FTAB .NET SDK User Manual

Project	FTAB
Title	FTAB .NET SDK User Manual
Reference	
Client Reference	
Author(s)	Nicholas Kopp
Date	March 15, 2012

REVISION HISTORY

Date	Changes Made	Issue	Initials
December 22, 2010	Initial version	2.0	Nko
May 6, 2011	Added FTAB SDE	2.1	Nko
March 15, 2012	Update contact info, minor changes	2.3	Nko

Hybrid DSP Systems
Koningin Emmalaan 3
2635HH Den Hoorn ZH
Netherlands

Tel: +31 (0) 15 8700817

Email: info@hybriddsp.nl
Web: www.hybriddsp.nl
Web: http://www.hybriddsp.com/Products/ESAFFTCSDK.aspx

References

[REF 1] ESA_1210_User_Manual_2_3_FFTC_SDK

1 Background

This document details the use of a set of .NET libraries for programming and simulating the Astrium FTAB development board. These libraries are built upon the ESA FFTC 2.0 libraries. It is highly recommended that users first acquaint themselves with the FFTC libraries before beginning to explore the FTAB specific libraries. The FFTC libraries are discussed in detail in *ESA_1210_User_Manual_2_3_FFTC_SDK*.

Resource	Description	Location
FFTC / FTAB SDK API Documentation (also included in installer)	Detailed API reference.	*http://www.hybriddsp.nl/esa/doc150312/ftabsde.chm*
Email address for questions / support		support@hybriddsp.nl

2 Introduction

2.1 Overview

The FTAB .NET SDK is a set of programmers' libraries and applications for assisting in the development of algorithms for the Astrium FTAB board which is based on the ESA FFTC space qualified processor.

The libraries make algorithm development and testing faster and easier. The learning curve for new users is reduced.

2.2 Benefits

Some of the benefits of the SDK are described below:

- **Easy to use** – The libraries are written for the Microsoft .NET framework. This is an easy to use run-time that is standard with Windows Vista and 7 and a free download for Windows XP. .NET allows developers to use a wide range of languages including C++, Visual Basic and C#.
- **Flexible** – By use of Mono the same binaries can be used also under Linux.
- **Powerful** – By using the libraries from an established framework and language, highly complex applications can be developed that integrate compile and run-time sections into one application.
- **Low cost** – Compilers and software development environments are available free of charge (e.g. Visual Studio Express, Eclipse and Sharp Develop).
- **Encourages development –** The SDK includes a simulator and performance profiler that allows algorithms to be run and benchmarked without the OPDP FFTC hardware.

2.3 Features

The SDK makes use of the ESA FFTC SDK and further adds FTAB specific functions.

2.4 Installing the SDK

The SDK comes with a Microsoft installer. Double click the installer and follow the steps. The software is installed under Program Files/ESA/FFTC.

2.4.1 Third Party Libraries

FFTW library (libfftw3f-3.dll) must be either in the executing directory, in the System32 directory or on search path. It is included by default with the installer.

If working with an actual board then the appropriate drivers, software and hardware **must** be installed on the host computer.

If you wish to compile the unit test projects then NUnit must be installed. This is a free unit testing framework. Ensure that these projects reference nunit.framework.dll.

2.4.2 Native Libraries

It is necessary to copy the native library esafftcsimwin32.dll to the directory you are running the executable from, the Windows System32 directory or somewhere else on the search path. The installer puts these files into the bin directory of the installation.

2.4.3 32-bit and 64-bit

By default the target is 32-bit since this is compatible with both 32-bit and 64-bit OS. However if you wish to target 64-bit because your application requires this then remember you will need 64-bit versions of all libraries including esafftcsimwin32.dll and libfftw3f-3.dll. Failure to use these will result in a bad image format exception. Another advantage with 64-bit is the performance of the simulator which can improve by up to 20%.

Please contact Hybrid DSP if you wish to use 64-bit.

3 Algorithm Library

The FTAB board directly uses FFTC control vectors. No further modification or encapsulation is required.

4 Device Interface

From .NET it is possible to interface with both an actual FTAB board and an FTAB simulator. The FTAB simulator encapsulates the FFTC simulator. Both provide the following interfaces:

- Byte order
- Device number
- Enable data generator (LFSR)
- Disable data generator
- Enable links
- Force 8-bit enabled
- Read data
- Write data
- Read register
- Reset
- Send control vector
- Write
- Flush
- Set output sample size
- Set up port connection
- Device settings
    - Force 8 bit
    - Byte order
    - LFSR initial value
    - LFSR XOR
    - LFSR sequence length
    - LFSR enabled

4.1 FTAB Device Interface

The relevant SpaceWire drivers and dll must be installed on target machine. These are available from STAR-Dundee.

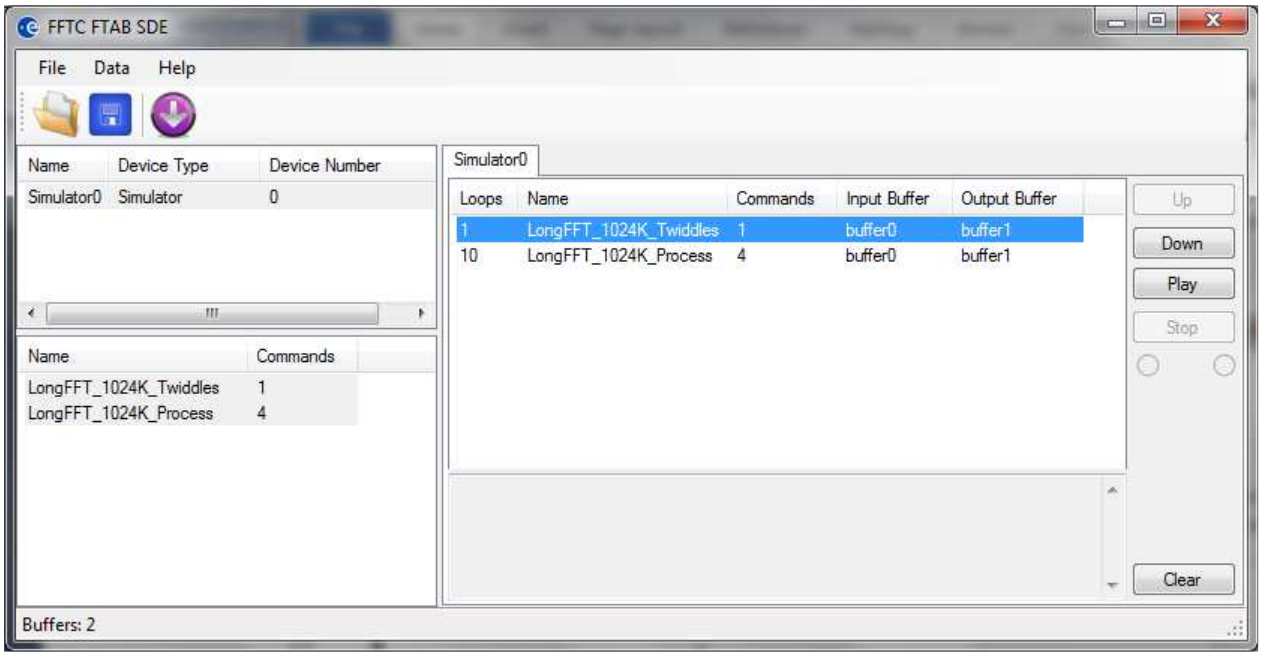
4.2 Example

```
public void TestLFSR()
{
    FFTCModule fftc = new FFTCModule();
    int w = 1024;
```

```
            int h = 1024;
            int bytes = w * h * 8;
            int loops = Benchmark ? 50 : 10;

            LFSR64 controlLFSR = new LFSR64(16, 15, (uint)(w * h));

            ControlVector cv = FFTCControlVectorWizard.Echo(w * h, FFTCModule.dIEEE_PAR,
FFTCModule.dIEEE_PAR, 1)[0];

            TimeSpan ts;
            Stopwatch sw = new Stopwatch();
            using (FTABDeviceSimulator fftcsim = new FTABDeviceSimulator())
            {
                byte[] outputBuffer = new byte[w * h * 8];
                fftcsim.InternalExceptionThrown += new
InternalDataflowSimulatorExceptionThrownDelegate(fftcsim_InternalExceptionThrown);
                fftcsim.EnableDataGenerator(16, 15, (uint)(w * h));

                if (Benchmark)
                {
                    sw.Start();
                    for (int x = 0; x < loops; x++)
                    {
                        fftcsim.Set(cv);
                        fftcsim.Read(outputBuffer, 0, outputBuffer.Length);
                    }
                    sw.Stop();
                }
                else
                {
                    for (int x = 0; x < loops; x++)
                    {
                        sw.Start();
                        fftcsim.Set(cv);
                        fftcsim.Read(outputBuffer, 0, outputBuffer.Length);
                        sw.Stop();
                        controlLFSR.Reset();
                        MemoryStream ms = new MemoryStream(outputBuffer);
                        BinaryReader br = new BinaryReader(ms);
                        for (int i = 0; i < w * h; i++)
                        {
                            ulong expected = controlLFSR.GetNextValue();
                            ulong actual = br.ReadUInt64();
                            Assert.AreEqual(expected, actual);
                        }
                    }
                }

                ts = sw.Elapsed;
            }
            Console.WriteLine(string.Format("Done in {0}s", ts.TotalSeconds));
            Console.WriteLine(string.Format("Samples/sec {0}", (long)(w * h) * loops /
ts.TotalSeconds));
        }

        void fftcsim_InternalExceptionThrown(object sender,
InternalDataflowSimulatorExceptionEventArgs args)
        {
            Console.WriteLine("fftcsim_InternalExceptionThrown");
            Console.WriteLine(args.Exception.ToString());
        }
```

5 FTAB SDE

The FTAB SDE is a graphical tool that permits an easy to use interface to the FTAB hardware or simulator. Control vectors can be generated, loaded and saved and can be executed on the hardware or simulator. Test data can be generated, loaded, saved and viewed and associated with control vector that define inputs or outputs.

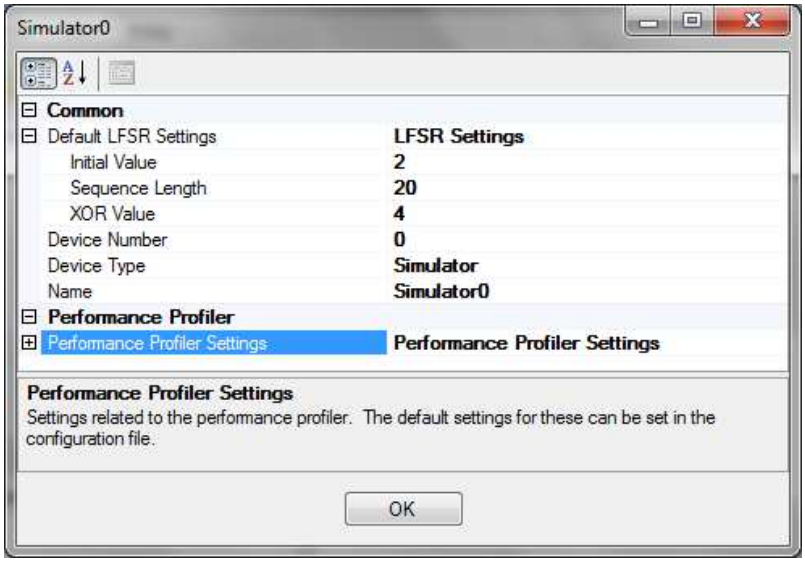


The user interface comprises three list views. These are:

- Devices
- Control Vectors
- Device Function Player Tabs
  - Functions

5.1 Devices

To add a device right click the top left list view and select **Add :: *Device Type***. You can edit a device by selecting and right clicking **Edit…** a device in the list.

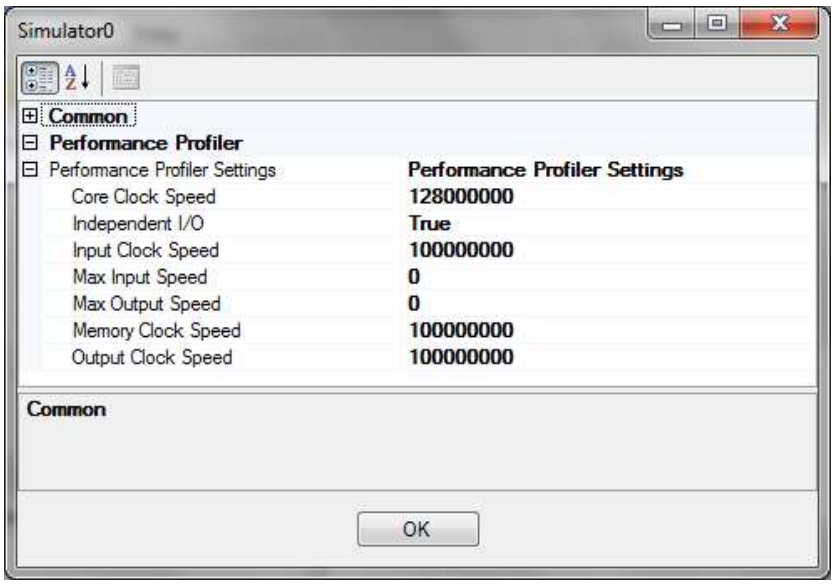


Settings are split into two categories: **Common** and **Performance Profiler**.

5.1.1 Common

The Common settings category contains parameters for LFSR data generator, device number, device type and name. When working with real devices as opposed to a simulator the device number refers to the device id enumeration as seen by the host operating system.

5.1.2 Performance Profiler



The subsequent category contains settings for the performance profiler. The values set here are used when calculating the timings. They are not reflecting what is

happening on actual hardware and are estimates only. Core Clock Speed, Input Clock Speed, Memory Clock Speed and Output Clock Speed are in Hz. Max Input Speed and Max Output Speed are in Bytes Per Second. Independent I/O is a Boolean setting that states whether P0 and P5 (the input and output of the FFTC) can run concurrently or not.

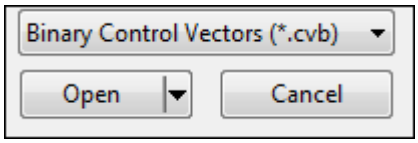
5.2 Control Vectors

At the bottom left of the FTAB SDE is a list view containing control vectors. Control vectors can be added by loading from file (xml or binary) or using the Control Vector Wizard. Control vectors can be saved to disk as xml or binary.

5.2.1 Loading and Saving

To load a control vector from disk use one of the following methods:

- Click **File :: Open Control Vector…**
- Click the Open Folder button on the tool bar
- Right click **Load…** in the Control Vector list view



Use the drop down menu in the open file dialog to select **xml** or **cvb**. To save a control vector select it in the list and then either:

- Click **File :: Save Control Vectors…**
- Click the Save disk button on the tool bar
- Right click **Save…** in the Control Vector list view

If multiple control vectors are selected then it is only possible to save as a control vector list (xml).

5.2.2 Control Vector Wizard

The Control Vector Wizard can create simple, non-optimized vectors to perform the following:

- Short 1D FFT (16 to 1024 point, Power of 2)
- Long 1D FFT (2048 to 1M point, Power of 2)
- 2D FFT (16x16 to 1024x1024 point, Power of 2)
- Echo (Transfers between 1K and 1M samples to and from the device)
- Write (Transfers between 1K and 1M samples to the device)
- Read (Transfers between 1K and 1M samples from the device)

Sizes, input and output data formats and where applicable FFT direction can be set. The Echo, Write and Read functions permit the specification of memory bank. Right click the control vector window and select Wizard followed by algorithm type.

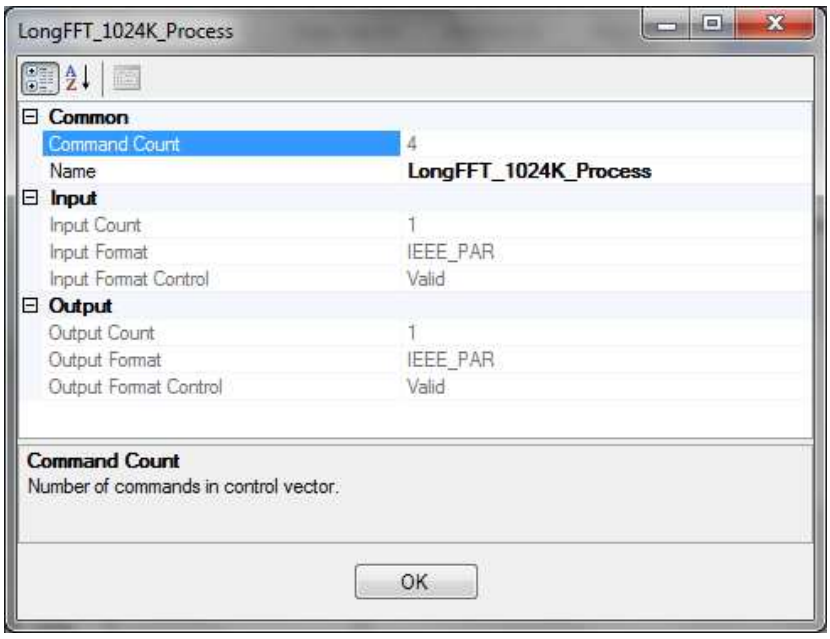
Note: When working with long FFTs two control vectors are created. The first is for loading the necessary twiddle vectors. These are of the same length as the FFT and arranged as near to a square as possible. Therefore a 1M-point FFT is arranged as 1K x 1K point. A 512K-point FFT is done as 1K x 512-point. You must generate twiddles to be loaded by this first control vector matching this shape.

5.2.3 Edit

Click **Edit…** to view summary details of the control vector and to alter the name. Note that the name can also be changed by clicking directly on the entry in the control vector window.

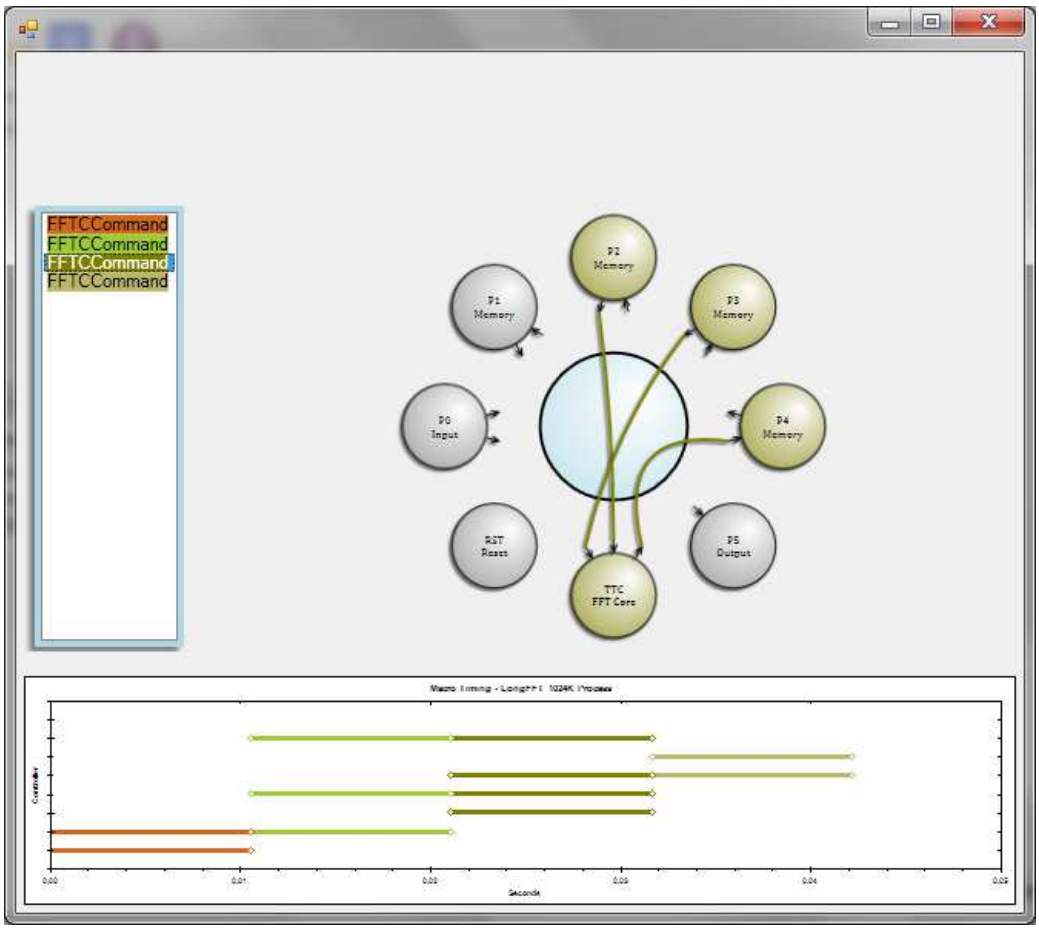
The other fields are read only. These include:

- Command Count – number of commands in control vector,
- Input / Output Count – the number of commands with inputs / outputs in control vector,
- Input / Output Format – the data format of the control vector (or Multiple in event that different data formats are present per command),
- Input / Output Format Control – If more than one format exists (Multiple) and the formats are of different sample sizes then the control vector cannot run from within the SDE. If formats are of same sample sizes then the control vector can run but the user must ensure that the data source is suitable.



5.2.4 View Dataflow

Right clicking and selecting **View Dataflow…** brings up a window showing the dataflow per command of the selected control vector plus its performance profile. On the left hand side is a list of commands found in the control vector. Multiple commands can be selected by use of the Ctrl key while clicking on the commands. Placing the cursor over an active controller (circles) shows its low level settings. At the bottom of the screen is the performance profile based on default settings. It is recommend to use the profiler from the player (tab) control.

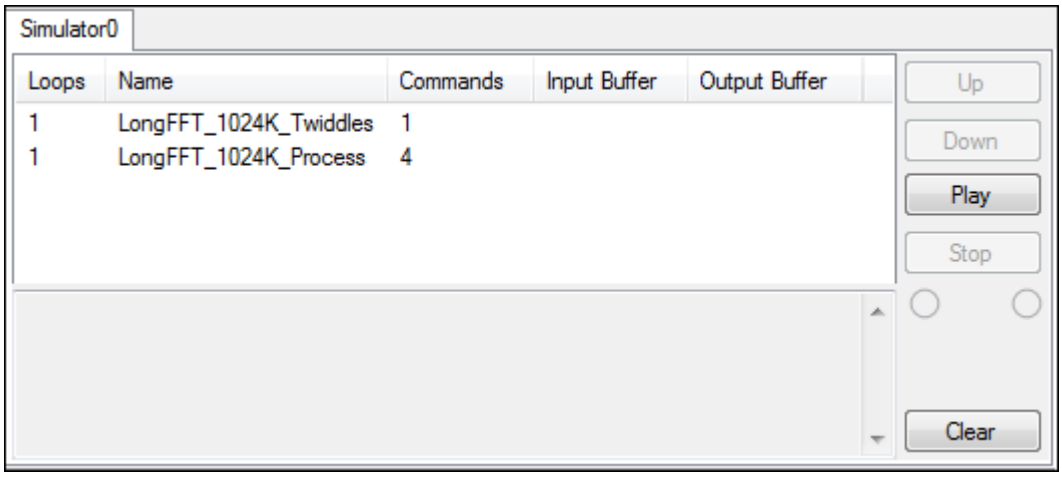


5.2.5 Add to Selected Devices

This function adds the highlighted control vectors to the devices that are currently selected in the device window. They will appear in the player window.

5.3 Player Window

Each device has a player window. This consists of a play list of ‘functions’. A function is simply a control vector that has been added to this list. A function differs from a control vector in that it is player specific, has a data sink and source and can be placed in a loop. It is also possible to view the performance profiling of a control vector based on the settings for the associated device.



5.3.1 Edit

Right click the play list of functions and select **Edit…** The edit window permits the user to set the follow:

- Output Buffer
- Input Buffer
- Loops
- Function Name
- LFSR Enabled
- LFSR Settings
- Performance Profile Settings

5.3.1.1 Output Buffer

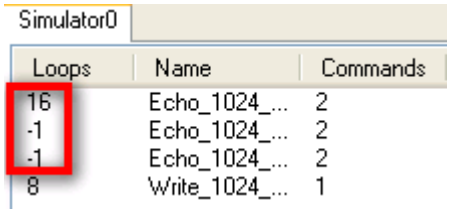
Select the desired buffer to write output data to. One or more buffers must have been created in the Buffer Manager.

5.3.1.2 Input Buffer

Select the desired buffer to read input data from. One or more buffers must have been created in the Buffer Manager.

5.3.1.3 Loops

The loops parameter determines how many times a function should be executed before proceeding to the next function in the list. To make multiple functions part of the same loop set the first function in the loop to the desired number of loops and set all subsequent functions to have a loop count of -1. The following example therefore loops the first three functions 16 times before proceeding to the fourth function.



It is also possible to edit the loop counts directly from the item list.

5.3.1.4 Function Name

The name of the function can be changed.

5.3.1.5 LFSR Enabled

Setting this flag to true enabled the 64-bit LFSR data generator on P0 input. All actual data sent to P0 will be discarded. The SDE does not send any data or increment data source pointers if the LFSR is enabled.

5.3.1.6 LFSR Settings

The default settings for this can be specified against the device – see **Error! Reference source not found.**. The settings are:

- Initial value
- Sequence length
- XOR value

5.3.2 Function Play Order

Using the **Up** and **Down** buttons the sequence of the functions can be re-ordered.

5.3.3 Playing

Once the functions are correctly initialized playing can be started by clicking **Play**. Any I/O activity (P0 and P5 transfers) are indicated by the LED icons and data transfers are summarized in the text box below the play list.

Stop can be pressed at any time. Play does however not stop automatically. This is because the host does not necessarily know when all processing is complete. Data passing through the FTAB device that does not interact with the host PC may still be running. Therefore it is necessary to manually stop processing.

The text box displays details of any I/O and system messages including errors.

5.3.4 Performance Profiling

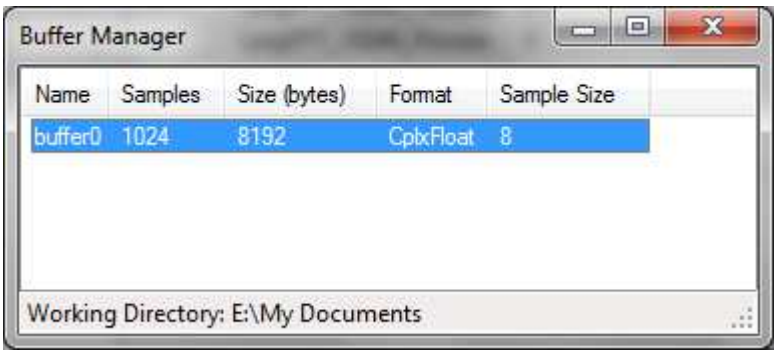
Right clicking on a function and selecting **Performance Profile** displays the graphical result of the hypothetical performance of a function based on the settings for the device (see the device window for settings). The profile is completely based on calculations in the simulator and have nothing to do with actual FTAB devices – this also applies to USB and PCI devices in the device window.

It is possible to change the settings on the fly and compare multiple profiles. Open the settings editor from the device window and begin changing profiler settings. The relevant profiles will update automatically. This is especially useful when investigating the effect of I/O or clock speeds or the enabling or disabling full-duplex I/O (independent). The various settings are detailed below:

Performance Profiler Settings	Description
Core Clock Speed	The speed of the FFT core in Hz
Independent I/O	Can P0 and P5 I/O take place concurrently
Input Clock Speed	The speed of the P0 input controller in Hz
Max Input Speed	The maximum input speed in bytes per second. Set to zero to ignore.
Max Output Speed	The maximum output speed in bytes per second. Set to zero to ignore.
Memory Clock Speed	The speed of the memory controllers P1-P4 in Hz
Output Clock Speed	The speed of the P5 output controller in Hz

5.4 Buffer Manager

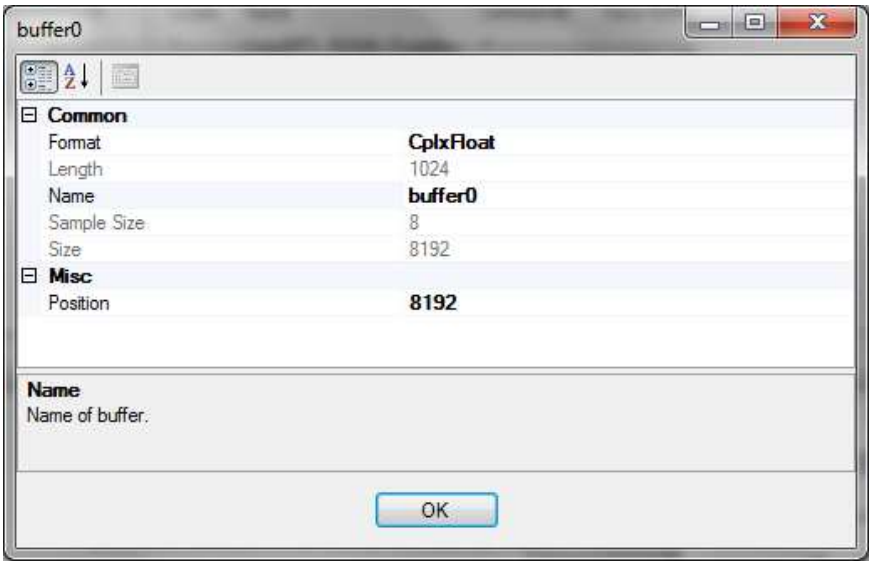
Under the main Data menu is a link to the Buffer Manager. You can also access it via the button on the tool menu. Data buffers are used as sources and sinks for the players. You can load and save data from files or create a limited range of data types from within the Buffer Manager.



It is possible to load and save data setups via the **File** menu. The data in buffers acting as sources is however not automatically saved.

Note: The maximum size of a buffer is likely to be around 1GByte. If you attempt to write more to a buffer an out of memory error will be encountered.

The Buffer Manager permits storing of data in memory and viewing of this data. To add a buffer right click and select **Add Buffer…**

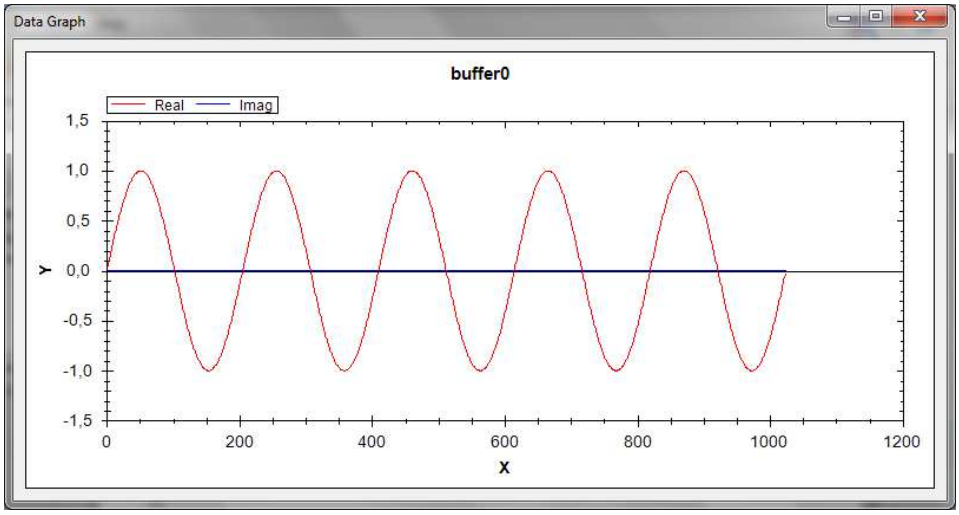


It is also possible to load and save data from file via the **Load…** and **Save…** commands. To save multiple files in one operation use the **Save** command and set a default directory by using **Set Working Directory…**

Clear empties a buffer without deleting it. **Edit…** permits the setting of **Data Format**, **Position**, data **Format** and **Name**. Finally **View…** brings up a graph of the data.

5.4.1 Viewer

The graph features zooming, panning, printing, saving of image, point value display and options to show real, imaginary and absolute values. These features are accessed via right clicking. Note that the performance of the viewer may suffer with very large data sets.

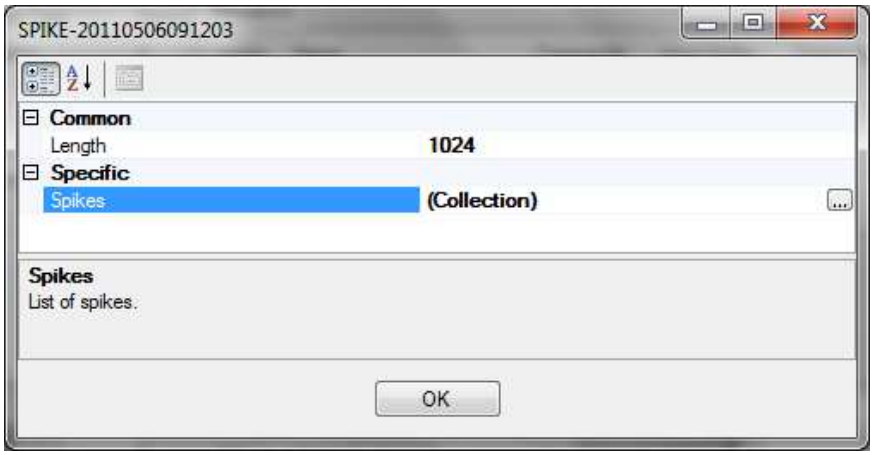


5.4.2 Create Data

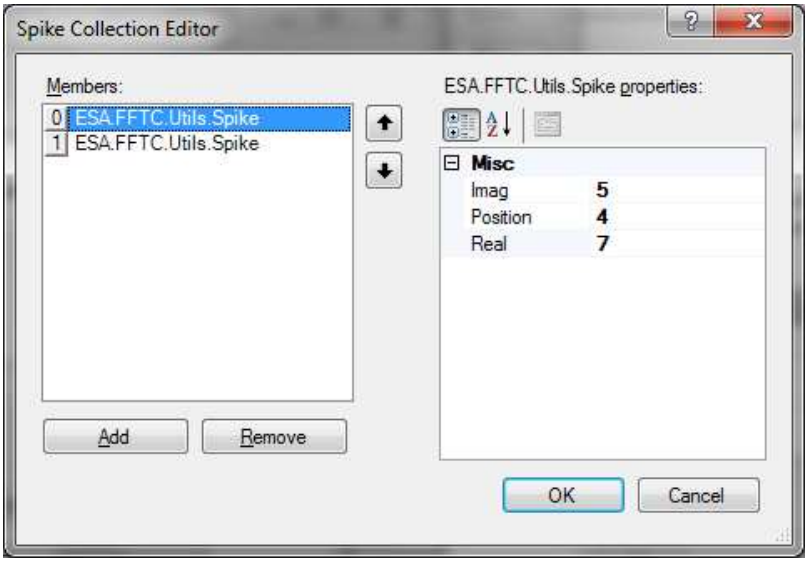
Add a new buffer and right click **Create Data** and then the choice of **Spike**, **Sine** or **Twiddles**. .

All three have a common setting of length in samples. The format of the data generated is as per the current buffer data type. Data can be real or complex. If the buffer is real data then the imaginary settings are ignored.

5.4.2.1 Spike

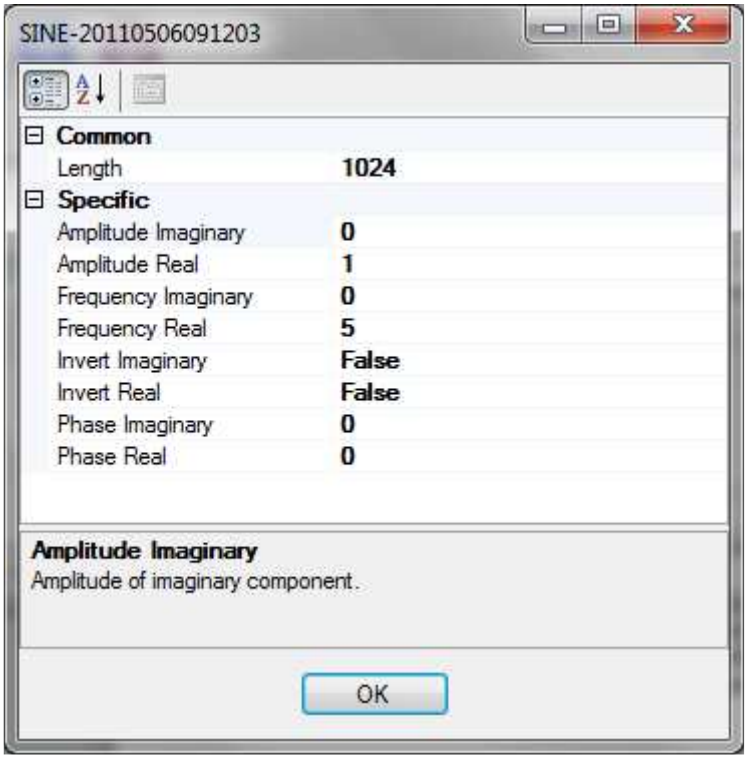


Spikes can be added, edited and removed by clicking the **...** button.



5.4.2.2 Sine

Amplitude, frequency, invert and phase can be specified for both real and imaginary parts.



5.4.2.3 Twiddles

Twiddle vectors are used for performing long 1D FFTs. The FFT core of the FFTC processor can perform FFTs of maximum length 1K points. In order to perform long FFTs the algorithm must be split into multiple passes through the core including

corner turning and multiplication with twiddle vectors. It is most efficient to arrange the long FFT in a form as close to a square as possible i.e. sides of equal length, **Width** = **Height**. This is what the control vector wizard does. If sides are not equal then the width takes precedence and is longer. For example 512K FFT is performed as a square of 1K by 512 point.