannel1LeftIn and iChannel1RightIn     //
//              respectively. The processed data should be stored in       //
//              iChannel0LeftOut, iChannel0RightOut, iChannel1LeftOut,     //
//              iChannel1RightOut, iChannel2LeftOut and iChannel2RightOut  //
//              respectively.                                              //
//-------------------------------------------------------------------------//
void Process_Data(void)
{
    iChannel0LeftOut = iChannel0LeftIn;
    iChannel0RightOut = iChannel0RightIn;

    iChannel1LeftOut = iChannel1LeftIn;
    iChannel1RightOut = iChannel1RightIn;

}
```

Channel0 connect

Channel1 connect

You've just connected:

- Channel0:
  - Channel0 Left input to Channel0 Left output
  - Channel0 Right input to Channel0 Right output
- Channel1:
  - Channel1 Left input to Channel0 Left output
  - Channel1 Right input to Channel1 Right output

- we'll use only Channel0, though.

Plug in some music!

- Plug an iPhone (iPad, iWhatever or any sound source) on the

  Channel0 input

and headphones on the

  Channel0 output

How to connect

- plug the RCA cable on Channel0 input (Left and Right)
- plug another RCA cable on Channel0 output (Left and Right)
- plug in a sound source on the RCA input cable
- use a female to female adapter at the output
- plug in a headphone


jack to RCA cable
signal source
jack to RCA cable

Build the project (F7)

Analog Devices VisualDSP++ - [Target: ADSP-BF533 EZ-KIT Lite via Debug Agent] - [Project:

File Edit Session View Project Register Memory Debug Settings Tools Window Help

C_Talkthrough_I2S | Debug

Build Project (F7)

Project: C_Talkthrough_I2S.dpj

- Project Group (1 project)
  - C_Talkthrough_I2S
    - Source Files
      - Initialize.c
      - ISR.c
      - main.c
      - Process_data.c
    - Linker Files
    - Header Files
      - Talkthrough.h
    - readme.txt

Process_data.c

```
#include "Talkthrough.h"

//-------------------------
// Function:     Process_Dat
//
// Description:  This functi
//               time a comp
//               input sampl
//               iChannel0Ri
//               respectivel
//               iChannel0Le
//               iChannel1Ri
//               respectivel
//-------------------------
void Process_Data(void)
{
```

Project

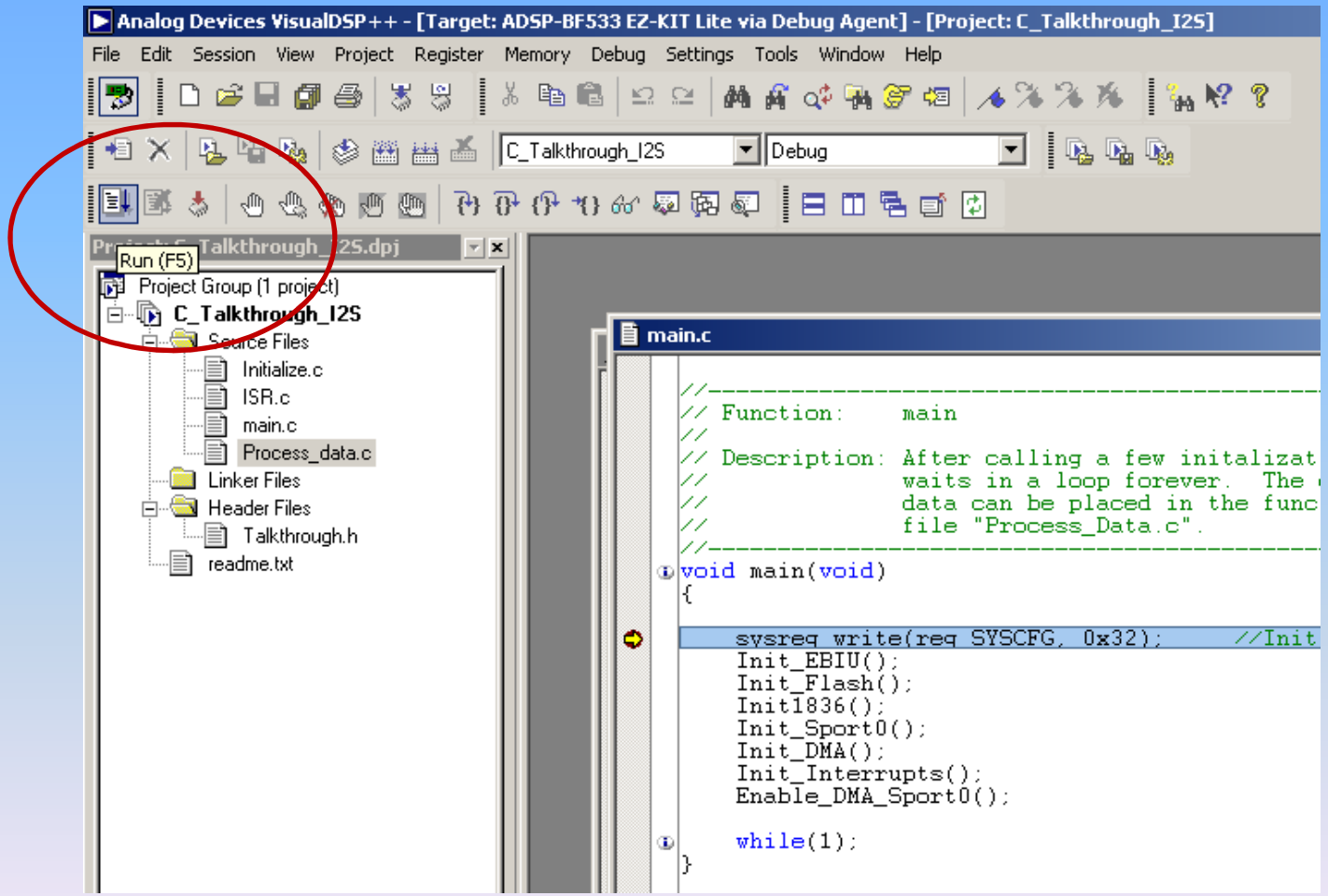
Output Window

```
Loading: "C:\stefan\DSP_lab\Audio Codec Talkthrough - I2S (C)\Debug\C_Talkthrough_I2S.dxe"...
Load complete.
```

Console | Build

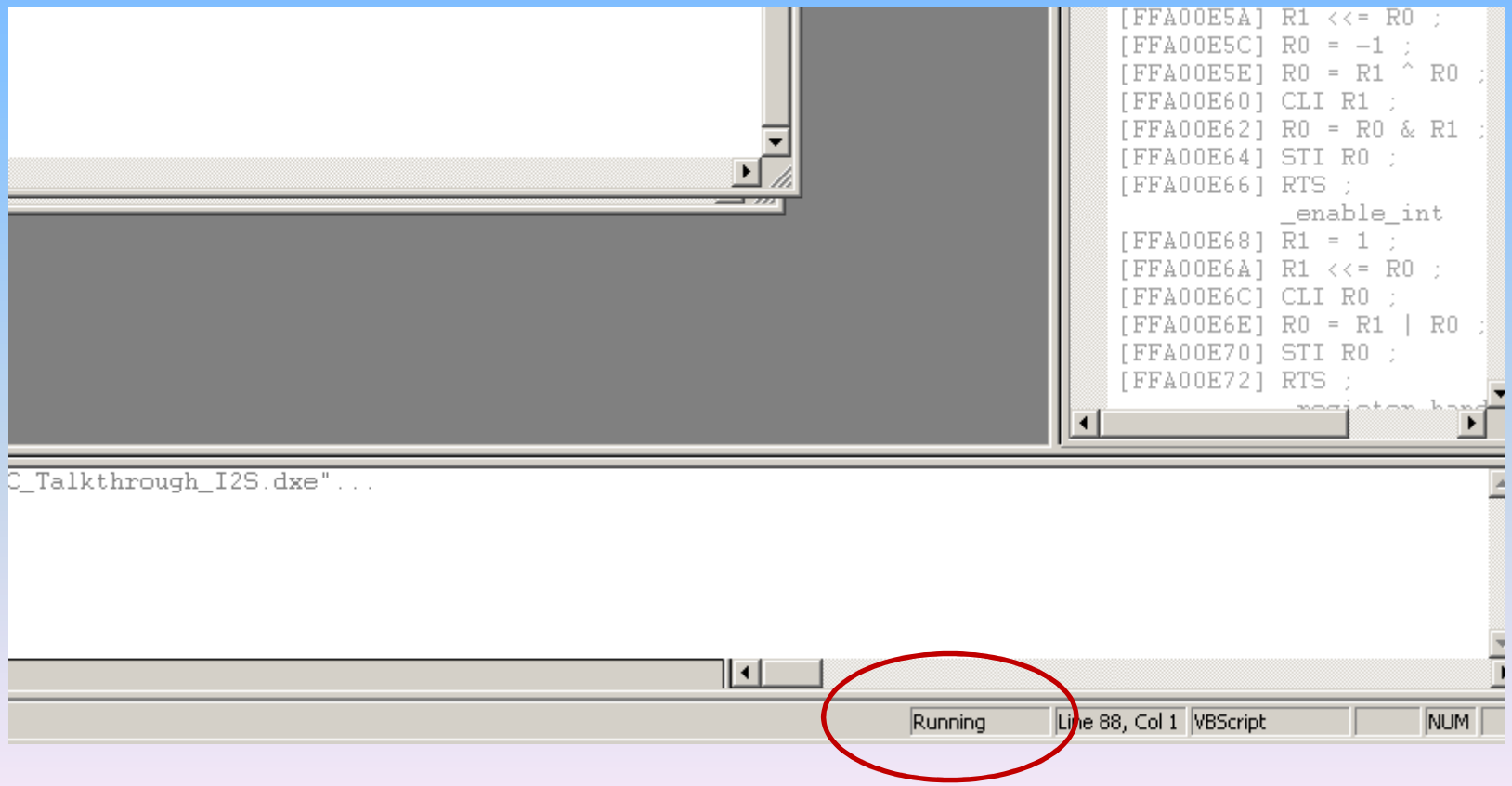
Ready

Run the project (F5)



While the project runs…

- you should hear music!



Let's DJ a little bit…

- on Channel0 left:
  - put the Channel0 right input
- on Channel0 right:
  - put the Channel0 left input

Let's DJ a little bit more…

- put on Channel0 left:
  - the sum of Channel0 left and right
- put on Channel0 right:
  - the difference of Channel0 left and right

The Process_Data.c file becomes:

Process_data.c

```c
#include "Talkthrough.h"

//-----------------------------------------------------------------------//
// Function:    Process_Data()                                           //
//                                                                       //
// Description: This function is called from inside the SPORT0 ISR every //
//              time a complete audio frame has been received. The new   //
//              input samples can be found in the variables iChannel0LeftIn,//
//              iChannel0RightIn, iChannel1LeftIn and iChannel1RightIn   //
//              respectively. The processed data should be stored in     //
//              iChannel0LeftOut, iChannel0RightOut, iChannel1LeftOut,   //
//              iChannel1RightOut, iChannel2LeftOut and iChannel2RightOut //
//              respectively.                                            //
//-----------------------------------------------------------------------//
void Process_Data(void)
{

    //iChannel0LeftOut = iChannel0LeftIn;
    //iChannel0RightOut = iChannel0RightIn;

    iChannel0LeftOut = iChannel0LeftIn + iChannel0RightIn;

    iChannel0RightOut = iChannel0LeftIn - iChannel0RightIn;


    iChannel1LeftOut = iChannel1LeftIn;
    iChannel1RightOut = iChannel1RightIn;


}
```

Your data is represented on 24 bits

- your registers are 32 bits wide
- therefore if you mask your audio data with:

0 0 F F F F F F

it should stay unchanged.

Try it!

Process_data.c *

```c
#include "Talkthrough.h"

//-----------------------------------------------------------------------//
// Function:    Process_Data()                                           //
//                                                                       //
// Description: This function is called from inside the SPORT0 ISR every //
//              time a complete audio frame has been received. The new   //
//              input samples can be found in the variables iChannel0LeftIn,//
//              iChannel0RightIn, iChannel1LeftIn and iChannel1RightIn   //
//              respectively. The processed data should be stored in     //
//              iChannel0LeftOut, iChannel0RightOut, iChannel1LeftOut,   //
//              iChannel1RightOut, iChannel2LeftOut and iChannel2RightOut //
//              respectively.                                            //
//-----------------------------------------------------------------------//
void Process_Data(void)
{

    iChannel0LeftOut = iChannel0LeftIn & 0x00FFFFFF;

    iChannel0RightOut = iChannel0RightIn & 0x00FFFFFF;


    iChannel1LeftOut = iChannel1LeftIn;
    iChannel1RightOut = iChannel1RightIn;


}
```

Masking your samples

- now change the mask. Try first:

0 0 F F F F 0 0

Question Set #1

- what is the meaning of this mask?
- what do you notice (in audio)?
- can you explain this result?

Masking your samples

- now you can try:

0 0 0 F F F F F

Question Set #2

- what is the meaning of this mask?
- what do you notice (in audio)?
- can you explain this result?

Louder, DJ, louder!

- how do you amplify your signal?

- reminder: in C you have this:

<< (shift left)

(for example `myVariable << 1;`)

- amplify the left channel!

Lost? Ok, this is how to do it:

Process_data.c

```c
#include "Talkthrough.h"

//-------------------------------------------------------------------------//
// Function:    Process_Data()                                             //
//                                                                         //
// Description: This function is called from inside the SPORT0 ISR every   //
//              time a complete audio frame has been received. The new     //
//              input samples can be found in the variables iChannel0LeftIn,//
//              iChannel0RightIn, iChannel1LeftIn and iChannel1RightIn     //
//              respectively. The processed data should be stored in       //
//              iChannel0LeftOut, iChannel0RightOut, iChannel1LeftOut,     //
//              iChannel1RightOut, iChannel2LeftOut and iChannel2RightOut  //
//              respectively.                                              //
//-------------------------------------------------------------------------//
void Process_Data(void)
{

    // left-shift Channel0LeftIn
    iChannel0LeftOut = iChannel0LeftIn <<1 & 0x00FFFF00;

    iChannel0RightOut = iChannel0RightIn & 0x00FFFFFF;


    iChannel1LeftOut = iChannel1LeftIn;
    iChannel1RightOut = iChannel1RightIn;


}
```

Is it louder, DJ?

If not, shift more…

Process_data.c

```
#include "Talkthrough.h"

//-----------------------------------------------------------------------------//
// Function:    Process_Data()                                                 //
//                                                                             //
// Description: This function is called from inside the SPORT0 ISR every       //
//              time a complete audio frame has been received. The new         //
//              input samples can be found in the variables iChannel0LeftIn,//
//              iChannel0RightIn, iChannel1LeftIn and iChannel1RightIn         //
//              respectively. The processed data should be stored in           //
//              iChannel0LeftOut, iChannel0RightOut, iChannel1LeftOut,         //
//              iChannel1RightOut, iChannel2LeftOut and iChannel2RightOut      //
//              respectively.                                                  //
//-----------------------------------------------------------------------------//
void Process_Data(void)
{

    // left-shift Channel0LeftIn
    iChannel0LeftOut = iChannel0LeftIn <<4 & 0x00FFFFFF;

    iChannel0RightOut = iChannel0RightIn & 0x00FFFFFF;


    iChannel1LeftOut = iChannel1LeftIn;
    iChannel1RightOut = iChannel1RightIn;


}
```

Question Set #3

- what happens if we try to make it even louder?
- can you explain why?

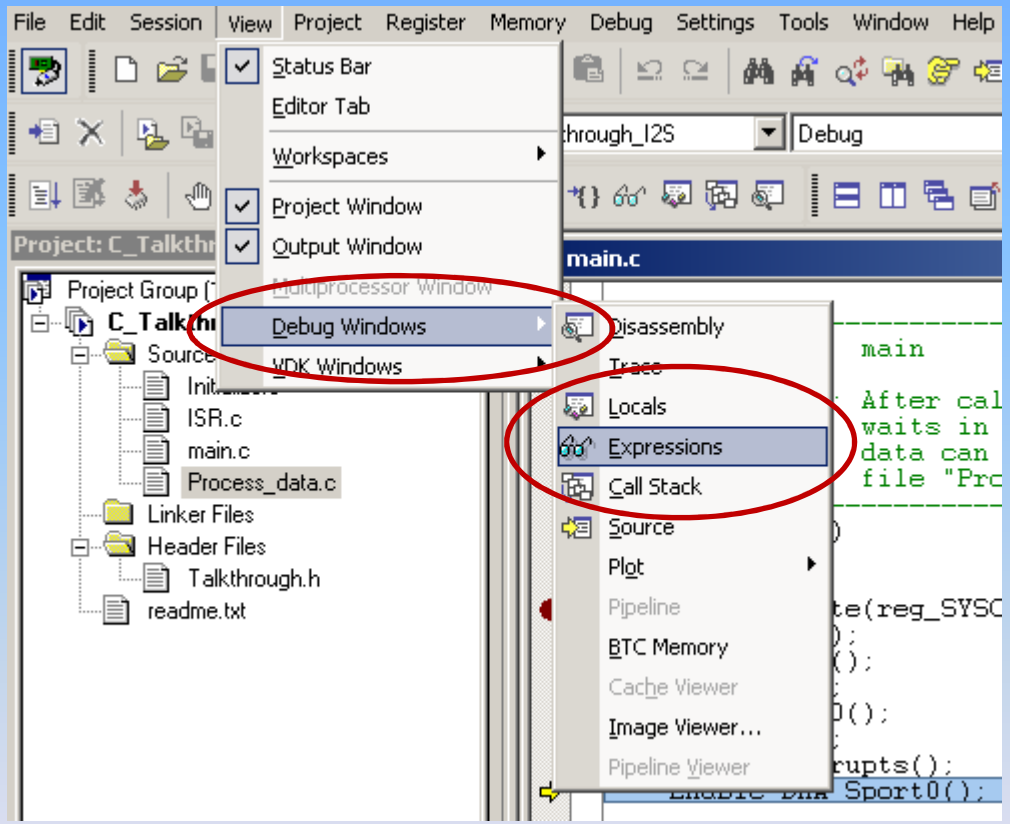
Debugging with the VisualDSP++

Breakpoints, Run, Step, Stop

How to debug:

- Set a breakpoint
- display the expressions you need
- run
- (stop at the breakpoint)
- step (if needed)

How to debug


File
Edit
Session
View
Project
Register
Memory
Debug
Settings
Tools
Window
Help
Status Bar
Editor Tab
Workspaces
Project Window
Output Window
Multiprocessor Window
Debug Windows
VDK Windows
Disassembly
Trace
Locals
Expressions
Call Stack
Source
Plot
Pipeline
BTC Memory
Cache Viewer
Image Viewer...
Pipeline Viewer
Debug
main.c
Project Group
Source
ISR.c
main.c
Process_data.c
Linker Files
Header Files
Talkthrough.h
readme.txt

Set the breakpoint

(double click in front of the code line)

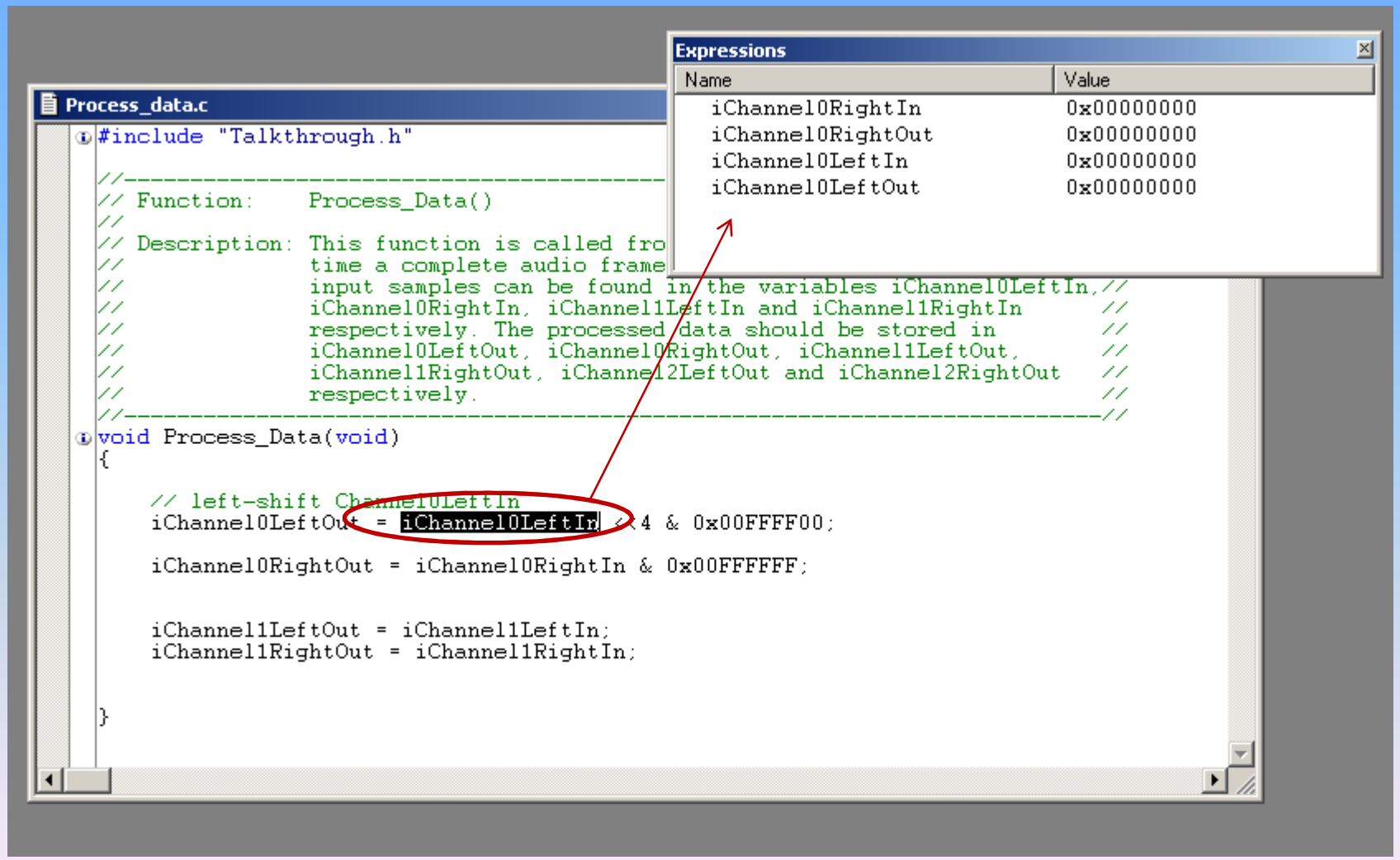
Expressions

Name	Value

Process_data.c

```
#include "Talkthrough.h"

//-----------------------------------------
// Function:     Process_Data()
//
// Description: This function is called fro
//              time a complete audio frame
//              input samples can be found in the variables iChannel0LeftIn,//
//              iChannel0RightIn, iChannel1LeftIn and iChannel1RightIn      //
//              respectively. The processed data should be stored in        //
//              iChannel0LeftOut, iChannel0RightOut, iChannel1LeftOut,      //
//              iChannel1RightOut, iChannel2LeftOut and iChannel2RightOut   //
//              respectively.                                               //
//-----------------------------------------------------------------------------//
void Process_Data(void)
{

    // Left-shift Channel0LeftIn
    iChannel0LeftOut = iChannel0LeftIn <<4 & 0x00FFFF00;

Breakpoint at "Process_data.c" 19, address 0xffa00e50.l0RightIn & 0x00FFFFFF;


    iChannel1LeftOut = iChannel1LeftIn;
    iChannel1RightOut = iChannel1RightIn;


}
```

Select and drag the variables you want to watch.



Now run! (F5)

(you will stop at the breakpoint, of course!)

Process_data.c

```
#include "Talkthrough.h"

//-----------------------------------------
// Function:     Process_Data()
//
// Description: This function is called fro
//              time a complete audio frame
//              input samples can be found in the variables iChannel0LeftIn,//
//              iChannel0RightIn, iChannel1LeftIn and iChannel1RightIn      //
//              respectively. The processed data should be stored in        //
//              iChannel0LeftOut, iChannel0RightOut, iChannel1LeftOut,      //
//              iChannel1RightOut, iChannel2LeftOut and iChannel2RightOut   //
//              respectively.                                               //
//-----------------------------------------------------------------------------//
void Process_Data(void)
{

    // left-shift Channel0LeftIn
    iChannel0LeftOut = iChannel0LeftIn <<4 & 0x00FFFF00;

    iChannel0RightOut = iChannel0RightIn & 0x00FFFFFF;


    iChannel1LeftOut = iChannel1LeftIn;
    iChannel1RightOut = iChannel1RightIn;


}
```

Expressions

Name	Value
iChannel0LeftIn	0x0005681c
iChannel0LeftOut	0x00000000
iChannel0RightIn	0x00fb0b13
iChannel0RightOut	0x00000000

Now step! (F11)

Process_data.c

```
#include "Talkthrough.h"

//-----------------------------------------
// Function:     Process_Data()
//
// Description: This function is called fro
//              time a complete audio frame
//              input samples can be found in the variables iChannel0LeftIn,//
//              iChannel0RightIn, iChannel1LeftIn and iChannel1RightIn      //
//              respectively. The processed data should be stored in        //
//              iChannel0LeftOut, iChannel0RightOut, iChannel1LeftOut,      //
//              iChannel1RightOut, iChannel2LeftOut and iChannel2RightOut   //
//              respectively.                                               //
//-----------------------------------------------------------------------------//
void Process_Data(void)
{

    // left-shift Channel0LeftIn
    iChannel0LeftOut = iChannel0LeftIn <<4 & 0x00FFFF00;

    iChannel0RightOut = iChannel0RightIn & 0x00FFFFFF;


    iChannel1LeftOut = iChannel1LeftIn;
    iChannel1RightOut = iChannel1RightIn;


}
```

Expressions

Name	Value
iChannel0LeftIn	0x0005681c
iChannel0LeftOut	0x00568100
iChannel0RightIn	0x00fb0b13
iChannel0RightOut	0x00000000

Step again!

Process_data.c

```
#include "Talkthrough.h"

//-----------------------------------------
// Function:     Process_Data()
//
// Description: This function is called fro
//              time a complete audio frame
//              input samples can be found in the variables iChannel0LeftIn,//
//              iChannel0RightIn, iChannel1LeftIn and iChannel1RightIn      //
//              respectively. The processed data should be stored in        //
//              iChannel0LeftOut, iChannel0RightOut, iChannel1LeftOut,      //
//              iChannel1RightOut, iChannel2LeftOut and iChannel2RightOut   //
//              respectively.                                               //
//-----------------------------------------------------------------------------//
void Process_Data(void)
{

    // left-shift Channel0LeftIn
    iChannel0LeftOut = iChannel0LeftIn <<4 & 0x00FFFF00;

    iChannel0RightOut = iChannel0RightIn & 0x00FFFFFF;


    iChannel1LeftOut = iChannel1LeftIn;
    iChannel1RightOut = iChannel1RightIn;


}
```

Expressions

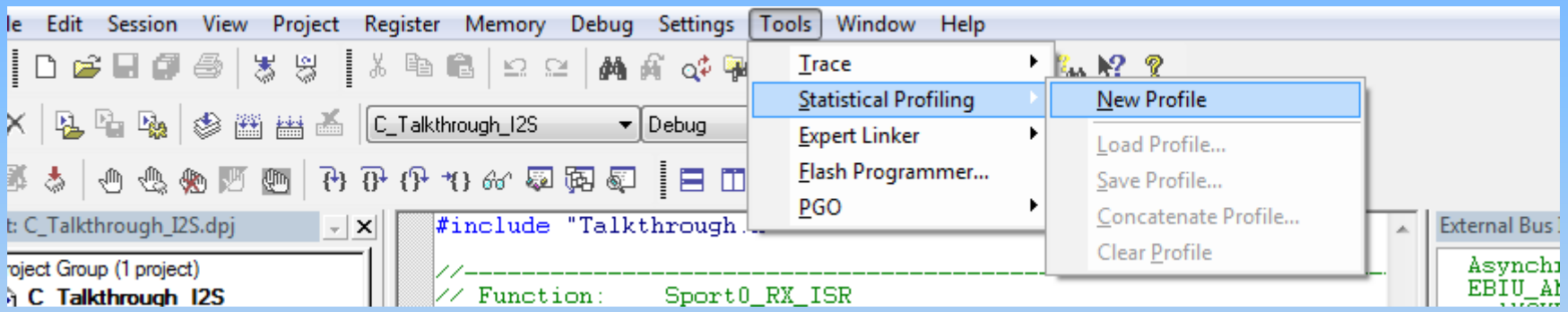
Name	Value
iChannel0LeftIn	0x0005681c
iChannel0LeftOut	0x00568100
iChannel0RightIn	0x00fb0b13
iChannel0RightOut	0x00fb0b13

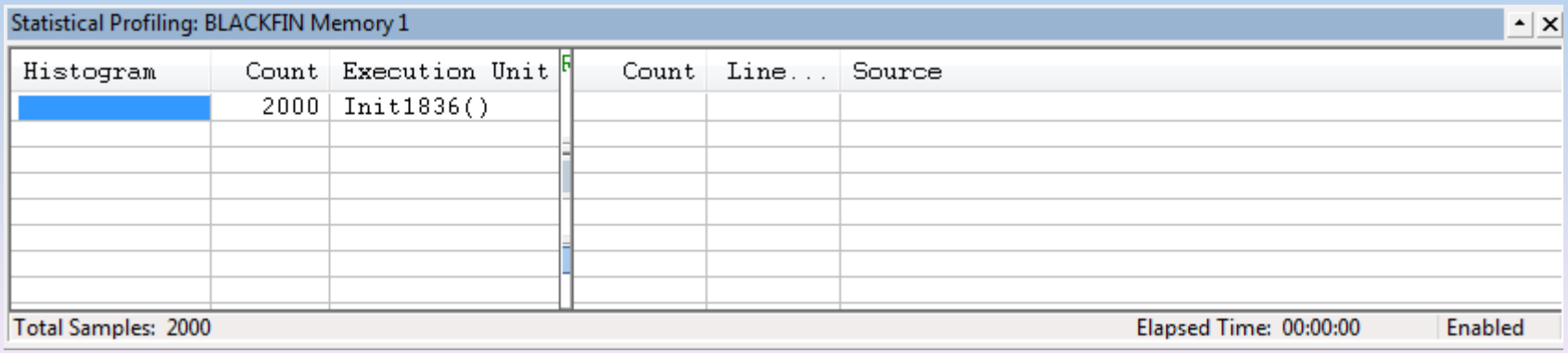
Question Set #4

- compare the values of **`iChannel0LeftOut`** and **`iChannel0LeftIn`**. Is this what you expect?
- do the same thing with **`iChannel0RightOut`** and **`iChannel0RightIn`**
- can you explain now why the signal got distorted?

Lenear Profile

- How to find instruction cycle count for each function ?





In conclusion:

- we have a complete setup for audio processing
- input/output samples from Channel0/1 are int type variables
- in order to implement filters, fractional representation will be needed.
- this will be the subject of the next labs.