AHEAD OF WHAT'S POSSIBLE™

SIGMASTUDIO FOR SHARC (ADSP-SC5XX/ADSP-215XX) - ALGORITHM DESIGNER GUIDE

Document Status	Approved
Approved by	ASH

ANALOG DEVICES, INC.

www.analog.com

Revision List

Table 1: Revision List

Revision	Date	Description
2.1	19.03.2015	Initial draft of the changes made to support ADSP-SC58x
2.2	20.05.2015	Updated with ADSP-SC58x Algorithm Designer details.
2.3	02.07.2015	Minor corrections/ updates made
2.4	10.07.2015	Minor corrections/ updates made
2.5	10.07.2015	Review comments incorporated
2.6	13.07.2015	Minor correction in section 5.5
3.0	13.07.2015	Approved and Base-lined for 3.2.0
3.1	05.10.2015	Minor addition in section 3.1.3 for 3.4.0 Release
4.0	07.10.2015	Approved and Base-lined for 3.4.0
4.1	01.02.2016	Updated for Release 3.5.0
4.2	02.02.2016	Review comments incorporated
5.0	02.02.2016	Approved and Baselined for 3.5.0
5.1	20.04.2016	Moved "Plug-In Version Compatibility Check" section from AE_42_SS4G_QuickStartGuide.pdf document.
6.0	21.04.2016	Approved and Base-lined for 3.6.0
6.1	05.10.2016	Updated for Release 3.8.0
6.2	06.10.2016	Review comments incorporated
7.0	07.10.2016	Approved and Base-lined for 3.8.0
7.1	19.12.2017	Correction in Section 6.
8.0	20.12.2017	Reviewed, approved and Base-lined for 3.12.0
8.1	18.12.2020	Updated for Release 4.6.0
8.2	22.12.2020	Addressed review comments for release 4.6.0
9.0	23.12.2020	Approved and baselined for release 4.6.0
9.1	08.04.2022	Updated for Release 4.7.0
10.0	14.04.2022	Approved and baselined for release 4.7.0

Copyright, Disclaimer & Trademark Statements

Table of Contents

List of Figures

List of Code Examples

1 Introduction

SigmaStudio™ is a development environment from Analog Devices for graphically programming ADI's DSPs. SigmaStudio for SHARC includes an extensive set of algorithms to perform audio processing tasks such as filtering and mixing, as well as basic low-level DSP functions, optimized to run on the SHARC family of processors. SigmaStudio for SHARC also provides support for Analog Devices Software Modules such as the Dolby® Digital AC3 Decoder. SHARC Software Modules can be obtained separately along with their respective SigmaStudio Plug-Ins.

The environment also extends parameter export and filter coefficient generation support for a host microcontroller. Automation API support is provided to connect with many other tools, such as Python, .NET application, Matlab®, and LabVIEW. An easy-to-use graphical interface allows users to create custom filters, compressors, and other audio-shaping algorithms to improve or change the characteristics of the audio. SigmaStudio for SHARC Algorithm Designer is provided to convert existing Software Modules or other SHARC libraries into SigmaStudio Plug-Ins. The environment is integrated with CrossCore® Embedded Studio.

1.1 Scope

Algorithm Designer helps in the integration and use of externally developed Algorithms within SigmaStudio. This document is intended to help engineers integrate externally developed ADSP-SC5xx algorithms with SigmaStudio. The external algorithms to be used with Algorithm Designer can be either developed as library using CrossCore Embedded Studio tools or entered as C source code in the Algorithm Designer Source Editor. The conventions to be followed while implementing the module are also described in this document.

Algorithm Designer can be inserted in the schematic as a generic C source module during the module development. Once the module is finalized, Algorithm Designer generates DLLs (Dynamic Link Libraries) which can be used by SigmaStudio for SHARC for redistributing the Plug-In module. Algorithm Designer can be used to determine the cell parameters and other functionality. Developers are assumed to be familiar with ADSP-SC5xx, SHARC programming, CrossCore Embedded Studio, and SigmaStudio. SigmaStudio for SHARC referred throughout this document is referring to ADSP-SC5xx in particular.

1.2 Organization of this Guide

Section 1: this section contains the introduction.

Section 2: this section gives an overview of the Algorithm Designer use cases.

Section 3: this section gives the details on Designer Control usage.

Section 4: this section gives the details on Plug-In generation.

Section 5: this section describes the coding conventions to be followed while implementing the algorithm.

Section 6: this section gives the module XML details.

Section 7: this section describes the various sub-windows and settings on the Designer.

Section 8: this section explains the Plug-In compatibility check mechanism.

Section 9: this section describes the steps to rebuild the example Plug-Ins.

1.3 Acronyms

ADI	Analog Devices Inc.
CCES	CrossCore Embedded Studio
DLB	DSP Library
DLL	Dynamic Link Library
GUI	Graphical User Interface
LDF	Linker Descriptive File
VISA	Variable Instruction Set Architecture
XML	Extensible Markup Language

1.4 References

[1] Analog Devices SigmaStudio Graphical Development Tool Help
[2] SigmaStudio for SHARC Integration Guide
Analog Devices, Inc.
[3] Microsoft Developer Network
http://msdn.microsoft.com/en-US/
[4] SigmaStudio for SHARC Quick Start Guide
Analog Devices, Inc.

1.5 Additional Information

For more information on the latest ADI processors, silicon errata, code examples, development tools, system services and device drivers, technical support and any other additional information, visit our website at www.analog.com/processors. For additional information on SigmaStudio visit www.analog.com/sigmastudio.

2 Overview

Algorithm Designer offers the flexibility to create a custom SigmaStudio block with runtime control over the parameters. SigmaStudio Algorithm Designer has two use cases:

- Algorithm Designer can be used as SHARC Designer Control module which could be inserted in the schematic as a generic module. SHARC Designer Control can implement any algorithm and offer provision for run-time parameter tuning. Since edits can be made to the source code as well as parameters in the Designer Control and the schematic can be immediately compiled and executed, this is useful during the development phase of the module.
- Algorithm Designer can generate a reusable SigmaStudio block in the form of a DLL from an Algorithm using CrossCore Embedded Studio tools. The resulting SigmaStudio Add-In library file (Plug-In) can be reused and redistributed just like built-in SigmaStudio Algorithms.

This document describes the steps required to use Algorithm Designer as SHARC Designer Control module as well as steps required to generate a custom SigmaStudio Plug-In using CrossCore Embedded Studio tools and Algorithm Designer.

3 SHARC Designer Control Module

SHARC Designer Control is a generic module which can be inserted in the schematic. Once inserted in the schematic, this module can be used to directly implement custom block processing algorithms without the need of generating a Plug-In. Run-time control of parameters are also supported on the Designer Control. An overview of SHARC Designer Control usage is shown below.

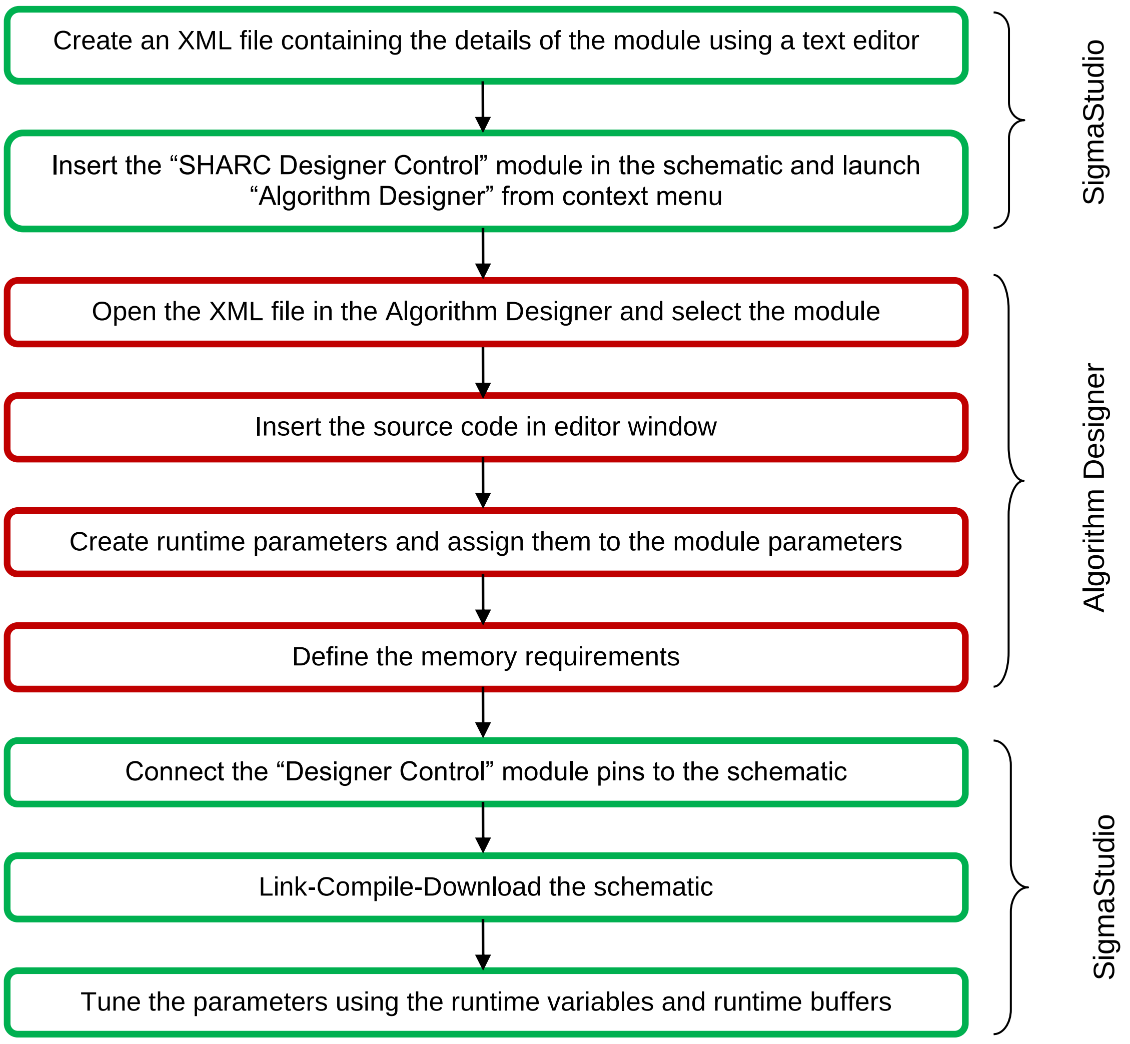

Create an XML file containing the details of the module using a text editor
Insert the “SHARC Designer Control” module in the schematic and launch “Algorithm Designer” from context menu
SigmaStudio
Open the XML file in the Algorithm Designer and select the module
Insert the source code in editor window
Create runtime parameters and assign them to the module parameters
Define the memory requirements
Algorithm Designer
Connect the “Designer Control” module pins to the schematic
Link-Compile-Download the schematic
Tune the parameters using the runtime variables and runtime buffers
SigmaStudio

Figure 1: Using SHARC Designer Control – Overview

Detailed usage of the SHARC Designer Control module is given below.

3.1 Implementing Custom Algorithm using Designer Control

3.1.1 Launch Algorithm Designer

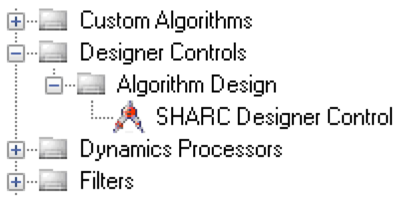


Figure 2: SHARC Designer Control in Toolbox

SHARC Designer Control module has to be first inserted in the schematic. This module is listed under “Designer Controls” category in the tool-box as shown in Figure 2. This module is available only on the block processing schematic. When inserted in the schematic, the module will have no input-output pins.

Algorithm Designer will be used to configure the algorithm source code and parameters of the SHARC Designer Control. The Algorithm Designer window can be launched by clicking “Show Design Window” on the context menu of the module.

3.1.2 Create Module XML

Every module implemented using Algorithm Designer needs an XML file as the starting point. This XML file defines the name of the module, whether the module is block processing or sample processing, all the input-output pins on the module, all the runtime parameters used by the module, source libraries (DLBs) required by the module. Refer to section 6 for more detail on the module XML.

3.1.3 Build the Source Libraries

Algorithm Designer can extract the source of the module from either one or more libraries (DLBs) listed in the Module XML file and from the C source code entered in the source editor of Algorithm Designer. Although it is mandatory to only have any one of these two options, a combination of both DLBs and source code is also supported. If libraries are use, then they should be coded following the coding conventions mentioned in section 5 and built as a library using CrossCore Embedded Studio.

3.1.4 Load Module XML

Once the module XML is created, it has to be loaded in Algorithm Designer as explained in section 7.1.1.

3.1.5 Select the Module

Once the XML is loaded, the module to be implemented has to be selected from the list as explained in section 7.1.2. Once the module is selected, input-output pins will appear on the SHARC Designer Control module inserted in the schematic as defined in the module XML file.

3.1.6 Enter Source Code

The C source code of the module to be implemented can be now entered in the source editor of the Algorithm Designer. Refer to section 7.2 for more details on using the source editor. The source code has to comply with the coding conventions mentioned in section 5. Since the source can also be used in the form of DLBs, entering code in the source editor is optional. C source code entered in the source editor is not supported for the sample processing modules.

3.1.7 Assign Parameters

All the parameters used by the selected module would be displayed in the “Parameter Control” sub-window within Algorithm Designer. We have to assign there parameters with constant values or formulas. In order to assign formulas, we define runtime variables and runtime buffers. These runtime variables and buffers can be modified during runtime. When the actual parameters of the modules are assigned using these runtime parameters, we can modify and tune the actual parameters by modifying the runtime parameters. Refer to section 7.3 for more details.

3.1.8 Enter Memory Requirement

Scratch and State memory required by the module should be entered in the “Memory Requirement” window. Refer to section 7.4 for more details.

3.2 Compiling the Schematic with Designer Control

Once the above steps are completed, the input-output pins on the Designer Control module can be connected to the schematic. Link-Compile-Download the schematic. The Algorithm

Designer window can be either left open or closed before compiling the schematic. Ensure to save the Designer project before closing the Algorithm Designer window.

3.3 Tuning Custom Algorithm using Designer Control

3.3.1 Tune Parameters

Module parameters can be modified by changing the runtime variables and runtime buffers. Since the module parameters are assigned with the runtime parameters, changing the runtime parameters changes the module parameters. Runtime variables can be modified using the numeric up-down control. Runtime buffers can be modified using the 'Edit" button. After the table entries and edited, press 'Update' to send the updated parameters to the target.

3.3.2 Modify Source Code

Changes to the module source code can be made in the source editor within Algorithm Designer. Note that it is required to recompile the schematic once the source code has been modified.

3.3.3 Modify Memory Requirements

Note that it is required to recompile the schematic once the memory requirements have been modified in the Algorithm Designer.

3.4 Finalize the Module

Once the module is finalized, it can be built as a Plug-In and distributed. The process is mentioned in the next section.

4 SigmaStudio Plug-In Generation

The overall process of implementing a new Algorithm Plug-In within SigmaStudio is shown below:

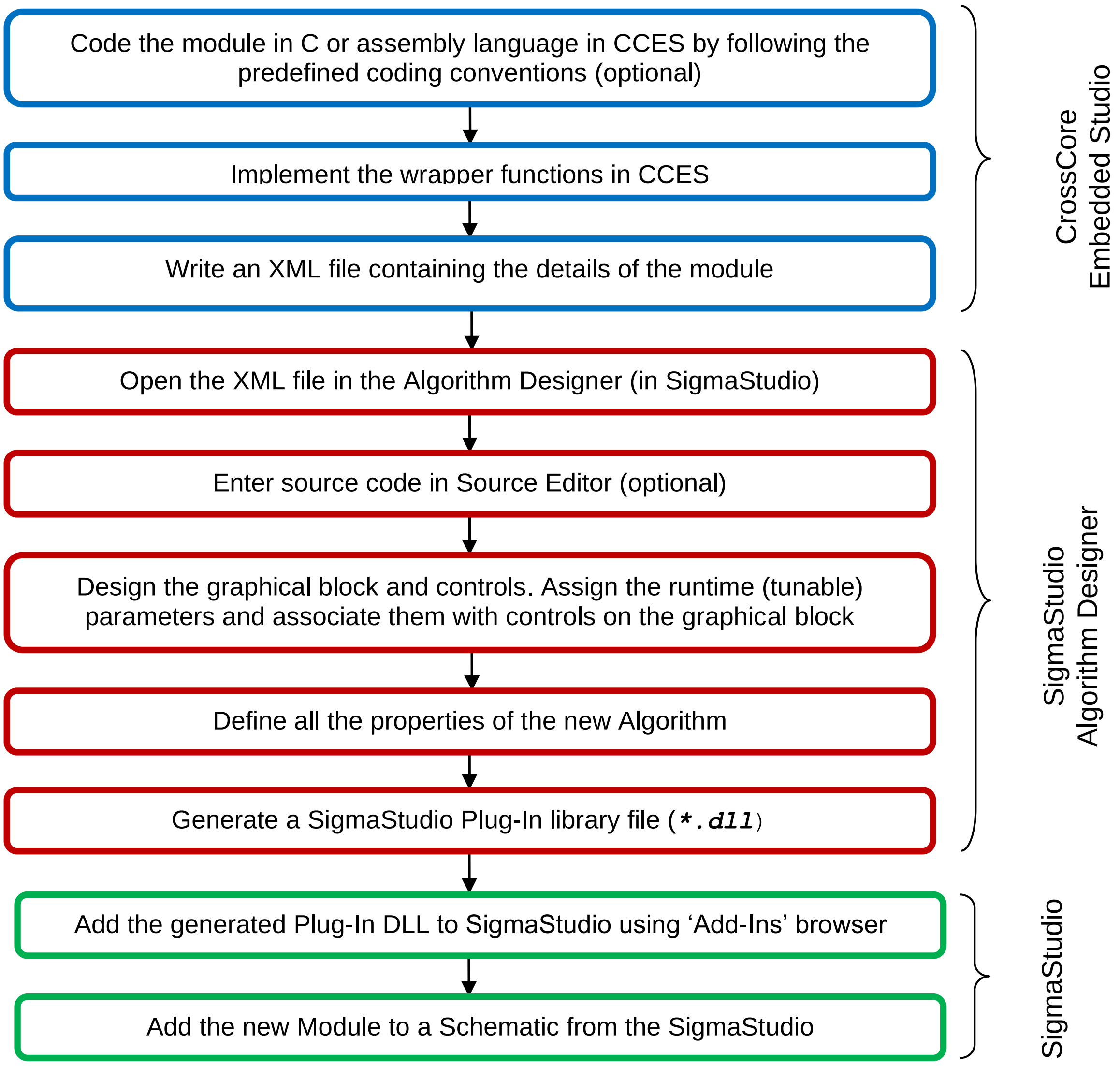

Code the module in C or assembly language in CCES by following the predefined coding conventions (optional)
Implement the wrapper functions in CCES
Write an XML file containing the details of the module
CrossCore Embedded Studio
Open the XML file in the Algorithm Designer (in SigmaStudio)
Enter source code in Source Editor (optional)
Design the graphical block and controls. Assign the runtime (tunable) parameters and associate them with controls on the graphical block
Define all the properties of the new Algorithm
Generate a SigmaStudio Plug-In library file (*.dll)
SigmaStudio Algorithm Designer
Add the generated Plug-In DLL to SigmaStudio using 'Add-Ins' browser
Add the new Module to a Schematic from the SigmaStudio
SigmaStudio

Figure 3: Plug-In Generation Overview

4.1 Designing Custom Module

4.1.1 Implementing the Libraries

Algorithm Designer can extract the source of the module from either one or more libraries (DLBs) listed in the Module XML file and from the C source code entered in the source editor of Algorithm Designer. Although it is mandatory to only have any one of these two option, a combination of both DLBs and source code is also supported. If libraries are use, then they should be coded following the coding conventions mentioned in section 5 and built as a library using CrossCore Embedded Studio.

4.1.2 Create Module XML

Every module implemented using Algorithm Designer needs an XML file as the starting point. This XML file defines the name of the module, whether the module is block processing or sample processing, all the input-output pins on the module, all the runtime parameters used by the module, any source libraries (DLBs) required by the module. Refer to section 6 for more detail on the module XML.

4.1.3 Launching Algorithm Designer

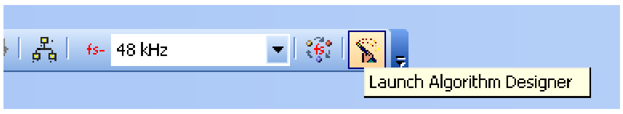


Figure 4: Schematic Control Toolbar

Algorithm Designer can be launched by selecting **Action ➔ Launch Algorithm Designer** or by clicking “Launch Algorithm Designer” tool on the Schematic Control Toolbar as shown in Figure 4. Algorithm Designer pops up as a separate window. The main Algorithm Designer window is shown in Figure 7.

4.1.4 Configure the Designer

The next step involved in generation of a SigmaStudio Plug-In is to configure the Algorithm Designer. This involves loading the module information, assigning source code and parameters, defining all settings etc. within the Algorithm Designer window. These steps are covered in detail in section 7. Various steps involved in this process along with the reference to the details for Plug-In generation are given below.

1. Select the module

    a. Load the Module XML (Refer to section 7.1.1)
    b. Select the Module (Refer to section 7.1.2)
2. Enter the source code (Refer to section 7.2). This step is applicable only to block processing modules. This step is also optional. If the entire source is in the form of libraries (DLBs), this step could be avoided.
3. Assign the parameters
    a. Define runtime parameters (Refer to section 7.3.1)
    b. Assign formulas and values (Refer to section 7.3.2)
4. Define the memory requirements (Refer to section 7.4)
5. Design the Cell GUI
    a. Insert GUI controls (Refer to section 7.5)
    b. Assign runtime control to parameters (Refer to section 7.5.1)
6. Define all the settings
    a. Define tool box settings (Refer to section 7.6.1)
    b. Define Growth and Add settings (Refer to section 7.6.2)
    c. Define Assembly Info settings (Refer to section 7.6.3)
    d. Define Pin Labels (Refer to section 7.6.4)

4.2 Generate Plug-In Assembly

There are 3 options available for plug-in generation;

1. Generate Assembly – Generates UI and Algorithm in a single plug-in DLL
2. Generate Cell Assembly – Generates UI-Only plug-in DLL.
3. Generate Cell and Algorithm Assembly – It either generates UI and Algorithm as separate plug-in DLLs or generates Algorithm-Only DLL using a referenced UI plug-in DLL.

4.2.1 Generate Assembly

This option generates a single plug-in DLL containing both UI and algorithm in it. Along with the plug-in, this option will also generate an interface XML whose details can be found in section C. The interface XML will have details of the plug-in interface.

This plug-in can either be generated by pressing the "Generate Assembly" button in the toolbar or by choosing **Action ➔ Generate Assembly** menu item.

4.2.2 Generate Cell Assembly

This option generates only the UI plug-in DLL. Along with the UI plug-in DLL, this option will also generate an interface XML whose details can be found in section C. The interface XML will have details of the plug-in interface.

This plug-in can either be generated by pressing the "Generate Cell Assembly" button in the toolbar or by choosing **Action ➔ Generate Cell Assembly** menu item. This option does not require the module selection, parameter assignment, growth settings, memory requirements and I/O pin labels to be defined.

4.2.3 Generate Cell and Algorithm Assembly

This involves 2 cases:

1. Generate UI and Algorithm as separate plug-in DLLs.
2. Generate Algorithm Only plug-in DLL using a referenced UI plug-in DLL.

4.2.3.1 Generate Separate UI and Algorithm Plug-In DLLs

This option generates separate plug-in DLLs for UI and algorithm. Along with the 2 plug-in DLLs, this option will also generate an interface XML whose details can be found in section C. The interface XML will have details of the plug-in interface.

This plug-in can either be generated by pressing the "Generate Cell and Algorithm Assembly" button in the toolbar or by choosing **Action ➔ Generate Cell and Algorithm Assembly** menu item.

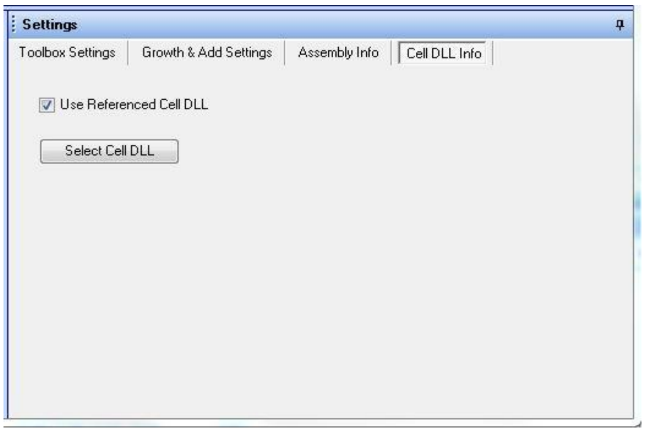
4.2.3.2 Generate Algorithm Only Plug-In DLL

It is possible to generate algorithm only Plug-In using a Referenced UI Plug-In DLL. In order to build an Algorithm Only plug-in DLL using a referenced UI plug-in, a UI plug-in DLL has to be selected first. This selection can be done in the "Cell DLL Info" tab present in the Algorithm

Designer as shown in Figure 5. The following steps have to be followed to select a UI plug-in DLL as reference.

1. The checkbox "Use Referenced Cell DLL" has to be checked.
2. On clicking the button "Select Cell DLL", a dialog box will pop-up prompting the user to select the UI plug-in DLL.

Note: Before selecting the referenced UI plug-in DLL, the "Name" field in the "Toolbox Settings" tab has to be filled with the correct cell name. The selected UI plug-in DLL will then be validated using the cell name entered in the "Name" field as the reference.



☑ Use Referenced Cell DLL

Figure 5: Selecting a Referenced UI Plug-in DLL

This plug-in can either be generated by pressing the "Generate Cell and Algorithm Assembly" button in the toolbar or by choosing **Action ➔ Generate Cell and Algorithm Assembly** menu item. This option does not require GUI Cell design, Control-parameter assignment, rest of the settings other than the cell name, Add settings to be defined. Please note that usage of run-time buffers is currently not supported in this case.

4.3 Using Custom Module in Schematic

4.3.1 Adding Generated DLL to SigmaStudio

The generated assembly file (*.dll) should be added through the Add-Ins mechanism of SigmaStudio. Refer to Annexure E of [4] for more details.

4.3.2 Adding Custom Module in Schematic

Once the generated assembly is added to SigmaStudio, the Algorithm is shown in the SigmaStudio toolbox under the category defined in the Algorithm Designer's settings tab.

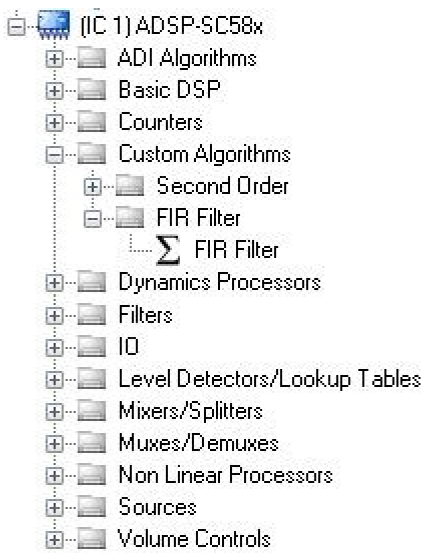


Figure 6: Custom Algorithm Listing in Toolbox

Note: In case Algorithm and UI Plug-In DLLs are separately available, both the DLLs have to be added to Add-Ins.

5 Coding Conventions

SigmaStudio modules can be of two types: Modules supporting sample/stream processing and Modules supporting block processing. Algorithm Designer can generate both sample processing and block processing Plug-Ins, whereas SHARC Designer Control module can be inserted only on block processing schematics and thus supports only block processing.

Algorithm source used for implementing a Module using SHARC Designer Control or for generating SigmaStudio Plug-Ins (***.dll**) for ADSP-SC5xx has to be either DLBs generated using CrossCore Embedded Studio or C source code directly entered in the source editor of Algorithm Designer or a combination of both DLB and source code. Direct insertion of C source code in the Algorithm Designer source editor is supported only for block processing Modules.

This section gives the general rules, guidelines and conventions to be followed while implementing modules compatible with the Algorithm Designer.

5.1 Include File

Include file '***adi_ss_extmod.h***' should be included by the source files.

5.2 Function Prototype

5.2.1 Sample Processing

The entry-point function must have a prefix '`PROCESS_`' to indicate that it is a sample processing Plug-In function for SigmaStudio. Sub-functions in the source file which are referenced only from the main Plug-In function or other sub-functions need not follow any convention. The general prototype of any Plug-In entry-point function performing sample processing is shown below.

```
void PROCESS_<algorithm_name> (float pInput[], float pOutput[], float pState[],
float pParameter[], int nRepCount)
```

Assembly entry-point function names must have the '`PROCESS`' prefix. The general template of any Plug-In sample processing assembly function should be as shown below. It is compulsory to end the function with '`PROCESS_<algorithm_name>..end`' in case of byte addressed Plug-Ins and '`_PROCESS_<algorithm_name>.end`' in case of word addressed Plug-Ins.

```
.section/pm seg_swco;
.global PROCESS_<algorithm_name>.;

PROCESS_<algorithm_name>.:
modify(i7,0xfffffffc); /* offset depends on the required frame space */

/* custom code */

i12 = dm(0xffffffff,i6);
jump (m9,i12)(db);
rframe;
nop;
PROCESS_<algorithm_name>..end:
```

Code 1: Sample Processing Assembly Function Template (Byte Addressed)

When word addressed mode is used, the template is as shown below.

```
.section/pm seg_swco;
.global _PROCESS_<algorithm_name>;

_PROCESS_<algorithm_name>:
modify(i7,0xfffffffc); /* offset depends on the required frame space */

/* custom code */

i12 = dm(0xffffffff,i6);
jump (m9,i12)(db);
rframe;
nop;
_PROCESS_<algorithm_name>.end:
```

Code 2: Sample Processing Assembly Function Template (Word Addressed)

General rules to be followed are:

1. Inputs samples, output samples, state buffer and parameter buffer should be accessed only through the pointers in the function arguments.
2. If there are multiple input or output channels for the Module, samples corresponding to these channels are interleaved within the input/output buffer.
3. The parameter/state buffer pointer is of type float. This pointer should be type casted to an integer pointer for accessing integer parameters/state variables. Refer to the example code in section 5.5.
4. The argument '`nRepCount`' represents the growth count. Growth count is a common terminology used in SigmaStudio and each Module has a different interpretation of growth count. For example, in General Second Order Filter, the growth count represents the number of cascaded biquad filters, whereas in Mute Algorithm the growth count represents number of channels. Refer to [1] for more details.

5.2.2 Block Processing

The entry-point function must have a prefix ‘BPROCESS_’ to indicate that it is a block processing Plug-In function for SigmaStudio. Sub-functions in the source file which are referenced only from the main Plug-In function or other sub-functions need not follow any convention. The general prototype of a block processing Plug-In entry-point function is as shown below.

```
void BPROCESS_<algorithm_name> (SSBlockAlgo* pBlockAlgo)
```

Assembly entry-point function names must have the prefix ‘BPROCESS_’. The general template of any Plug-In block processing assembly function should be as shown below. It is compulsory to end the function with ‘BPROCESS_<algorithm_name>..end’ in case of byte addressed Plug-Ins and ‘_BPROCESS_<algorithm_name>.end’ in case of word addressed Plug-Ins.

```
.section/pm seg_swco;
.global BPROCESS_<algorithm_name>.;

BPROCESS_<algorithm_name>.:
modify(i7,0xfffffffc); /* offset depends on the required frame space */

/* custom code */

i12 = dm(0xffffffff,i6);
jump (m9,i12)(db);
rframe;
nop;
BPROCESS_<algorithm_name>..end:
```

Code 3: Block Processing Assembly Function Template (Byte Addressed)

Same function when implemented in word addressed mode must follow the below template.

```
.section/pm seg_pmco;
.global _BPROCESS_<algorithm_name>;

_BPROCESS_<algorithm_name>:
modify(i7,0xfffffffc); /* offset depends on the required frame space */

/* custom code */

i12 = dm(0xffffffff,i6);
jump (m9,i12)(db);
rframe;
nop;
_BPROCESS_<algorithm_name>.end:
```

Code 4: Block Processing Assembly Function Template (Word Addressed)

In addition to the process function, block Algorithms can also have an initialization function. Initialization functions are used to initialize the Module in case of decoders and 3rd party post processing Modules. The function name must have the prefix '`INIT_`' to indicate that it is the initialization function of the Module. The general prototype of a block processing Plug-In initialization function is as shown below.

```
void INIT_<algorithm_name> (SSBlockAlgo* pBlockAlgo)
```

If a Module must process a specific number of samples that is different to what SigmaStudio passes in, then BPROCESS wrapper function should deal with it by internal buffering.

5.3 Interface Structures

5.3.1 SSBlockAlgo

```
typedef struct _SSBlockAlgo
{
    int32_t     nInputs;
    int32_t     nOutputs;
    Block      *pInputs;
    Block      *pOutputs;
    int32_t     nGrowth;
    int32_t     nGrowthB;
    void       *pParam;
    float32_t  *pState;
    float32_t  *pScratchDM;
    float32_t  *pScratchPM;
    float32_t  *pStateB;
    float32_t  *pStateC;
    float32_t  *pExtPreState;
    int32_t    *pExtSymbols;

}SSBlockAlgo;
```

Description

The structure `SSBlockAlgo` is used to pass Module properties as arguments to a block processing Plug-In.

Fields

- `nInputs`
  Number of input pins/channels.

- `nOutputs`
  Number of output pins/channels.
- `pInputs`
  Pointer to the array of input block structure.
- `pOutputs`
  Pointer to the array of output block structure.
- `nGrowth`
  Growth count of the Module.
- `nGrowthB`
  Output growth count of the Module when 2D growth is enabled.
- `pParam`
  Pointer to the parameter memory to be used by the Module.
- `pState`
  Pointer to the primary state memory to be used by the Module.
- `pScratchDM`
  Pointer to the scratch memory in DM block to be used by the Module.
- `pScratchPM`
  Pointer to the scratch memory in PM block to be used by the Module.
- `pStateB`
  Pointer to the second state memory to be used by the Module.
- `pStateC`
  Pointer to the third state memory to be used by the Module.
- `pExtPreState`
  Pointer to the extended precision state memory to be used by the Module.
- `pExtSymbols`
  Not used for ADSP-SC5xx.

5.3.2 Block

```
typedef struct _Block
{
    BlockProperties *pBlockProperties;
    float           *pSamples;

} Block;
```

Description

The structure `Block` is used to define the properties of each input/output channel.

Fields

- `pBlockProperties`
  Pointer to the block properties structure which defines the properties of the block input/output channel.
- `pSamples`
  Pointer to the block of samples.

5.3.3 BlockProperties

```
typedef struct _BlockProperties
{
    int32_t     nSamplingRate;
    int32_t     nBlockSize;
    int32_t     nReserve0;
    int32_t     nReserve1;

} BlockProperties;
```

Description

The structure `BlockProperties` is used to define the properties of each memory block.

Fields

- `nSamplingRate`
  Sampling rate of the input channel expressed in Hz.

- `nBlockSize`
  Module Block Size in number of samples.
- `nReserve0`
  Reserved. Not supported in this version.
- `nReserve1`
  Reserved. Not supported in this version.

5.4 SigmaStudio Memory Types

Four different types of data memory are used in the externally coded Algorithm. They are explained below.

5.4.1 Parameters

Parameters are data generated based on the control data received from the Cell. Runtime Tuning is achieved by modifying the parameter data. Parameters are stored in the parameter buffer and memory is allocated per instance of the Module. Memory for storing parameters in the parameter buffer is allocated by SigmaStudio and the pointer to the memory is passed as an argument to the Plug-In function.

5.4.2 State

State memory is used to store the internal state of the Algorithm. Every instance of the function/Algorithm has dedicated state memory. This memory is allocated by SigmaStudio and the pointer to the memory is passed as an argument to the Plug-In function. The size of the state memory required by the Algorithm should be mentioned in the Algorithm Designer. Refer to section 7.4 for more details. The state memory is initialized with zeros by SigmaStudio every time before downloading a new Schematic. Four state buffers (pointers) are available for the Plug-In to use.

5.4.3 Constant Tables

All global variables and global tables in the Module are treated as constant tables by SigmaStudio. In case of ADSP-SC5xx only one instance of this memory is allocated per schematic. This memory is mapped on to the parameter buffer.

Note: Modification of this memory is allowed even though this is treated as constant table by SigmaStudio. This memory is shared across all modules in any given schematic.

5.4.4 Scratch

Temporary buffers or variables required by the Algorithm can be mapped to the scratch memory. This memory is shared by all the Modules in the Schematic. There are 2 types of Scratch memory; memory mapped to DM block and memory mapped to PM block. An Algorithm can choose to use either or both based on the requirement. Both scratch memories are allocated by SigmaStudio and the pointer to the memory is passed as a parameter to the Plug-In function. The size of the scratch memory required by the Algorithm should be specified in the Algorithm Designer. Refer to section 7.4 for more details. Scratch memory is available only for block processing Algorithms. Total amount of scratch Memory reserved by SigmaStudio is the maximum of scratch memory requirement by Modules that are part of the Schematic.

5.5 General Guidelines

1. All global variables/buffers/tables accessed from an Algorithm are shared across instances of all the Algorithms inserted in the schematic.
2. Code and data can be placed in any section.
3. Only extended precision state memory is supported. Tables or parameters cannot be in extended precision.
4. Extended precision access must be cleared at the beginning and restored before exiting if the Plug-In uses either State B or State C memory buffers. This is because Applications may place the 32-bit state memory buffers in the same block as the Extended Precision state memory buffer.
5. Since the input buffers of a Plug-In can be used, with the help of a T-connector, by many subsequent Modules in a Schematic, care should be taken to preserve the input buffers and to avoid overwriting of these buffers.
6. If there are 2 functions with the same name defined in a single file, the linker gives a "Multiply defined symbol" error when the schematic which uses the function is link-compile-downloaded. If the functions with same name are defined in different files, the linker doesn't throw any error during the schematic compilation and linking. It uses whichever function definition it finds first during linking. Hence, it is advisable to use unique function names if it is not intended for the functions to be shared across modules or with the target application.

5.6 Example

The example code below shows a block processing implementation of a volume control function.

```
#include "adi_ss_extmod.h"

#pragma section("seg_pmco")
void BPROCESS_Scale (SSBlockAlgo* pBlkAlgoInfo)
{
        int index, sample, gain, blockSize, repCount;
        float *pInput, *pOutput;

        repCount = pBlkAlgoInfo->nGrowth;

        for(index = 0; index < repCount; index++)
        {
                blockSize = pBlkAlgoInfo->pInputs[index].pBlockProperties->nBlockSize;
                gain = ((float *)pBlkAlgoInfo->pParam)[index];

                pInput = pBlkAlgoInfo->pInputs[index].pSamples;
                pOutput = pBlkAlgoInfo->pOutputs[index].pSamples;

                for(sample = 0; sample < blockSize; sample++)
                {
                        pOutput[sample] = pInput[sample] * gain;
                }
        }
}
```

Code 5: Volume Control Implemented as Block Processing Plug-In

The example code below shows a sample processing implementation of a volume control function.

```
#include "adi_ss_extmod.h"

#pragma section("seg_pmco")
void PROCESS_Scale (float pInput[], float pOutput[], float pState[], float pParameter[], int
nRepCount)
{
        int index;
        float gain;

        for(index = 0; index < nRepCount; index++)
        {
                gain = pParameter[index];
                pOutput[index] = pInput[index] * gain;
        }
}
```

Code 6: Volume Control Implemented as Sample Processing Plug-In

Other example implementations can be found under '***`Host\Examples\Sample Plug-Ins`***' folder.

6 Module XML

The Module XML file is an input to the Algorithm Designer and contains the source definitions for the Module which includes name and path of the module binary, parameter details, pin details etc. Each XML can contain Module source definitions for one or more Modules. The following items are included in the source definition.

- Name, path and target processor of DLBs of the Module (ADSP-SC5xx).
- List and details of all the runtime parameters of the Module.
- Details of the input and output pins.

Below example shows the same 'Scaler' XML for ADSP-SC5xx implementation.

```
<?xml version="1.0" standalone="yes"?>
<SS4SH name="ss4sh_module_xml" description="SigmaStudio for SHARC Algorithm Designer"
version="3.4.0.0">
       <module name="Scaler" block="TRUE">
              <dlb name="Scaler_wrapper.dlb" target="ADSP-SC58x" embed = "FALSE" path="..\lib\"/>
              <dlb name="Scaler.dlb" target="ADSP-SC58x" embed = "FALSE" path="..\lib\"/>
              <parameter name="Gain" isbuffer="NO" format="INT" />
              <pin type="DATA" direction="INPUT" />
              <pin type="DATA" direction="OUTPUT" />
       </module>
</SS4SH>
```

Code 7: Example XML for a Scaler Module (ADSP-SC5xx)

6.1 XML Elements

6.1.1 <SS4SH>

`SS4SH` is the outermost element of the XML. This can appear only once in the XML and should be exactly as shown in the example above.

The attributes of `SS4SH` element are:

- `name`
  String indicating the name of the xml. This should always be "ss4sh_module_xml". If the name is different, Algorithm Designer treats this as an invalid XML file.
- `description`
  Description of the xml of type string.
- `version`
  Version of the SigmaStudio for SHARC package in a.b.c.d format.

The contents of the `SS4SH` element are:

- `module`
  Refer to section 6.1.2 for details. One `SS4SH` element can have any number of module elements.

6.1.2 <module>

The `module` element defines a Plug-In. More than one Plug-In can be defined in a single XML, i.e., there can be more than one `module` element inside an XML.

The attributes of `module` element are:

- `name`
  String indicating the name of the Module. If the Plug-In process function is `BPROCESS_MyModule`, then the name should be "MyModule".

- `block`
  "TRUE" if the Module is block processing and "FALSE" if the Module is sample processing.

The contents of the `module` element are:

- `dlb`
  Refer to section 6.1.3 for details. One `module` element can have any number of `dlb` elements. It is mandatory to have at least one `dlb` element if the Module is sample processing. There is provision to directly insert C source code in the Designer window in the case of Block processing Module and hence need not have a `dlb` element.

- `parameter`
  Refer to section 6.1.4 for details. One `module` element can have any number of `parameter` elements.

- `pin`
  Refer to section 6.1.5 for details. One `module` element can have any number of `pin` elements. It is mandatory to have at least one `pin` element.

6.1.3 <dlb>

The `dlb` element defines the DLB containing the source of the Module. If there are more than one DLBs per module, each one should be defined in a separate `dlb` element. One Module can have more than one `dlb` elements. `dlb` element is mandatory for sample processing Module and optional for Block processing module.

The attributes of `dlb` element are:

- `name`
  String indicating the name of the DLB file. Each `dlb` element should have a unique name.
- `embed`
  'TRUE' if the DLB is to be embedded inside the Plug-In DLL and 'FALSE' if the DLB is going to be referenced from an absolute or relative location.
- `target`
  Indicates the target processor. This could be specific targets like 'ADSP-SC589' or a family like 'ADSP-SC58x'.
- `path`
  String indicating the folder path of the DLB file. The path can either be an absolute location or a relative location. While using a relative location, the path should be relative to the XML path when `embed` is 'True'. When `embed` is 'False' and build as a Plug-In, the path is relative to Plug-In DLL location. When `embed` is 'False' and used as a SHARC Designer Control, the path is relative to schematic. Use back-slash '\' to separate folders in the path.

The `dlb` element cannot have any content.

6.1.4 <parameter>

The `parameter` element defines the parameters of the Module. One `parameter` element is required for each runtime parameter of the Module. There can be any number of `parameter` elements inside a `module` element.

The attributes of `parameter` element are:

- `name`
  String indicating the name of the parameter.
- `isbuffer`
  "YES" if the parameter is a table or "NO" if the parameter is a single variable.
- `format`
  Indicates the format of the parameter. Can be "INT", "FLOAT", "32.0" (same as "INT"), 31.1, 30.2…… 0.32 etc.

The `parameter` element cannot have any content.

6.1.5 <pin>

The `pin` element defines the input/output pins on the Module. It is mandatory for any Module to have at least one pin. Hence there should be at least one `pin` element. Pins are inserted in the order of the `pin` element.

The attributes of `pin` element are:

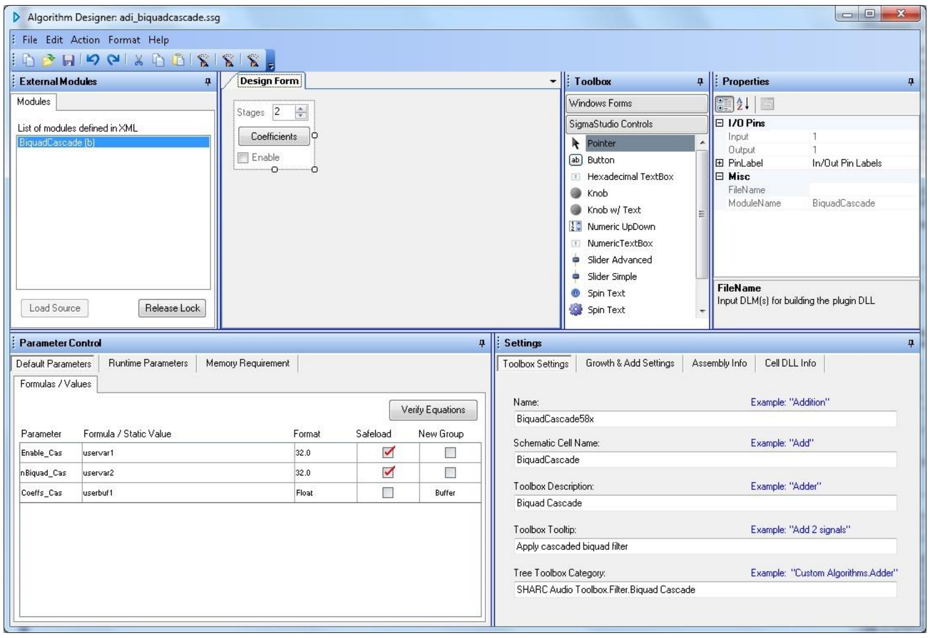
- `type`
  String indicating the type of the pin. The options are:
  - `DATA`: Pin carries analog audio samples.
  - `CONTROL`: Pin carries control signals.
- `direction`
  "INPUT" if the pin is an input pin and "OUTPUT" if the pin is an output pin.

The `pin` element cannot have any content.

7 Designer Window

Algorithm Designer is a custom 'Windows Forms Designer' control built into the SigmaStudio. A variety of Windows and SigmaStudio graphic controls can be assembled and arranged to create a custom user interface for the externally developed Algorithms.

The process of using SHARC Designer Control in the schematic as well as the process of generating a Plug-In assembly for SigmaStudio involves reading Module details from a Module XML file, entering source code in the editor window, assigning parameters, and defining runtime controls and settings. Each step is explained below in detail.



Parameter	Formula / Static Value	Format	Safeload	New Group
Enable_Cas	uservar1	32.0	✓	☐
nBiquad_Cas	uservar2	32.0	✓	☐
Coeffs_Cas	userbuf1	Float	☐	Buffer

Name: BiquadCascade58x — Example: "Addition"
Schematic Cell Name: BiquadCascade — Example: "Add"
Toolbox Description: Biquad Cascade — Example: "Adder"
Toolbox Tooltip: Apply cascaded biquad filter — Example: "Add 2 signals"
Tree Toolbox Category: SHARC Audio Toolbox.Filter.Biquad Cascade — Example: "Custom Algorithms.Adder"

Figure 7: Algorithm Designer Window

The sub windows within the Algorithm Designer are as listed below:

- **External Modules** - This window is used to select the Plug-In which is to be converted to SigmaStudio assembly using the Algorithm Designer.
- **Parameter Control** - This window lists all the parameters within the selected Plug-In. The user can create runtime variables and assign them to Plug-In parameters.
- **Settings** - The name, description, toolbox attributes and other settings of the Plug-In are assigned in the settings window.
- **Toolbox** - The Toolbox displays a list of controls available for use in the user interface design.

- **Properties** - This window displays the properties of the Plug-Ins. While using the toolbox controls, this window can be used to view and edit the properties of the toolbox controls.
- **Design Form** - The Design Form pane in the middle is the background for the custom design interface on which all controls are arranged.

The following steps in section 7.1 to section 7.4 are to be followed for using Algorithm Designer as SHARC Designer Control module in schematic. Steps in section 7.1 to section 7.6 are to be followed for generating Cell and Algorithm in a single plug-in DLL.

7.1 Module Selection

7.1.1 Load the Module XML

The first step is to export the Module details and code to Algorithm Designer by loading the source XML. Click the ‘Load Source’ button on the ‘External Module’ window. This opens an ‘Open’ dialog in which the user can select an XML file for source import.

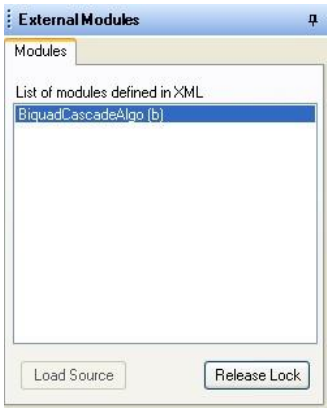


Figure 8: External Module Selection Window

Note: The letter (b) or (s) at the end of the Algorithm name signifies the type of the Algorithm; either block processing or sample processing respectively.

7.1.2 Select the Plug-In

Once an XML file is selected in the ‘External Modules’ window, the list box within the ‘External Module’ window lists the Modules defined in the XML file. Select any Module from the list by clicking on the list. Upon selection, the ‘Properties’ window lists the properties of the selected

Module and the 'Parameter Control' window displays all the parameters associated with the selected Module.

Note: Only one Module can be selected at a time and hence there can be only one SigmaStudio Plug-In Module in any SigmaStudio Add-In library file (`*.dll`) generated by the Algorithm Designer. Use external DLL merging utilities to combine multiple DLLs generated by Algorithm Designer into a single DLL with multiple Plug-Ins.

7.2 Insert C Source Code

Algorithm Designer has a source editor where C source code of the Module could be directly entered. The source editor can be launched from the 'Action' menu of the Algorithm Designer as shown in Figure 9 below.

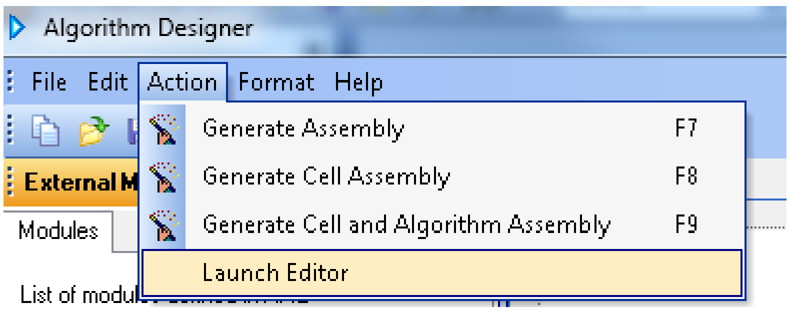
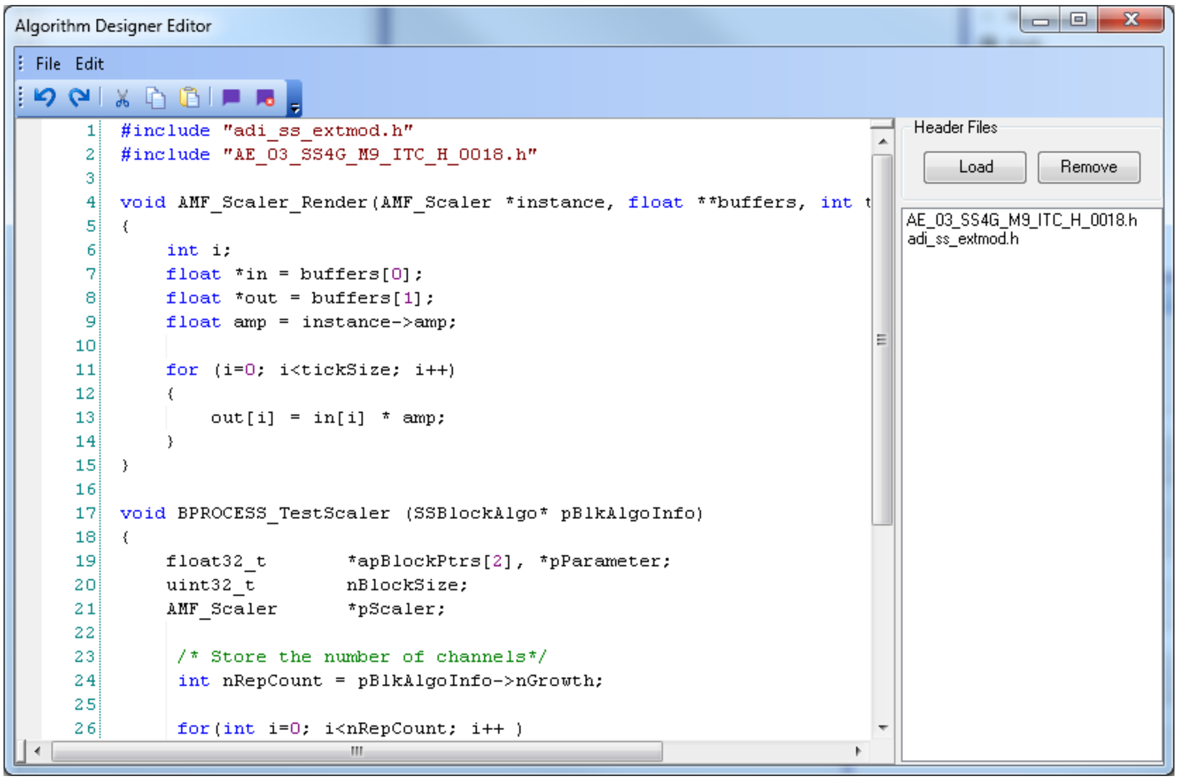


Figure 9: Launching Source Editor

Algorithm Designer source editor is shown in Figure 10.



```
#include "adi_ss_extmod.h"
#include "AE_03_SS4G_M9_ITC_H_0018.h"

void AMF_Scaler_Render(AMF_Scaler *instance, float **buffers, int t
{
    int i;
    float *in = buffers[0];
    float *out = buffers[1];
    float amp = instance->amp;

    for (i=0; i<tickSize; i++)
    {
        out[i] = in[i] * amp;
    }
}

void BPROCESS_TestScaler (SSBlockAlgo* pBlkAlgoInfo)
{
    float32_t       *apBlockPtrs[2], *pParameter;
    uint32_t        nBlockSize;
    AMF_Scaler      *pScaler;

     /* Store the number of channels*/
     int nRepCount = pBlkAlgoInfo->nGrowth;

     for(int i=0; i<nRepCount; i++ )
```

Figure 10: Algorithm Designer Source Editor

C source code can be inserted in the editor on the left side. The code has to follow the conventions mentioned in section 5. The ‘Header Files’ group box on the right side of the window is used to load the header files included in the source code on the right side. Header files can be added/updated using the ‘Load’ button. Use the ‘Remove’ button to remove the selected header file from the list. The source entered in the editor and header files selected will get saved in the Algorithm Designer project. Source code entred in the Algorithm Designer source editor is not supported for sample processing schematics. It is not mandatory to have source code entered in the editor window since block processing Module supports a combinatipn of libraries and C source code. Follwoing combinations are supported.

- Module implemented exclusively using DLBs
- Module implemented using only source code entered in the source editor.
- Module implemented using a combination of source code and libraries. Entry-point function is in one of the libraries.
- Module implemented using a combination of source code and libraries. Entry-point function is implemented in the source editor.

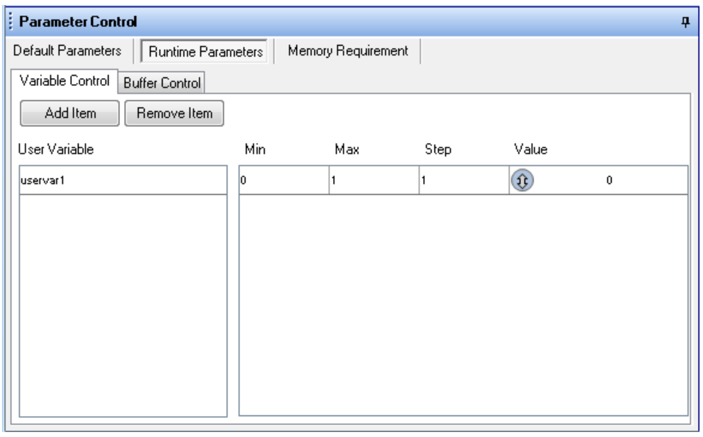
7.3 Parameter Assignment

7.3.1 Define Runtime Parameters

The Designer's 'Runtime Parameter' feature allows the user to define runtime modifiable parameters and link them to one or more Algorithm parameters. Runtime parameters are of two types: runtime variables and runtime buffers.

7.3.1.1 Variable Control

Runtime variables are added/removed using the 'Variable Control' window shown in the figure below.



User Variable	Min	Max	Step	Value
uservar1	0	1	1	0

Figure 11: Runtime Variable Control Window

Following are the columns in the Variable Control grid.

- **User Variable** – Name of the runtime variable. This can be renamed.
- **Min** – Minimum value of the runtime variable control.
- **Max** – Maximum value of the runtime variable control.
- **Step** – Step size of the runtime variable control.
- **Value** – Current value of the user variable and numeric up-down control to modify the value.

The run-time control using the numeric up-down is active only when used as SHARC Designer Control module in a schematic. When the module is built as a SigmaStudio Plug-In, the value at the time of Plug-In generation remains the default value of the user variables. When a GUI control is assigned to the user variable, the default value set on the assigned GUI control overrides the default value set for the user variable.

7.3.1.2 Buffer Control

Runtime buffers are added/removed using the ‘Buffer Control’ window shown in the figure below. The second column in each row is the buffer size.

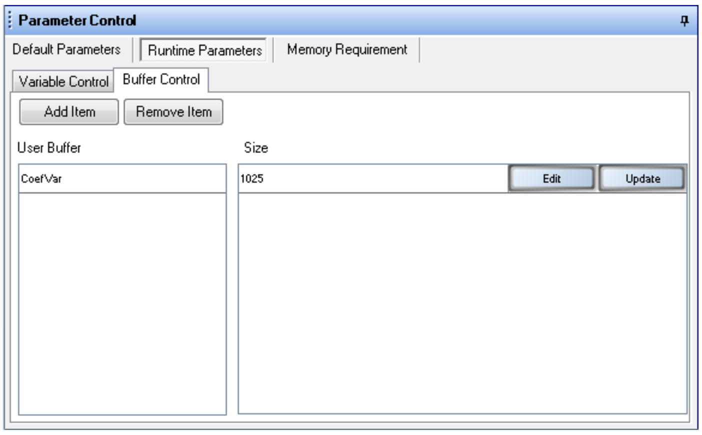


Figure 12: Runtime Buffer Control Window

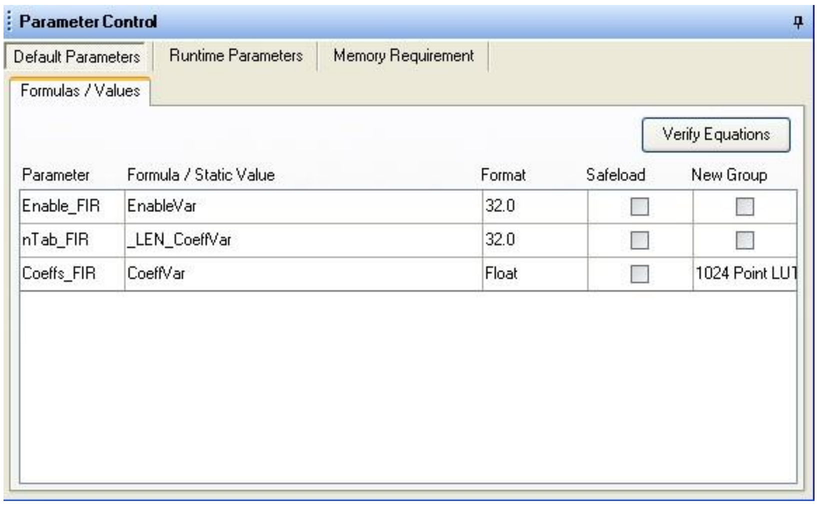
Following are the columns in the Buffer Control grid.

- **User Buffer** – Name of the runtime buffer. This can be renamed.
- **Size** – Size of the buffer.
- **Edit** – Use this button to edit the buffer entries.
- **Update** – Use this button to update the buffer with the edited entries.

Note that ‘Edit’ and ‘Update’ are applicable only when used as SHARC Designer Control module in a schematic. When a table control is assigned to the runtime buffer, the size of the runtime buffer is set to the size of the table control. When the module is built as a SigmaStudio Plug-In and the buffer assigned to a Table control, the default value set on the assigned GUI control overrides the default value set for the user buffer.

7.3.2 Assigning Formulas and Values

The parameters and tables used within the selected Plug-In are listed in the ‘Default Parameter’ tab within ‘Parameter Control’ window. The values and properties of the parameters can be assigned through this window.



Parameter	Formula / Static Value	Format	Safeload	New Group
Enable_FIR	EnableVar	32.0	☐	☐
nTab_FIR	_LEN_CoeffVar	32.0	☐	☐
Coeffs_FIR	CoeffVar	Float	☐	1024 Point LUT

Figure 13: Default Module Parameters

The different columns in the parameter grid are as described below:

7.3.2.1 Parameter

This is the name of the parameter used in the Module as described in the XML. This field cannot be modified by the user within Algorithm Designer.

7.3.2.2 Formula / Static Value

This is the field where the values are assigned to the parameters. This field is different for individual parameters and tables.

- **Individual Parameter** - The user can either set a static value to the parameter or assign a formula which is a function of one or more runtime variables and size of runtime buffers. To set a formula, the user needs to define one or more runtime variables or runtime buffers. See section 7.3.1.1 and 7.3.1.2 for more details on runtime variables and runtime buffers. ‘`FS`’, ‘`RepCount`’ and ‘`BlockSize`’ are predefined keywords and can be used in the formula to represent the ‘Schematic Sampling Rate’, ‘Growth Count’ and ‘Schematic Block Size’ (for block processing Modules) respectively. In order to use the size of a runtime buffer as a variable use the buffer name with a prefix ‘`_LEN_`’. For e.g. size of a runtime buffer ‘`LUT`’ is denoted by ‘`_LEN_LUT`’.

  Examples:

  - `3.14`
  - `uservar1 + 10 + RepCount`
  - `uservar1 + uservar2 * 4 + _LEN_userbuf1`

- `uservar1 + (BlockSize>256?20:10)`
- `uservar1 + (BlockSize % 256 >= 0?1:0)`
- `uservar1 + ((BlockSize/2)^2) * _LEN_userbuf1`
- `uservar1 + Math.Round(Math.Sqrt(BlockSize)) + Math.Min(uservar1, BlockSize) + Math.Log(BlockSize, 2)`

In the above examples uservar1, uservar2 and userbuf1 are runtime variables and buffers.

Note: Other Math class functions such as Math.Abs, Math.Acos, Math.Asin, Math.Atan, Math.Cos, Math.Cosh, Math.Exp, Math.Log10, Math.Max, Math.Sign, Math.Sin, Math.Sinh, Math.Tan, Math.Tanh, Math.Truncate and constants such as Math.E and Math.PI are also supported.

- **Tables** - The user can set a runtime buffer to the table. See section 7.3.1.2 for more details on runtime buffers. **Formula or constants are not supported and should not be assigned to Table parameters**.

7.3.2.3 Format

This field displays the binary representation format (float/fixed point) of the parameter. The format is set in the XML and cannot be modified in the Algorithm Designer.

7.3.2.4 Safeload

Some Algorithms require that a set of parameters be updated at once and the update in the parameter memory happens while the Algorithm is not being executed.

For example, in a Bi-quad filter, 5 filter coefficients form the parameter set. To successfully update the parameters of the Bi-quad filter, it is mandatory to satisfy the following conditions:

- All the 5 coefficients are updated at once.
- Coefficients are updated while the filter is not being executed.

It is required to ensure that the filter is executed with a set of stable coefficients. If either of the above two conditions are not met, the resultant parameter set used in the Bi-quad filter can be a mix of coefficients from two different parameter sets and this can lead to filter instability.

If the 'Safeload' checkbox is selected, the Algorithm Designer ensures that the second condition mentioned above is met, i.e., corresponding parameter is updated in the parameter memory while the Algorithm is not being executed.

In case of Tables, when 'Safeload' is enabled, 5 parameters are sent to the SHARC Target at the same time and are updated in the SHARC Target memory while the Algorithm is not getting executed.

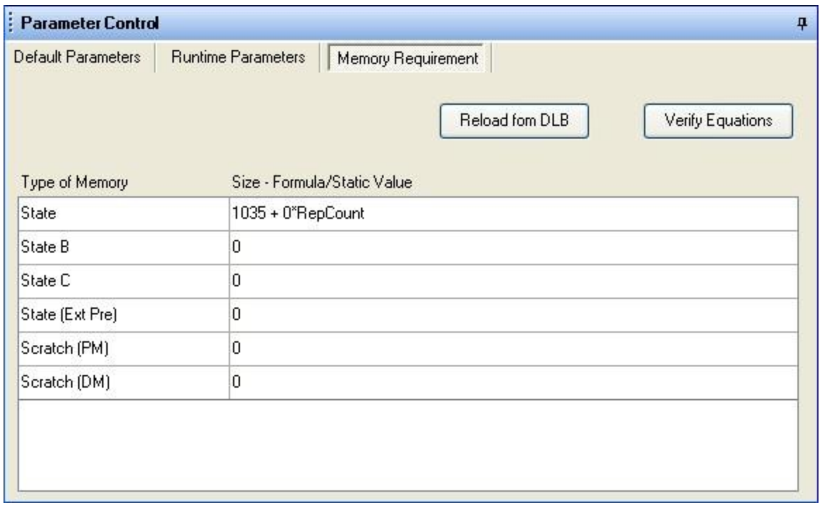
7.3.2.5 New Group

The 'New Group' checkbox is used to define a parameter group. By mapping the parameters to a group, the Algorithm Designer ensures that the first condition mentioned in section 7.3.2.4 is met, i.e., all parameters within a group are updated at once in the parameter memory. A group is defined based on the following rules:

- The parameter with the 'New Group' enabled is the first parameter in the group.
- A group can have maximum of 5 parameters.
- A group cannot have parameter table as its member.
- Only parameters with 'Safeload' enabled can be group members.
- A group can have only floating point or only fixed-point (any fixed-point format) parameters.
- Group members should be in adjacent rows in the parameter grid.
- Group ends upon reaching 5 members or upon encountering a parameter which does not satisfy any of the above conditions.
- The 'Safeload' and 'New Group' fields of all members in a group except the first member become inactive. This could be used to identify a group.

7.4 Memory Requirement

State and Scratch Memory required by the Module should be assigned on this window.



Type of Memory	Size - Formula/Static Value
State	1035 + 0*RepCount
State B	0
State C	0
State (Ext Pre)	0
Scratch (PM)	0
Scratch (DM)	0

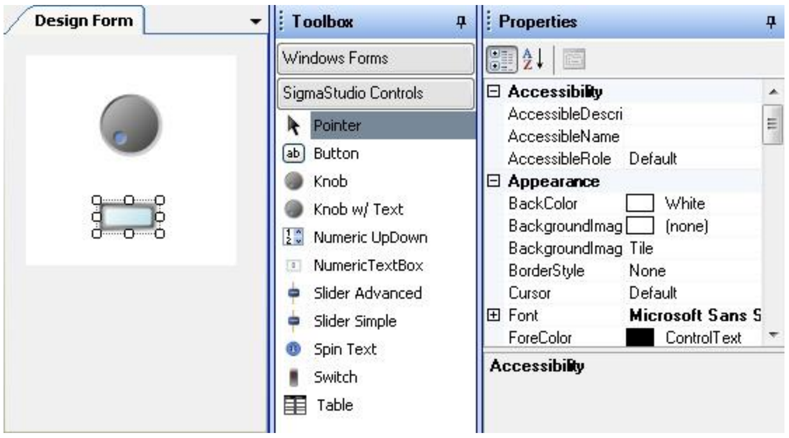
Figure 14: Memory Assignment Window

The user can assign a static value to the memory size or assign a formula which is a function of one or more runtime variables and size of runtime buffers. '`FS`', '`RepCount`' and '`BlockSize`' are predefined keywords and can be used in the formula to represent the 'Schematic Sampling Rate', 'Growth Count' and 'Schematic Block Size' (for block processing Modules) respectively. In order to use the size of a runtime buffer as a variable use the buffer name with a prefix '`_LEN_`'. For e.g. size of a runtime buffer '`LUT`' is denoted by '`_LEN_LUT`'. Ternary operators can also be used to define the size. For example; "`1024 + (RepCount>256?20:10)`"

Note: The value of the memory size is in words (4 Bytes).

7.5 GUI Cell Design

The GUI design involves three windows: the Toolbox window, the 'Design Form' pane, and the Property window as shown in Figure 15.



AccessibleDescri	
AccessibleName	
AccessibleRole	Default
BackColor	White
BackgroundImag	(none)
BackgroundImag	Tile
BorderStyle	None
Cursor	Default
Font	Microsoft Sans S
ForeColor	ControlText

Figure 15: GUI Design

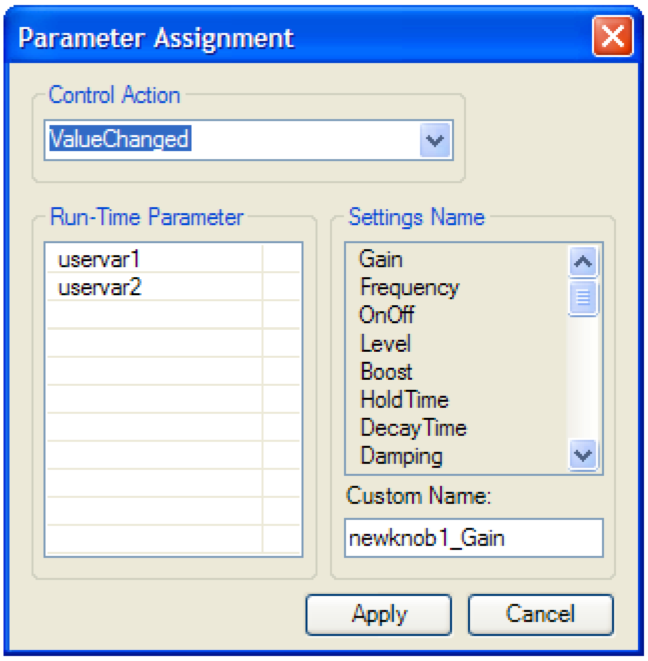
The Toolbox window displays a list of components available for use in the user interface design. There are two toolbox categories, one containing standard 'Windows Controls' and the other containing custom 'SigmaStudio Controls'.

Controls may be dragged from the Toolbox onto the Form surface. The Form pane in the middle is the background for the custom interface on which all the controls are arranged. To select a control on the form, click on it with the left mouse button. The selected control's properties are listed in the Property window. The standard Undo, Redo, Cut, Copy, and Paste actions are supported in the Form designer. Using the Format menu commands, the user can adjust the layout and alignment of the selected controls on the form. The Property window can be used to view and change the design-time properties of selected objects that are located in design.

After adding a control to the form design, the control has to be associated with an Algorithm parameter, using 'Runtime Parameters'.

7.5.1 Control-Parameter Assignment

The controls in the GUI are an extension to the runtime modifiable variables/buffers. Each custom GUI control is manually assigned to one runtime variable/buffer. After adding a control to the form design, the control has to be associated with an Algorithm parameter, using one of the 'Runtime Parameters'. To associate a control with a particular parameter right-click the control in the form design. This opens the Parameter Assignment dialog as shown in Figure 16.



Control Action: ValueChanged

Run-Time Parameter: uservar1, uservar2

Settings Name: Gain, Frequency, OnOff, Level, Boost, HoldTime, DecayTime, Damping

Custom Name: newknob1_Gain

Figure 16: Parameter Assignment Window

1. Select a “Control Action” in the Control Action list. Typically, the “ValueChanged” action has to be assigned to one of the “Run-Time Parameter” items.
2. Select the Run-time Parameters to which the control’s value has to be assigned.
3. Define a “Settings Name” for the control action or use the default settings name. See below for more on Settings Names.
4. Press the “Apply” button.

It is possible to release (remove the Run-time parameter assignment from the control) a Run-time parameter from a control. Perform the following steps to release an already assigned Run-time parameter.

1. Select the “Control Action” from the drop down list. If a parameter is already assigned to the action, it is highlighted in the “Run-time Parameter” list.
2. Click on the small column next to the highlighted parameter. This releases the parameter assignment. Alternatively, the user can select another parameter to assign to the control action.
3. Press the “Apply” button.

Note that based on the type of the control, either ‘Runtime Variables’ or ‘Runtime Buffers’ only (not both) gets listed in the ‘Run-Time Parameter’ list-box.

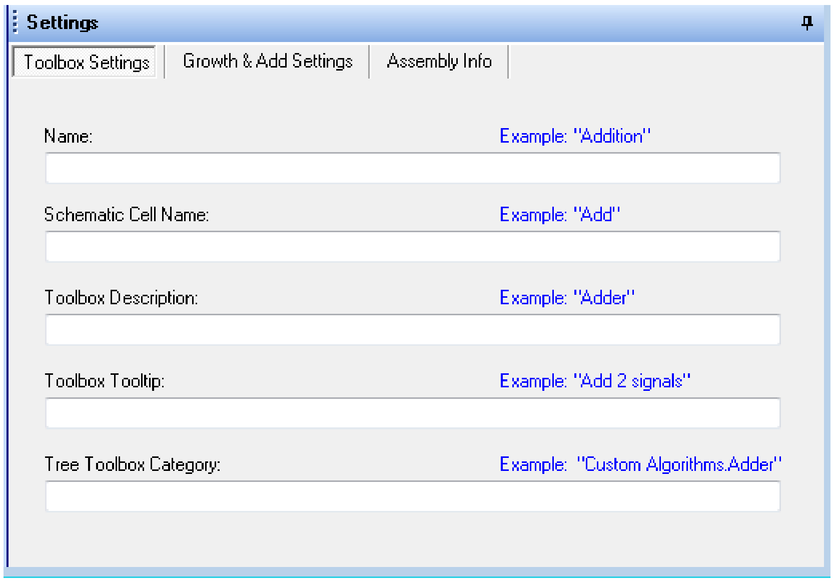
7.5.1.1 Settings Name

The Settings Name for a particular control attribute is used during save, open, and paste of SigmaStudio project Settings. It is possible to create Cell Settings, Board Settings, and Project Settings in SigmaStudio. These settings files can be copied between Cells of different types if (and only if) they have matching Settings Names. The standard settings names are defined in the Parameter Assignment dialog. Use one of these standard names, to have the setting interoperability between the custom Cell and the built in SigmaStudio blocks. Otherwise ignore the Settings Name and stick with the default. This field should not be left blank.

7.6 Define Module Setting

Use the ‘Settings’ window to define the custom Cell and Algorithm names, toolbox descriptions and Growth settings of the Plug-In. These settings are used during assembly generation and to identify the Cell when added to SigmaStudio.

7.6.1 Toolbox Settings


Settings

Toolbox Settings | Growth & Add Settings | Assembly Info

Name: Example: "Addition"

Schematic Cell Name: Example: "Add"

Toolbox Description: Example: "Adder"

Toolbox Tooltip: Example: "Add 2 signals"

Tree Toolbox Category: Example: "Custom Algorithms.Adder"

Figure 17: Toolbox Settings Window

7.6.1.1 Name

The 'Name' uniquely identifies the Cell and Algorithm in the SigmaStudio Add-Ins collection and SigmaStudio project file. This name is used to define a .NET object type.

Note: the name must be selected carefully. This name must be unique among all custom SigmaStudio blocks. If duplicate names exist, the custom Module cannot be added to SigmaStudio. In short, select unique names.

7.6.1.2 Schematic Cell Name

The 'Schematic Cell Name' is the default name that is used when a Cell is inserted into a Schematic design. It should be a short descriptive identifier; it does not need to be unique.

7.6.1.3 Toolbox Description

The 'Toolbox Description' is the name used in the SigmaStudio toolbox for this Cell.

7.6.1.4 Toolbox Tooltip

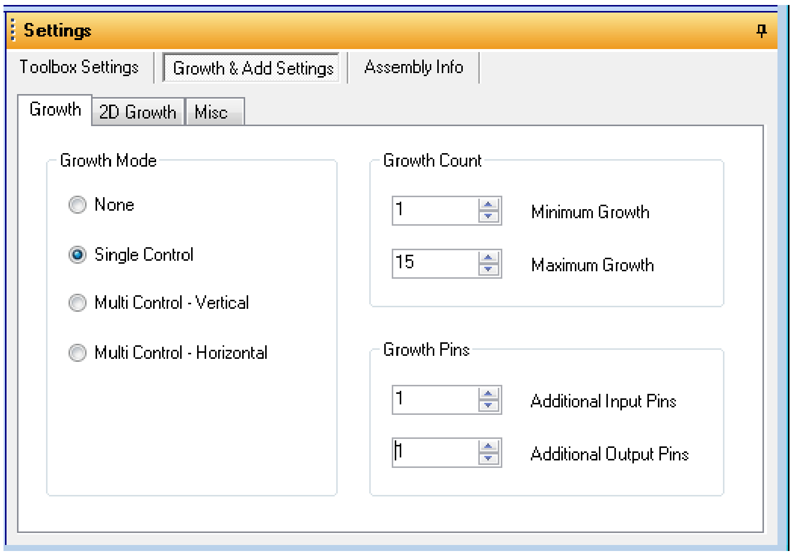
The 'Toolbox Tooltip' is the optional tooltip that is displayed when mouse is placed over a Module entry in the SigmaStudio toolbox.

7.6.1.5 Tree Toolbox Category

The 'Tree Toolbox Category' defines the hierarchy levels or the branches when the Module is listed in Tree Toolbox. If there are multiple levels, use dot to separate levels; for example "Filters.Second Order.2 Channel.Custom Implementation". The representation is from top to bottom.

7.6.2 Growth and Add Settings

7.6.2.1 Growth Settings



Growth Mode
( ) None
(●) Single Control
( ) Multi Control - Vertical
( ) Multi Control - Horizontal

Growth Count
1 Minimum Growth
15 Maximum Growth

Growth Pins
1 Additional Input Pins
1 Additional Output Pins

Figure 18: Growth Settings Window

7.6.2.1.1 Growth Mode

Use the 'Growth Mode' control to specify the "Growth" behavior of the Cell in SigmaStudio.

- **None** – Growth is not supported by the Module.
- **Single Control** – Control doesn't change with growth.
- **Multi Control – Vertical** – An additional GUI control is created for each growth, stacked vertically within the Cell.
- **Multi Control – Horizontal** – An additional GUI control is created for each growth, arranged horizontally within the Cell.

7.6.2.1.2 Growth Count

This defines the minimum and maximum growth count for the Module.

7.6.2.1.3 Growth Pins

This setting defines the additional number of pins to be inserted on the input and output for every growth. Additional pin count for both input and output must be less than the number of pins on the base Algorithm (with growth as 1).

7.6.2.2 Two Dimensional Growth Settings

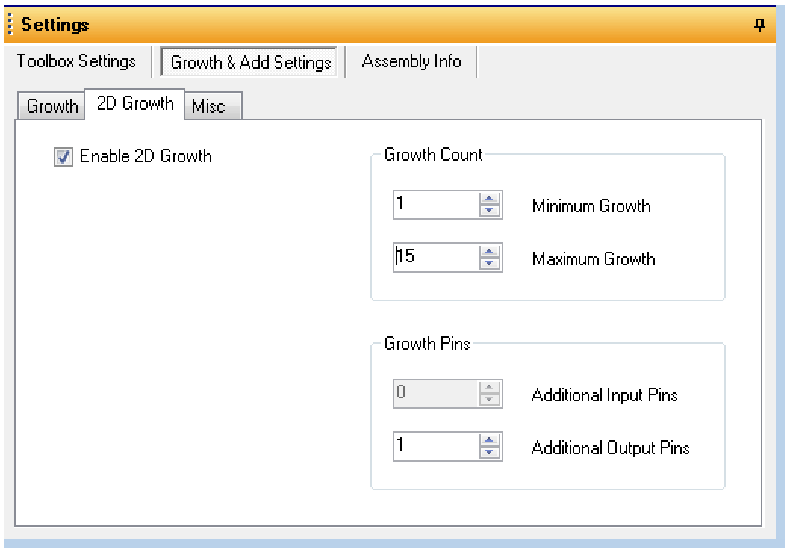

Settings
Toolbox Settings | Growth & Add Settings | Assembly Info
Growth | 2D Growth | Misc
☑ Enable 2D Growth
Growth Count
1 Minimum Growth
15 Maximum Growth
Growth Pins
0 Additional Input Pins
1 Additional Output Pins

Figure 19: 2D Growth Settings Window

7.6.2.2.1 Enable 2D Growth

Enable or disable two dimensional growths. In 2D growth, the input and output growth can be independently controlled.

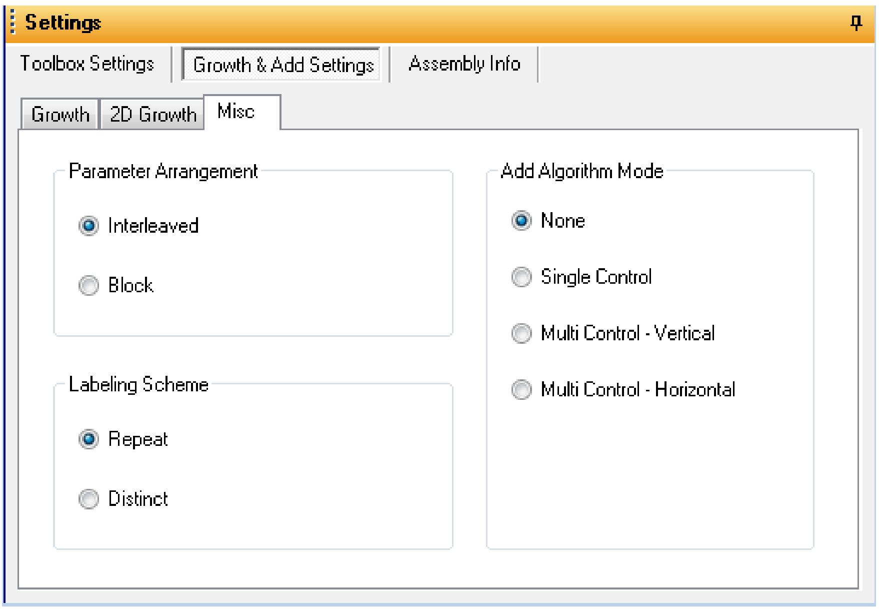
7.6.2.2.2 Growth Count

This defines the minimum and maximum count for the second growth dimension of the Module.

7.6.2.2.3 Growth Pins

This setting defines the additional number of pins to be inserted on the input and output for every growth on the second dimension. Additional pin count for both input and output must be less than the number of pins on the base Algorithm (with growth as 1).

7.6.2.3 Miscellaneous Settings



Parameter Arrangement: [✓ Interleaved] Block

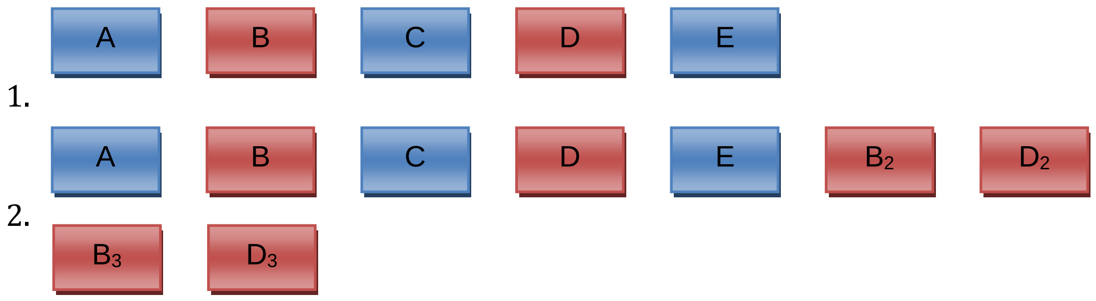
Labeling Scheme: [✓ Repeat] Distinct

Add Algorithm Mode: [✓ None] Single Control, Multi Control - Vertical, Multi Control - Horizontal

Figure 20: Miscellaneous Settings Window

7.6.2.3.1 Parameter Arrangement

Parameters are arranged in the memory in the order in which they appear in the parameter control window. Parameters are classified as stacking parameters and non-stacking parameters. Stacking parameters are those that are linked to control that replicates when grown. Non-stacking parameters are parameters, that are not linked to any controls (assigned with constants) or parameters linked to controls that do not replicate when grown. When the Module is grown, additional parameters are generated for stacking parameters. The arrangement of these additional parameters are shown in the below illustration.



3.

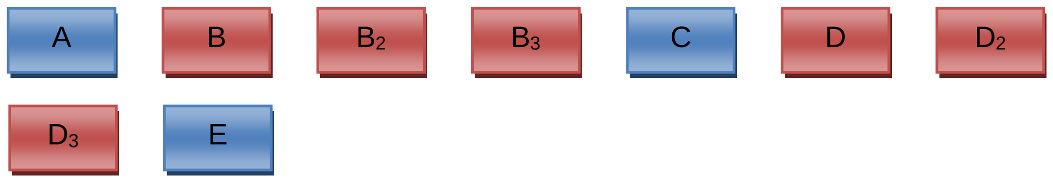
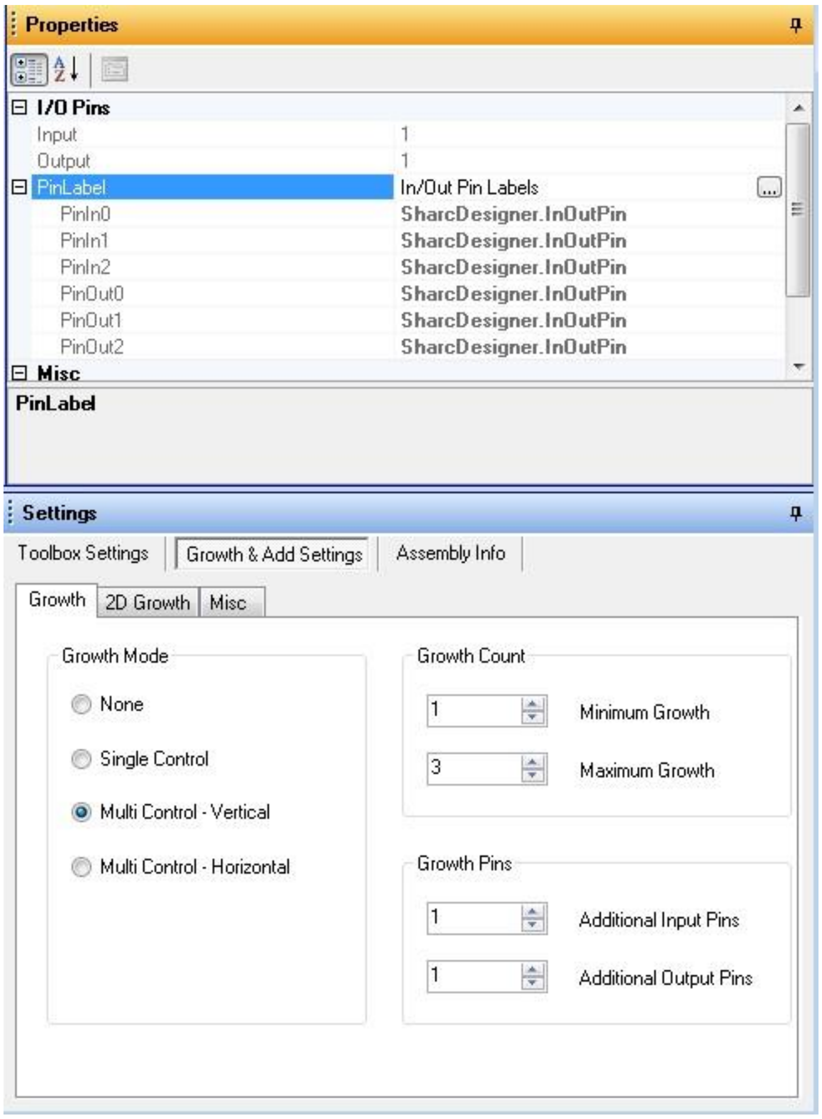


Figure 21: Parameter Arrangement in Memory

Illustration 1 shows the parameters of an example Module without growth. Red color boxes represent stacking parameters and blue color boxes represent non-stacking parameters. Illustrations 2 and 3 shows 2 different arrangements of parameters when the Modules are grown additionally by 2 (total growth to 3). Illustration 2 is 'Interleaved' arrangement where the blocking parameters are interleaved and arranged per additional growth. Illustration 3 is 'Block' arrangement where all the growth copies of each stacking parameters are arranged as a block.

7.6.2.3.2 Labeling Scheme

Each of the input/output pins on the Plug-In can be given a user-defined name. When the SigmaStudio Module is grown, pins are duplicated and inserted on the Module. The labels of the additional pins inserted by the growth can be controlled using the 'Labeling Scheme' setting. 'Repeat' stands for repeating the labels of the base pin labels upon growth. 'Distinct' stands for distinct label for each additional pin inserted upon growth. Labels are defined on the grid within 'Properties' window in Algorithm Designer. Refer to section 7.6.4 for more details. When 'Distinct' is selected, there are distinct entries in the grid for each of the pins for the maximum growth case, in order, with input pins (till the maximum growth) followed by the output pins (till the maximum growth).


Properties

I/O Pins

Input	1
Output	1
PinLabel	In/Out Pin Labels
PinIn0	SharcDesigner.InOutPin
PinIn1	SharcDesigner.InOutPin
PinIn2	SharcDesigner.InOutPin
PinOut0	SharcDesigner.InOutPin
PinOut1	SharcDesigner.InOutPin
PinOut2	SharcDesigner.InOutPin

Misc

PinLabel

Settings

Toolbox Settings | Growth & Add Settings | Assembly Info

Growth | 2D Growth | Misc

Growth Mode: ( ) None ( ) Single Control [✓ Multi Control - Vertical] ( ) Multi Control - Horizontal

Growth Count: Minimum Growth 1, Maximum Growth 3

Growth Pins: Additional Input Pins 1, Additional Output Pins 1

Figure 22: Distinct Pin Labeling Scheme

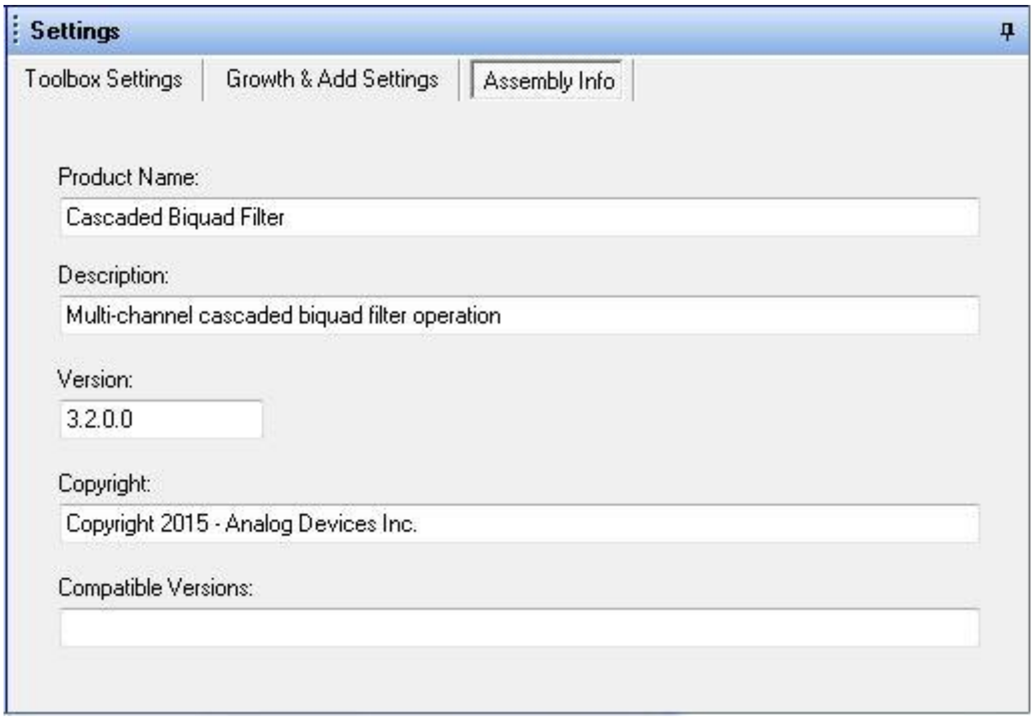
7.6.2.3.3 Add Algorithm Mode

Use the ‘Add Algorithm Mode’ control to specify the “Add Algorithm” behavior of the Cell in SigmaStudio.

- **None** – Disables the Add Algorithm function, the user cannot add more than one Algorithm instance per Cell.
- **Single Control** – A single GUI control is shared by all Algorithm instances.

- **Multi Control – Vertical** – An additional GUI control is created for each Algorithm instance, stacked vertically within the Cell.
- **Multi Control – Horizontal** – An additional GUI control is created for each Algorithm instance, arranged horizontally within the Cell.

7.6.3 Assembly Info Settings



Product Name: Cascaded Biquad Filter

Description: Multi-channel cascaded biquad filter operation

Version: 3.2.0.0

Copyright: Copyright 2015 - Analog Devices Inc.

Compatible Versions:

Figure 23: Assembly Info Settings Window

7.6.3.1 Product Name

The name of the Module, for which assembly info has to be generated, has to be given as the Product Name. This is of string type, without any special characters.

7.6.3.2 Description

The description of the Module which has to be captured in the assembly info of the generated DLL has to be given here.

7.6.3.3 Version

This entry is the version number of the Plug-In DLL to be generated by the Algorithm Designer. The format to be followed is either a.b or a.b.c or a.b.c.d.

Note that trailing 0's are ignored, therefore versions a.b, a.b.0 and a.b.0.0 are equivalent.

Invalid version numbers if entered are reset to 1.0.0.0 by the Algorithm Designer after informing the user. The version entered also goes through an auto-correction mechanism. For example,

11..2..a.3 entered is reset to 1.0.0.0

0001.0002.0003 entered is corrected as 1.2.3

2. 4.   5 entered is corrected as 2.4.5

Note: Version numbers '0.0', '0.0.0' and '0.0.0.0' are reserved and should not be used. The tool doesn't restrict the usage of these version numbers and this should be handled by the user.

7.6.3.4 Copyright

The copyright information of the Module has to be given in this field.

7.6.3.5 Compatible Versions

SigmaStudio Schematics (.dspproj files) should be opened using the same or a compatible version of the Plug-Ins with which the Schematic was created. If the Schematic is opened using a version of the Plug-In that is not compatible, the Schematic may malfunction and become corrupted. When a Schematic is opened, SigmaStudio checks the compatibility of the Plug-Ins used in the Schematic and informs the users regarding any incompatibility.

The 'Compatible Versions' field of the Algorithm Designer Assembly Info window is used to enter the list of existing versions of the Plug-In DLL to which the new Plug-In which is about to be generated is compatible. SigmaStudio uses this information to perform the version compatibility check. Refer to Annexure F of the Quick Start Guide for more details on Plug-In version compatibility checking.

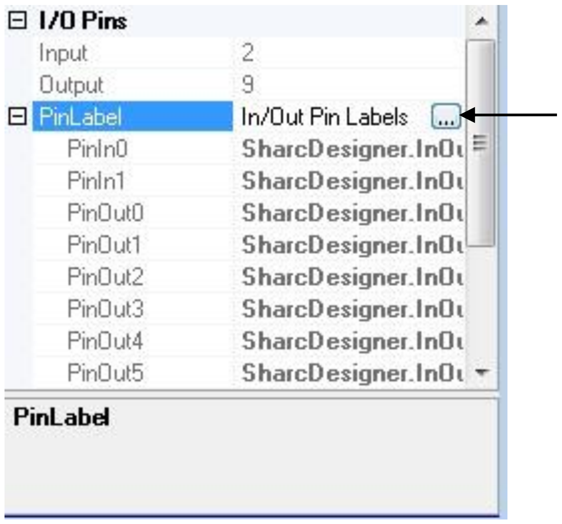
Multiple version numbers can be entered in this field and each version number should be separated by a semi-colon ';'. Only the first three digits (*major.minor.build*) in the version number are used for compatibility checking. Therefore it is not mandatory to enter the fourth digit (*revision*) of the version number. The fourth digit, if entered, is ignored.

For example 1.2.3.4 and 1.2.3 are treated as compatible versions. Similarly versions 1.2.3.4 and 1.2.3.5 are treated as compatible versions. However, 1.0.0 and 1.0.1 would be treated as incompatible versions. Invalid entries, such as unsupported characters, invalid version formats, or version numbers followed by an unsupported separator, in the list are removed by the Algorithm Designer. One example for a valid entry is given below.

2.0.0.0; 2.1.0;2.2.0;2.5.0;3.0.0

7.6.4 Input/Output Pin Labels

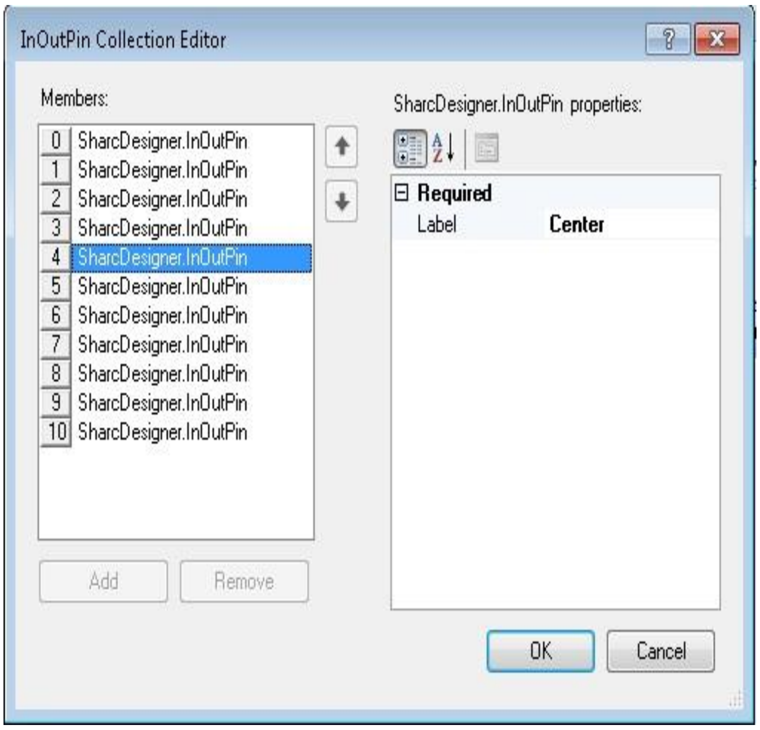
Upon selecting a Module in the “External Module” window, the ‘Properties’ window lists the properties of the selected Module. When the Ellipses button with 3 dots, corresponding to the ‘PinLabel’ as shown in Figure 24 is clicked, the ‘InOutPin Collection Editor’ window opens. The ‘Label’ for each pin on the Cell can be entered and edited on the ‘InOutPin Collection Editor’ window. See Figure 25 below for more details.



I/O Pins	
Input	2
Output	9
PinLabel	In/Out Pin Labels
PinIn0	SharcDesigner.InOu
PinIn1	SharcDesigner.InOu
PinOut0	SharcDesigner.InOu
PinOut1	SharcDesigner.InOu
PinOut2	SharcDesigner.InOu
PinOut3	SharcDesigner.InOu
PinOut4	SharcDesigner.InOu
PinOut5	SharcDesigner.InOu

PinLabel

Figure 24: Pin Label Properties Window


InOutPin Collection Editor

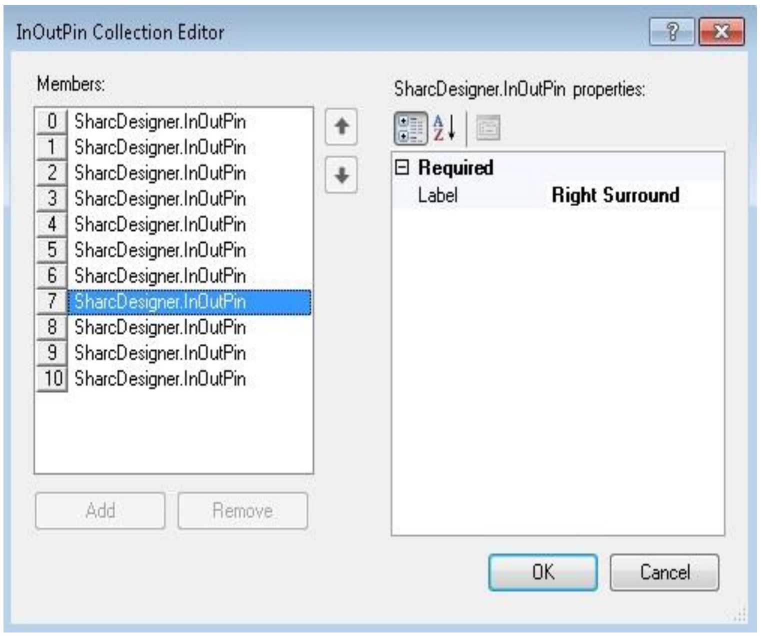
Members:

0 SharcDesigner.InOutPin
1 SharcDesigner.InOutPin
2 SharcDesigner.InOutPin
3 SharcDesigner.InOutPin
4 SharcDesigner.InOutPin
5 SharcDesigner.InOutPin
6 SharcDesigner.InOutPin
7 SharcDesigner.InOutPin
8 SharcDesigner.InOutPin
9 SharcDesigner.InOutPin
10 SharcDesigner.InOutPin

SharcDesigner.InOutPin properties:

⊟ Required

Label	Center


InOutPin Collection Editor

Members:

0 SharcDesigner.InOutPin
1 SharcDesigner.InOutPin
2 SharcDesigner.InOutPin
3 SharcDesigner.InOutPin
4 SharcDesigner.InOutPin
5 SharcDesigner.InOutPin
6 SharcDesigner.InOutPin
7 SharcDesigner.InOutPin
8 SharcDesigner.InOutPin
9 SharcDesigner.InOutPin
10 SharcDesigner.InOutPin

SharcDesigner.InOutPin properties:

⊟ Required

Label	Right Surround

Figure 25: Assigning Label for Input-Output Pins

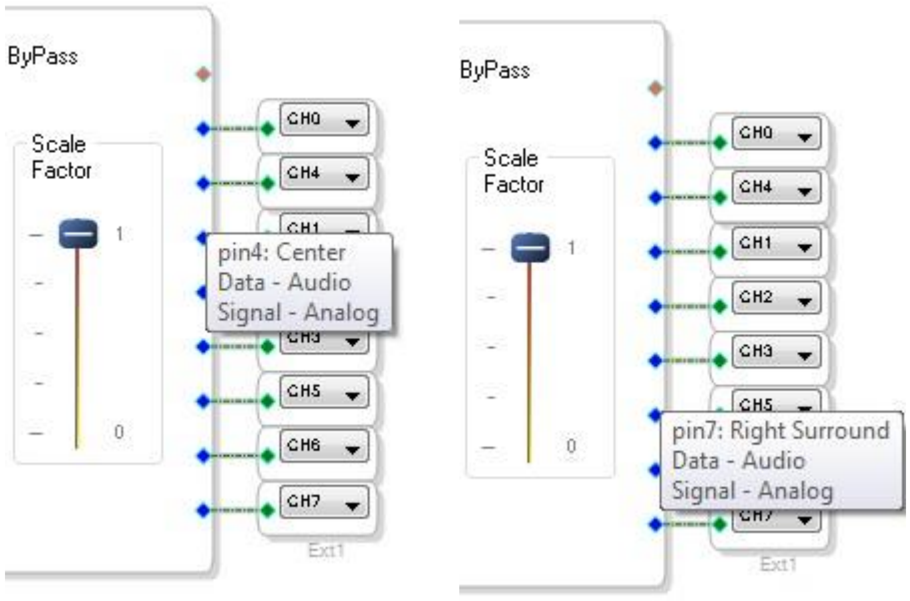


Figure 26: Labels Displayed on the Cell

7.7 Save/Open Design

The Algorithm Designer GUI design, settings and parameters can be saved to a file for reuse. Choose **File ➔ Save** menu item. This opens the 'Save' dialog and the user can save the design as a "SigmaStudio GUI Design (*.ssg)" file.

After saving a SigmaStudio Design file, it is not advised to further modify the type, format, size and number of parameters inside the XML. The user is free to modify the code and rebuild the library.

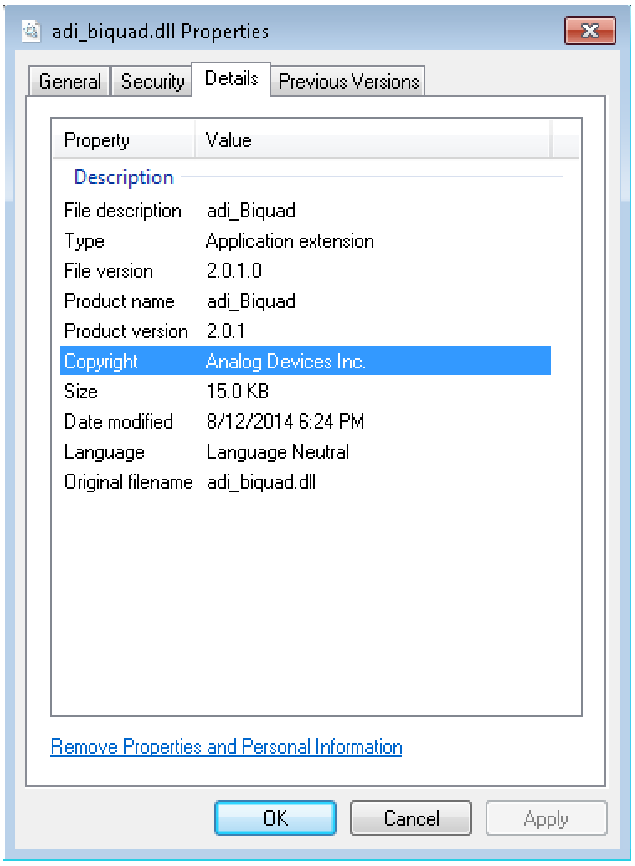
7.7.1 Open Design

The saved Algorithm Designer file (*.ssg) can be opened by clicking on the Open button or by selecting **File ➔ Open** menu item. If the XML file used in the design is not found or is modified, the Algorithm Designer prompts the user. The user can either continue or cancel the operation. Upon successful load of all the settings, the "External Module" window is disabled to prevent the user from modifying the selection of Algorithm. The user is free to modify all other settings. Use the 'Release Lock' button to enable the selection of Algorithms from the list.

8 Plug-In Version Compatibility Check

SigmaStudio for SHARC Schematics should be opened with the same version or a compatible version of any Plug-Ins with which the Schematic was created. If the Schematic is opened using a version of the Plug-In that is not compatible, the Schematic may become corrupted and possibly malfunction.

SigmaStudio performs the Plug-In version compatibility check when a Schematic containing Plug-Ins is opened. The Plug-In version compatibility check is performed to ensure that the enabled Plug-Ins are compatible with the Schematic. The user can view the list of enabled Plug-Ins from the SigmaStudio Add-Ins window. The version number of the Plug-In is used to perform the version compatibility check. The Plug-In version is listed in the “Details” tab which can be found by right- clicking the Plug-In DLL file with Windows Explorer and selecting “Properties” as shown in Figure 27.

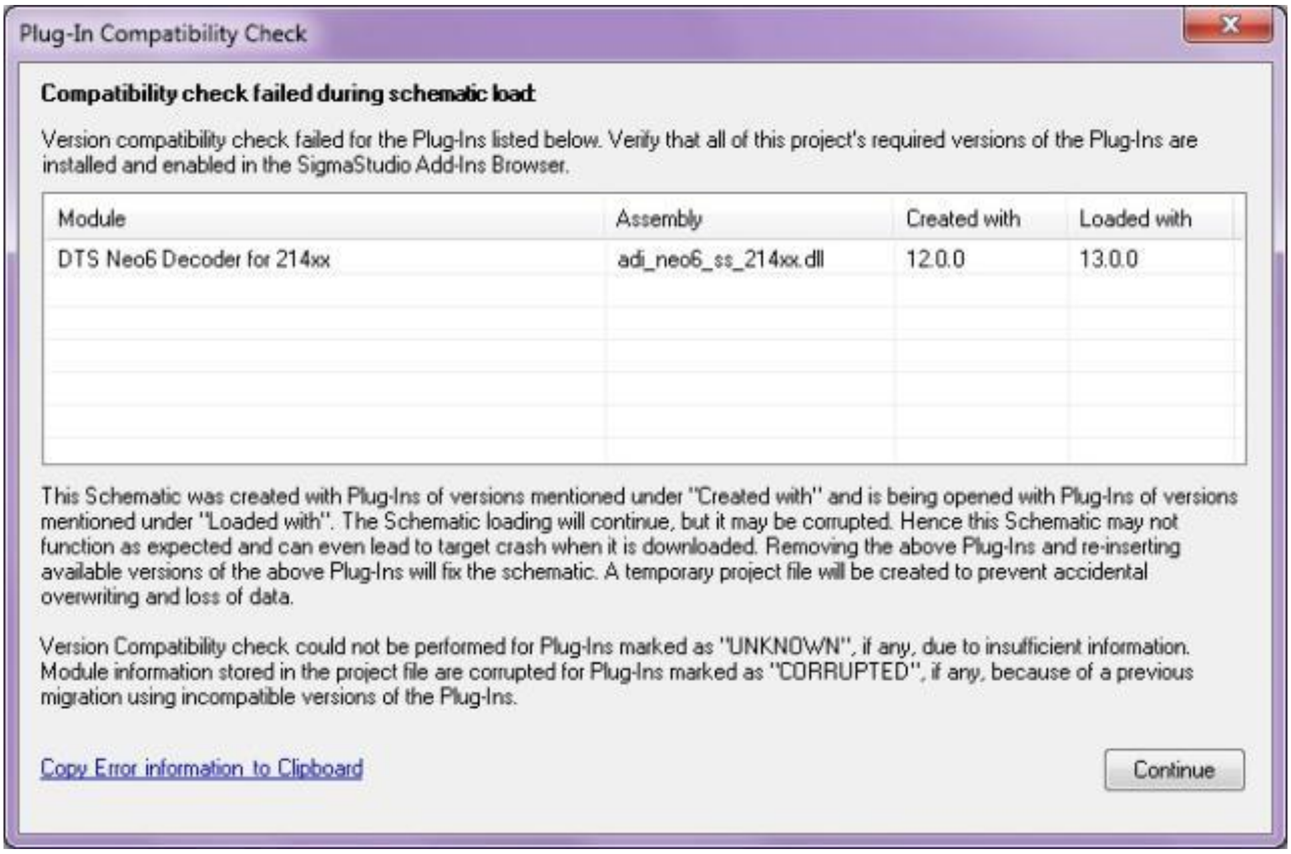

adi_biquad.dll Properties
General
Security
Details
Previous Versions

Property	Value
Description	
File description	adi_Biquad
Type	Application extension
File version	2.0.1.0
Product name	adi_Biquad
Product version	2.0.1
Copyright	Analog Devices Inc.
Size	15.0 KB
Date modified	8/12/2014 6:24 PM
Language	Language Neutral
Original filename	adi_biquad.dll

Remove Properties and Personal Information
OK
Cancel
Apply

Figure 27: Details of a DLL

If the Schematic does not pass the Plug-In version compatibility check, a message window as shown in Figure 28 is displayed. This message includes the details of the Plug-Ins which did not pass the compatibility check. Plug-In version compatibility check is supported only for SigmaStudio for SHARC Plug-Ins.



Compatibility check failed during schematic load

Version compatibility check failed for the Plug-Ins listed below. Verify that all of this project's required versions of the Plug-Ins are installed and enabled in the SigmaStudio Add-Ins Browser.

Module	Assembly	Created with	Loaded with
DTS Neo6 Decoder for 214xx	adi_neo6_ss_214xx.dll	12.0.0	13.0.0

This Schematic was created with Plug-Ins of versions mentioned under "Created with" and is being opened with Plug-Ins of versions mentioned under "Loaded with". The Schematic loading will continue, but it may be corrupted. Hence this Schematic may not function as expected and can even lead to target crash when it is downloaded. Removing the above Plug-Ins and re-inserting available versions of the above Plug-Ins will fix the schematic. A temporary project file will be created to prevent accidental overwriting and loss of data.

Version Compatibility check could not be performed for Plug-Ins marked as "UNKNOWN", if any, due to insufficient information. Module information stored in the project file are corrupted for Plug-Ins marked as "CORRUPTED", if any, because of a previous migration using incompatible versions of the Plug-Ins.

Figure 28: Plug-In Version Compatibility Check Message

Note: The versions of the DTS Neo6 Decoder Plug-In in the above figure are only for illustrative purpose and are not valid.

The 'Module' column of the tables in the above image lists the name of the Module as displayed in the SigmaStudio Toolbox. 'Assembly' column gives the name of the Plug-In DLL from which the Module is loaded. 'Created with' column shows the version of the Plug-In with which the Schematic was created and 'Loaded with' column shows the version of the Plug-In with which the Schematic is opened. Only the first 3 digits (ie. a.b.c) of the version number are used for version compatibility check and hence only 3 digits are displayed in the table.

8.1 Plug-In Compatibility

When a Schematic is created, the Plug-Ins are loaded from the respective Plug-In DLLs whereas user configuration, Plug-In parameters and connection details are saved in the project file (*.dspproj). When the Schematic is re-opened, the Plug-Ins are loaded from Plug-In

DLLs which are currently enabled in SigmaStudio whereas its associated connection details, Plug-In parameters and other configurations are read from the project file. Before opening the Schematic it should be ensured that the Plug-In DLLs enabled in the SigmaStudio Add-Ins window are compatible with the Plug-In DLLs used for creating the project file. If the Plug-In versions are not compatible, the configurations and parameters read from the project file will not be compatible with the Plug-In loaded from the Plug-In DLL. This will lead to Schematic corruption. A corrupted Schematic doesn't function as expected and may even cause the SHARC Target to crash.

Hence, it is recommended to open a Schematic with the same or compatible version of the Plug-Ins with which the Schematic was created. The version compatibility check feature in SigmaStudio performs the compatibility check of the Plug-Ins and reports any detected anomalies to the user.

8.2 Define Compatibility Details

The Compatibility details of a Plug-In are defined using the Algorithm Designer while generating the Plug-In. The 'Version' field in the Algorithm Designer is used to define the version of the Plug-In. The 'Compatible Versions' field in the Algorithm Designer is used to enter the list of other versions of the Plug-In DLL with which the new version which is about to be generated is compatible. SigmaStudio uses this information to perform version compatibility check. Refer to section 5.5.3 of **[2]** for more details.

8.3 Compatibility Check

SigmaStudio doesn't show any message when the compatibility check passes for all Plug-Ins in the Schematic. A window with failure details pops-up only when the compatibility check for one or more Plug-Ins has failed. Compatibility check failures are listed in the table in the pop-up window. In either of the cases, the expected and loaded versions of all Plug-Ins used in the Schematic are displayed in the output window after the Schematic is loaded. Also, a message *"* indicates that the version compatibility check has been skipped/failed for the Plug-In. In such cases, the Schematic may be corrupted and may not function as expected. This can even lead to target crash when it is downloaded. Removing failed/skipped Plug-Ins from the Schematic and re-inserting available versions of such Plug-Ins will fix the Schematic."* is displayed in the output window in either of the cases. However, this message can be ignored for those Plug-Ins whose loaded version is not preceded be a '*'.

SigmaStudio doesn't allow the user to save and update a Schematic for which the compatibility check has failed. SigmaStudio always prompts a "Save As" dialogue when the user attempts to save a corrupted Schematic. This is to prevent loss of data or accidental overwriting of Schematic.

A Schematic for which the compatibility check failed can still be compiled. The Schematic may be corrupted and can even cause the SHARC Target to crash. Refer to the following section for details on how to fix a corrupted Schematic.

8.4 Fixing a Corrupted Schematic

A Schematic for which the compatibility check has failed may be corrupted. A corrupted Schematic once saved under a different name remains corrupted even if re-opened later with a compatible version of the Plug-In. This is because the Schematic information in the project file became corrupted when it was saved. Such Plug-Ins are marked as “CORRUPTED” in the table in the pop-up window. In order to fix the Schematic, the user should delete the corrupted Plug-Ins' cells from the Schematic, drag/drop from the Toolbox to the Schematic, and reconnect the inputs and outputs.

8.5 Legacy Schematics and Plug-Ins

Sufficient compatibility details may not be defined in the Schematic if either the Schematic or the Plug-In was created using 2.1.0 or an earlier version of SigmaStudio for SHARC. In such cases, the compatibility check cannot be performed and are skipped. The Plug-Ins for which the compatibility check has been skipped are marked as “UNKNOWN” in the table in compatibility pop-up window. Such Plug-Ins may be incompatible and hence the Schematic may be corrupted. Hence SigmaStudio unconditionally treats such Schematics as corrupted Schematics. See section 8.4 for how to correct a corrupted Schematic.

8.6 Frozen Schematics

Frozen Schematics are already compiled and the program and parameter data are embedded in the Schematic. Hence it is mandatory for Frozen Schematics to use the same version of Plug-Ins for saving as well as opening the Schematic. The compatibility fails even when the versions are compatible, but different.

9 Steps to Rebuild the Example Plug-Ins

Follow the steps given below to rebuild the example Plug-Ins that are part of the package. Refer to Annexure E of [4] for more details on using these example Plug-Ins.

9.1 Rebuild DLBs

1. Launch CrossCore Embedded Studio. Using the option "Import -> Existing Projects into Workspace" open the CCES project in the folder "<*SigmaStudio for SHARC Installation folder*>\Host\ Sample Plug-Ins\ADSP-SC5xx\<Example Plug-In>\Src".
2. Make changes to the source code if any.
3. Rebuild the project.

9.2 Rebuild DLL

1. Launch Algorithm Designer and open the designer project file (*.ssg) from "<*SigmaStudio for SHARC Installation folder*>\Host\ Sample Plug-Ins\ADSP-SC5xx\<Example Plug-In>" folder.
2. Make changes to the GUI or settings if needed.
3. Press the "Generate Add-In Assembly" button in the toolbar or choose **Action ➔ Generate Assembly**.

A. Algorithm Designer Controls

Properties of the custom SigmaStudio controls are listed below. Refer to MSDN [3] for details on Windows Forms controls.

A.1 Button

Sl No	Property		Description
1.	Accessibility	AccessibleDescription	The description that will be reported to accessibility clients.
		AccessibleName	The name that will be reported to accessibility clients.
		AccessibleRole	The role that will be reported to accessibility clients.
2.	Appearance	BackColor	The background color of the component.
		BackgroundImage	The background image used for the component.
		BackgroundImageLayout	The background image layout used for the component.
		BorderStyle	Indicates whether the panel should have the border.
		Cursor	The cursor that appears when pointer moves over the control.
		Font	The font used to display text in control.
		Fore color	The foreground color of this component, which is used to display text.
		RightToLeft	Indicates whether the component should draw right to left for RTL language.
		UseWaitCursor	When this property is true, the cursor property of the control and its child are set to WaitCursor.
3.	Behavior	AllowDrop	Indicates whether the control can accept data that the user drags on to it.
		AutoValidate	Indicates whether the controls in the container will be automatically validated when the focus changes
		ContextMenuStrip	The Shortcut menu to display when the user right clicks on the control.
		Enabled	Indicates whether the control is enabled.
		ImeMode	Determines the IME(Input Method Editor) status of the object when selected.
		TabIndex	Determines the index in the TAB order that this control will occupy.
		TabStop	Indicates whether the user can use the TAB key to give focus to the control.
		Visible	Determines whether the control is visible or hidden.
4.	Data	DataBindings	The data bindings for the control.
		• (Advanced)	Advanced binding allows the user to bind properties of the

			control.
		• Tag	
		Tag	User-defined data associated with the object.
5.	Design	Name	Indicates the name used in the code to identify the object.
		GenerateMember	Indicates if a member variable will be generated for this component.
		Locked	The locked property determines if we can move or resize the control.
		Modifiers	Indicates the visibility level of the object.
6.	Focus	CausesValidation	Indicates whether this component raises validation events.
7.	Layout	Anchor	Defines the edges of the container to which a certain control is bound. When a control is anchored to an edge, the distance between the control's closest edge and the specified edge will remain constant.
		AutoScroll	Indicates whether scroll bars automatically appear when the control contents are larger than the visible area.
		AutoScrollMargin	The margin around controls during auto scroll.
		AutoScrollMinSize	The minimum logical size for the auto scroll region.
		AutoSize	Specifies whether a control will automatically size itself to fit its contents.
		AutoSizeMode	Specifies the mode by which the user interface element automatically resizes itself.
		Dock	Defines which borders of the control are bound to the container.
		Location	The co-ordinates of the upper-left corner of the control relative to the upper-left corner of the container.
		Margin	Specifies space between this control and another control's margin.
		MaximumSize	Specifies the maximum size of the control.
		MinimumSize	Specifies the minimum size of the control.
		Padding	Specifies the interior spacing of the control.
		Size	The size of the control in pixels.
8.	Misc	ArrowUpOrDown	When ShowText is false, up/down arrow is displayed if ShowArrow is true.
		Btext	Text displayed in the button, set 'Show Text' to true to display the text.
		ButtonColor	Button color and ButtonDark determine the color of the button.
		ButtonDark	Button color and ButtonDark determine the color of the button.
		ShowArrow	Shows Up/Down arrow when true.
		ShowText	Shows text entered in BText when true.

A.2 Knob

Sl No	Property		Description
1.	Accessibility	AccessibleDescription	The description that will be reported to accessibility clients.
		AccessibleName	The name that will be reported to accessibility clients.

		AccessibleRole	The role that will be reported to accessibility clients.
2.	Appearance	BackColor	The background color of the component.
		BackgroundImage	The background image used for the component.
		BackgroundImageLayout	The background image layout used for the component.
		BorderStyle	Indicates whether the panel should have the border.
		Cursor	The cursor that appears when pointer moves over the control.
		Font	The font used to display text in control.
		Fore color	The foreground color of this component, which is used to display text.
		RightToLeft	Indicates whether the component should draw right to left for RTL language.
		UseWaitCursor	When this property is true, the cursor property of the control and its child are set to WaitCursor.
3.	Behavior	AllowDrop	Indicates whether the control can accept data that the user drags on to it.
		AutoValidate	Indicates whether the controls in the container will be automatically validated when the focus changes
		ContextMenuStrip	The Shortcut menu to display when the user right clicks on the control.
		Enabled	Indicates whether the control is enabled.
		ImeMode	Determines the IME(Input Method Editor) status of the object when selected.
		TabIndex	Determines the index in the TAB order that this control will occupy.
		TabStop	Indicates whether the user can use the TAB key to give focus to the control.
		Visible	Determines whether the control is visible or hidden.
4.	Data	DataBindings	The data bindings for the control.
		• (Advanced)	Advanced binding allows the user to bind properties of the control.
		• Tag	
		Tag	User-defined data associated with the object.
5.	Design	Name	Indicates the name used in the code to identify the object.
		GenerateMember	Indicates if a member variable will be generated for this component.
		Locked	The locked property determines if we can move or resize the control.
		Modifiers	Indicates the visibility level of the object.
6.	Focus	CausesValidation	Indicates whether this component raises validation events.
7.	Layout	Anchor	Defines the edges of the container to which a certain control is bound. When a control is anchored to an edge, the distance between the control's closest edge and the specified edge will remain constant.
		AutoScroll	Indicates whether scroll bars automatically appear when the control contents are larger than the visible area.
		AutoScrollMargin	The margin around controls during auto scroll.

		AutoScrollMinSize	The minimum logical size for the auto scroll region.
		AutoSize	Specifies whether a control will automatically size itself to fit its contents.
		AutoSizeMode	Specifies the mode by which the user interface element automatically resizes itself.
		Dock	Defines which borders of the control are bound to the container.
		Location	The co-ordinates of the upper-left corner of the control relative to the upper-left corner of the container.
		Margin	Specifies space between this control and another control's margin.
		MaximumSize	Specifies the maximum size of the control.
		MinimumSize	Specifies the minimum size of the control.
		Size	The size of the control in pixels.
8.	Misc	ActiveColor	Color of the knob when selected and active.
		CanEdit	Allow the knob's Min, Max or Resolution values to be user adjusted.
		CanEditMax	Allow user adjustment of the Maximum knob value.
		CanEditMin	Allow user adjustment of the Minimum knob value.
		CanEditRes	Allow user adjustment of knob resolution.
		InactiveColor	Color of the knob when un-selected and in-active.
		MajorTickFrequency	Number of major tick marks to display. 'Major Ticks' must be set accordingly.
		MajorTicks	Display Major tick-marks. 'ShowLabels' must also be set accordingly.
		Maximum	Maximum value. Knob position fully clock-wise.
		Minimum	Minimum value. Knob position fully counter clock-wise.
		MinorTickDivisions	Number of minor tick divisions between each major tick mark. 'MinorTicks' must be set accordingly.
		MinorTicks	Display Minor tick-marks. 'ShowLabels' must also be set accordingly.
		Padding	Changes the size of the knob.
		Resolution	Resolution of the knob value range. The number of discrete values between min and max.
		ShowLabels	Show or hide the knob value labels and tick marks.
		ThumbRadiusFactor	Knob size.
		ThumbWidthFactor	Knob size.
		TickLengthRatio	Length of ticks.
		UseEditConversion	Enable to specify a conversion from the value range to display range [e.g knob's min/max is 0-10, but displayed value is -10dB to +10dB]. If enabled the user must specify ConversionMin, ConversionMax and ConversionStep.
		Value	Default value of the knob.

A.3 Knob With Text

Sl No	Property		Description
1.	Accessibility	AccessibleDescription	The description that will be reported to accessibility clients.
		AccessibleName	The name that will be reported to accessibility clients.

		AccessibleRole	The role that will be reported to accessibility clients.
2.	Appearance	BackColor	The background color of the component.
		BackgroundImage	The background image used for the component.
		BackgroundImageLayout	The background image layout used for the component.
		BorderStyle	Indicates whether the panel should have the border.
		Cursor	The cursor that appears when pointer moves over the control.
		Font	The font used to display text in control.
		Fore color	The foreground color of this component, which is used to display text.
		RightToLeft	Indicates whether the component should draw right to left for RTL language.
		UseWaitCursor	When this property is true, the cursor property of the control and its child are set to WaitCursor.
3.	Behavior	AllowDrop	Indicates whether the control can accept data that the user drags on to it.
		AutoValidate	Indicates whether the controls in the container will be automatically validated when the focus changes
		ContextMenuStrip	The Shortcut menu to display when the user right clicks on the control.
		Enabled	Indicates whether the control is enabled.
		ImeMode	Determines the IME(Input Method Editor) status of the object when selected.
		TabIndex	Determines the index in the TAB order that this control will occupy.
		TabStop	Indicates whether the user can use the TAB key to give focus to the control.
		Visible	Determines whether the control is visible or hidden.
4.	Data	DataBindings	The data bindings for the control.
		• (Advanced)	Advanced binding allows the user to bind properties of the control.
		• Tag	
		Tag	User-defined data associated with the object.
5.	Design	Name	Indicates the name used in the code to identify the object.
		GenerateMember	Indicates if a member variable will be generated for this component.
		Locked	The locked property determines if we can move or resize the control.
		Modifiers	Indicates the visibility level of the object.
6.	Focus	CausesValidation	Indicates whether this component raises validation events.
7.	Layout	Anchor	Defines the edges of the container to which a certain control is bound. When a control is anchored to an edge, the distance between the control's closest edge and the specified edge will remain constant.
		AutoScroll	Indicates whether scroll bars automatically appear when the control contents are larger than the visible area.
		AutoScrollMargin	The margin around controls during auto scroll.

		AutoScrollMinSize	The minimum logical size for the auto scroll region.
		AutoSize	Specifies whether a control will automatically size itself to fit its contents.
		AutoSizeMode	Specifies the mode by which the user interface element automatically resizes itself.
		Dock	Defines which borders of the control are bound to the container.
		Location	The co-ordinates of the upper-left corner of the control relative to the upper-left corner of the container.
		Margin	Specifies space between this control and another control's margin.
		MaximumSize	Specifies the maximum size of the control.
		MinimumSize	Specifies the minimum size of the control.
		Padding	Specifies the interior spacing of the control.
		Size	The size of the control in pixels.
8.	Misc	Maximum	Knob/Text Control Maximum Value.
		Minimum	Knob/Text Control Minimum Value.
		Resolution	Knob's Resolution, the number of unique positions between min/max.
		Suffix	Units suffix to display in text field [e.g dB].
		TColorA	Active color of knob indicator.
		TColorI	Inactive color of knob indicator.
		Value	Default Knob/Text Value.

A.4 NumericUpDown

Sl No	Property		Description
1.	Accessibility	AccessibleDescription	The description that will be reported to accessibility clients.
		AccessibleName	The name that will be reported to accessibility clients.
		AccessibleRole	The role that will be reported to accessibility clients.
2.	Appearance	BackColor	The background color of the component.
		BorderStyle	Indicates whether the panel should have the border.
		Cursor	The cursor that appears when pointer moves over the control.
		Font	The font used to display text in control.
		Fore color	The foreground color of this component, which is used to display text.
		Hexadecimal	Indicates whether the Numeric up-down should display its value in hexadecimal.
		RightToLeft	Indicates whether the component should draw right to left for RTL language.
		TextAlign	Indicates how the text should be aligned in the edit box.
		UpDownAlign	Indicates how the up-down control will position the up and down buttons relative to its edit box.
		UseWaitCursor	When this property is true, the cursor property of the control and its child are set to WaitCursor.
		Value	The Current value of the numeric up-down control.
3.	Behavior	AllowDrop	Indicates whether the control can accept data that the user drags on to it.
		ContextMenuStrip	The Shortcut menu to display when the user right clicks on the control.

		Enabled	Indicates whether the control is enabled.
		ImeMode	Determines the IME(Input Method Editor) status of the object when selected.
		InterceptArrowKeys	Indicates whether the up-down control will increment and decrement the value when the UP ARROW and DOWN ARROW keys are pressed.
		ReadOnly	Indicates whether the edit-box is read-only.
		TabIndex	Determines the index in the TAB order that this control will occupy.
		TabStop	Indicates whether the user can use the TAB key to give focus to the control.
		Visible	Determines whether the control is visible or hidden.
4.	Data	DataBindings	The data bindings for the control.
		• (Advanced)	Advanced binding allows the user to bind properties of the control.
		• Tag	
		• Value	
		DecimalPlaces	Indicates number of decimal places to display.
		Increment	Indicates the amount to increment or decrement on each button click.
		Maximum	Indicates the maximum value for the numeric up-down control.
		Minimum	Indicates the minimum value for the numeric up-down control.
		Tag	User-defined data associated with the object.
		ThousandSeparator	Indicates whether the thousands separator will be inserted between every three decimal digits.
5.	Design	Name	Indicates the name used in the code to identify the object.
		GenerateMember	Indicates if a member variable will be generated for this component.
		Locked	The locked property determines if we can move or resize the control.
		Modifiers	Indicates the visibility level of the object.
6.	Focus	CausesValidation	Indicates whether this component raises validation events.
7.	Layout	Anchor	Defines the edges of the container to which a certain control is bound. When a control is anchored to an edge, the distance between the control's closest edge and the specified edge will remain constant.
		AutoSize	Specifies whether a control will automatically size itself to fit its contents.
		Dock	Defines which borders of the control are bound to the container.
		Location	The co-ordinates of the upper-left corner of the control relative to the upper-left corner of the container.
		Margin	Specifies space between this control and another control's margin.
		MaximumSize	Specifies the maximum size of the control.
		MinimumSize	Specifies the minimum size of the control.
		Size	The size of the control in pixels.
8.	Misc	ValueDouble	Default value of the control[e.g 1.0]

A.5 NumericTextBox

Sl No	Property		Description
1.	Accessibility	AccessibleDescription	The description that will be reported to accessibility clients.

		AccessibleName	The name that will be reported to accessibility clients.
		AccessibleRole	The role that will be reported to accessibility clients.
2.	Appearance	BackColor	The background color of the component.
		BorderStyle	Indicates whether the panel should have the border.
		Cursor	The cursor that appears when pointer moves over the control.
		Font	The font used to display text in control.
		Fore color	The foreground color of this component, which is used to display text.
		Lines	The lines of text in a multi-line edit, as an array of String values.
		RightToLeft	Indicates whether the component should draw right to left for RTL language.
		ScrollBars	Indicates, for multiline edit controls, which scroll bars will be shown for this control.
		Text	The text associated with this control.
		TextAlign	Indicates how the text should be aligned for edit controls.
		UseWaitCursor	When this property is true, the cursor property of the control and its child are set to WaitCursor.
3.	Behavior	AcceptsReturn	Indicates if return characters are accepted as input for multiline edit controls.
		AcceptsTab	Indicates if tab characters are accepted as input for multiline edit controls.
		AllowDrop	Indicates whether the control can accept data that the user drags on to it.
		CharacterCasing	Indicates if all characters should be left alone or converted to uppercase or lowercase.
		ContextMenuStrip	The Shortcut menu to display when the user right clicks on the control.
		Enabled	Indicates whether the control is enabled.
		HideSelection	Indicates that the selection should be hidden when the edit control loses focus.
		ImeMode	Determines the IME(Input Method Editor) status of the object when selected.
		MaxLength	Specifies the maximum number of characters that can be entered in to the edit control.
		Multiline	Controls whether the text of the edit control can span more than one line.
		PasswordChar	Indicates the character to display for password input for single-line edit controls.
		ReadOnly	ReadOnly disabled user editing of numeric value.
		ShortcutsEnabled	Indicates whether shortcuts defined for the control are enabled.
		TabIndex	Determines the index in the TAB order that this control will occupy.
		TabStop	Indicates whether the user can use the TAB key to give focus to the control.
		UseSystemPasswordChar	Indicates if the text in the edit control should appear as the default password character.
		Visible	Determines whether the control is visible or hidden.
		WordWrap	Indicates if lines are automatically word-wrapped for multi-line edit controls.
4.	Data	DataBindings	The data bindings for the control.
		• (Advanced)	Advanced binding allows the user to bind properties of the control.
		• Tag	

		• Text	
		Tag	User-defined data associated with the object.
5.	Design	Name	Indicates the name used in the code to identify the object.
		GenerateMember	Indicates if a member variable will be generated for this component.
		Locked	The locked property determines if we can move or resize the control.
		Modifiers	Indicates the visibility level of the object.
6.	Focus	CausesValidation	Indicates whether this component raises validation events.
7.	Layout	Anchor	Defines the edges of the container to which a certain control is bound. When a control is anchored to an edge, the distance between the control's closest edge and the specified edge will remain constant.
		Dock	Defines which borders of the control are bound to the container.
		Location	The co-ordinates of the upper-left corner of the control relative to the upper-left corner of the container.
		Margin	Specifies space between this control and another control's margin.
		MaximumSize	Specifies the maximum size of the control.
		MinimumSize	Specifies the minimum size of the control.
		Size	The size of the control in pixels.
8.	Misc	AutoCompleteCustomSource	The StringCollection to use when the AutoCompleteSource property is set to CustomSource.
		AutoCompleteMode	Indicates the text completion behavior of the text box.
		AutoCompleteSource	The autocompletesource, which can be one of the values from AutoCompleteSource enumeration.
		Decimals	Number of decimal places to the right of the decimal point to display.
		Maximum	Text control maximum value.
		Minimum	Text control minimum value.
		Suffix	Units suffix to display in Text field [e.g dB]
		Value	Default text box value.

A.6 Hexadecimal TextBox

Sl No	Property		Description
1.	Accessibility	AccessibleDescription	The description that will be reported to accessibility clients.
		AccessibleName	The name that will be reported to accessibility clients.
		AccessibleRole	The role that will be reported to accessibility clients.
2.	Appearance	BackColor	The background color of the component.
		BorderStyle	Indicates whether the panel should have the border.
		Cursor	The cursor that appears when pointer moves over the control.
		Font	The font used to display text in control.
		Fore color	The foreground color of this component, which is used to display text.
		Lines	The lines of text in a multi-line edit, as an array of String values.
		RightToLeft	Indicates whether the component should draw right to left for RTL language.
		ScrollBars	Indicates, for multiline edit controls, which scroll bars will be shown for this control.

		Text	The text associated with this control.
		TextAlign	Indicates how the text should be aligned for edit controls.
		UseWaitCursor	When this property is true, the cursor property of the control and its child are set to WaitCursor.
3.	Behavior	AcceptsReturn	Indicates if return characters are accepted as input for multiline edit controls.
		AcceptsTab	Indicates if tab characters are accepted as input for multiline edit controls.
		AllowDrop	Indicates whether the control can accept data that the user drags on to it.
		CharacterCasing	Indicates if all characters should be left alone or converted to uppercase or lowercase.
		ContextMenuStrip	The Shortcut menu to display when the user right clicks on the control.
		Enabled	Indicates whether the control is enabled.
		HideSelection	Indicates that the selection should be hidden when the edit control loses focus.
		ImeMode	Determines the IME(Input Method Editor) status of the object when selected.
		MaxLength	Specifies the maximum number of characters that can be entered in to the edit control.
		Multiline	Controls whether the text of the edit control can span more than one line.
		PasswordChar	Indicates the character to display for password input for single-line edit controls.
		ReadOnly	ReadOnly disabled user editing of numeric value.
		ShortcutsEnabled	Indicates whether shortcuts defined for the control are enabled.
		TabIndex	Determines the index in the TAB order that this control will occupy.
		TabStop	Indicates whether the user can use the TAB key to give focus to the control.
		UseSystemPasswordChar	Indicates if the text in the edit control should appear as the default password character.
		Visible	Determines whether the control is visible or hidden.
		WordWrap	Indicates if lines are automatically word-wrapped for multi-line edit controls.
4.	Data	DataBindings	The data bindings for the control.
		• (Advanced)	Advanced binding allows the user to bind properties of the control.
		• Tag	
		• Text	
		Tag	User-defined data associated with the object.
5.	Design	Name	Indicates the name used in the code to identify the object.
		GenerateMember	Indicates if a member variable will be generated for this component.
		Locked	The locked property determines if we can move or resize the control.
		Modifiers	Indicates the visibility level of the object.
6.	Focus	CausesValidation	Indicates whether this component raises validation events.
7.	Layout	Anchor	Defines the edges of the container to which a certain control is bound. When a control is anchored to an edge, the distance between the control's closest edge and the specified edge will remain constant.
		Dock	Defines which borders of the control are bound to the container.

		Location	The co-ordinates of the upper-left corner of the control relative to the upper-left corner of the container.
		Margin	Specifies space between this control and another control's margin.
		MaximumSize	Specifies the maximum size of the control.
		MinimumSize	Specifies the minimum size of the control.
		Size	The size of the control in pixels.
8.	Misc	AutoCompleteCustomSource	The StringCollection to use when the AutoCompleteSource property is set to CustomSource.
		AutoCompleteMode	Indicates the text completion behavior of the text box.
		AutoCompleteSource	The autocompletesource, which can be one of the values from AutoCompleteSource enumeration.
		DecimalValue	Textbox value in integer
		Maximum	Text control maximum value.
		Minimum	Text control minimum value.
		Value	Default text box value in hexadecimal.

A.7 SliderSimple

Sl No	Property		Description
1.	Accessibility	AccessibleDescription	The description that will be reported to accessibility clients.
		AccessibleName	The name that will be reported to accessibility clients.
		AccessibleRole	The role that will be reported to accessibility clients.
2.	Appearance	BackColor	The background color of the component.
		BackgroundImage	The background image used for the component.
		BackgroundImageLayout	The background image layout used for the component.
		BorderStyle	Indicates whether the panel should have the border.
		Cursor	The cursor that appears when pointer moves over the control.
		Font	The font used to display text in control.
		Fore color	The foreground color of this component, which is used to display text.
		RightToLeft	Indicates whether the component should draw right to left for RTL language.
		UseWaitCursor	When this property is true, the cursor property of the control and its child are set to WaitCursor.
3.	Behavior	AllowDrop	Indicates whether the control can accept data that the user drags on to it.
		AutoValidate	Indicates whether the controls in the container will be automatically validated when the focus changes
		ContextMenuStrip	The Shortcut menu to display when the user right clicks on the control.
		Enabled	Indicates whether the control is enabled.
		ImeMode	Determines the IME(Input Method Editor) status of the object when selected.
		TabIndex	Determines the index in the TAB order that this control will occupy.
		TabStop	Indicates whether the user can use the TAB key to give focus to the control.
		Visible	Determines whether the control is visible or hidden.
4.	Data	DataBindings	The data bindings for the control.
		• (Advanced)	Advanced binding allows the user to bind properties of the control.

		• Tag	
		Tag	User-defined data associated with the object.
5.	Design	Name	Indicates the name used in the code to identify the object.
		GenerateMember	Indicates if a member variable will be generated for this component.
		Locked	The locked property determines if we can move or resize the control.
		Modifiers	Indicates the visibility level of the object.
6.	Focus	CausesValidation	Indicates whether this component raises validation events.
7.	Layout	Anchor	Defines the edges of the container to which a certain control is bound. When a control is anchored to an edge, the distance between the control's closest edge and the specified edge will remain constant.
		AutoScroll	Indicates whether scroll bars automatically appear when the control contents are larger than the visible area.
		AutoScrollMargin	The margin around controls during auto scroll.
		AutoScrollMinSize	The minimum logical size for the auto scroll region.
		AutoSize	Specifies whether a control will automatically size itself to fit its contents.
		AutoSizeMode	Specifies the mode by which the user interface element automatically resizes itself.
		Dock	Defines which borders of the control are bound to the container.
		Location	The co-ordinates of the upper-left corner of the control relative to the upper-left corner of the container.
		Margin	Specifies space between this control and another control's margin.
		MaximumSize	Specifies the maximum size of the control.
		MinimumSize	Specifies the minimum size of the control.
		Padding	Specifies the interior spacing of the control.
		Size	The size of the control in pixels.
8.	Misc	CanEdit	Allow the slider Min.Max. or Resolution values to be user adjusted.
		CanEditMax	Allow user adjustment of the Maximum vertical (Y) slider range.
		CanEditMin	Allow user adjustment of the Minimum vertical (Y) slider range.
		CanEditRes	Allow user adjustment of the vertical (Y) slider resolution.
		dB Selected	dB scale selected.
		LinearEnabled	Allow user to set linear scales.
		LinearSelected	Linear scale selected.
		MajorTickFrequency	Number of major tick marks to display. 'MajorTicks' must be set accordingly.
		MajorTicks	Display 'Major' tick marks. 'ShowLabels' must also be set accordingly.
		MaximumY	Maximum value of vertical(Y) slider, the value at top of slider position.
		MinimumY	Minimum value of vertical(Y) slider, the value at slider bottom position.
		MinorTickDivisions	Number of Minor Tick Divisions between each major tick mark. 'MinorTicks' must be set accordingly.
		MinorTicks	Display 'Minor' tick marks. 'ShowLabels' must also be set accordingly.
		Resolution	Resolution of the vertical(Y) slider range, the number of discrete values in the min to max range.
		ShowLabels	Show or Hide the slider labels and tick marks.

		TickStyle2	Display tick marks on either the left or right side of the slider.
		Value	Default value of vertical(Y) slider, between MinimumY and MaximumY.

A.8 SliderAdvanced

Sl No	Property		Description
1.	Accessibility	AccessibleDescription	The description that will be reported to accessibility clients.
		AccessibleName	The name that will be reported to accessibility clients.
		AccessibleRole	The role that will be reported to accessibility clients.
2.	Appearance	BackColor	The background color of the component.
		BackgroundImage	The background image used for the component.
		BackgroundImageLayout	The background image layout used for the component.
		BorderStyle	Indicates whether the panel should have the border.
		Cursor	The cursor that appears when pointer moves over the control.
		Font	The font used to display text in control.
		Fore color	The foreground color of this component, which is used to display text.
		RightToLeft	Indicates whether the component should draw right to left for RTL language.
		UseWaitCursor	When this property is true, the cursor property of the control and its child are set to WaitCursor.
3.	Behavior	AllowDrop	Indicates whether the control can accept data that the user drags on to it.
		AutoValidate	Indicates whether the controls in the container will be automatically validated when the focus changes
		ContextMenuStrip	The Shortcut menu to display when the user right clicks on the control.
		Enabled	Indicates whether the control is enabled.
		ImeMode	Determines the IME(Input Method Editor) status of the object when selected.
		TabIndex	Determines the index in the TAB order that this control will occupy.
		TabStop	Indicates whether the user can use the TAB key to give focus to the control.
		Visible	Determines whether the control is visible or hidden.
4.	Data	DataBindings	The data bindings for the control.
		• (Advanced)	Advanced binding allows the user to bind properties of the control.
		• Tag	
		Tag	User-defined data associated with the object.
5.	Design	Name	Indicates the name used in the code to identify the object.
		GenerateMember	Indicates if a member variable will be generated for this component.
		Locked	The locked property determines if we can move or resize the control.
		Modifiers	Indicates the visibility level of the object.
6.	Focus	CausesValidation	Indicates whether this component raises validation events.
7.	Layout	Anchor	Defines the edges of the container to which a certain control is bound. When a control is anchored to an edge, the distance between the control's closest edge and the

			specified edge will remain constant.
		AutoScroll	Indicates whether scroll bars automatically appear when the control contents are larger than the visible area.
		AutoScrollMargin	The margin around controls during auto scroll.
		AutoScrollMinSize	The minimum logical size for the auto scroll region.
		AutoSize	Specifies whether a control will automatically size itself to fit its contents.
		AutoSizeMode	Specifies the mode by which the user interface element automatically resizes itself.
		Dock	Defines which borders of the control are bound to the container.
		Location	The co-ordinates of the upper-left corner of the control relative to the upper-left corner of the container.
		Margin	Specifies space between this control and another control's margin.
		MaximumSize	Specifies the maximum size of the control.
		MinimumSize	Specifies the minimum size of the control.
		Padding	Specifies the interior spacing of the control.
		Size	The size of the control in pixels.
8.	Misc	CanEdit	Allow the slider Min.Max. or Resolution values to be user adjusted.
		CanEditMax	Allow user adjustment of the Maximum vertical (Y) slider range.
		CanEditMin	Allow user adjustment of the Minimum vertical (Y) slider range.
		CanEditRes	Allow user adjustment of the vertical (Y) slider resolution.
		dB Selected	dB scale selected.
		FreqLock	Lock the text control, Horizontal(X) value. When locked, the value cannot be edited in the text control.
		HorizontalIncrement	Increment amount when dragging the Horizontal(X) control.
		InitialXValue	Default value for the Horizontal(X) Control.
		LinearEnabled	Allow user to set linear scales.
		LinearSelected	Linear scale selected.
		MajorTickFrequency	Number of major tick marks to display. 'MajorTicks' must be set accordingly.
		MajorTicks	Display 'Major' tick marks. 'ShowLabels' must also be set accordingly.
		MaximumX	Maximum value for horizontal (X) control.
		MaximumY	Maximum value of vertical(Y) slider, the value at top of slider position.
		MinimumX	Minimum value for horizontal(X) control.
		MinimumY	Minimum value of vertical(Y) slider, the value at slider bottom position.
		MinorTickDivisions	Number of Minor Tick Divisions between each major tick mark. 'MinorTicks' must be set accordingly.
		MinorTicks	Display 'Minor' tick marks. 'ShowLabels' must also be set accordingly.
		Resolution	Resolution of the vertical(Y) slider range, the number of discrete values in the min to max range.
		ShowLabels	Show or Hide the slider labels and tick marks.
		TickStyle2	Display tick marks on either the left or right side of the slider.
		Value	Default value of vertical(Y) slider, between MinimumY and MaximumY.

A.9 Switch

Sl No	Property		Description
1.	Accessibility	AccessibleDescription	The description that will be reported to accessibility clients.
		AccessibleName	The name that will be reported to accessibility clients.
		AccessibleRole	The role that will be reported to accessibility clients.
2.	Appearance	BackColor	The background color of the component.
		BackgroundImage	The background image used for the component.
		BackgroundImageLayout	The background image layout used for the component.
		BorderStyle	Indicates whether the panel should have the border.
		Cursor	The cursor that appears when pointer moves over the control.
		Font	The font used to display text in control.
		Fore color	The foreground color of this component, which is used to display text.
		RightToLeft	Indicates whether the component should draw right to left for RTL language.
		UseWaitCursor	When this property is true, the cursor property of the control and its child are set to WaitCursor.
3.	Behavior	AllowDrop	Indicates whether the control can accept data that the user drags on to it.
		AutoValidate	Indicates whether the controls in the container will be automatically validated when the focus changes
		ContextMenuStrip	The Shortcut menu to display when the user right clicks on the control.
		Enabled	Indicates whether the control is enabled.
		ImeMode	Determines the IME(Input Method Editor) status of the object when selected.
		TabIndex	Determines the index in the TAB order that this control will occupy.
		TabStop	Indicates whether the user can use the TAB key to give focus to the control.
		Visible	Determines whether the control is visible or hidden.
4.	Data	DataBindings	The data bindings for the control.
		• (Advanced)	Advanced binding allows the user to bind properties of the control.
		• Tag	
		Tag	User-defined data associated with the object.
5.	Design	Name	Indicates the name used in the code to identify the object.
		GenerateMember	Indicates if a member variable will be generated for this component.
		Locked	The locked property determines if we can move or resize the control.
		Modifiers	Indicates the visibility level of the object.
6.	Focus	CausesValidation	Indicates whether this component raises validation events.
7.	Layout	Anchor	Defines the edges of the container to which a certain control is bound. When a control is anchored to an edge, the distance between the control's closest edge and the specified edge will remain constant.
		AutoScroll	Indicates whether scroll bars automatically appear when the control contents are larger than the visible area.
		AutoScrollMargin	The margin around controls during auto scroll.
		AutoScrollMinSize	The minimum logical size for the auto scroll region.
		AutoSize	Specifies whether a control will automatically size itself to fit

			its contents.
		AutoSizeMode	Specifies the mode by which the user interface element automatically resizes itself.
		Dock	Defines which borders of the control are bound to the container.
		Location	The co-ordinates of the upper-left corner of the control relative to the upper-left corner of the container.
		Margin	Specifies space between this control and another control's margin.
		MaximumSize	Specifies the maximum size of the control.
		MinimumSize	Specifies the minimum size of the control.
		Padding	Specifies the interior spacing of the control.
		Size	The size of the control in pixels.
8.	Misc	OffColor	Switch indicator's color when in the 'OFF' position.
		On	Set the switch position 'ON'.
		OnColor	Switch indicator's color when in the 'ON' position.

A.10 SpinText

Sl No	Property		Description
1.	Accessibility	AccessibleDescription	The description that will be reported to accessibility clients.
		AccessibleName	The name that will be reported to accessibility clients.
		AccessibleRole	The role that will be reported to accessibility clients.
2.	Appearance	BackColor	The background color of the component.
		BackgroundImage	The background image used for the component.
		BackgroundImageLayout	The background image layout used for the component.
		BorderStyle	Indicates whether the panel should have the border.
		Cursor	The cursor that appears when pointer moves over the control.
		Font	The font used to display text in control.
		Fore color	The foreground color of this component, which is used to display text.
		RightToLeft	Indicates whether the component should draw right to left for RTL language.
		UseWaitCursor	When this property is true, the cursor property of the control and its child are set to WaitCursor.
3.	Behavior	AllowDrop	Indicates whether the control can accept data that the user drags on to it.
		AutoValidate	Indicates whether the controls in the container will be automatically validated when the focus changes
		ContextMenuStrip	The Shortcut menu to display when the user right clicks on the control.
		Enabled	Indicates whether the control is enabled.
		ImeMode	Determines the IME(Input Method Editor) status of the object when selected.
		TabIndex	Determines the index in the TAB order that this control will occupy.
		TabStop	Indicates whether the user can use the TAB key to give focus to the control.
		Visible	Determines whether the control is visible or hidden.
4.	Data	DataBindings	The data bindings for the control.
		• (Advanced)	Advanced binding allows the user to bind properties of the control.

		• Tag	
		Tag	User-defined data associated with the object.
5.	Design	Name	Indicates the name used in the code to identify the object.
		GenerateMember	Indicates if a member variable will be generated for this component.
		Locked	The locked property determines if we can move or resize the control.
		Modifiers	Indicates the visibility level of the object.
6.	Focus	CausesValidation	Indicates whether this component raises validation events.
7.	Layout	Anchor	Defines the edges of the container to which a certain control is bound. When a control is anchored to an edge, the distance between the control's closest edge and the specified edge will remain constant.
		AutoScroll	Indicates whether scroll bars automatically appear when the control contents are larger than the visible area.
		AutoScrollMargin	The margin around controls during auto scroll.
		AutoScrollMinSize	The minimum logical size for the auto scroll region.
		AutoSize	Specifies whether a control will automatically size itself to fit its contents.
		AutoSizeMode	Specifies the mode by which the user interface element automatically resizes itself.
		Dock	Defines which borders of the control are bound to the container.
		Location	The co-ordinates of the upper-left corner of the control relative to the upper-left corner of the container.
		Margin	Specifies space between this control and another control's margin.
		MaximumSize	Specifies the maximum size of the control.
		MinimumSize	Specifies the minimum size of the control.
		Padding	Specifies the interior spacing of the control.
		Size	The size of the control in pixels.
8.	Misc	ButtonPosition	Spin control position, left or right of the text field.
		Decimals	Number of decimal places to display.
		Increment	Amount of increment/decrement per click.
		SetArrowDirection	Up/Down or Left/Right.
		SetButtonColor	Color of Spin control.
		SetInitialValue	Default Spin Text Value.
		SetMax	Maximum SpinText range value.
		SetMin	Minimum SpinText range value.
		SetSelectColor	Color of the arrow.
		SetSpeed	The increment/decrement 'Speed', adjusts how quickly the value changes when arrow is held down [1 - 10, 10 is fastest]
		SetTextBoxColor	Background color of the text-box.
		TextBoxFont	Font of text inside the text box.

A.11 Table

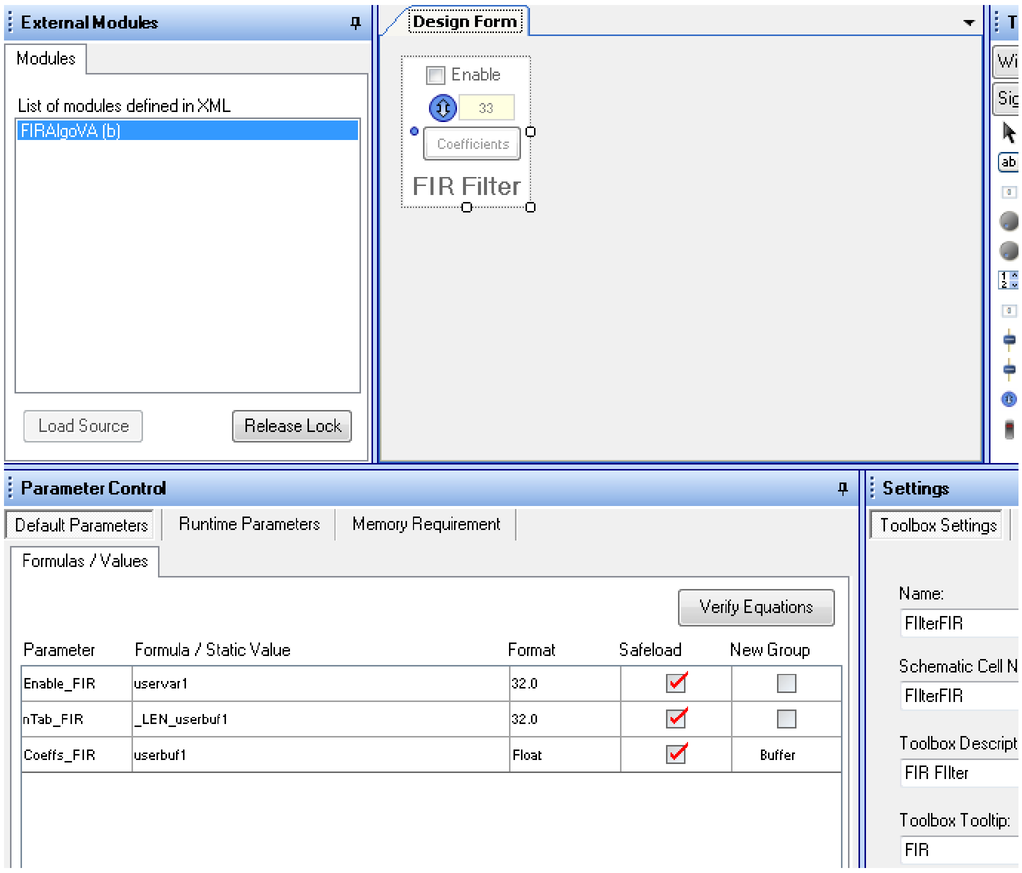
Sl No	Property		Description
1.	Accessibility	AccessibleDescription	The description that will be reported to accessibility clients.
		AccessibleName	The name that will be reported to accessibility clients.
		AccessibleRole	The role that will be reported to accessibility clients.

2.	Appearance	BackColor	The background color of the component.
		BackgroundImage	The background image used for the component.
		BackgroundImageLayout	The background image layout used for the component.
		BorderStyle	Indicates whether the panel should have the border.
		Cursor	The cursor that appears when pointer moves over the control.
		Font	The font used to display text in control.
		Fore color	The foreground color of this component, which is used to display text.
		RightToLeft	Indicates whether the component should draw right to left for RTL language.
		UseWaitCursor	When this property is true, the cursor property of the control and its child are set to WaitCursor.
3.	Behavior	AllowDrop	Indicates whether the control can accept data that the user drags on to it.
		AutoValidate	Indicates whether the controls in the container will be automatically validated when the focus changes
		ContextMenuStrip	The Shortcut menu to display when the user right clicks on the control.
		Enabled	Indicates whether the control is enabled.
		ImeMode	Determines the IME(Input Method Editor) status of the object when selected.
		TabIndex	Determines the index in the TAB order that this control will occupy.
		TabStop	Indicates whether the user can use the TAB key to give focus to the control.
		Visible	Determines whether the control is visible or hidden.
4.	Data	DataBindings	The data bindings for the control.
		• (Advanced)	Advanced binding allows the user to bind properties of the control.
		• Tag	
		Tag	User-defined data associated with the object.
5.	Design	Name	Indicates the name used in the code to identify the object.
		GenerateMember	Indicates if a member variable will be generated for this component.
		Locked	The locked property determines if we can move or resize the control.
		Modifiers	Indicates the visibility level of the object.
6.	Focus	CausesValidation	Indicates whether this component raises validation events.
7.	Layout	Anchor	Defines the edges of the container to which a certain control is bound. When a control is anchored to an edge, the distance between the control's closest edge and the specified edge will remain constant.
		AutoScroll	Indicates whether scroll bars automatically appear when the control contents are larger than the visible area.
		AutoScrollMargin	The margin around controls during auto scroll.
		AutoScrollMinSize	The minimum logical size for the auto scroll region.
		AutoSize	Specifies whether a control will automatically size itself to fit its contents.
		AutoSizeMode	Specifies the mode by which the user interface element automatically resizes itself.
		Dock	Defines which borders of the control are bound to the container.
		Location	The co-ordinates of the upper-left corner of the control relative to the upper-left corner of the container.
		Margin	Specifies space between this control and another control's

			margin.
		MaximumSize	Specifies the maximum size of the control.
		MinimumSize	Specifies the minimum size of the control.
		Padding	Specifies the interior spacing of the control.
		Size	The size of the control in pixels.
8.	Misc	ButtonAutoSize	
		ButtonPosition	Position of the button.
		ButtonSize	Size of the button.
		ButtonText	Text on the button.
		ModifySize	True/False
		NumValues	Number of values in the table.
		SizeBoxposition	Position of the text-box.
		SpinTextMax	Maximum size of the table allowed.
		Values	Values of table.

B. Table Control Usage

Use the '`ModifySize`' property of the Table control to make the length of the Table fixed or variable. When '`ModifySize`' is 'false', the size is fixed and is equal to the '`NumValues`' property of the control. When '`ModifySize`' property is set to 'true', the size can be varied during Schematic design time. The default size is equal to the value of '`NumValues`' property of the control. Users can make use of the size of the Table as a variable in defining the parameters and memory sizes by making use of `_LEN_<runtime_buffer>` where 'runtime_buffer' is the runtime buffer to which the Table control is assigned. The screenshots below illustrates the usage of `_LEN_<runtime_buffer>`. In the example below the Table control is assigned to the runtime buffer '`userbuf1`'.



Parameter	Formula / Static Value	Format	Safeload	New Group
Enable_FIR	uservar1	32.0	✓	☐
nTab_FIR	_LEN_userbuf1	32.0	✓	☐
Coeffs_FIR	userbuf1	Float	✓	Buffer

Name:
FIlterFIR
Schematic Cell N
FIlterFIR
Toolbox Descript
FIR FIlter
Toolbox Tooltip:
FIR

Figure 29: Table Size as a Variable in Defining Parameter

C. Plug-In Interface XML

The below interface XML will be generated along with the UI plug-in. This interface XML will provide details about the cell name and the real-time variable names to be used in case of Algorithm only plug-in DLL generation using UI plug-in DLL as the reference.

```
<?xml version="1.0" standalone="yes"?>
<SS4SH name="ss4sh_moduleinterface_xml" description="SigmaStudio for SHARC
Algorithm Designer" version="3.6.0.0">
   <interface name="IScaler" />
   <Settings name = "Scaler" />
   <InterfaceFunction name="ReceiveData_uservar1" returntype="int">
     <Parameter name=″arr″ type=″Arraylist″ />
     <Parameter name=″repcount″ type=″int″ />
   </ InterfaceFunction >

     <InterfaceFunction name="ReceiveData_uservar2" returntype="int">
     <Parameter name=″arr″ type=″Arraylist″ />
     <Parameter name=″repcount″ type=″int″ />
   </ InterfaceFunction >

   <Variable name=″uservar1″ control=″Knob1″/>
   <Variable name=″uservar2″ control=″Knob2″/>

</SS4SH>
```

Code 8: Example Interface XML for a Scaler Module (ADSP-SC5xx)

C.1 Interface

C.1.1 Name

This is the name of the plug-in interface which should be implemented by the plug-in algorithm.

C.2 Settings

C.2.1 Name

This is the “Name” field in Settings menu of Algorithm Designer. This will be used to form the interface name and validate the referenced UI DLL in case of generation of Algorithm-Only Plug-in.

C.3 Interface Function

This section of the XML gives details of the various interface functions.

C.3.1 Name

This represents the name of the interface function.

C.3.2 Return Type

This represents the return type of the interface function.

C.3.3 Parameter

This sub-section of the interface function elements of the interface XML gives details of the various parameters of the interface function.

C.3.3.1 Name

This represents the name of the parameter of the interface function

C.3.3.2 Type

This represents the type of the interface function parameter.

C.4 Variable

This section of the interface XML gives details about the run-time variables/ buffers used and the real-time controls which they are bound to.

C.4.1 Name

This gives the name of the run-time variable/ buffer.

C.4.2 Control

This represents the control which the variable/ buffer is assigned to in the cell.

References

Table 2: References

Reference No.	Description
[1]	SigmaStudio for SHARC Release Note(*ReleaseNotes.pdf)*
[2]	SigmaStudio for SHARC Algorithm Designer Guide(AE_42_SS4G_AlgorithmDesignerGuide.pdf)