BME-MIT

# DSP development technologies

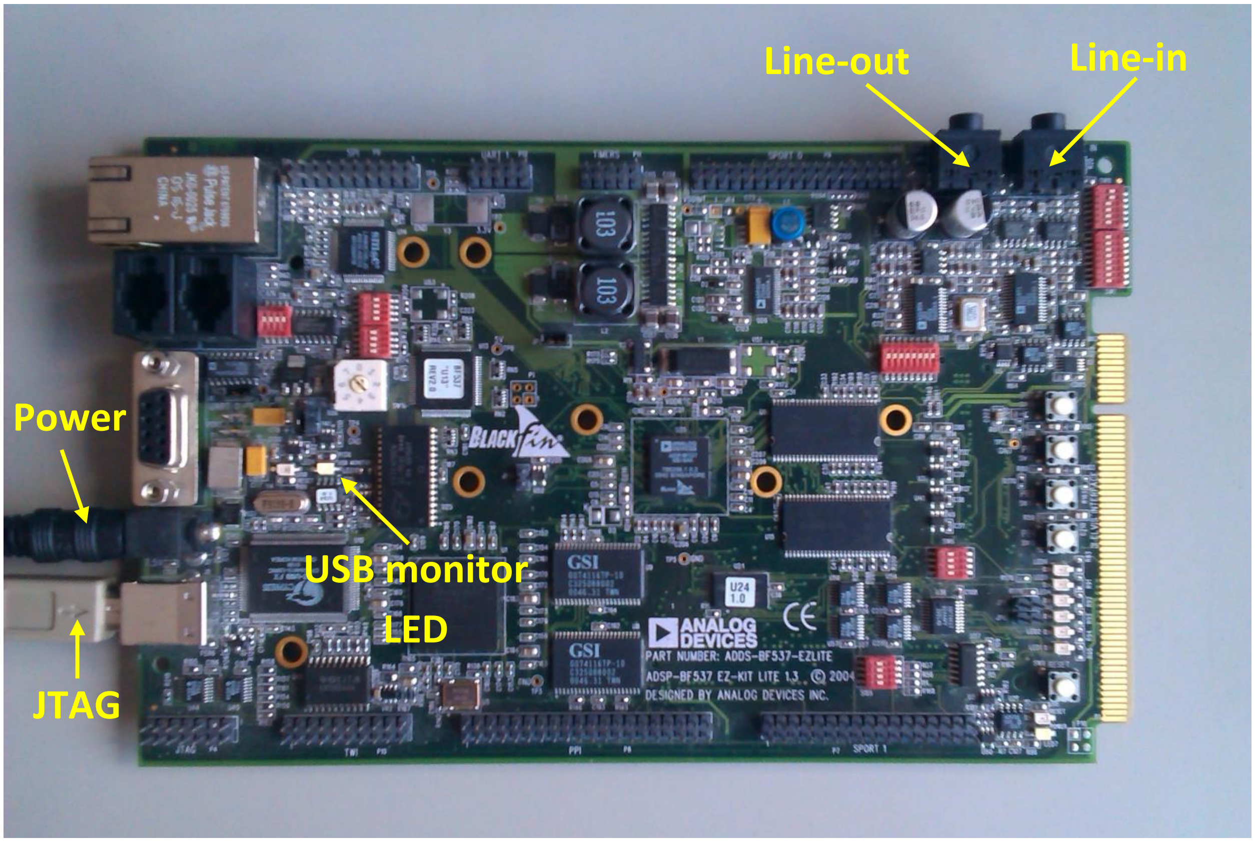
## Introduction to DSP development systems

**Bence Ország, György Orosz**

**2023**

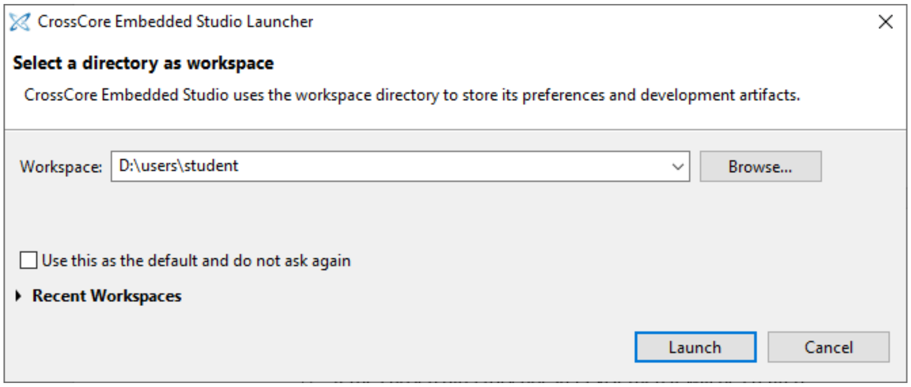
## Preparation of the DSP board and the development environment

- Connect the DSP card to the PC. **Attention!** Plug the power cord first, then the USB JTAG cable. Stick to this order if you need to reset the DSP card.
- Wait until the „USB monitor” LED lights up! It means that the card has successfully connected to the computer.



- Start the development environment on the PC: CrossCore Embedded Studio

- Choose a directory for your workspace where your projects will reside.
  - If the chosen directory doesn't exist, then it will be created.
  - All of your settings will be saved here, so feel free to change them.
  - Don't use special characters in the name and make sure your group is easily identifiable.



CrossCore Embedded Studio Launcher

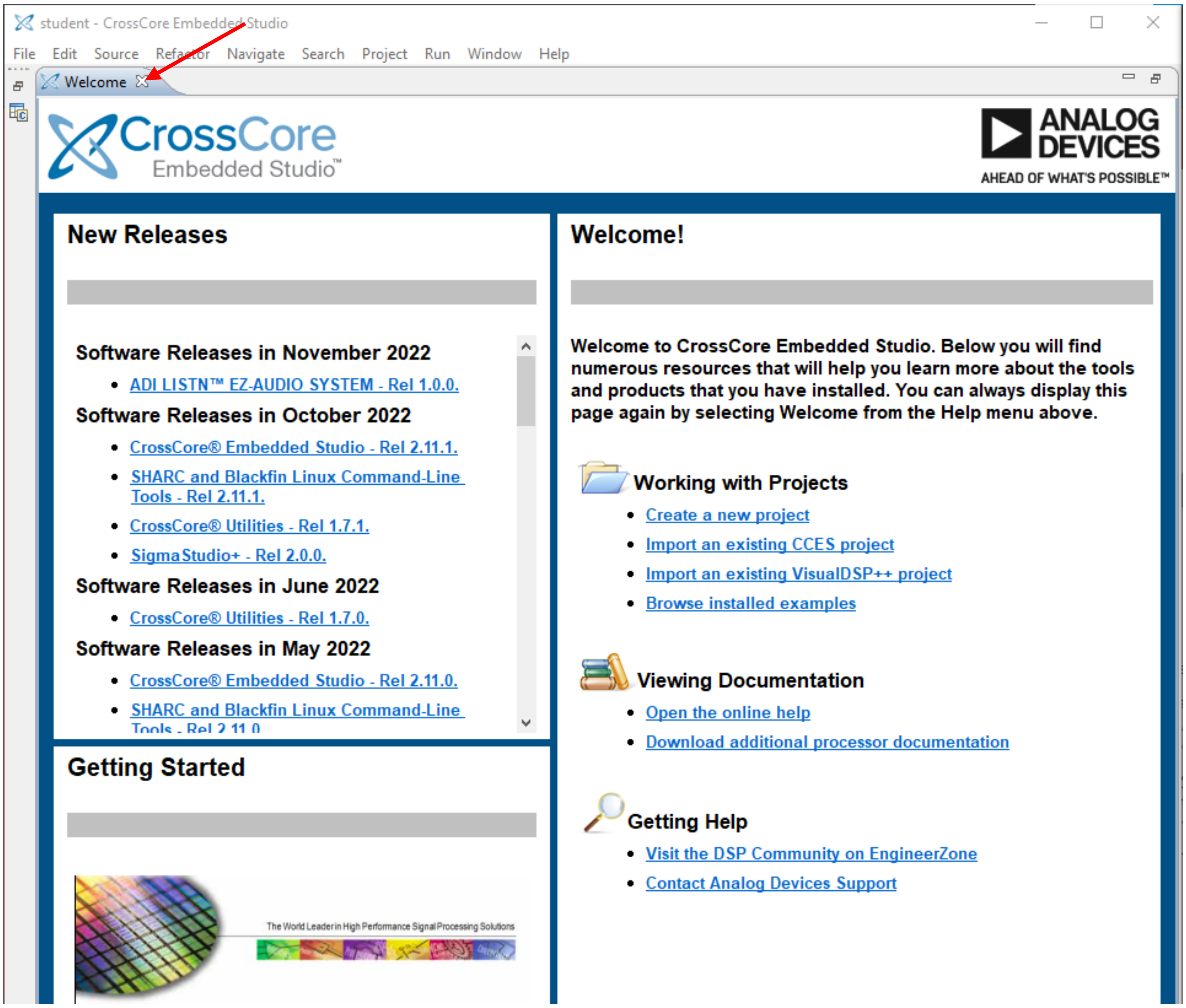
**Select a directory as workspace**

CrossCore Embedded Studio uses the workspace directory to store its preferences and development artifacts.

Workspace: D:\users\student

□ Use this as the default and do not ask again

- Close the welcome screen.



student - CrossCore Embedded Studio

**New Releases**

**Software Releases in November 2022**

- ADI LISTN™ EZ-AUDIO SYSTEM - Rel 1.0.0.

**Software Releases in October 2022**

- CrossCore® Embedded Studio - Rel 2.11.1.
- SHARC and Blackfin Linux Command-Line Tools - Rel 2.11.1.
- CrossCore® Utilities - Rel 1.7.1.
- SigmaStudio+ - Rel 2.0.0.

**Software Releases in June 2022**

- CrossCore® Utilities - Rel 1.7.0.

**Software Releases in May 2022**

- CrossCore® Embedded Studio - Rel 2.11.0.
- SHARC and Blackfin Linux Command-Line Tools - Rel 2.11.0

**Getting Started**

**Welcome!**

Welcome to CrossCore Embedded Studio. Below you will find numerous resources that will help you learn more about the tools and products that you have installed. You can always display this page again by selecting Welcome from the Help menu above.

**Working with Projects**

- Create a new project
- Import an existing CCES project
- Import an existing VisualDSP++ project
- Browse installed examples

**Viewing Documentation**

- Open the online help
- Download additional processor documentation

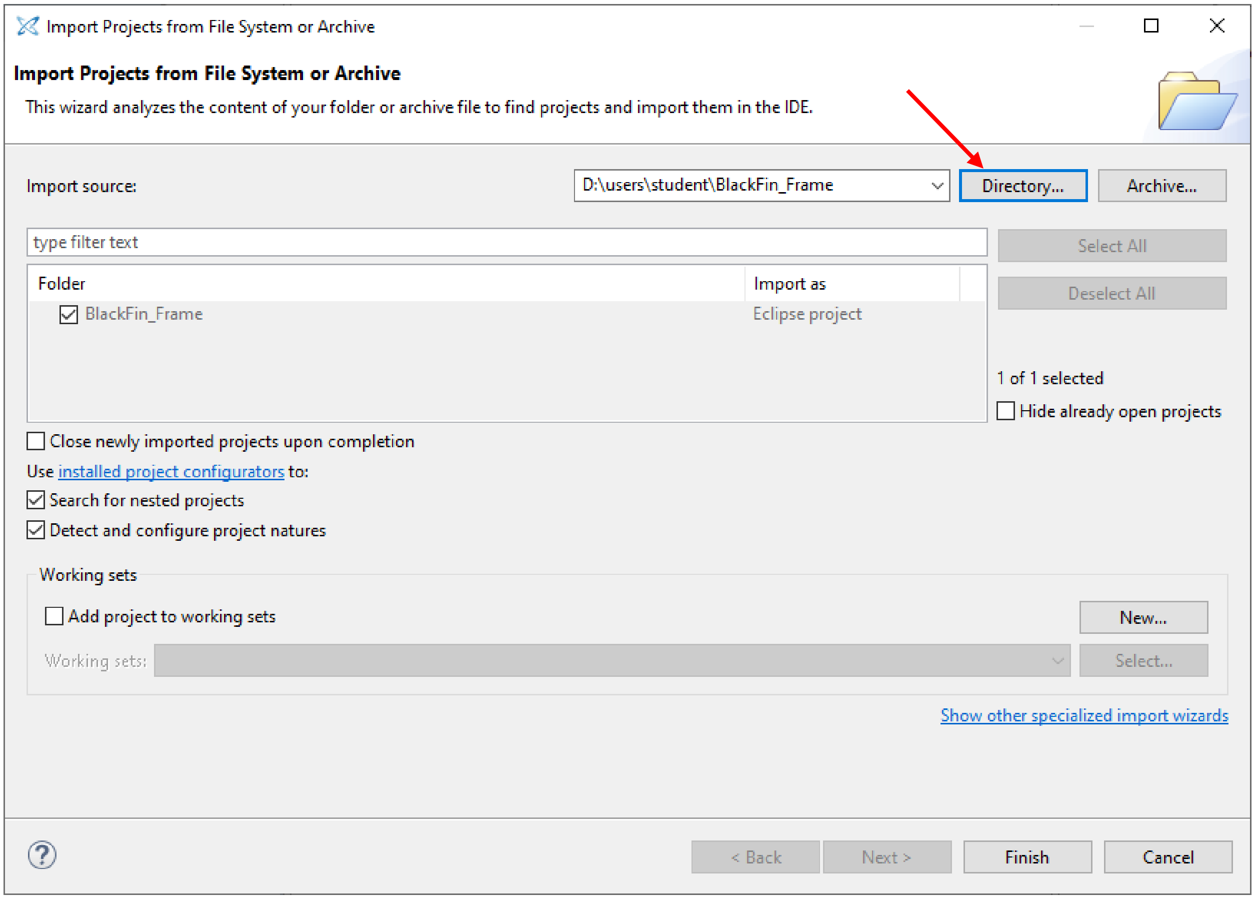
**Getting Help**

- Visit the DSP Community on EngineerZone
- Contact Analog Devices Support

- Download the sample project from the webpage of the laboratory course.
  Unzip the file into your workspace. Take care to unzip it into a subfolder with the name of the project! This is necessary because the IDE has a one project per folder policy.
- Connect the jack – BNC cables to the Line-in and Line-out connectors of the DSP card.
  - **Attention:**
    - Take care of the labels on the input and output connectors (Line-in/Line-out)! **Do not connect the Line out to the output of the signal generator!**
    - The maximum input on the line in channel is 3 $V_{pp}$. Make sure the output of the function generator is less so the input won't be overdriven! On Hameg generators the -20 dB button should be pressed! (Hameg function generator: max. 20 $V_{pp}$)
  - Connect the „Line out" output to the oscilloscope (both channels).
  - Connect the „Line in" input to the signal generator (the channel does not matter). Set a sine wave output signal on the generator.

## Opening a project

- Open the BlackFin_Frame project in the previously unzipped folder! To open a project: File → Open Projects from File System..., and here the project folder has to be selected!



Import Projects from File System or Archive

**Import Projects from File System or Archive**

This wizard analyzes the content of your folder or archive file to find projects and import them in the IDE.

Import source: D:\users\student\BlackFin_Frame

type filter text

| Folder | Import as |
|---|---|
| ☑ BlackFin_Frame | Eclipse project |

1 of 1 selected

□ Hide already open projects

□ Close newly imported projects upon completion

Use installed project configurators to:

☑ Search for nested projects

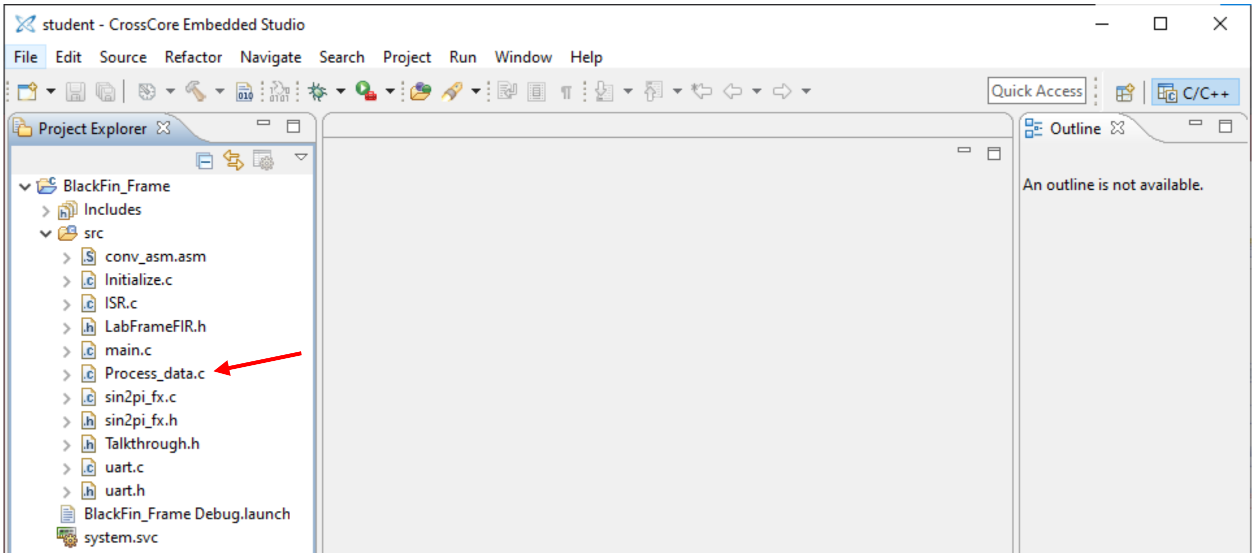
☑ Detect and configure project natures

Working sets

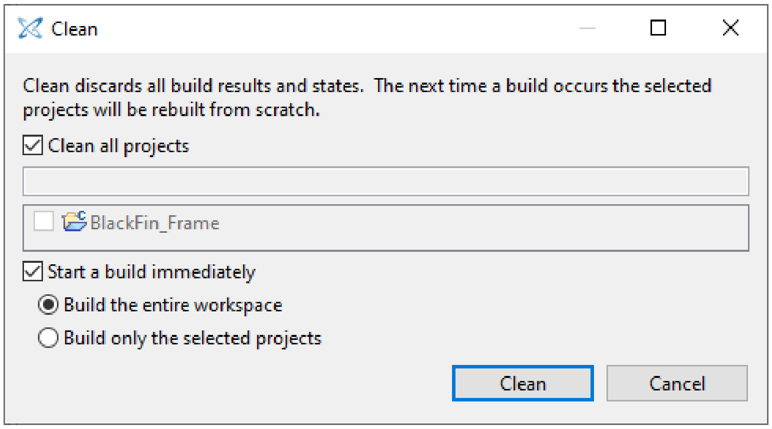
□ Add project to working sets

Working sets:

- Now you can see the Process_data.c file in the project window under the BlackFin_Frame project among other source files. The Process_data.c file contains the following signal processing method:
  - `void Process_Data(void)` runs if a new sample is available from the ADC.
  - The `iChannel0RightIn` and `iChannel0LeftIn` variables contain the sample of the left and right channels.
  - `iChannel0RightOut` and `iChannel0LeftOut` contain the data sent to the DAC.
  - The UART peripheral signals the availability of a new character through the `uart_rx_flag`. It must be cleared to receive the next one! Use `uart_getch()` to read it and `uart_putch()` or `uart_print()` to write to the serial port.



- Compile the project! Compiling options:
  - F7 key *or*
  - Project menu → Build project *or*
  - click on the 🔨 button.
- If you want to rebuild the project (and clean all previous files):
  - Project menu → Clean…



Clean discards all build results and states. The next time a build occurs the selected projects will be rebuilt from scratch.

☑ Clean all projects

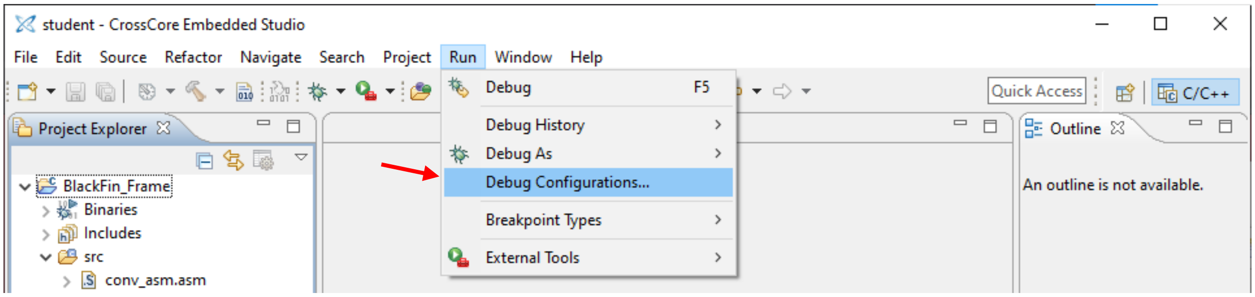
□ BlackFin_Frame

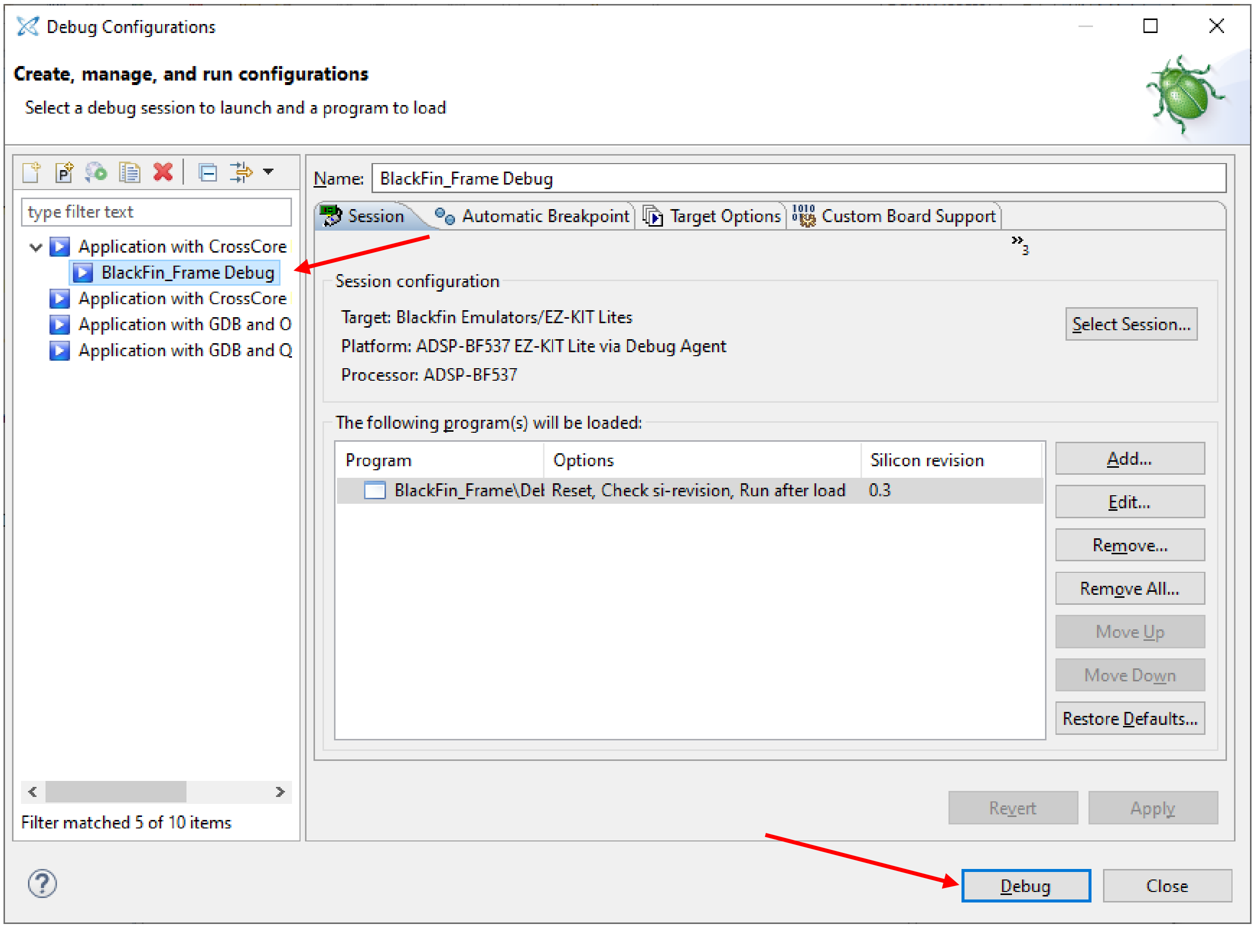
☑ Start a build immediately

● Build the entire workspace

○ Build only the selected projects

- Start a debug session and upload the project!
  - Check that the DSP board is indeed connected!
  - For the first time you need to select a debug configuration: Run → Debug Configurations… Then select Application with CrossCore Debugger → BlackFin_Frame Debug and click on Debug!
  - F5 *or*
  - Run → Debug *or*
  - click on the 🐞 button.





Name: BlackFin_Frame Debug

Session configuration

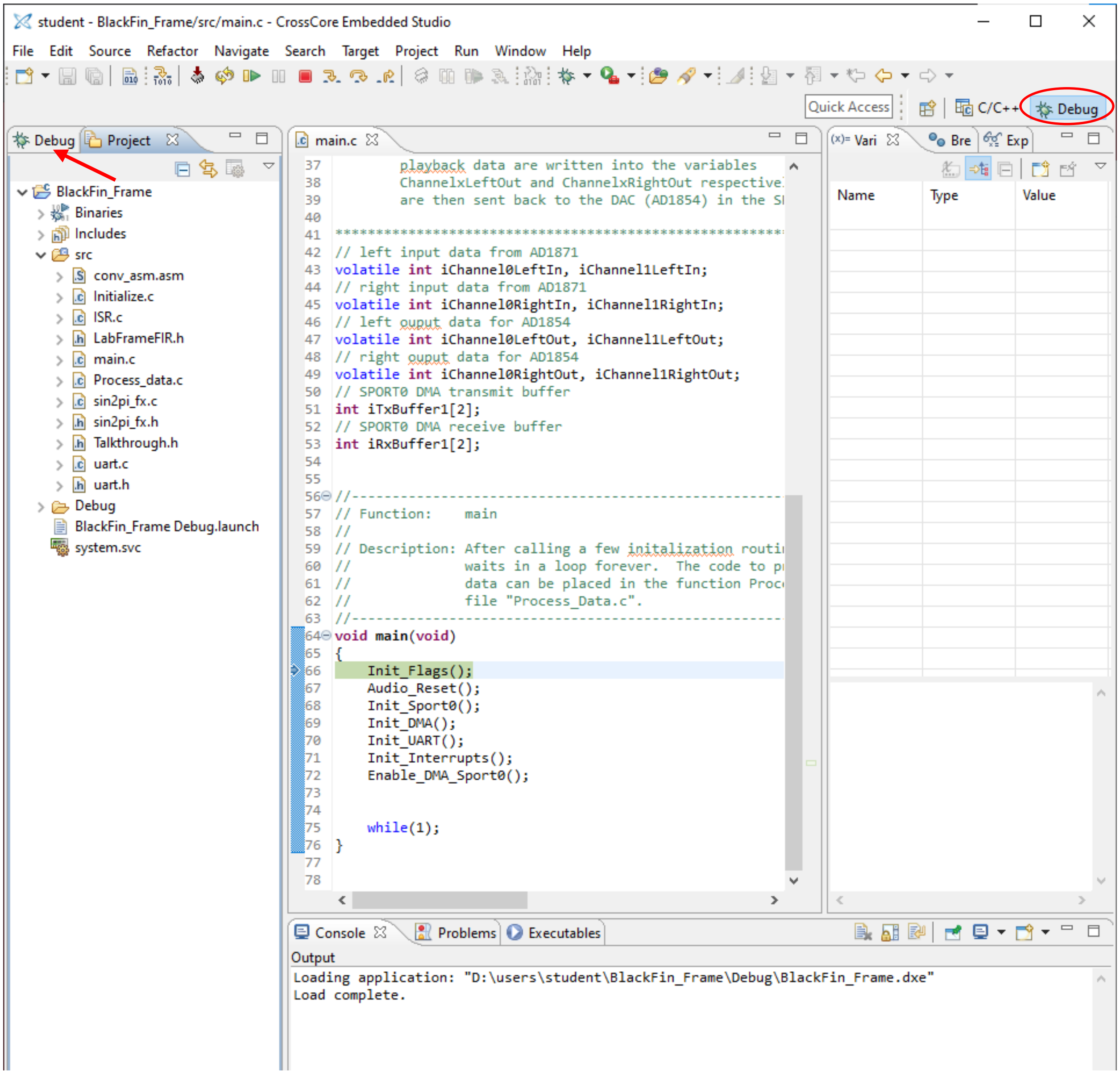
Target: Blackfin Emulators/EZ-KIT Lites
Platform: ADSP-BF537 EZ-KIT Lite via Debug Agent
Processor: ADSP-BF537

The following program(s) will be loaded:

| Program | Options | Silicon revision |
|---|---|---|
| BlackFin_Frame\Del | Reset, Check si-revision, Run after load | 0.3 |

- Run the project! (After uploading the program, it will be automatically halted at the first instruction). You should be in the debug perspective for this. To run the program:
  - F5 *or*
  - Run menu → Resume *or*
  - click on the ▶ button. (If this button is greyed out you need to click on the Debug window next to the Project explorer tab.)

  To stop the program:
  - Shift + F5 *or*
  - Run menu → Suspend *or*
  - click on the ⏸ button.
- The project simply forwards the signal of the right (left) input channel to the right (left) outputs.



```
        playback data are written into the variables
        ChannelxLeftOut and ChannelxRightOut respective
        are then sent back to the DAC (AD1854) in the SI

*******************************************************
// left input data from AD1871
volatile int iChannel0LeftIn, iChannel1LeftIn;
// right input data from AD1871
volatile int iChannel0RightIn, iChannel1RightIn;
// left ouput data for AD1854
volatile int iChannel0LeftOut, iChannel1LeftOut;
// right ouput data for AD1854
volatile int iChannel0RightOut, iChannel1RightOut;
// SPORT0 DMA transmit buffer
int iTxBuffer1[2];
// SPORT0 DMA receive buffer
int iRxBuffer1[2];


//-----------------------------------------------------
// Function:    main
//
// Description: After calling a few initalization routi
//              waits in a loop forever.  The code to pr
//              data can be placed in the function Proc
//              file "Process_Data.c".
//-----------------------------------------------------
void main(void)
{
    Init_Flags();
    Audio_Reset();
    Init_Sport0();
    Init_DMA();
    Init_UART();
    Init_Interrupts();
    Enable_DMA_Sport0();


    while(1);
}
```

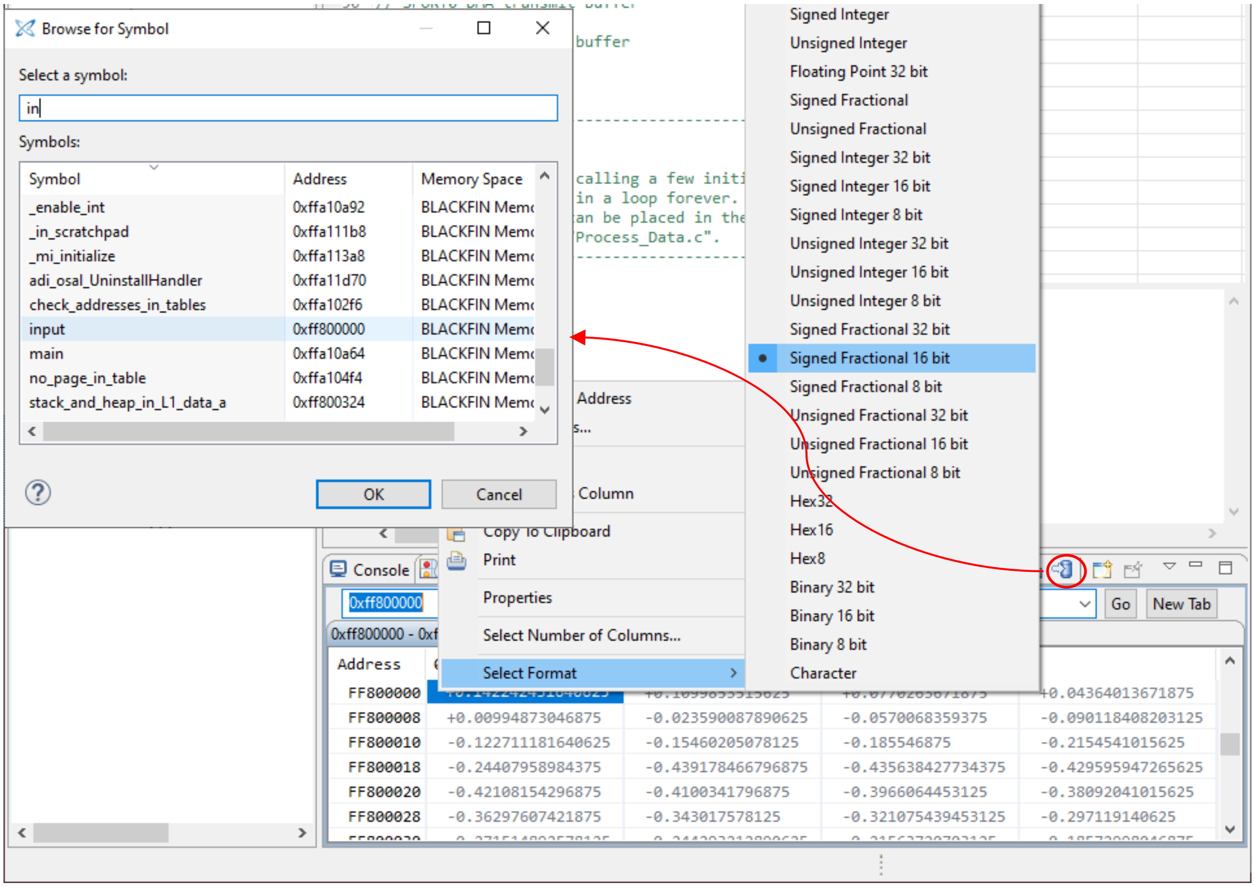
```
Loading application: "D:\users\student\BlackFin_Frame\Debug\BlackFin_Frame.dxe"
Load complete.
```

# Debug functions of the IDE

- For this part open the BlackFin_FIR_asm project. The program copies the right input to the left output channel while the right channel outputs the filtered right input.

## Reading and writing variables in the memory.

- In order to access the memory of the DSP you need to open the Memory Browser: Window menu → Show view → Memory Browser
- To select variables to monitor click on . On the newly opened window search for the variable to monitor.
- By right-clicking on a memory region and selecting „Select Format” you can choose the format of the data representation. For example, choose „Signed Fractional 16 bit” for fractional numbers.
- By double clicking on a variable its value can be modified. After entering the new value, confirm it by hitting the Enter key. Then it will be updated in the memory of the DSP.
- These features can be used only if the DSP is in „Halt” state, so the running program must be halted first.
- To study some variables in the program, insert Breakpoints by double clicking next to the selected line of the code on the grey bar on the left. When the program arrives to the selected line, its state will change automatically from running to halted.



Browse for Symbol

Select a symbol: in

Symbols:

| Symbol | Address | Memory Space |
|---|---|---|
| _enable_int | 0xffa10a92 | BLACKFIN Memo |
| _in_scratchpad | 0xffa111b8 | BLACKFIN Memo |
| _mi_initialize | 0xffa113a8 | BLACKFIN Memo |
| adi_osal_UninstallHandler | 0xffa11d70 | BLACKFIN Memo |
| check_addresses_in_tables | 0xffa102f6 | BLACKFIN Memo |
| input | 0xff800000 | BLACKFIN Memo |
| main | 0xffa10a64 | BLACKFIN Memo |
| no_page_in_table | 0xffa104f4 | BLACKFIN Memo |
| stack_and_heap_in_L1_data_a | 0xff800324 | BLACKFIN Memo |

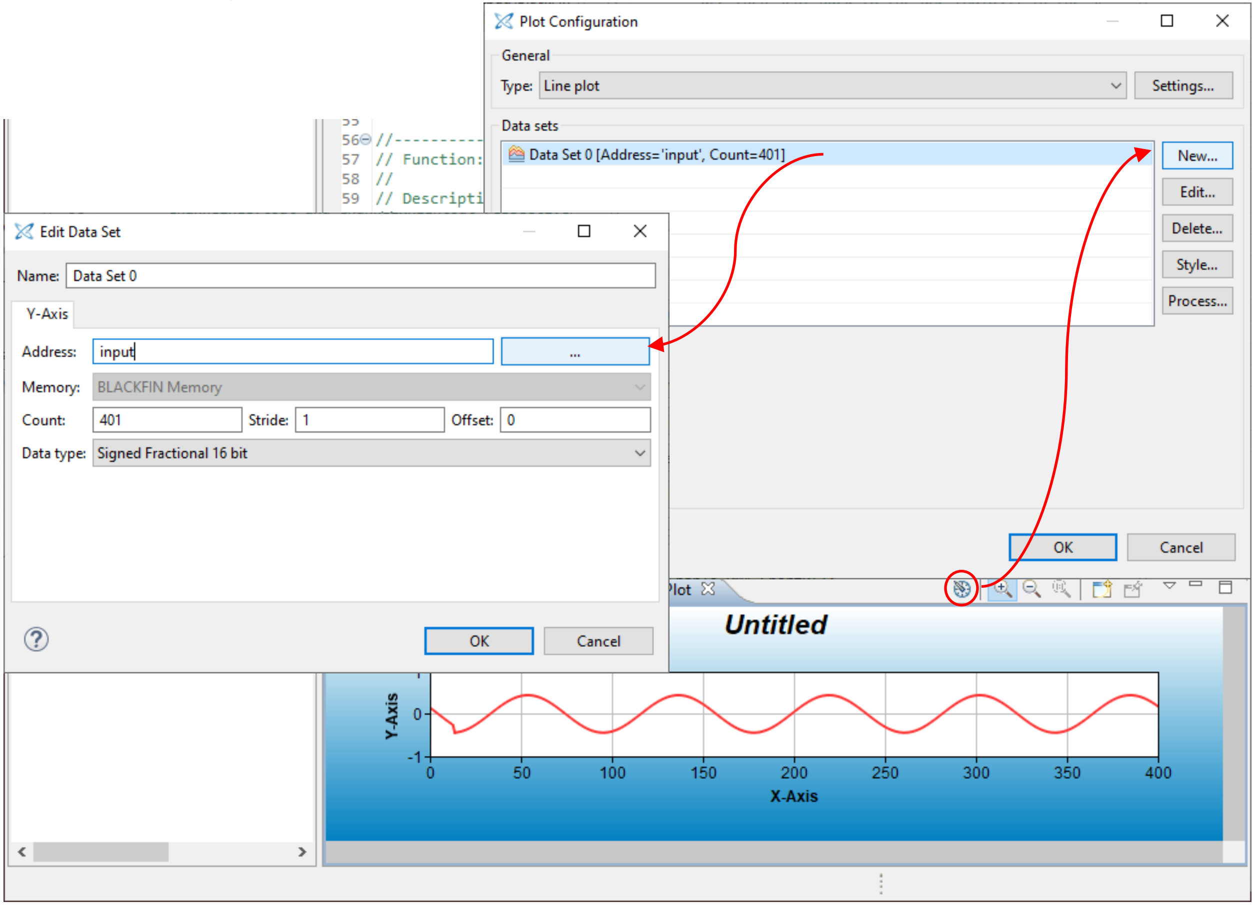
```
buffer

calling a few initi
in a loop forever.
an be placed in the
Process_Data.c".
```

Memory Browser: 0xff800000

```
Address
FF800000                                                                  +0.04364013671875
FF800008  +0.00994873046875   -0.023590087890625  -0.0570068359375    -0.090118408203125
FF800010  -0.122711181640625  -0.15460205078125   -0.185546875        -0.2154541015625
FF800018  -0.24407958984375   -0.439178466796875  -0.435638427734375  -0.429595947265625
FF800020  -0.42108154296875   -0.4100341796875    -0.3966064453125    -0.38092041015625
FF800028  -0.36297607421875   -0.343017578125     -0.321075439453125  -0.297119140625
```

## Plotting data arrays.

- Select Window menu → Show view → Plot. This opens a new window.
- Click on ⚙ to open the plot configuration. Here you can add new datasets to the plot.
- After clicking on the button New the Edit Data Set window opens. Here you can specify the name, length and data type of the array you want to plot. Click the button Add after specifying these parameters. The array can be selected with the Browse button. First select the input variable.
- Enter the length of the array to the Count field. For the input array this is 401.
- Stride defines the step size. Its value should be 1.
- Select the number format in the Data list. Choose short.
- The DSP must be in Halt state to use these features otherwise it will use the values saved during the previous halt. Use breakpoint(s) or the Pause debug button to halt the processor.
- The sampling frequency of the ADC and DAC is 48 kHz. Adjust the frequency of the input signal until the sampling is coherent (an integer number of periods is measured of the input sine) and plot the result (e.g. measure 7 periods of the signal). Next measure a sine with noncoherent sampling by changing the signal frequency. Plot the result and explain it (circularity of the buffer).
- Modify some elements of the input array and plot it again.
- Plot and study the filter coefficients (coefs variable), explain the filter characteristics (bandpass filter = lowpass filter modulated by a sine wave).



Plot Configuration

General

Type: Line plot

Data sets

Data Set 0 [Address='input', Count=401]

Edit Data Set

Name: Data Set 0

Address: input

Memory: BLACKFIN Memory

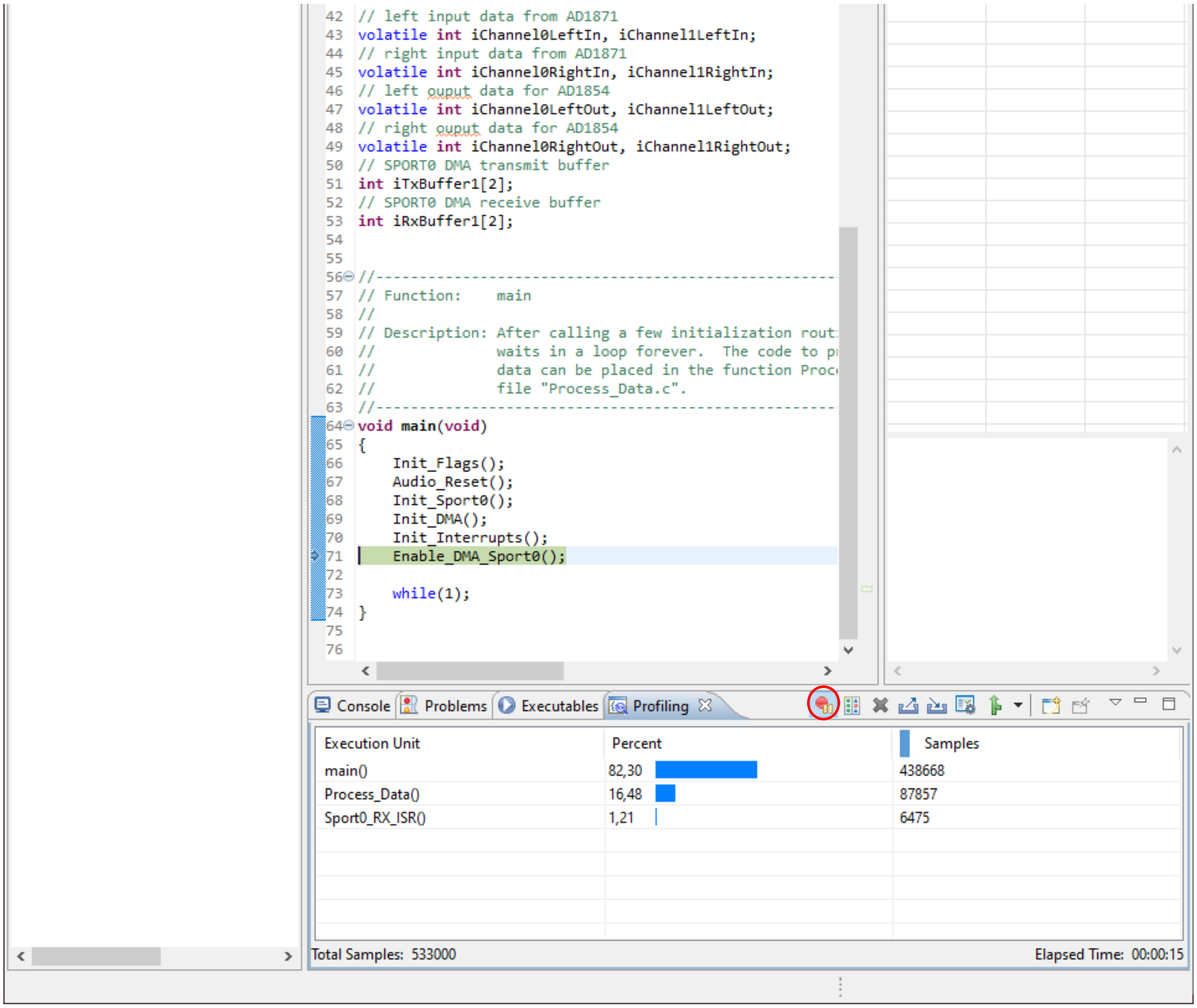
Count: 401 Stride: 1 Offset: 0

Data type: Signed Fractional 16 bit

```
56⊖ //---------
57  // Function:
58  //
59  // Descripti
```

## Runtime-analysis

- Select Window menu → Show view → Profiling and open a new runtime statistics.
- Launch the program and collect data by clicking on . Analyze the the run-time of different functions. Measure the run-time percentage of the `conv_asm` function. (Optionally visualize it with a pie chart by clicking on one of the buttons.)
- Modify the memory section of the input or the coefs variables (from `section("L1_data_a")` to `section("L1_data_b"),` or vice versa). The variables should be in the same memory section.
- Measure the run-time percentages again. What is the difference? (To delete the previously measured statistics manually click on  while the processor is halted.)



```
// left input data from AD1871
volatile int iChannel0LeftIn, iChannel1LeftIn;
// right input data from AD1871
volatile int iChannel0RightIn, iChannel1RightIn;
// left ouput data for AD1854
volatile int iChannel0LeftOut, iChannel1LeftOut;
// right ouput data for AD1854
volatile int iChannel0RightOut, iChannel1RightOut;
// SPORT0 DMA transmit buffer
int iTxBuffer1[2];
// SPORT0 DMA receive buffer
int iRxBuffer1[2];


//-----------------------------------------------------
// Function:    main
//
// Description: After calling a few initialization rout
//              waits in a loop forever.  The code to p
//              data can be placed in the function Proc
//              file "Process_Data.c".
//-----------------------------------------------------
void main(void)
{
    Init_Flags();
    Audio_Reset();
    Init_Sport0();
    Init_DMA();
    Init_Interrupts();
    Enable_DMA_Sport0();

    while(1);
}


```

| Execution Unit | Percent | Samples |
|---|---|---|
| main() | 82,30 | 438668 |
| Process_Data() | 16,48 | 87857 |
| Sport0_RX_ISR() | 1,21 | 6475 |

Total Samples: 533000

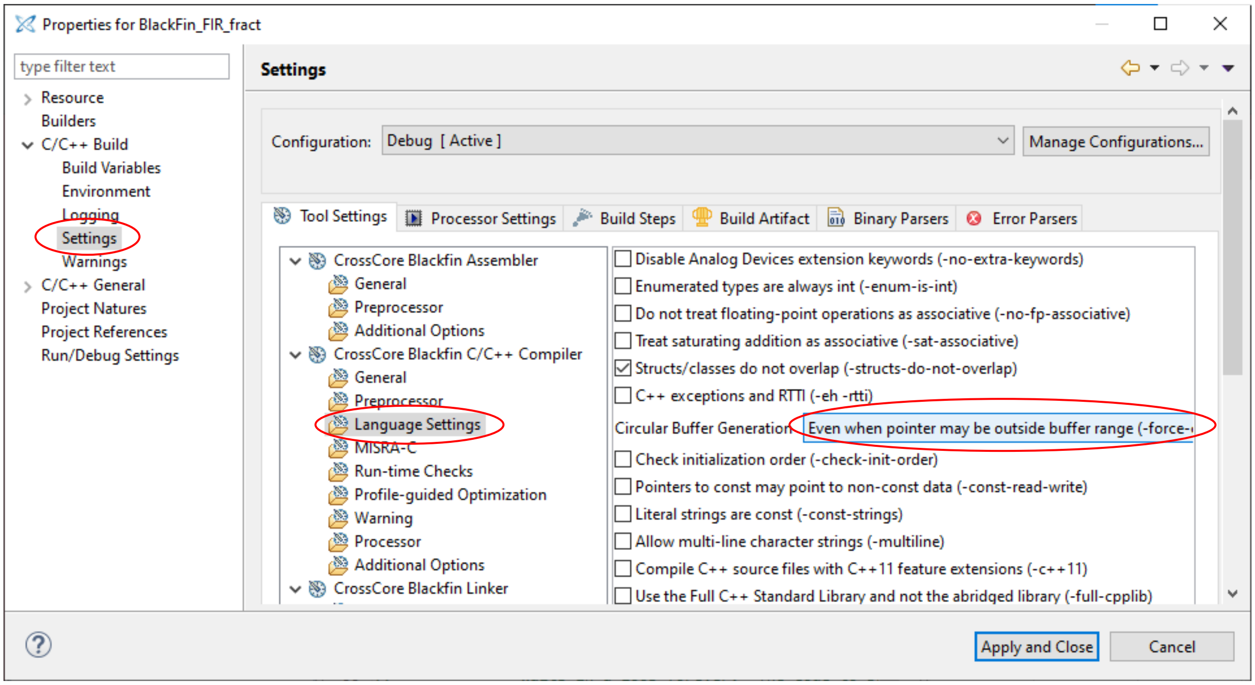
Elapsed Time: 00:00:15

## FIR filter with assembly routine

- Open the BlackFin_FIR_asm project! This is a FIR filter application: it filters the signal on the right channel of the input. The filtered signal is forwarded to the left channel of the output. On the right channel the original signal appears without being filtered.
- Study and understand the code. Notes:
    - The `iChannel0RightIn >> 8` and `out <<8` commands are shifting the data because the ADC and DAC are of 24 bits, but the DSP works with 16-bit data.
    - The filtering is done in the conv_asm function.
- Compile and run the project.
- Measure the cutoff frequencies of the filter using the signal generator and the oscilloscope. What type of filter is implemented? Compare the filter with the original one designed in Matlab: load the coefficients with the `h = load('bp_fract.dat');` command then create a figure of the transfer function with `freqz(h,1,2^16,48000);`.

## Implementing FIR filter in C

- Open the BlackFin_FIR_fract project.
- Study the implementation of the circular buffer.
- Measure the run-time percentage of the signal processing procedure.
- Select Project menu → Properties. In the new window select C/C++ Build → Settings and then on the Tool Settings tab select Crosscore Blackfin C/C++ Compiler → Language Settings. From the list select Never (-no-circbuf).
- Compile the project again and launch run-time statistics. Study the results.
- Enable circular buffer generation option again in the options.



Configuration: Debug [ Active ]

☐ Disable Analog Devices extension keywords (-no-extra-keywords)

☐ Enumerated types are always int (-enum-is-int)

☐ Do not treat floating-point operations as associative (-no-fp-associative)

☐ Treat saturating addition as associative (-sat-associative)

☑ Structs/classes do not overlap (-structs-do-not-overlap)

☐ C++ exceptions and RTTI (-eh -rtti)

Circular Buffer Generation: Even when pointer may be outside buffer range (-force-

☐ Check initialization order (-check-init-order)

☐ Pointers to const may point to non-const data (-const-read-write)

☐ Literal strings are const (-const-strings)

☐ Allow multi-line character strings (-multiline)

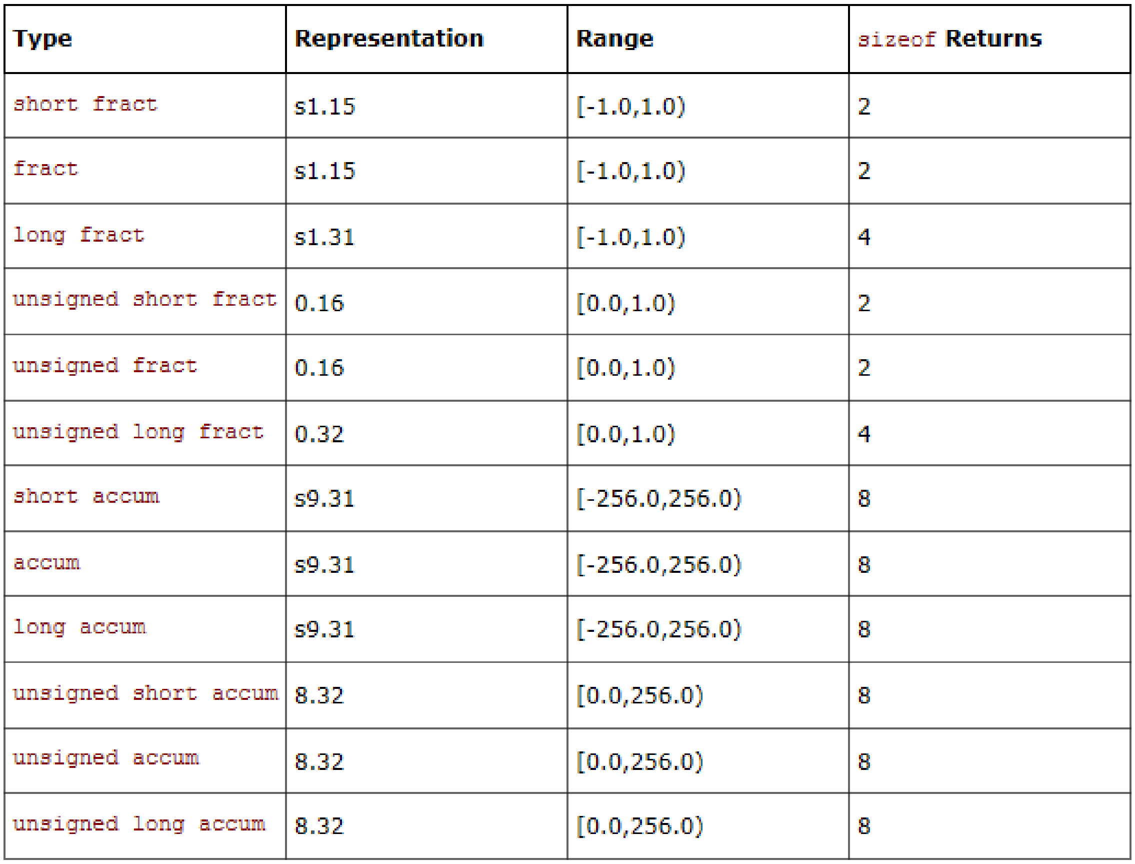
☐ Compile C++ source files with C++11 feature extensions (-c++11)

☐ Use the Full C++ Standard Library and not the abridged library (-full-cpplib)

# Appendix

## Using the built in fractional type

- The compiler has a built in fract type, which is a native type like int or char. It is provided by the compiler (as per *Extensions to support embedded processors* ISO/IEC Technical Report 18037). It has variants like int, so the following are valid examples: short fract, unsigned fract, etc. Signed types are in the [-1,1) range while unsigned types are in the [0, 1) range. The table below summarizes the fixpoint representations.
- To access the fract type one needs to include stdfix.h!
- Be aware that fractional calculations always saturate! This might cause problems in certain cases so the accum type can be used to avoid those. It uses more integer bits, the representable numbers are in the [-256, 256) range.
- For fract constants use the r suffix (i.e. `0.75r`), for accum constant use k (i.e. `3.25k`).
- There are functions to convert between 16 bit integers and fractional numbers. They are listed in stdfix.h and in the documentation of the compiler (CrossCore Embedded Studio → Blackfin Development Tools Documentation → C/C++ Compiler and Library Manual → Compiler → Using Native Fixed Point Types → Bit-Pattern Conversion Functions):
    - `bitsr(f)` : fract → short
    - `rbits(s)` : short → fract
- Type casting is the recommended method of conversion between fract and float values.



| **Type** | **Representation** | **Range** | `sizeof` **Returns** |
|---|---|---|---|
| `short fract` | s1.15 | [-1.0,1.0) | 2 |
| `fract` | s1.15 | [-1.0,1.0) | 2 |
| `long fract` | s1.31 | [-1.0,1.0) | 4 |
| `unsigned short fract` | 0.16 | [0.0,1.0) | 2 |
| `unsigned fract` | 0.16 | [0.0,1.0) | 2 |
| `unsigned long fract` | 0.32 | [0.0,1.0) | 4 |
| `short accum` | s9.31 | [-256.0,256.0) | 8 |
| `accum` | s9.31 | [-256.0,256.0) | 8 |
| `long accum` | s9.31 | [-256.0,256.0) | 8 |
| `unsigned short accum` | 8.32 | [0.0,256.0) | 8 |
| `unsigned accum` | 8.32 | [0.0,256.0) | 8 |
| `unsigned long accum` | 8.32 | [0.0,256.0) | 8 |

## Common bugs of the IDE

- After loading the program to the development board the debug toolbars' icons are greyed out so the program cannot be started.
  - Switch to Debug perspective.
  - On the left bring the Debug tab into foreground (instead of the Project Explorer).
- Failed to resolve « MACRO » error. The project can be compiled but the problem tab shows an error.
  - Rebuild the index by clicking on Project menu → C/C++ Index → Rebuild.
- When launching a new debug session there is an error with the text "Error in launch sequence. Failed to connect to processor. Failed passing SOM test." This is caused by a previously failed launch attempt where the development board wasn't connected or powered on.
  - Restart the IDE.