

PurePath Studio

Graphical Development Environment

(GDE)

User Guide

for

Portable Audio Devices

Version
7 June 2010

Table of Contents

1.	INTRODUCTION	4
1.1	DEFINITIONS	4
2.	TOOLSET OVERVIEW	5
2.1	GDE OVERVIEW	5
2.2	APPLICATION DEVELOPMENT	6
2.3	GDE ARCHITECTURE	10
2.4	USING THE GDE	12
2.5	ADDING AUDIO COMPONENTS TO THE DIAGRAM	14
2.6	SETTING COMPONENT PROPERTIES	15
2.7	CONFIGURATIONS	17
2.8	SETTING COMPONENT INTERFACE PROPERTIES	19
2.9	SETTING ALIASOF PROPERTY	21
2.10	LOADING AND SAVING PROPERTY SETTINGS	23
2.11	CREATING CONNECTIONS BETWEEN COMPONENTS	24
2.12	SETTING CONNECTION APPEARANCE PROPERTIES	26
2.13	GENERATING APPLICATION CODE	27
2.14	EXECUTING AND CONTROLLING YOUR APPLICATION	28
2.15	RUN MODE AND EDIT MODE	29
2.16	SAVING THE DIAGRAM	30
2.17	FRAMEWORK SWITCHING	31
2.18	TOOLS OPTIONS	33
2.19	PRINTING THE DIAGRAM	37
2.20	FLOATING AND DOCKING WINDOWS	38
2.21	COMPONENT LIBRARY	40
2.22	I ² C LOGGING	41
3.	DETAILED SPECIFICATIONS	42
3.1	GDE MAIN WINDOW	42
3.1.1	Menu Bar	42
3.1.2	Toolbar	44
3.1.3	Command Line Switches	45
3.1.4	Child Window Docking and Floating	45
3.2	PROCESS FLOW EDITOR	48
3.2.1	Components Window	49
3.2.2	Diagram Window	51
3.2.3	Dragging Components to the Diagram	52
3.2.4	Component Help files	52
3.2.5	Connecting Components with Links	52
3.2.6	Moving Components and Links	53
3.2.7	Component Cut/Copy/Paste	53
3.2.8	Framework Components	54
3.2.9	Property Window	54
3.2.10	Property Settings	55
3.2.11	Output Window	56
3.2.12	Pan / Zoom Window	57
3.2.13	Resource Window	57

3.2.14	Source Code Window	59
3.2.15	Process Flow File Management.....	59
3.2.16	Component Interface Overview	60
3.2.17	Sample Rate Support	60
3.3	PRINTING	61
3.3.1	Printing	61
3.3.2	Page Setup	62
3.3.3	Print Preview	63
3.4	GENERATING APPLICATION CODE	63
3.4.1	Process Listing File	64
3.5	EDIT MODE AND RUN MODE.....	64
3.5.1	miniDSP Adaptive Mode	64
3.6	COMPONENT INTERFACES	65
3.6.1	Component Interfaces	65
3.7	CONFIG FILE EXECUTION.....	67
3.7.1	Command File Execution	67
3.8	I ² C MEMORY TOOL.....	67
3.8.1	Peek/Poke Support	68
3.8.2	Peek/Poke Support in Adaptive Mode	71
3.8.3	I2C Read/Write Support.....	73
3.8.4	I2C Command File Support for miniDSP	74
3.9	GDE OPTIONS	76
3.9.1	Display Tab	77
3.9.2	Build Tab.....	78
3.9.3	Target Tab	79
3.10	ONLINE HELP	79
3.11	SUPPORTING MULTIPLE TARGET CHIPSETS.....	79
3.12	COMPONENT LIBRARY.....	80
3.12.1	System Component Library	80
3.12.2	User Component Library	80
3.12.3	Component Cache	80
3.13	PROCESS FLOW CONTROLLERS.....	80
3.14	POPUP GUIs	80

1. INTRODUCTION

This is the User Guide for the PurePath Studio Graphical Development Environment (GDE). It provides an overview of the GDE and related tools.

The tools and capabilities discussed in this User Guide are:

- The Graphical Development Environment (GDE) application, Gde.exe including:
 - Process flow creation and editing
 - Build/download and execution on the EVM
 - Control of running process flows

1.1 DEFINITIONS

Component	A reusable, parameterized portion of audio processing software logic
Component Instance	An instantiated Component, with actual values given for parameters
Component Library	Repository of components for the GDE
Component Palette	Area of GDE that displays the available components.
GDE	Graphical Development Environment
Process Flow	A diagram of components and links that graphically represents an audio application
Process Flow Document	The 'document' that is edited by the GDE
Template	A portion of a component that is copied during component instantiation, with actual values replacing parameter references

2. TOOLSET OVERVIEW

The GDE supports application development on the AIC3254, AIC36, TSC2117, and ADC3x01 (ADC3101 and ADC3001) target processors.

2.1 GDE OVERVIEW

The GDE's easy to use graphical interface provides a complete toolset that helps the user build miniDSP processing applications from existing Components. The user can combine the power of several algorithms into one application by just dragging and dropping those Components to create the desired process flow.

The PurePath Studio (Portable Audio) - Graphical Development Environment (GDE) provides the means to create and edit process flows, and then generate code from these process flows.

Within the GDE, you can:

- drag reusable audio components from a Components window and place them in a Diagram window
- specify the output-to-input connection between components by linking them with lines on the diagram
- edit the configuration properties for each component in the Properties window
- generate code required for the application to execute
- download and run the application on PurePath Studio (Portable Audio) hardware
- control the program during execution via interactive symbols in the diagram

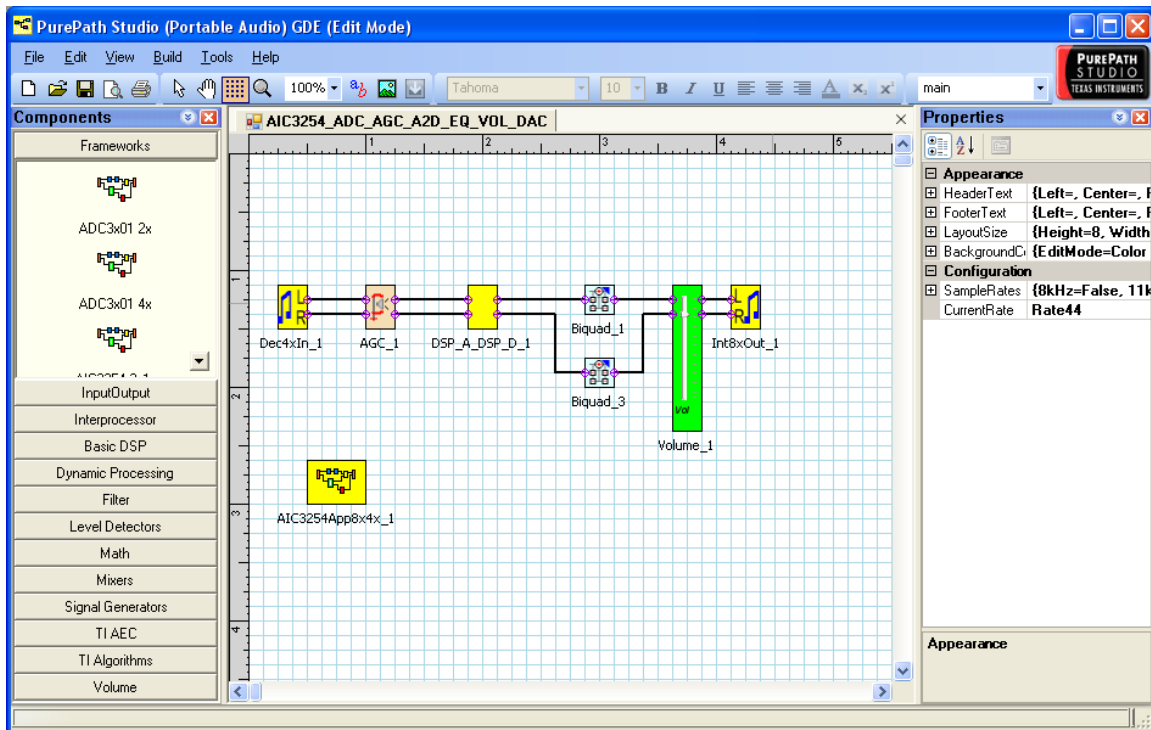
In the GDE, audio applications are created via three interfaces: the Components window, the Diagram window, and the Properties window.

The Components window contains reusable application and audio components for building the application. The Diagram window provides a workspace to create the process flow diagram of the components. The Properties window displays configurable properties for a selected component.

The steps below provide an overview of the process to create, run, and control an audio application with the GDE.

1. Add an application framework from the Components window and place it in the Diagram window.
2. Drag components from the Components window and place them in the Diagram window.
3. Create connections between components ("wire" the components) to depict the data flow.
4. Generate code for the application.
5. Download, execute, and control the application on the board.

2.2 APPLICATION DEVELOPMENT

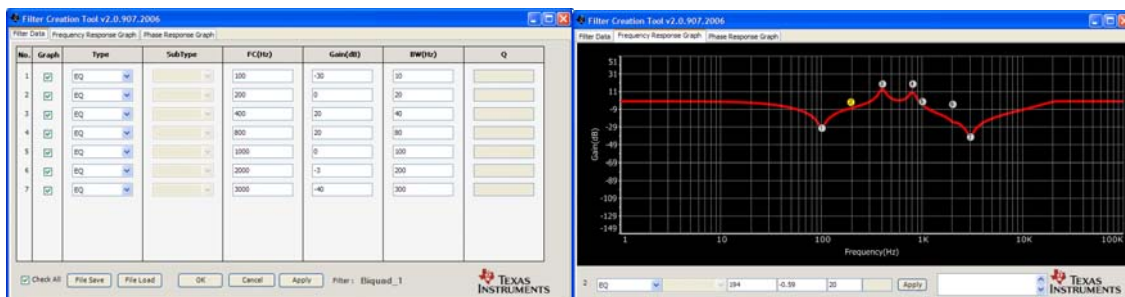


The GDE includes a palette of components and supports a very easy to use drag-and-drop programming model.

Components are listed in a Component Window. One or more instances of components can be dragged and dropped to a *process flow* diagramming area.

Component instances on the process flow can then be linked together to define the data flow. Each component instance has *properties* which can be used to configure the behavior of the component instance. For example, a mixer component can have mixer coefficients set.

Sophisticated components may have component-specific pop-up GUIs to set property values. An example is the Biquad component, which has a pop-up filter design tool for setting the biquad coefficient values.

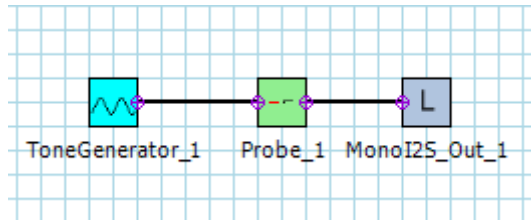


The GDE supports building an application that implements the given process flow and downloading the application to the EVM hardware for execution.

Probepoint Support in the GDE

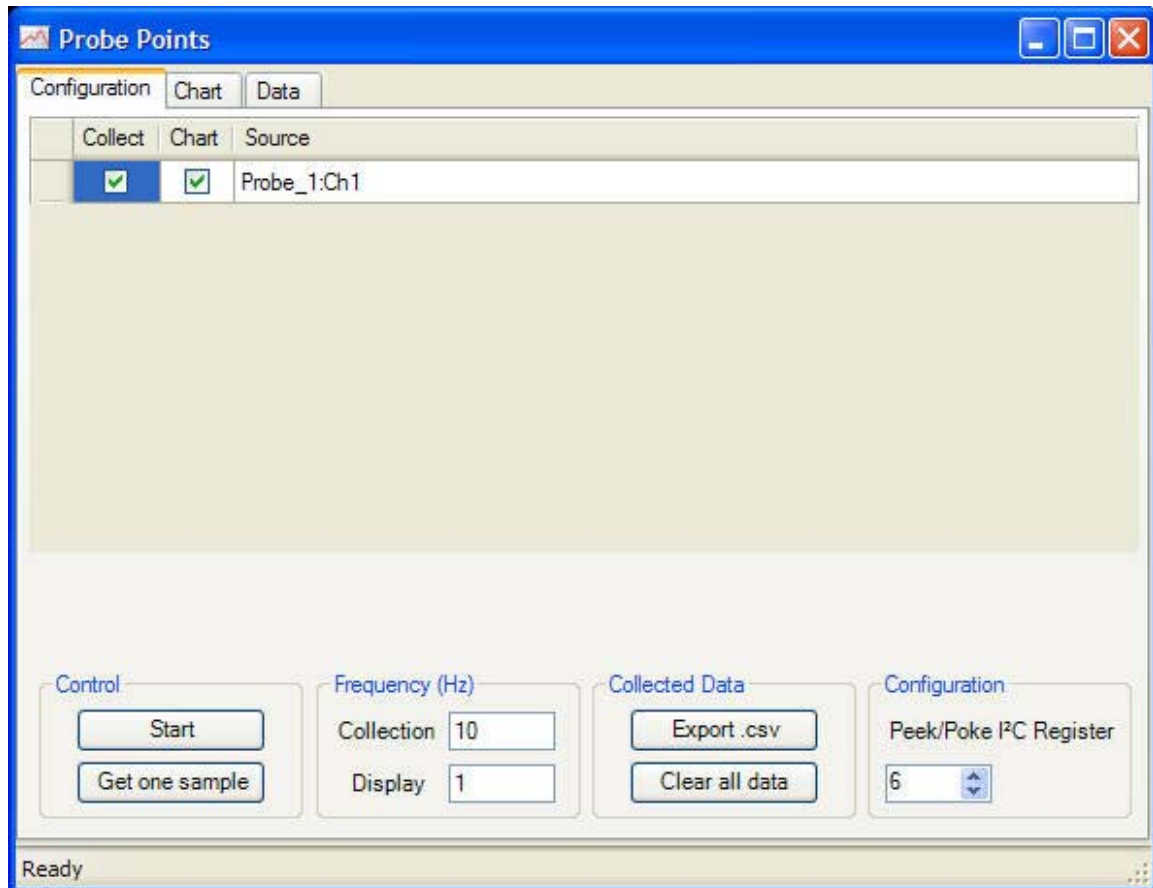
The GDE now supports the use of probepoints to read data from the target while running.

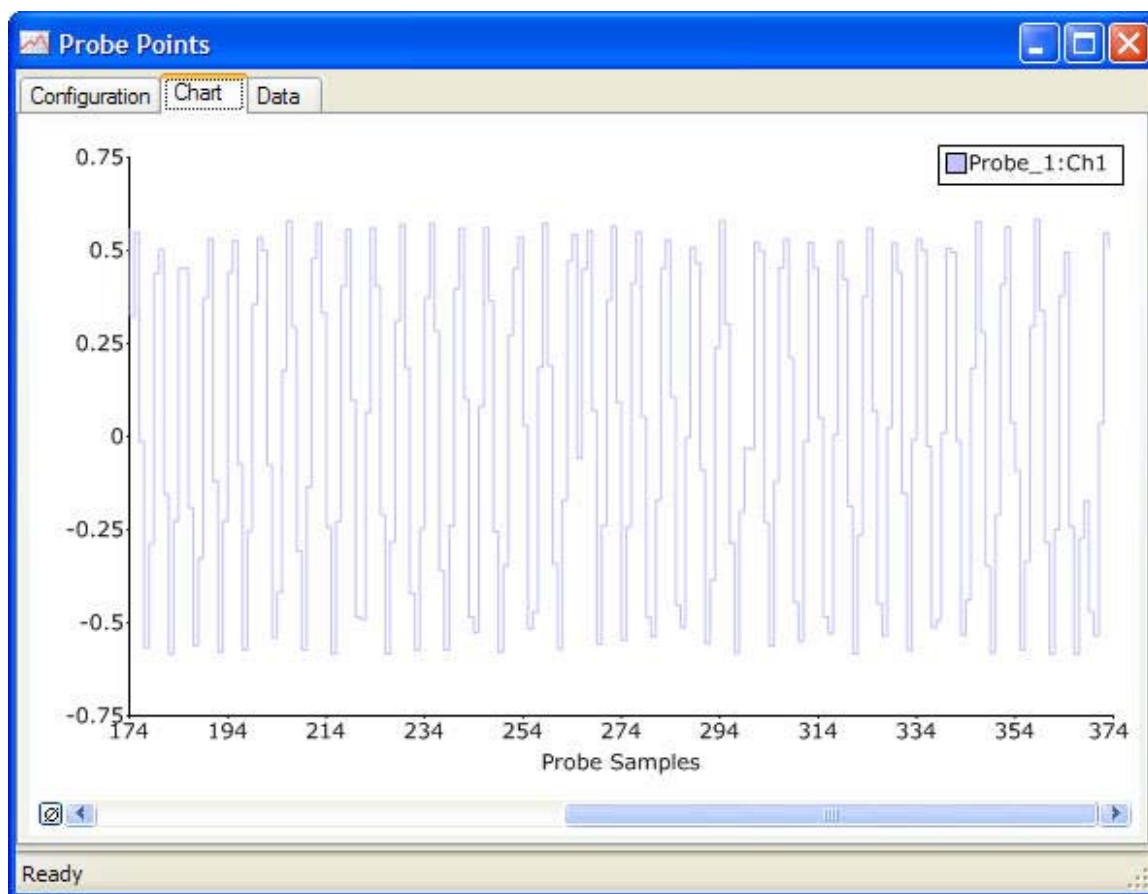
The new 'Probepoint' component may be placed anywhere in the process flow.

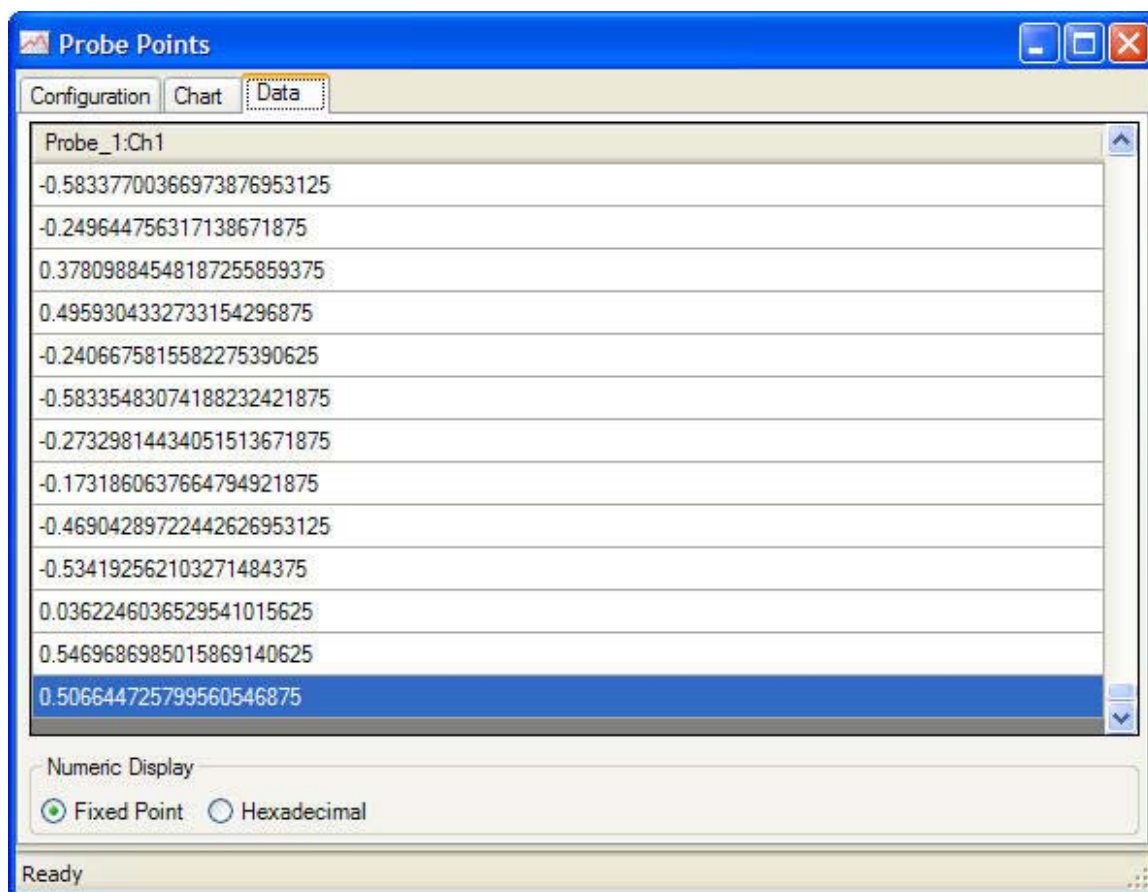


Then, the Probepoint dialog can choose which probepoints to collect and graph while the target is running. See the online help for more information.

The GDE supports data acquisition while the process flow is running using the GDE ProbePoints.



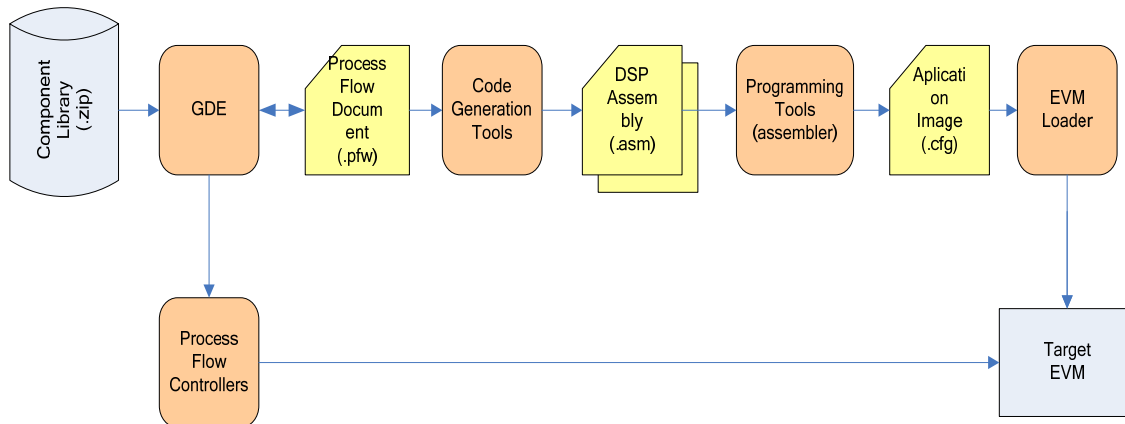




While running an application, the GDE will permit the user to modify the running application by communicating with the application thru a communication link such as I²C.

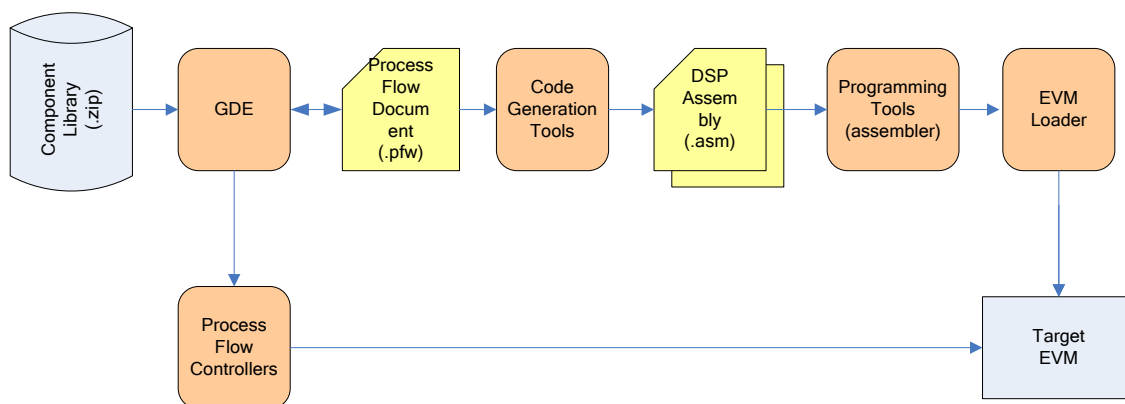
2.3 GDE ARCHITECTURE

The PurePath Studio toolset is a set of applications, both graphical and console-mode. The GDE generates DSP assembly source code, which is assembled and built into an executable application. After loading on the board, the GDE controls the process flow thru the use of Process Flow Controller DLLs.



The basic unit of application software in the GDE is the **component**. A component is a reusable, configurable algorithm that can be combined with other components to form a process flow. A component can be as small as a single operation such as Multiply, or as large as a complex processing unit such as Surround Sound.

At this time, the GDE supports multiple audio processors from multiple hardware families. The design of the GDE allows for the retargeting of the GDE to other application areas that are data-flow based.



The GDE toolset includes the tools for creating and editing graphical *process flows*, as well as tools for generating DSP code from a process flow, where applicable. In addition, tools for creating new components are included in the GDE toolset.



The GDE toolset uses tools that are defined in the Tools spec to build the source code and communicate with the EVM.

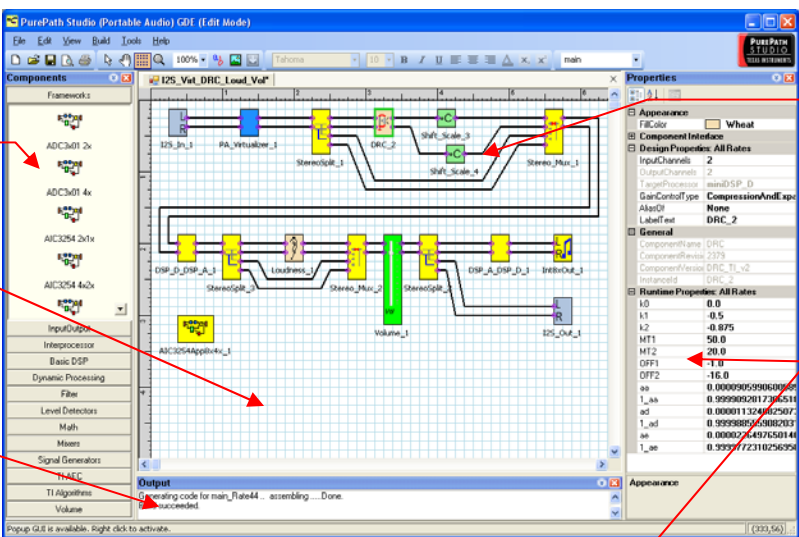
Each of the components of the GDE toolset is described in Section 0 below.



2.4 USING THE GDE

The GDE provides a graphical, component based, drag and drop programming environment for data-flow based applications such as audio processing.

Process flows are built by dragging components from a palette to a drawing area, then linking the components together to represent the data flow. The GDE then builds an application to represent the process flow. The source code for the generated application is viewable by the user.



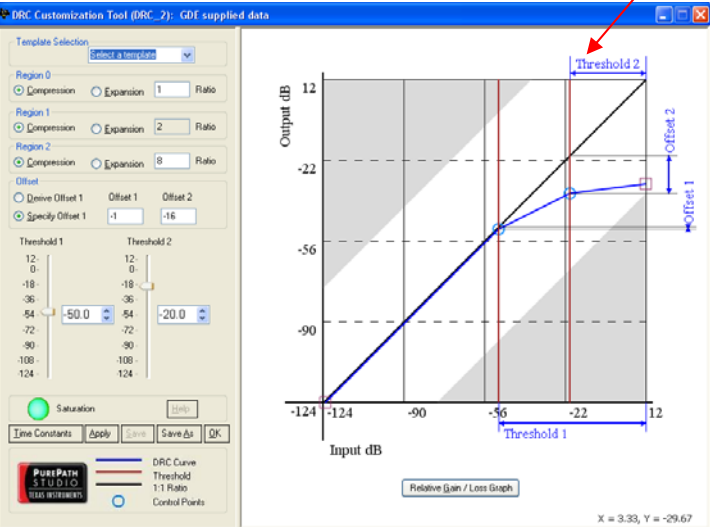
Palette of Components

Visio-like drawing area

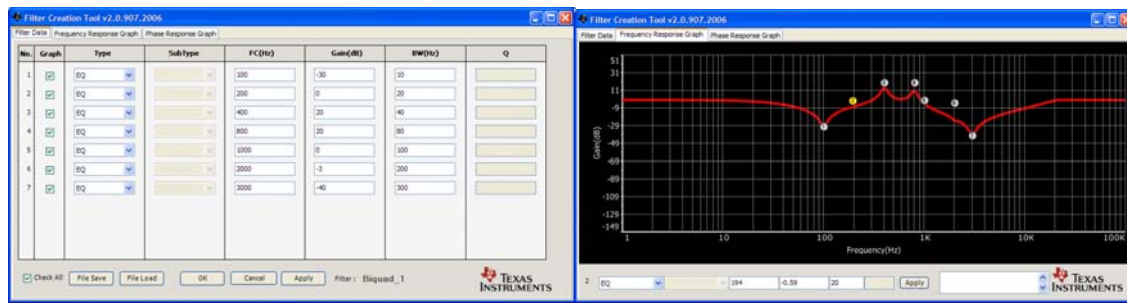
Generates user-readable source code

Drag/Drop components, then wire together

Set coefficients directly or with pop-up GUI's



Component settings, such as filter coefficients can be directly entered, or can be computed by graphical add-ins such as filter generators.



The GDE automatically generates an interface for control of the component at runtime. These interfaces often match exactly the interface that will be used to control the fielded application.

The GDE supports download and execution of the process flow on EVM hardware, and control of a running process flow.

The basic unit of application software in the GDE is the **component**. A component is a reusable, configurable algorithm that can be combined with other components to form a process flow. A component can be as small as a single operation such a Multiply, or as large as a complex processing unit such as Surround Sound

At this time, the GDE supports multiple audio processors from multiple hardware families. The design of the GDE allows for the retargeting of the GDE to other application areas that are data-flow based.

2.5 ADDING AUDIO COMPONENTS TO THE DIAGRAM

Audio components are selected in the Components window and placed in the Diagram window using drag-and-drop functionality.

In the Components window, the category headings represent the various types of components available.

Adding an Application Framework

Before adding any other component, you must first add an application framework component to the diagram. An application framework provides a skeleton for an audio application: a code and data infrastructure utilized by all components within the process flow diagram.

1. In the Components window, click the Framework heading.
2. Select a framework component.
3. Drag the component into the Diagram window and place it as desired.

The framework component:

- cannot be deleted unless all the other components are deleted.
- cannot be cut, copied, or pasted.
- does not have input or output connections to the rest of the diagram.

Adding Other Components

To add other components to the diagram:

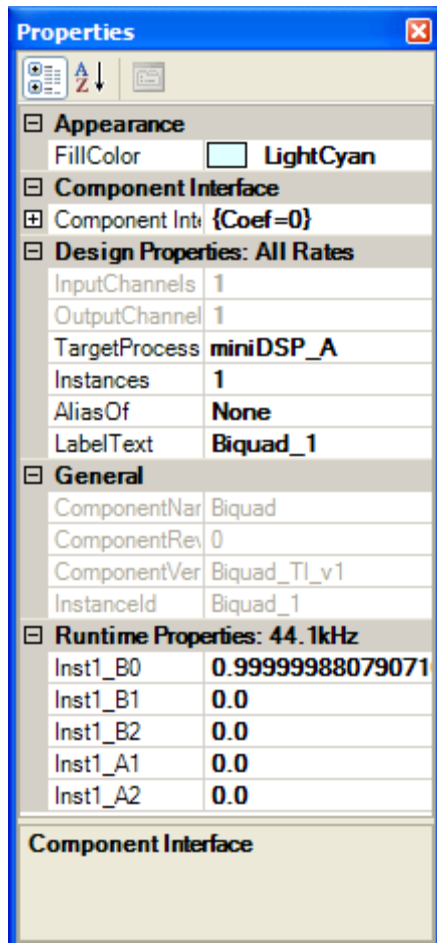
1. Click on a heading in the Components window to view the available components of that type.
2. Select a component.
3. Drag the component into the Diagram window and place it as desired. The component remains selected in the diagram until you select something else, so you can view and/or modify its properties in the Properties window.

Component Names

Components appear in the Diagram window with the name of the component displayed below the symbol. The GDE automatically names each component instance with a unique name. For example, the first Volume component placed on the diagram will be named Volume_1; the second Volume component will be named Volume_2, etc.

2.6 SETTING COMPONENT PROPERTIES

The Properties window displays the properties for the component currently selected in the Diagram window.



Appearance properties control the display of the component in the diagram.

Component Interface properties display all component interface properties.

Design Properties configure the generated code for a component and are common to all supported sample rates for the component.

General properties display the component name and instance identifier.

Rate properties (Rate: All or Rate: *value*) have unique values for each supported sample rate for the component.

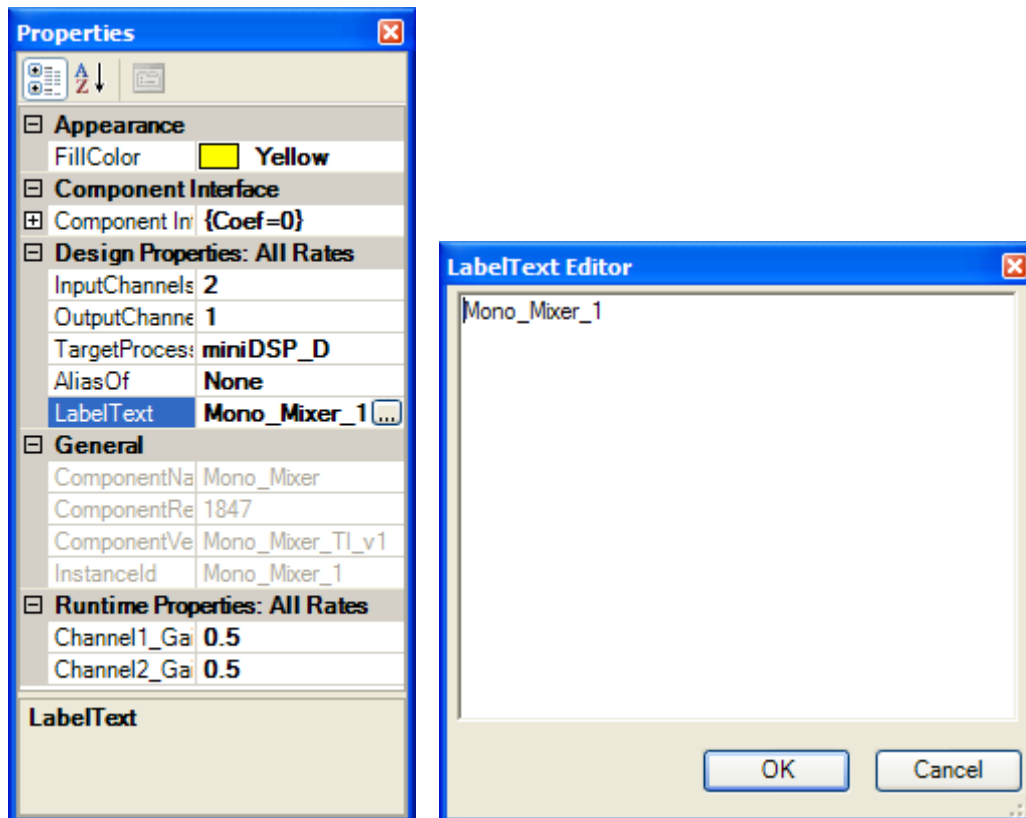
Note: Changing the CurrentRate property in the GDE affects only the calculation of coefficient values. It does not modify the sampling rate on the EVM. In order to modify the EVM's sampling rate, the firmware on the EVM must be reprogrammed and the EVM re-enumerated on the USB.

See the online help Reference Information section for information on changing the EVM sampling rate.

To modify the value of a property:

1. In the property's field, type in or select the desired value.
2. Press Enter.

String properties can be edited using a popup text editor. The text editor supports simple edit and cursor movement with the mouse or keyboard. The editor also supports cut, copy, paste, undo and redo using the standard shortcut keys. To display the popup text editor, click the ellipsis button in the right corner of the property cell.



Properties that are read-only are shown in grey and cannot be edited. Some properties are always read-only, while others may be editable according to the current mode. Rate: All and Rate: *value* properties can be edited while in Run mode. All properties can be edited in Edit mode.

When no component on the diagram is selected, the Properties window displays the properties for the Diagram window.

Certain components contain controls such as sliders or check boxes. These controls are connected to certain property values. For such a component, property values can be modified by manipulating the controls or by editing the value in the Properties window.

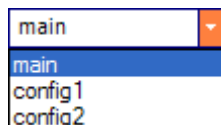
Click on a property's name to view descriptive text in the text box at the bottom of the Properties window.

2.7 CONFIGURATIONS

The GDE supports multiple configurations of a diagram. All configurations contain the same components in the diagram, connected in the same way. Configurations may be used to define a process flow with multiple behaviors. Each configuration has its own set of properties for every component. Properties that are shared between configurations may be modified only by the main configuration. All other configurations will display the property as read-only. All new diagrams start with one configuration, named **main**.

Configurations can be added or removed using the Configuration Editor dialog.

The currently active configuration is displayed in the configuration dropdown list. You can switch from one configuration to another using the configuration dropdown list.



Switching Configurations

Switching the active configuration is achieved by using the pull-down list on the GDE toolbar. In order to change configurations, no pop-up GUIs can be open. If any are open, the user is instructed to first close them.

Upon switching the active configuration, the Properties Window is refreshed with the currently selected component's property values in the new configuration.

The active configuration can only be changed while the GDE is in Edit mode.

Code Generation with the Generate Complete Applications option

If **Generate Complete Applications** is chosen in the Options dialog, the GDE will generate complete application images for each of the configurations and sample rates in the process flow..

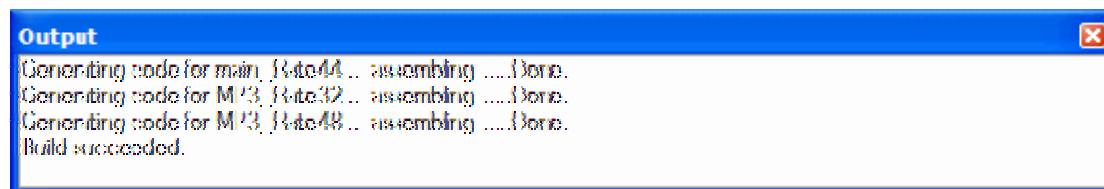
For example, consider a process flow with:

- § A **main** configuration with one process flow defined with one supported rate: 44.1Khz
- § An **MP3** configuration defined with supported rates 32Khz and 48Khz.

Code generation for this process flow would result in the following applications being generated:

- § Main_Rate44.cfg
- § MP3_Rate32.cfg
- § MP3_Rate48.cfg

The GDE Output window will show the build for each configuration and sample rate, then the final build of the image file.



Code Generation with the Generate Application Patches option

If **Generate Application Patches** is chosen in the Options dialog, the GDE will generate a complete application image for the current configuration and sample rate and generate application patches for each of the configurations and sample rates in the process flow. The application patches can be used to modify the running application to match each configuration and sample rate.

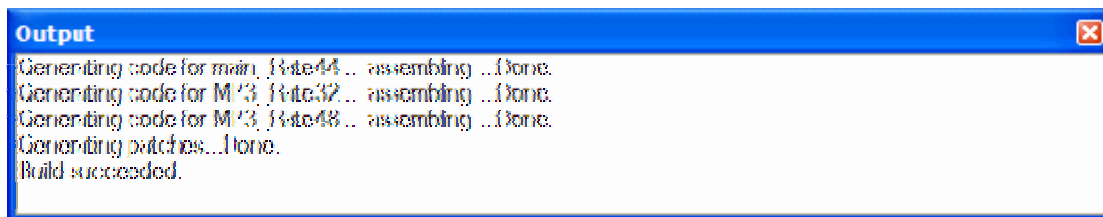
For example, consider a process flow with:

- § A **main** configuration with one process flow defined with one supported rate: 44.1Khz
- § An **MP3** configuration defined with supported rates 32Khz and 48Khz.

Code generation for this process flow would result in the following applications being generated:

- § Main_Rate44.cfg
- § Patch_Main_Rate44.cfg
- § Patch_MP3_Rate32.cfg
- § Patch_MP3_Rate48.cfg

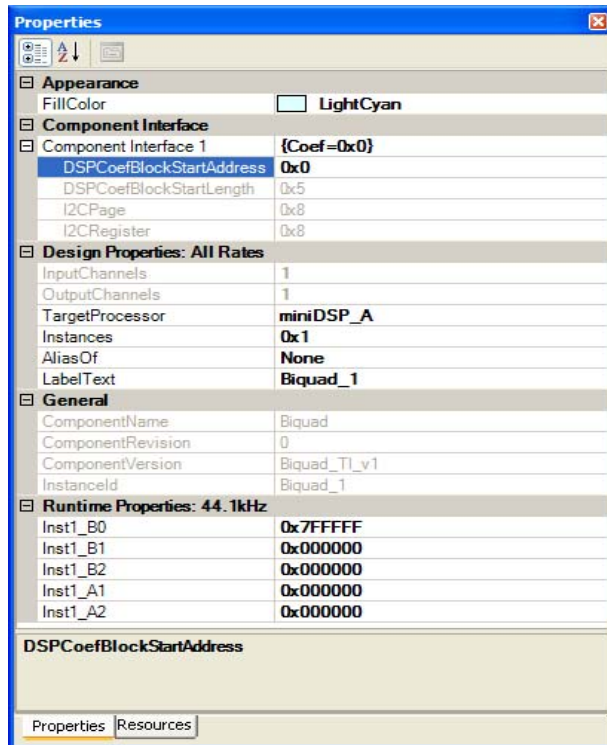
The GDE Output window will show the build for each configuration and sample rate, then the final build of the image file.



2.8 SETTING COMPONENT INTERFACE PROPERTIES

During code execution, components are dynamically controlled by the GDE over the I²C connection.

The component interface for a component is defined in the *Component Interface* section of the component's properties.



Each component interface contains the following information:

Title	Component Property	Description
DSPCoefBlockStartAddresses	DSPCoefBlockStartN	Address of the block of coefficient memory used to hold the values transferred from I ² C
DSPCoefBlockStartLength	DSPCoefBlockLengthN	Length of the block, in words, of coefficient memory .
I2CPage	I2CPageN	The starting I ² C page corresponding to the first coefficient address of the component interface.
I2CRegister	I2CRegisterN	The starting I ² C register corresponding to the first coefficient address of the component interface.

Coef Address Assignment

When a new component is dropped onto the process flow, the GDE assigns any needed coefficient memory to each interface in the component. The information may be modified by the GDE user. However care should be taken not to overlap existing memory in the application.

Aliased vs. Unique Interface Control

If several components are meant to share the same interface, they can be linked together with the AliasOf property. Aliasing is useful for single-channel components such as BiQuads or FIR filters. Several components, one on each audio channel, can share the same coefficients and other properties with this feature. To create an alias of another component select the name of the master component (i.e. Biquad_1) in the AliasOf property for the desired aliased component.

Component Interface Overview Window

The Component Interface Overview window is used to view the overall usages of coefficient memory address settings in a diagram.

This window displays the following information:

- component's Label Text
- interface description
- processor
- coefficient start address
- coefficient length (in words)
- I²C page
- I²C register

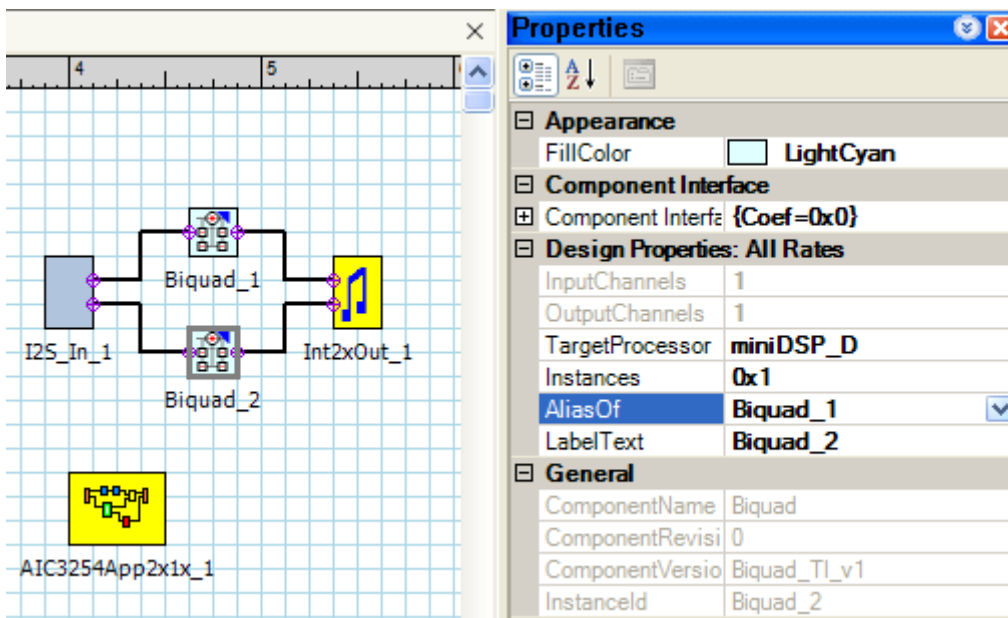


2.9 SETTING ALIASOF PROPERTY

Components of the same type can be linked together using the AliasOf property. An alias consists of a single "master" component and one or more aliased components. This allows components to share property values and resources such as I²C controls.

Creating an Alias

To create an alias of another component, select the name of the "master" component from the dropdown list of the AliasOf property. The property values of the aliased component (Biquad_2 in the example below) will be set to the values of the "master" component (Biquad_1 in the example below). Once the alias is established, property changes to the "master" or any aliased component will be propagated to all components involved in aliasing.

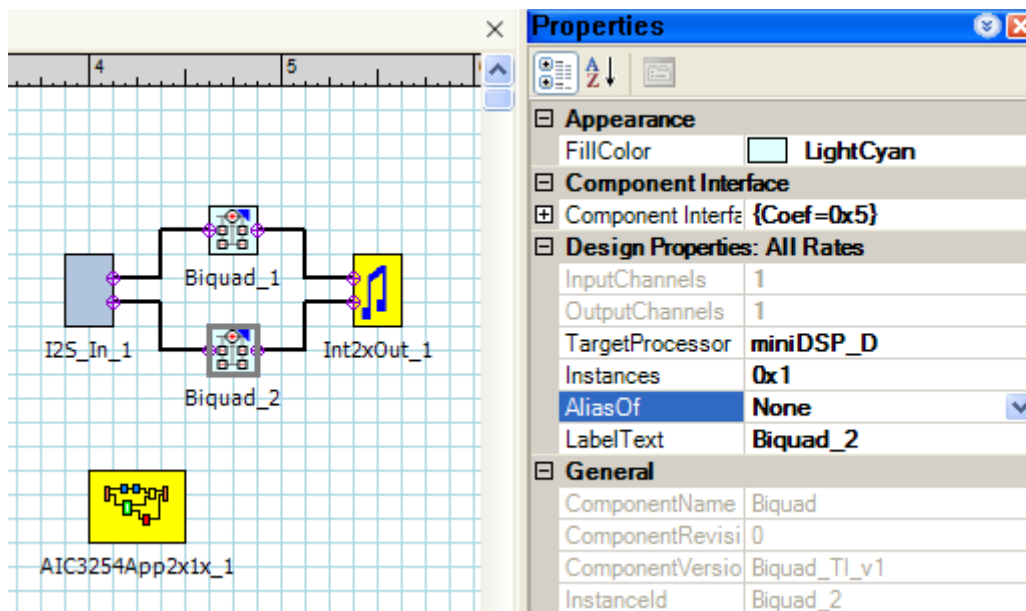


The screenshot displays the TI-RTOS IDE interface. On the left, a component diagram shows an I2S_In_1 component connected to two Biquad components (Biquad_1 and Biquad_2), which are then connected to an Int2xOut_1 component. A yellow component labeled AIC3254App2x1x_1 is also visible. On the right, the Properties window is open, showing the properties for the selected component, Biquad_2. The AliasOf property is set to Biquad_1.

Properties	
Appearance	
FillColor	LightCyan
Component Interface	
Component Interface	{Coef=0x0}
Design Properties: All Rates	
InputChannels	1
OutputChannels	1
TargetProcessor	miniDSP_D
Instances	0x1
AliasOf	Biquad_1
LabelText	Biquad_2
General	
ComponentName	Biquad
ComponentRevision	0
ComponentVersion	Biquad_TI_v1
InstanceID	Biquad_2

Clearing an Alias

To clear an alias, select "None" from the dropdown list of the AliasOf property. If the component has I²C properties, new I²C addresses will be assigned. All other property values remain unchanged. In addition, all aliases are cleared if the "master" component is deleted.



Alias Restrictions

1. All components involved in an Alias must be the same type.
2. A component cannot be an Alias of itself.
3. Aliased components must have the same value for the following properties:
 - Number of channels
 - TargetProcessor
 - Instances (for Biquad components)
 - Taps (for FIR components)
4. Aliased components cannot be chained. That is, a component cannot be declared an alias of an alias. It should be declared an alias of the original component.

2.10 LOADING AND SAVING PROPERTY SETTINGS

Loading Property Settings

Runtime property values can be modified by loading a property settings file (.set). The property settings file contains one or more command lines with the following format:

SetProperty [component_name] [property_name] [property_value]

1. The component name is the value of the component's InstanceId property.
2. The property name is the name of a property that belongs to the specified component.
3. The property value is the new value for the property.

To load the property settings file:

1. Select the File->Load Property Settings menu item
2. Select the property settings file to load using the Load Property Settings dialog.

Saving Property Settings

The runtime property values for the active process flow can be saved to a property settings file (.set).

To save the runtime property values:

1. Select the File->Save Property Settings menu item.
2. Specify the settings file using the Save Property Settings dialog.

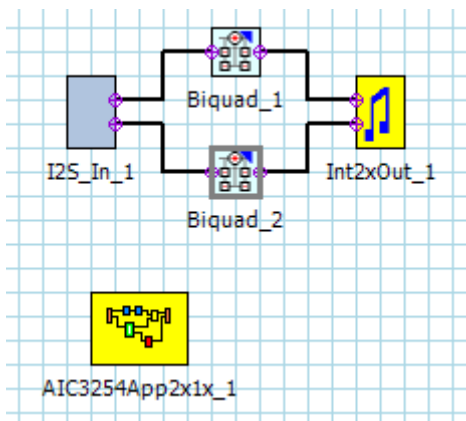
2.11 CREATING CONNECTIONS BETWEEN COMPONENTS

After adding components to the diagram, you must connect them via their ports. Connecting components in the diagram is referred to as *wiring* or *linking* the components. Components must be wired in order to depict data flow.

To wire components:

1. Click on a port icon. After you click on the port, the cursor changes to a drawing pointer (crosshair).
2. Use the pointer to draw the line to the destination port. To complete the connection, click on the destination port.

Links are straight by default. To route a link while drawing it, click at any interim point to create a vertex, and continue to do so as desired until the destination port is reached. The example below shows links that have been routed.



The following wiring rules are enforced:

- Only output ports (located on the right side of the component) may be wired to input ports (located on the left side of the component).
- Cycles are not permitted, either direct (A wired to A) or indirect (A wired to B wired to A).
- All component input ports must be connected to something.

If you create a connection that violates any of the rules stated above, an error message will appear in the Output window and the line will be colored red in the Diagram window.

Cancelling the Line

To stop routing the line after you've already started drawing it, press the <Esc> button.

Moving Components and Links

Components with links can be moved. The links will move with the component.

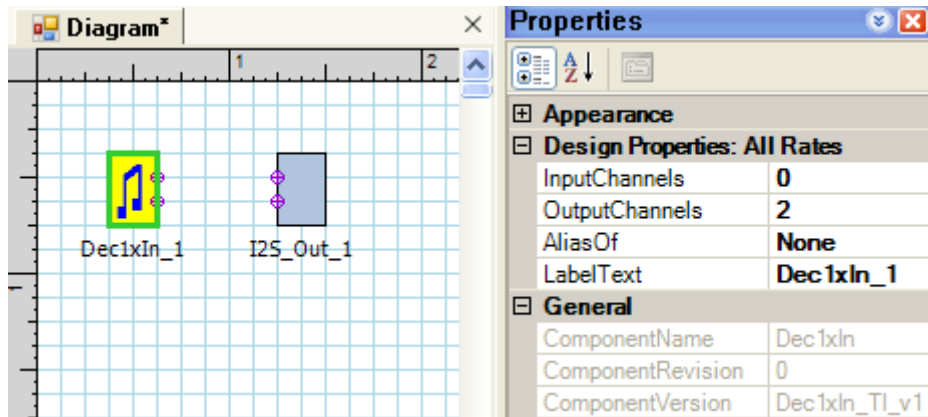
Editing the Link

To remove the line after it's been drawn, click on the line. When the line is selected (designated by square icons at each end), from the **Edit** menu, select **Delete**, or press the <Delete> key.

To move a vertex and modify the routing of a link, select the line and drag the vertex. You can then move the vertex as desired.

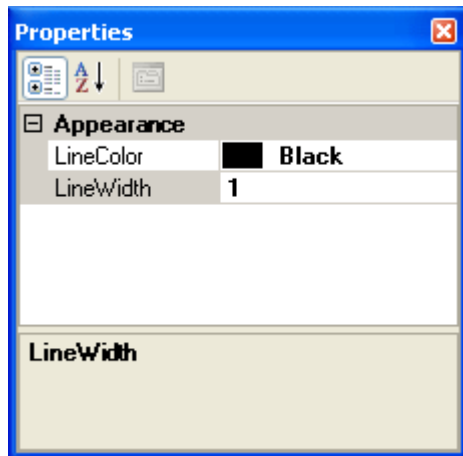
Number of Ports

Components contain the number of port icons specified by the component's **InputChannels** and **OutputChannels** property. In the example below, Dec1xIn_1 contains 2 output port icons and 0 input port icons. Some components have a variable number of channels and ports for input and/or output. If this is the case, the corresponding channels property holds a drop-down list of the supported numbers of channels.



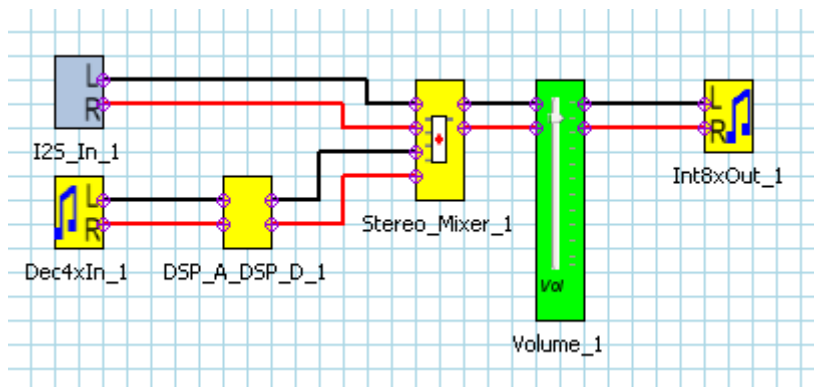
2.12 SETTING CONNECTION APPEARANCE PROPERTIES

The appearance of the connections (wiring) between the components is controlled by the Appearance Property section of the Properties window when a connection is selected. Line color and width can be set to highlight particular aspects of the process flow.



For instance:

- The left and right channels could be shown in different colors.
- The default path could be shown as bold by making the width larger.



2.13 GENERATING APPLICATION CODE

You can use the GDE to generate application code for the process flow

From the **Build** menu, select **Generate Code**.

The following actions will then occur:

- If you haven't already saved the diagram, the Save As dialog box will appear. You must save the diagram in a specified directory before generating the code, because the diagram file and generated code will reside in the same directory. The saved diagram will have the .pfw extension.
- If the current diagram has been modified since it was last saved, it will be saved without requesting confirmation.
- As code is generated, status messages appear in the Output window.
- If no errors are detected, a file named *Config_Rate.cfg*, where **Config** is the current configuration and **Rate** is the current sample rate, will be created in the process flow directory (the directory with the same root name as the .pfw file). If errors are detected, a *Config_Rate.cfg* file will not be created, and any existing *Config_Rate.cfg* file in the process flow directory will have been removed.
- After code is generated, the Resources window appears. This window displays the resources consumed by the current application and the resources still available. These amounts are updated each time the application is generated.

2.14 EXECUTING AND CONTROLLING YOUR APPLICATION

After you've generated code, the GDE allows you to download, execute, and control your application on the board.

- From the **Build** menu, select **Download Code**.

This option downloads the application code (.cfg file) to the target and runs the application. Any status messages will appear in the Output window.

While the code is running the GDE is in Run mode. You can control certain components by manipulating controls within the Diagram window. Properties that cannot be controlled during execution appear in grey.

To disconnect the GDE from the board and return the GDE to Edit mode, do the following:

- From the **Build** menu, select **Disconnect**.
-

2.15 RUN MODE AND EDIT MODE

While you are creating the diagram, the GDE is in Edit mode. The GDE enters Run mode when you select the **Download** command from the Build menu.

When the GDE is in Run mode:

- The diagram in the Diagram window cannot be modified.
- Symbols cannot be deleted, added, or moved.
- Only Rate: All or Rate: *value* properties can be modified in the Properties window.
- The background color of the Diagram window is changed as specified in the Diagram properties.
- On the Build menu, the Disconnect command appears.

To exit Run mode:

From the **Build** menu, select **Disconnect**. The GDE then returns to Edit mode.

Adaptive Mode

Each PurePath Studio (Portable Audio) framework component has two properties miniDSP_A_Adaptive and miniDSP_D_Adaptive that each have two settings, Enabled and Disabled.

When Enabled, the following actions are made:

- Code is generated in the .regs[] section of the framework to place each processor mADC or mDAC into adaptive mode when the processor is started.
- Code is assembled with the /miniDSP_A_Adaptive or /miniDSP_D_Adaptive switch as needed. These switches to the assembler cause the coefficient areas for both parts of adaptive memory to be initialized.
- During Run mode, the following occurs:
 - The title bar specifies (Run Mode – Adaptive)
 - The GDE allows runtime properties to be modified in run mode.
 - The Write State menu button and toolbar button are enabled.
 - Changes are immediately written to the coefficient bank that is connected to the control interface. The Build/Write State menu item and toolbar button swap coefficient banks to effect any current changes.

When AdaptiveMode is Disabled, the following actions are made.

- Run mode does not allow properties to be modified in the GDE. The Write State menu item and toolbar button are not enabled.

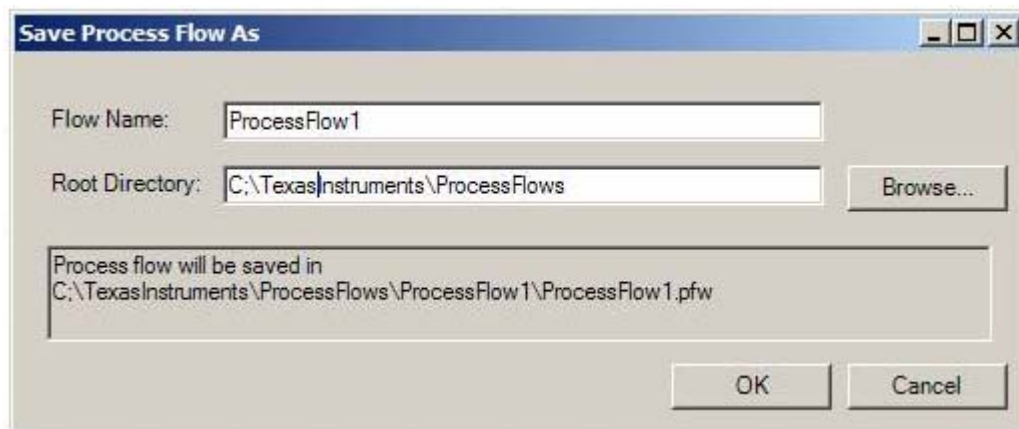
2.16 SAVING THE DIAGRAM

The diagram (process flow) must be saved to a specified directory before you can generate code, because the GDE will place generated code in the same directory. If you try to generate code and haven't manually saved the diagram, the GDE will force you to save it by opening the Save Process Flow As dialog. The diagram is saved as a .pfw file.

There can only be one .pfw file per directory.

To save the diagram:

1. From the **File** menu, select **Save**. The *Save Process Flow As* dialog appears.
2. In the Save Process Flow As dialog, in the Flow Name field, specify the file name.
3. In the Root Directory field, specify the directory. When the diagram is saved, a subdirectory is automatically created. The name of the subdirectory is the name specified in the Flow Name field.
4. Click **OK**.

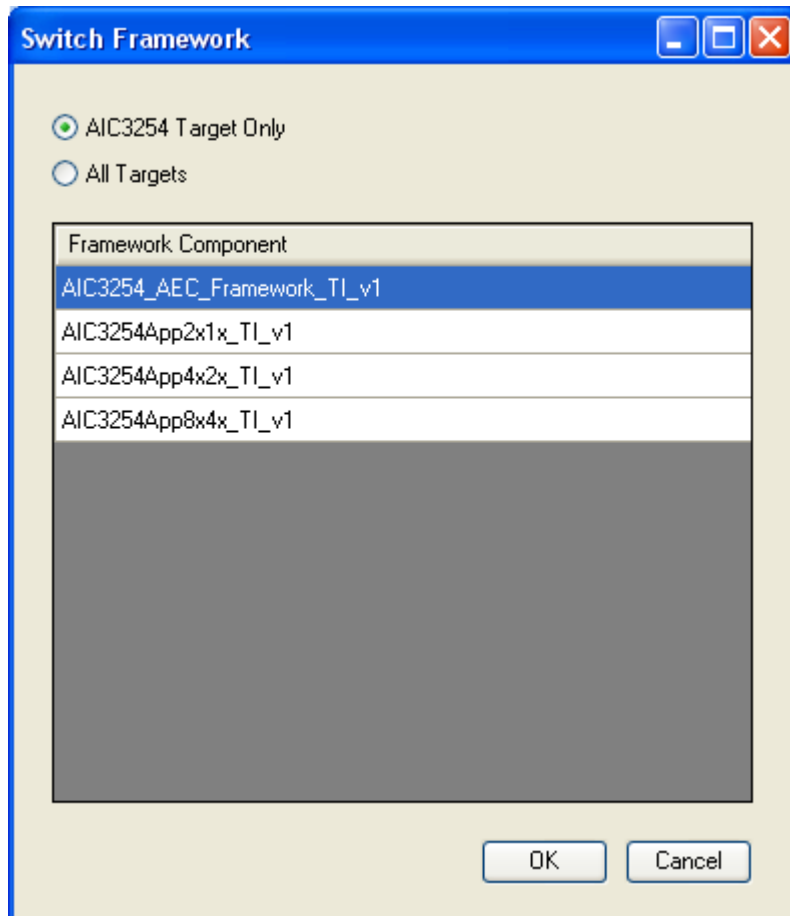


2.17 FRAMEWORK SWITCHING

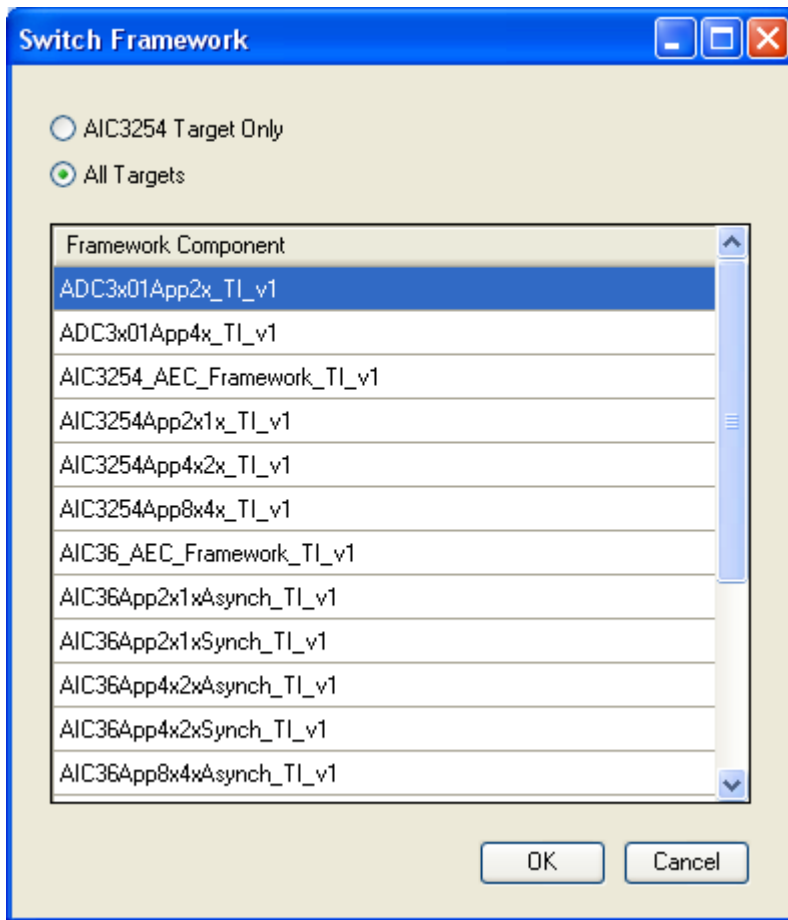
The Switch Framework dialog allows you to switch to a different framework than what was originally selected in the process flow in the GDE.

To open the Switch Framework dialog:

From the **Tools** menu, select **Switch Framework**.



Framework choices may be viewed for the current codec only (as above) or for all codec targets (as below).



The Switch Framework dialog contains the following:

<Target> Target Only	Select to display the frameworks within the same target group.
All Targets	Select to display all available frameworks.
Framework Component List	A list of frameworks are shown depending on the option chosen. Click once to select and highlight the desired framework.
OK	Select the OK button to switch the framework in the process flow on the GDE diagram to the one chosen here. A dialog box will be displayed asking the user to verify the change.
Cancel	Select the Cancel button to exit the dialog without making any changes to the process flow.

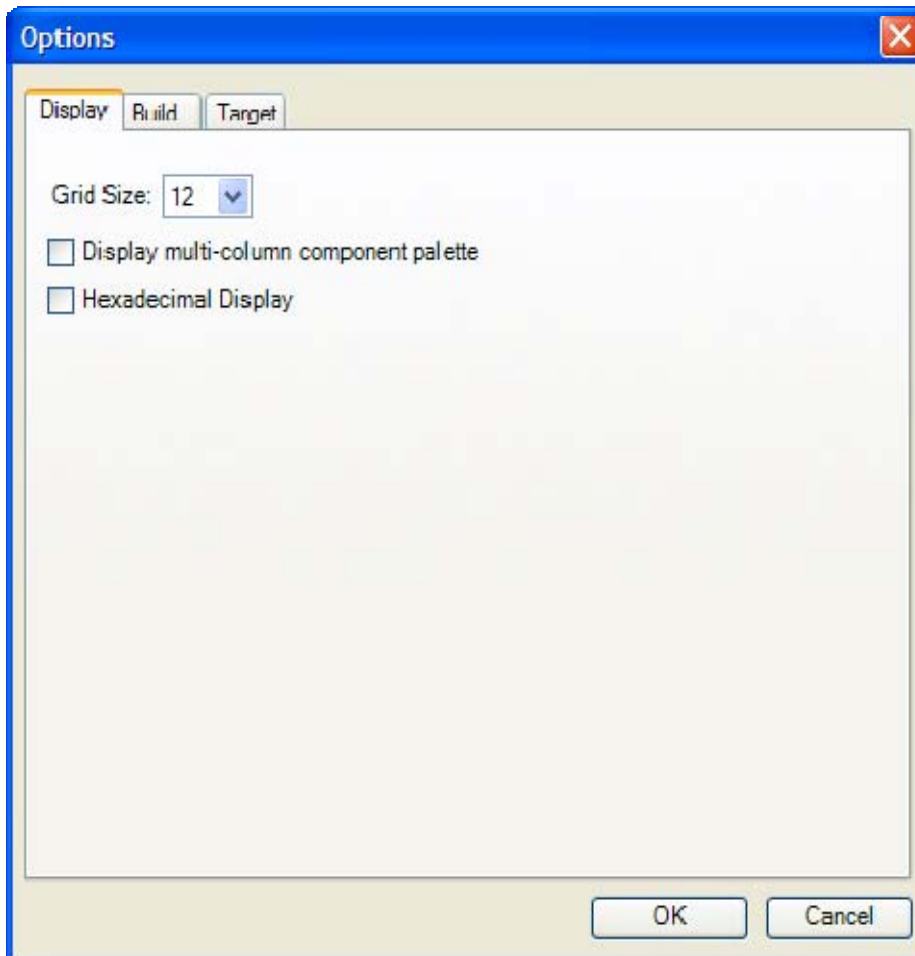
Once a diagram has been opened and a framework has been selected, the menu item will be enabled, allowing the user to switch to a different framework. Switching to a new framework may cause components and/or links in the process flow to be removed. Verify the accuracy of the process flow after switching to a different framework.

2.18 TOOLS OPTIONS

The Tools Options dialog allows you to configure various aspects of the GDE.

To open the Tools Options dialog:

From the **Tools** menu, select **Options**.

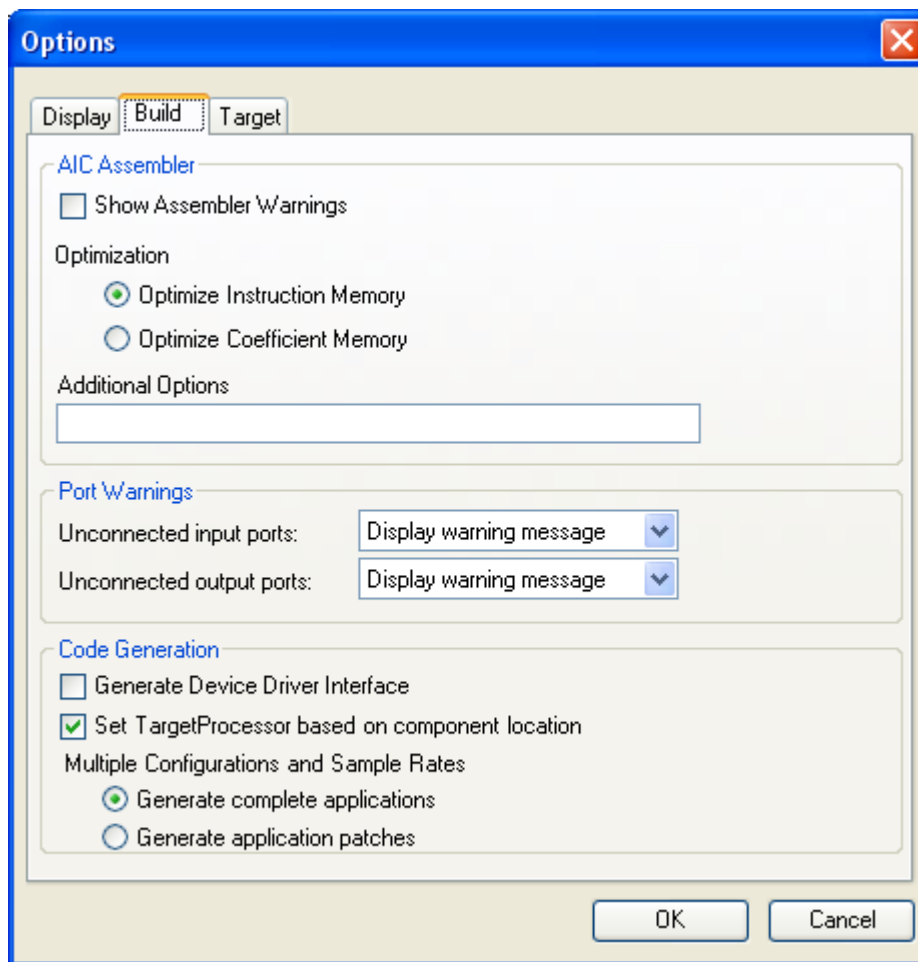


The **Display** tab contains the following:

Grid Size	Select the size of the grid that appears in the Diagram window. Choices are 4, 6 and 12 pixels, with 12 pixels as the default.
Display multi-column component palette	Display the component icons (within each component heading) in multiple columns instead of just one column.
Hexadecimal Display	If checked, integer and fixed point values in the Property window are displayed in hexadecimal instead of decimal. Regardless of this setting, new values may be entered in the Property window as either decimal, or hexadecimal using a 0x prefix.

An example of multi-column component palette is shown below:

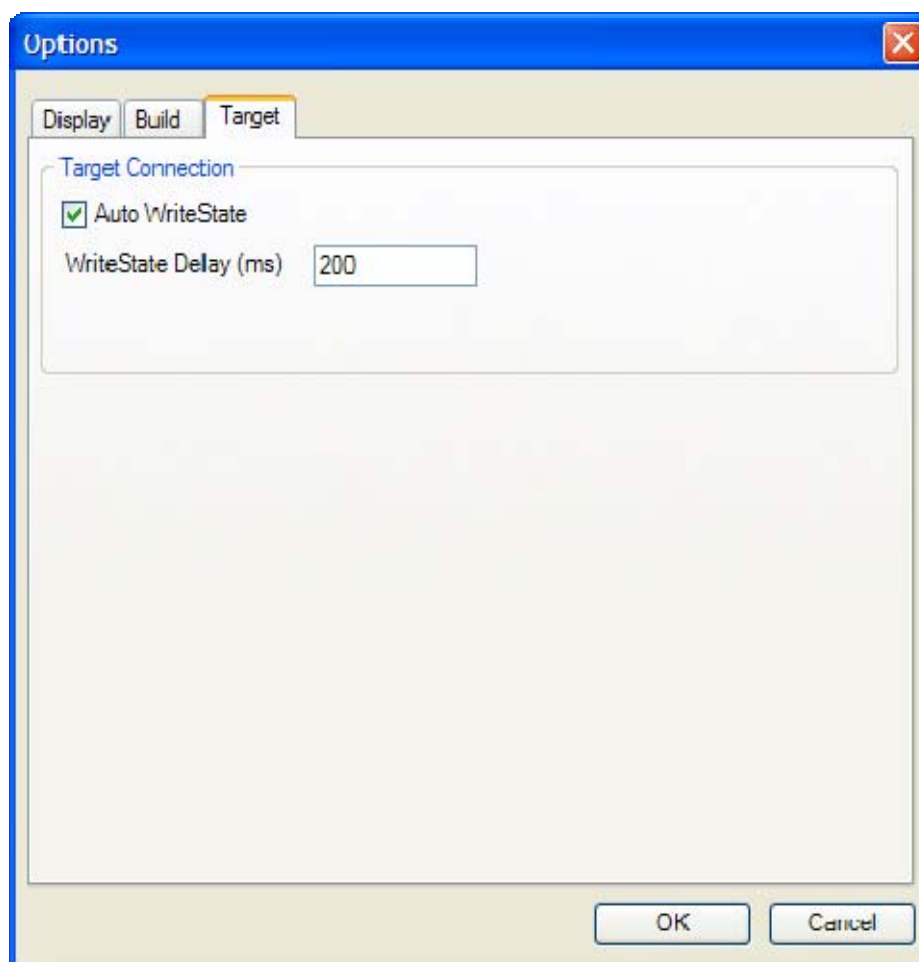
Default component listing (single column):	Listing with multiple Columns:
<div><div>Basic DSP</div><div><div>CC</div><div>AddCoef</div><div><div>X+C</div><div>AddCtoD</div><div><div>X+Y</div><div>AddData</div><div><div></div><div>A-law Decoder</div><div><div></div></div></div></div></div></div></div>	<div><div>Basic DSP</div><div><div>CC</div><div>X+C</div><div>X+Y</div><div><div></div></div><div><div>X-Y</div><div>C²</div><div>CC</div><div>C</div><div>C→</div><div><div></div></div><div><div>→C</div><div><div></div></div><div>CC</div><div>X-Y</div><div>(-)</div><div> X ·C</div><div>*C</div><div>*C</div><div>SGN×</div></div></div></div></div>



The **Build** tab contains the following:

Show Assembler Warnings	Specifies if assembler warnings should be displayed.
Optimization	Specifies whether to Optimize Instruction Memory or Coefficient Memory - Optimize Instruction Memory - Gives the /noploops switch to the assembler <default> - Optimize Coefficient Memory - Does not give the /noploops switch to the assembler
Additional Options	Additional options to send to the assembler.
Unconnected input ports	Specifies whether an unconnected input port should generate a warning message, an error message, or be ignored.
Unconnected output ports	Specifies whether an unconnected output port should generate a warning message, an error message, or be ignored.
Generate Device Driver Interface	Specifies if the device driver interface file should be generated.
Set TargetProcessor property	Specifies if the TargetProcessor property should be set based on the location of the component in the process flow.

Generate complete applications	Generate a complete application for each configuration and sample rate.
Generate application patches	Generate a complete application for the current configuration and sample rate only, and patches for the remaining configurations and sample rates .



The **Target** tab contains the following:

Auto WriteState	Specifies if the AutoWrite State capability is enabled.
WriteState Delay	Specifies the number of milliseconds that the writestate capability waits for additional target writes.



2.19 PRINTING THE DIAGRAM

To print the diagram:

1. From the **File** menu, select **Print**.
2. In the **Print** dialog, specify the desired print properties.
3. Click **OK**.


To view a preview of the printed page:

1. From the **File** menu, select **Print Preview**.
2. To close the Print Preview window, click **Close**.

2.20 FLOATING AND DOCKING WINDOWS

There are several ways to float and dock windows within the GDE.

Context Menu

Click on a double-arrow icon  on a window's title bar to view a context menu of floating and docking options.

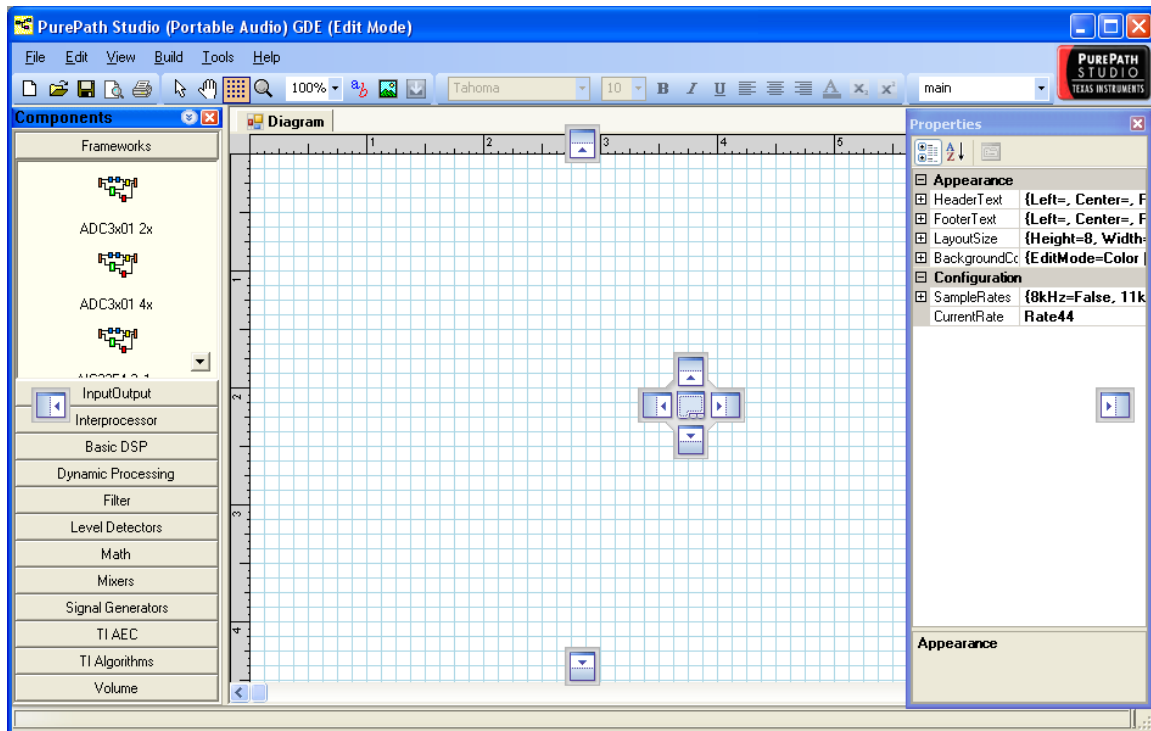
Floating

To float a window, select Floating from the context menu.

To dock the window again, right-click on the window's title bar and select Dockable from the context menu. Then right-click again and select the desired Dock To option from the context menu.

Docking

To move a window and dock it, grab the window's title bar with the mouse and drag it. Docking controls will be displayed, which allow you to dock on any of the four sides of the GDE, as a tab of any other window within the GDE, or as a tab on the main window. Windows can be stacked horizontally or vertically when docking.



Windows can be docked together as a tabbed window. For example, you can click on the title bar of the Properties window and drag it on top of the Components window. When you release the mouse, the Properties window will be on top of the Components window with tabs for each window at the bottom of the screen.

Hiding Windows



A window can be hidden by clicking the push-pin icon on its title bar. A tab for the window will appear on one side of the main window. If you hover over the tab with the cursor, the window will appear temporarily. You can restore the window at this point by clicking again on its push-pin.





2.21 COMPONENT LIBRARY

The components used by the GDE are stored in a Component Library. Components are packaged in .zip files.

System Component Library

The system component library resides in the GDE installation directory in a subdirectory named ComponentLibrary. This subdirectory holds the components that are shipped with the GDE, as well as third-party components that may be added later to the GDE.

User Component Library

The user component library resides in the GDE installation directory in a subdirectory named UserComponentLibrary. If a component exists in both ComponentLibrary and UserComponentLibrary, UserComponentLibrary takes priority, and the component will be used from this location.

The Component Cache

Before using components, the GDE temporarily unpacks components from the System Component Library and the User Component Library into a Component Cache. When the GDE is invoked, the Component Cache will be updated with any component having a .zip file in a component library that is newer than the component currently in the Component Cache.

Clearing the Component Cache:

The Component Cache may be cleared by the user by selecting the "Clear GDE cache and state" item from the PurePathStudio / Utilities start menu. This clears the GDE cache and causes the GDE to completely unpack the cache the next time the GDE is run.

2.22 I²C LOGGING

The I²C Logging Tool allows you to log I²C writes that are written to a board that is connected.

From the **Tools** menu, select **I²C Logging** to turn it on. To turn it off, simply uncheck it.

When this option is selected, a log file will be created in the process flow folder with the same name as the process flow. If a known component is related to the write, the log will record the component's name and description along with the values that will be written to the component's address. If the component is unknown, the name and description for the log will be //Unknown. Only one process flow may have I²C Logging on at a time.

Log Format and Example

Format:

//Component name: Component Description
Address, Subaddress, Value(s) written

Example:

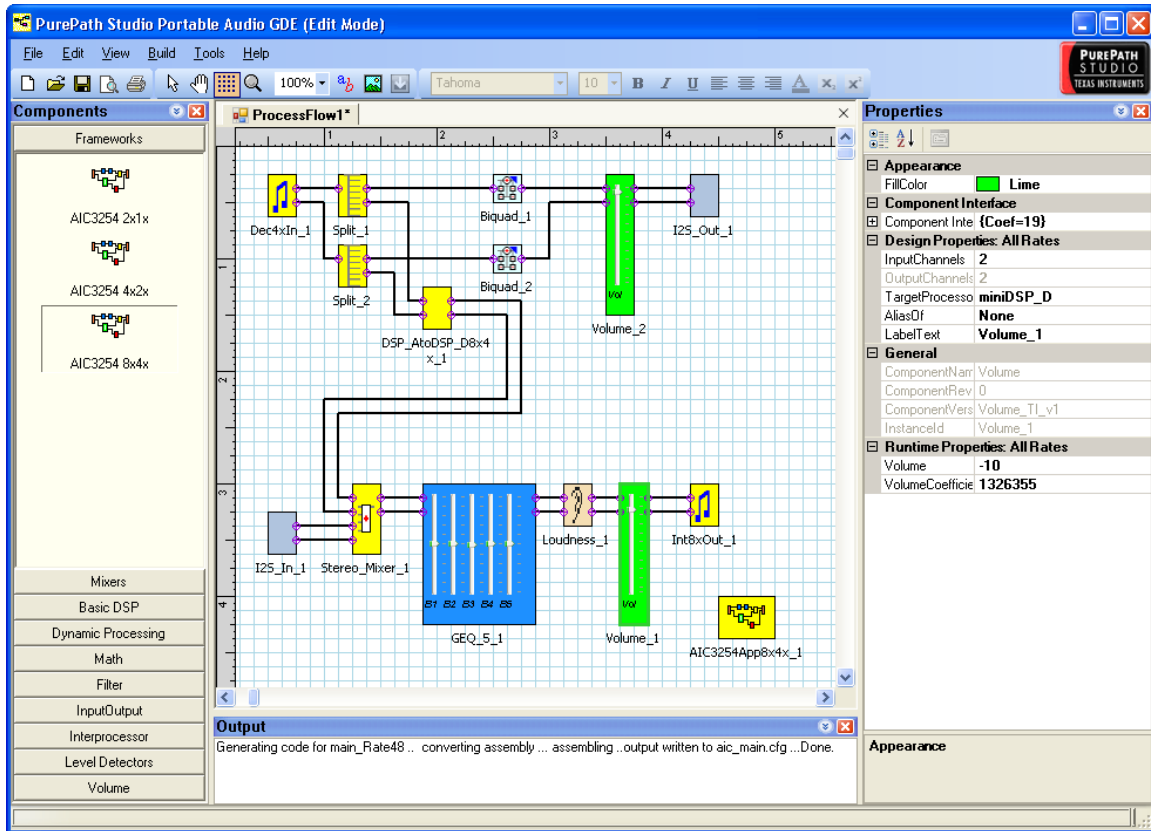
//ToneFast_1 : Bass and Treble
0X68, 0X21, 0X0, 0X0, 0XFF, 0X1

3. DETAILED SPECIFICATIONS

The individual components of the GDE toolset are described below.




3.1 GDE MAIN WINDOW



The GDE GUI is a MS Windows application with a menu bar, toolbar and child windows. Each component of the GDE GUI is described in below.



3.1.1 Menu Bar

The menu bar for the GDE is described below:

File Menu		
N ew	Ctrl+N	Create a new GDE document. The equivalent toolbar button is  .
O pen...	Ctrl+O	Open an existing GDE document (.pfw). The equivalent toolbar button is  .
L oad Property Settings		Open a property settings file (.set) and load the property values.
C lose		Close the current GDE document
S ave	Ctrl+S	Save the current GDE document. The equivalent toolbar button is  .
S ave A s...		Save the current GDE document to a different name.

File Menu		
Save Property Settings		Save the runtime property values to a property settings file (.set).
Page Setup...		Display the Page Setup dialog to define the size of the diagram.
Print Preview		Display the Print Preview window to preview a page before printing. The equivalent toolbar button is  .
Print...	Ctrl+P	Display the Print dialog to print the diagram. The equivalent toolbar button is  .
Recent Files		Display the files most recently accessed. Selecting a file will open the file in the GDE.
Exit		Exit the GDE. If the current document has not been saved, the Save As dialog will appear.

Edit Menu		
Undo <action>	Ctrl+Z	Undo the last action
Redo <action>	Ctrl+Y	Redo the last action that was undone
Select All	Ctrl+A	Select all of the objects in the current process flow
Cut	Ctrl+X	Cut the selected item(s) from the process flow diagram and place the item(s) in the clipboard
Copy	Ctrl+C	Copy the selected item(s) from the process flow diagram and place the item(s) in the clipboard
Paste	Ctrl+V	Paste the item(s) in the clipboard to the process flow diagram
Delete	Del	Delete the selected item(s) from the process flow diagram

View Menu	TAS3x0x target
Components Window	Show or hide the Components Window (see Section 3.2.1)
Properties Window	Show or hide the Property Window (see Section 3.2.8)
Source Code	Show the DSP source code listing (Developer Version) or the exported coefficients listing (Standard Version) for the active diagram (see Section 3.2.14)
Pan Zoom Window	Show or hide the Pan Zoom Window (see Section 3.2.12)
Output Window	Show or hide the Output Window (see Section 3.2.11)
Resource Window	Show or hide the Resource Window (see Section 3.2.13)
Ruler	Show or hide the ruler on the Diagram Windows

Build Menu	miniDSP target
Generate Code	Generate code for the current diagram
Download Code	Download code to the EVM
Write State	Writes the current process flow state to the target's shadow memory and swaps to that memory.






Tools Menu	
Component Interface Overview	Display the Component Interface Overview Window (see Section 3.2.16)
I ² C Command Tool...	Display the I ² C Command Tool (see Section 3.7.1)
I ² C Memory Tool...	Display the Memory Tool (see Tools functional specification)
Configuration Editor	Display the Configuration Editor Window (see Section 3.2.17)
I ² C Logging	Toggle on/off the I ² C logging support
Switch Framework...	Allows you to switch to a different framework than what was originally selected
Options	Display the Options Dialog for configuring the GDE (see Section 3.9)









Help Menu	
-----------	--




Help Menu		
Contents	F1	Show the Help Contents list (see Section 3.10)
Release Notes...		Show the Release Notes for this version
Tip of the Day		Show the Tip of the Day dialog
About		Show the About box




3.1.2 Toolbar

The following Toolbars are available:

Standard Toolbar		
New		See File/New
Open		See File/Open
Save		See File/Save
Print Preview		See File/Print Preview
Print		See File/Print

View/ Action Toolbar		
Pointer Cursor		Change cursor to the 'Pointer' which allows the user to select components on the diagram.
Hand Cursor		Change cursor to the 'Hand' which allows the user to grab the diagram and scroll it by panning.
Grid		Turn on/off the grid display
Zoom Cursor		Change cursor to the 'Zoom' magnifier cursor. Left click with this cursor zooms in, right click zooms out.
Zoom Percent		Choose the zoom percentage from a list of preset values
Rich Text Cursor		Change cursor to a text box creator. Click and mark a region to contain a rich text box for comments on the diagram
Image Insert		Insert an image into the diagram.
Write State		Writes the current process flow state to the target's shadow memory and swaps to that memory.

Text Formatting Toolbar		
Font Family		Choose the font family for a rich text box
Font Size		Choose the font size for a rich text box
Bold, Italic, Underline		Set the font properties of the selected text in the rich text box

Text Formatting Toolbar		
Alignment		Set the alignment of text in the rich text box
Font Color		Set the font color of the selected text in the rich text box
Subscript, Superscript		Set the subscript or superscript properties of text in the rich text box

3.1.3 Command Line Switches

The GDE accepts the following command line switches. Switches are case insensitive

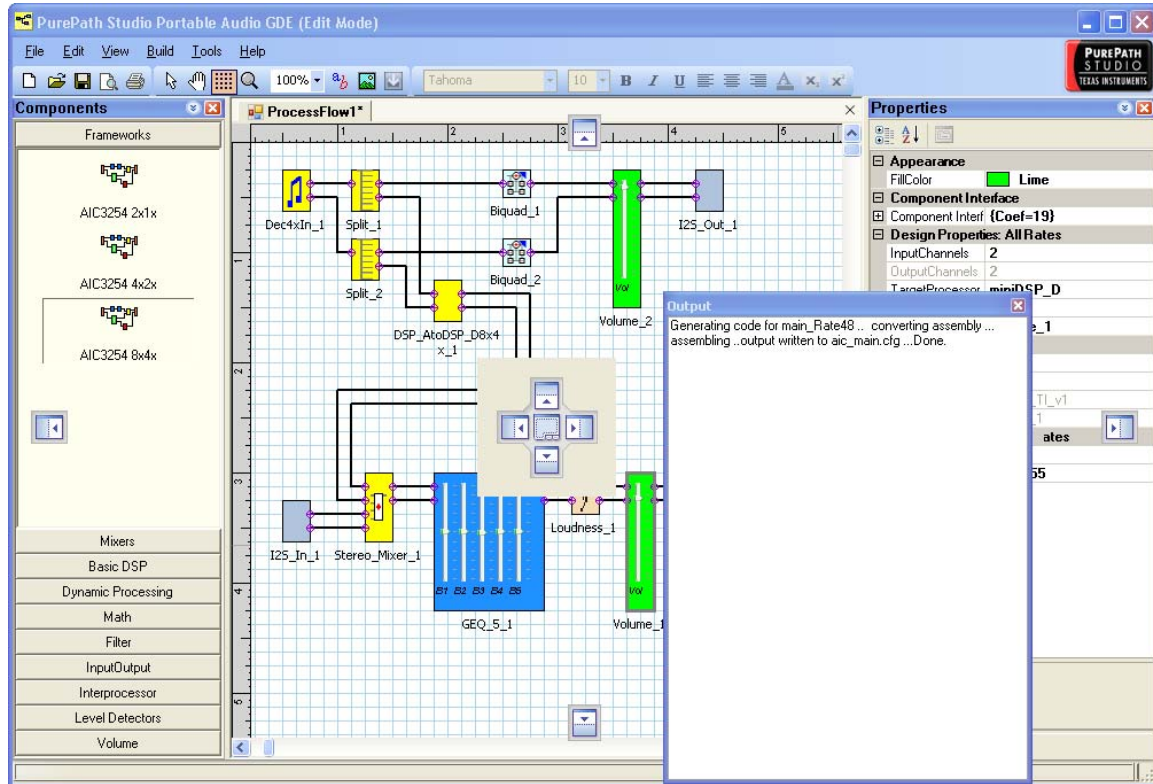
```
GDE.exe [/input=file.pfw] [/build] [/notips] [/testmode] [/i2clog=file.log]
        [/ppsasm] [/driver] [/patches]
```

Switch	Description
/input=file.pfw	If given, the GDE loads the given process flow file
/build	If given, the GDE does not show the user interface. The /input switch must also be given. The GDE builds the process flow file and exits.
/notips	If given, the GDE does not show the tool tips dialog, irrespective of what the settings for showing the tooltips currently are in the currently persisted GDE settings.
/testmode	If given, the GDE sets the target interface to 'test mode'. That is, the target interface behaves as if it were connected to a board, when it is not. Not included in user documentation.
/i2clog=file.log	If given, enables I2C logging and sets the I2C log file to the given file name. If the file name is a simple file name or relative path, it is relative to the current working directory of the GDE.
/ppsasm	If given, the GDE uses the PurePath Studio assembler, irrespective of what assembler is chosen in the currently persisted GDE settings. Not included in user documentation.
/driver	If given, the GDE will generate pps_driver.h files, irrespective of what option is chosen in the currently persisted GDE settings. Not included in user documentation.
/patches	If given, the GDE will generate patch files instead of complete applications for other configuration and rates, irrespective of what option is chosen in the currently persisted GDE settings. Not included in user documentation.

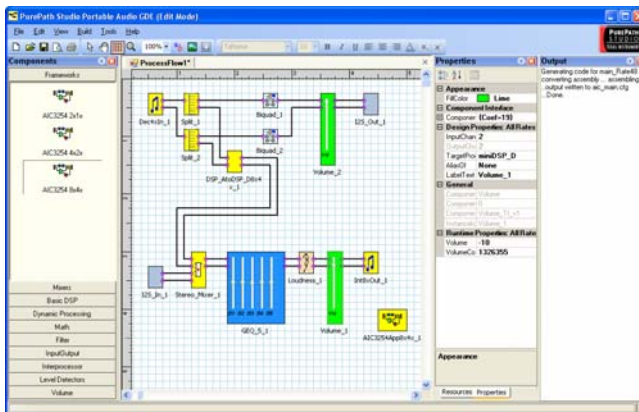
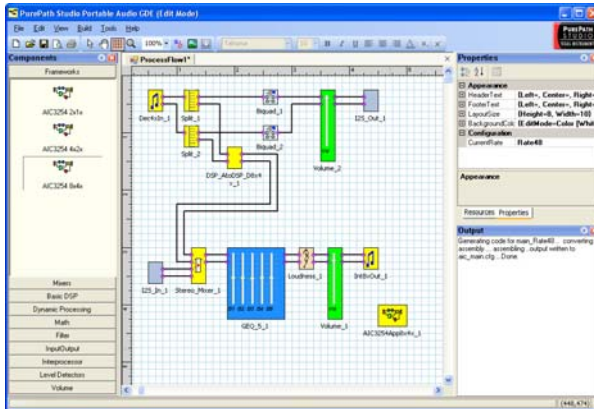
3.1.4 Child Window Docking and Floating

Each of the child windows in the GDE (Components, Properties, Resources, and Output) may be docked to the main window, or floating on the user's desktop.

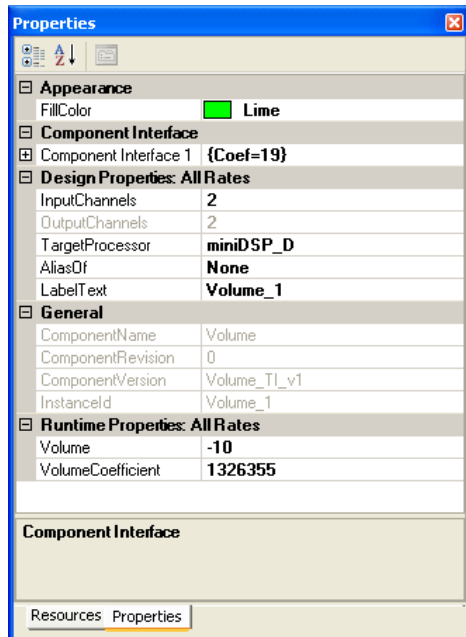
Docking is achieved by grabbing the window title with the mouse and dragging. A standard .Net 2.0 *docking control* is shown to the user, allowing docking on any of the four sides, as a tab of any other child window, or as a tab on the main window.



Windows may be stacked horizontally or vertically when docking



Child windows may be docked together as a tabbed window, and floated as one window.

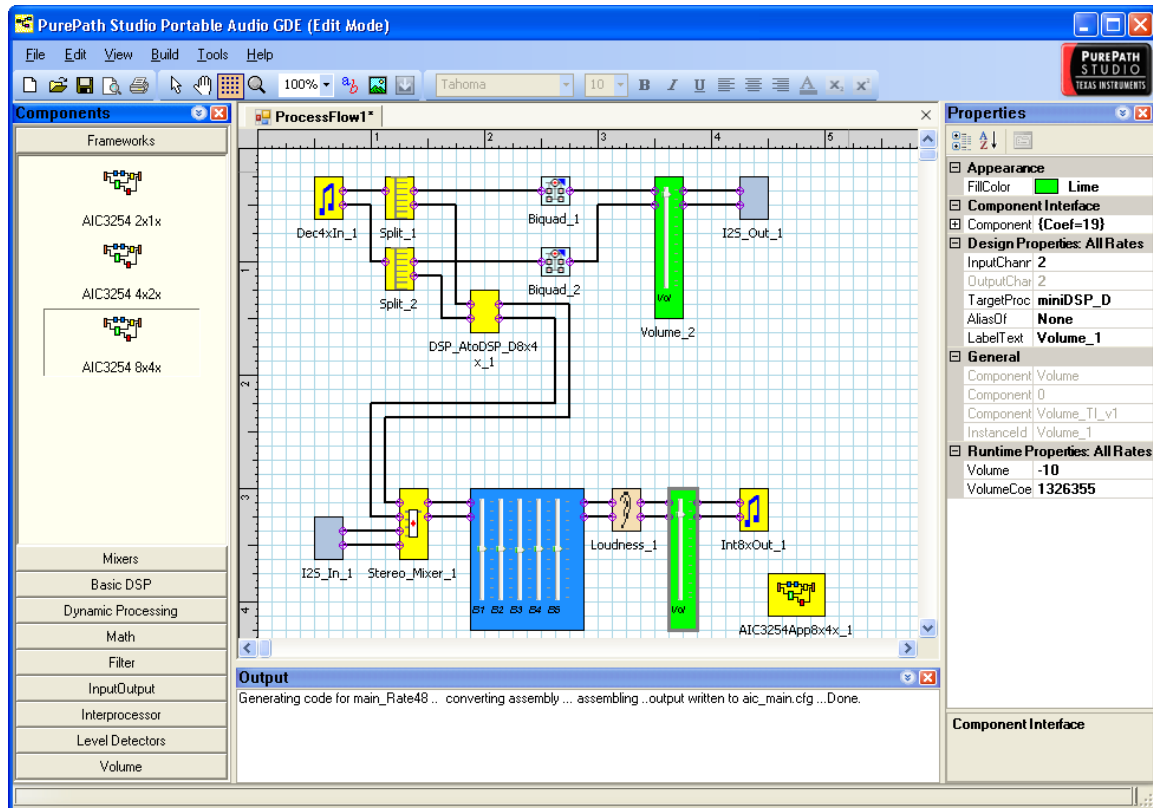


3.2 PROCESS FLOW EDITOR

The GDE application is primarily an editor for *process flow diagrams*. A process flow diagram represents audio components, and the connections between them.

The three major portions of the Process Flow Diagram are shown below:

- Components Window
- Diagram Windows
- Property Window



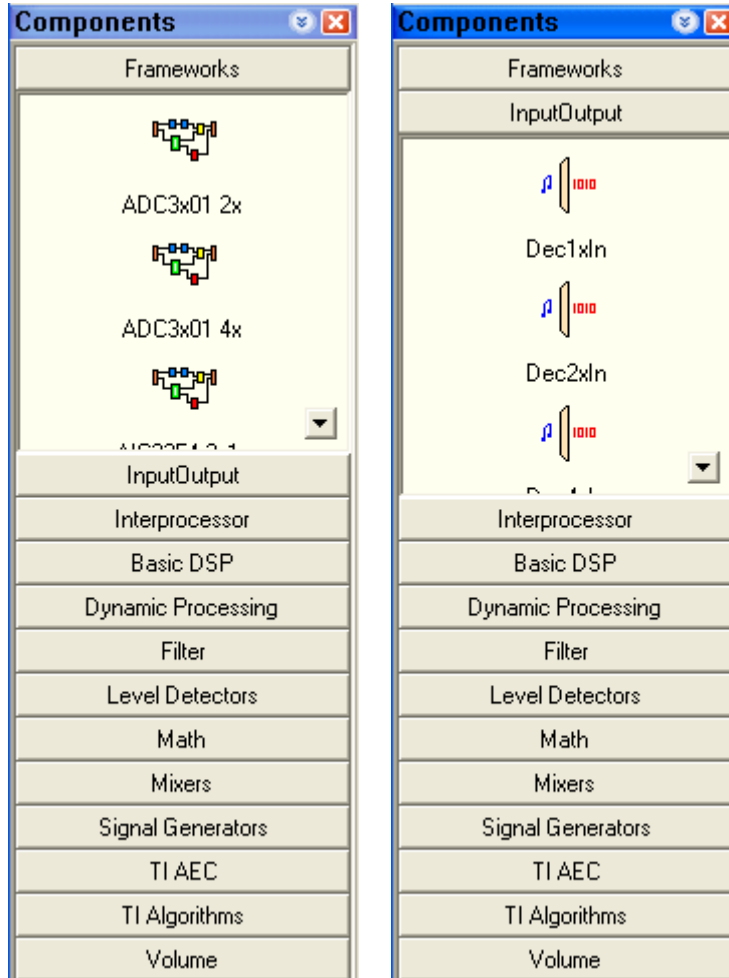
Components are dragged/dropped from the Component Palette to a Diagram Window. The components are then connected with *links* in the diagram window. *Component properties* may be examined and modified in the Property Window.

3.2.1 Components Window

The Components Window displays all of the components that are currently available in the Component Library.

The Components Window may be viewed with either large icons, one per row, with text labels, or small icons, many per row, and text labels via pop-ups.

See the Display tab of the Options Dialog (Section 3.9.1) for details.



Components are organized into palettes in the Components Window. The specific palette for a component is defined when the component is created.

The current GDE palette contains all the components available to the GDE. A future GDE release will be able to dynamically prune the palette contents based on the framework that is chosen by the user. For example, if an AIC3254 framework is chosen, only the components that are usable on an AIC3254 will be displayed. The others will be hidden. This feature is particularly useful for the Input/Output components, which are target-dependent.

Components are chosen from the Components window and placed on the Diagram window by a mouse drag/drop action.

Components have two representations:

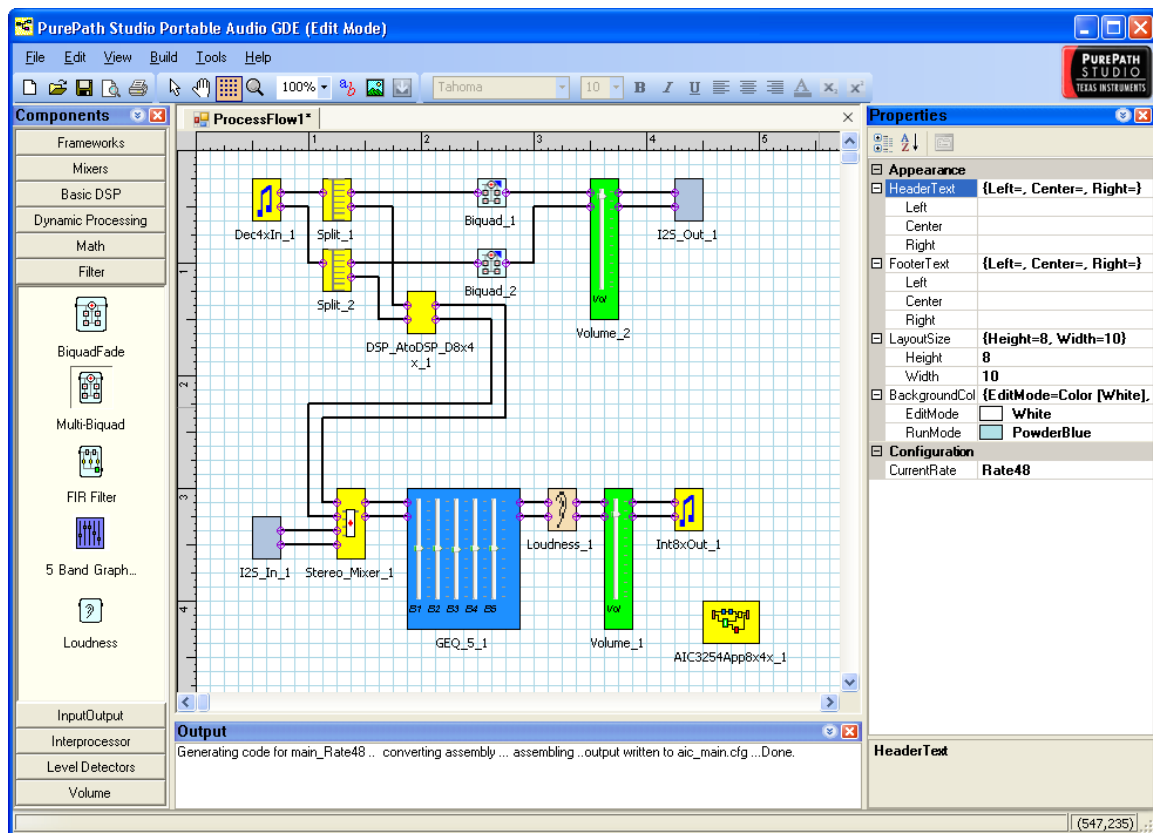
- The **icon** is the representation of the component shown in the Components Window palette.
- The **symbol** is the representation of the component on the Diagram Window. The symbol includes either:
 - A **graphic** that visually represents the component
 - A **control** such as a slider, spin control or checkbox that both visually represents the component and provides control of properties

At this time, component symbols are restricted to be rectangular in shape. The placement of ports on the component are restricted to the left side and right side, starting at the top of the component, except for control (feedback) ports, which are along the bottom. Future versions of the GDE will support various shapes of component symbols.

The appearance of components are currently static during the execution of the GDE. Future versions of the GDE will support the changing of the component graphic based on the values of component properties. This will allow the component developer to visually represent the state of the component (such as 'enabled' versus 'disabled')."

3.2.2 Diagram Window

The Diagram Window displays the process flow diagram. Components are dragged/dropped onto the process flow diagram window, or process flows may be loaded from disk and displayed on the diagram window.





The diagram window has the following properties which may be set in the Property Window when the diagram window background is selected.

- **Layout Size** – The X and Y size of the chart, in inches
- **Footer Text** – Text to display on the footer of printed diagrams
- **Background Colors** – Background colors for both edit and run mode
- **Header Text** – Text to display on the header of printed diagrams
- **Current Rate** – The current sample rate in use in the diagram. This is a drop-down list of all the supported sample rates in the GDE. Only one sample rate can be used at a time in the GDE at this time. This value is read-only during run mode.

When the GDE is first started, an 'Untitled' diagram window is displayed. If the user edits this diagram, they are prompted to save it with a name when exiting the GDE.

3.2.3 Dragging Components to the Diagram

As components are dragged from the palettes on the Components Window to the diagram, they are automatically 'snapped' to the grid of the Diagram Window. There is no option to turn off the snap-to-grid, which is necessary to make the links between components draw neatly.

Once a component is dragged to the diagram, it remains selected in the diagram and the properties for the component may be modified.

Components may not be dragged between diagrams; however, they can be moved with Cut/Copy and Paste.

3.2.4 Component Help files

Each component has a help file that describes the component operation and its configuration options. The help file is accessed by right clicking on the component and selecting help.

3.2.5 Connecting Components with Links

Most components have 'ports' located on them which may be linked to other components. Ports are either input, output, or control. By convention, input ports are on the left side of a component, output ports on the right side, and control ports are along the bottom.

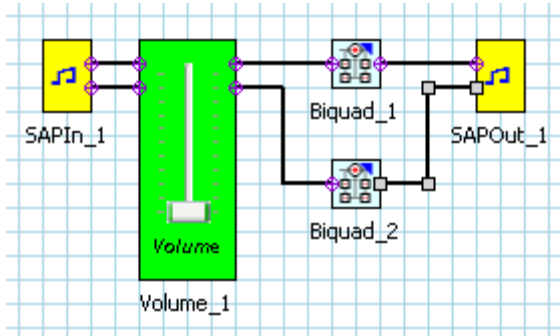
3.2.5.1 Creating Links

Connecting components is achieved by drawing 'links' between them. A link is drawn by the following sequence of actions

- Click on a port. The mouse cursor turns to a crosshair, and a dotted line shows the link being created.
- Moving to another port. The mouse cursor changes to a port symbol, noting that a link may be made. Clicking completes the link.

3.2.5.2 Link Routing

Links are straight by default. Links may be routed by the user during the creation of the link by clicking on any interim point where the user wishes the link to have a vertex. The example below shows links that have been routed.



Note: The original GDE specification discussed automatic routing of links by the GDE. This feature was found to be of little value and was discarded in favor of the Link Routing described in Section 3.2.5.2.

3.2.5.3 *Signal Feedback*

The GDE supports the creation of feedback loops thru the following mechanisms:

- Components can define 'control' ports that accept input. A control port, displayed along the bottom of a component allows specialized control input to be fed into a component.

3.2.6 **Moving Components and Links**

Components that have links attached to them may be moved, and the links will move with the components.

Links that have multiple vertices may be moved at each vertex, to modify the routing of the link. Hovering over a vertex will turn the mouse pointer to a crosshair. The vertex can then be moved with the mouse.

As discussed above, there is no automatic re-routing of links as components are moved. The links are kept connected, but the segment connected to the component is the only segment stretched.

3.2.7 **Component Cut/Copy/Paste**

Components and links may be copied and pasted on a diagram or between diagrams. Multiple components and links may be selected in two ways:

- By holding down the Ctrl key and selecting components and links
- By drawing a 'box' around a set of components and links on the diagram.
-

Links cannot be copied and pasted by themselves; however, links are copied and pasted along with the components that they are linking.

When components and links are pasted, the newly pasted items remain selected, to permit easy moving of the pasted items.

3.2.8 Framework Components

Components that are framework components have special rules that relate to them.

- A framework must be the first component in a framework. Dragging/dropping any other component before a framework is inserted will give an error.
- The framework in a diagram cannot be deleted unless all the other components are deleted.
- A framework component cannot be cut, copied or pasted.

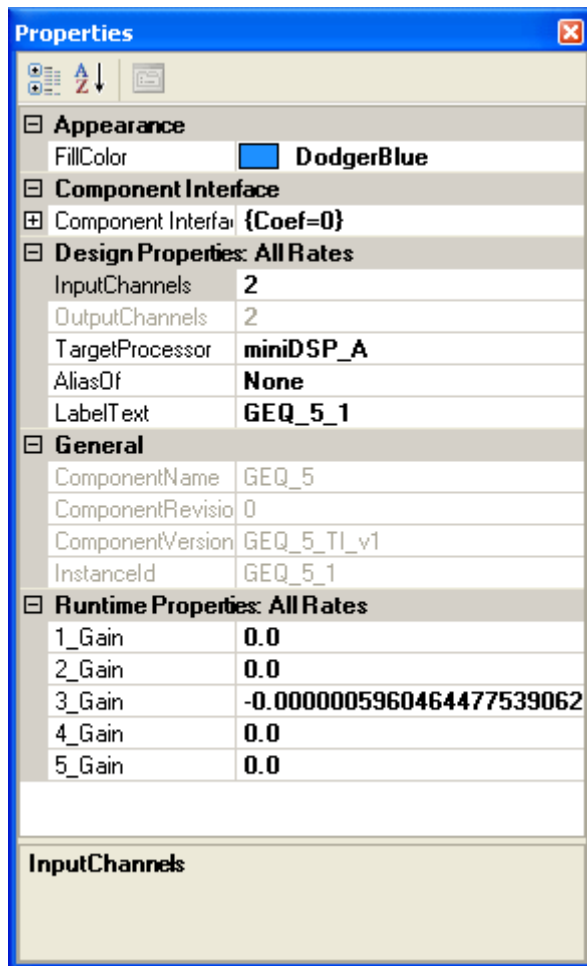
Generally, a framework component represents either a specific hardware device, such as AIC3254, or a closely related hardware family.

3.2.9 Property Window

The Property Window supports the editing of attributes for the selected component. When no component on the diagram is selected, the window displays the properties for the diagram as a whole.

Properties that are read-only are shown in grey and may not be edited. Some properties are always read-only and others are writable during edit mode and read-only during run mode.

Properties that are specific to a particular sample rate are shown in the Property Window grouped by sample rate.



3.2.10 Property Settings

Runtime properties can be loaded or saved using a property settings file (.set). The property settings file contains one or more command lines with the following format:

SetProperty [component_name] [property_name] [value]

1. component_name – The name of the component. The component name is the value of the InstancedId property.
2. property_name – The name of a property that belongs to the specified component.
3. value – The new value for the property.

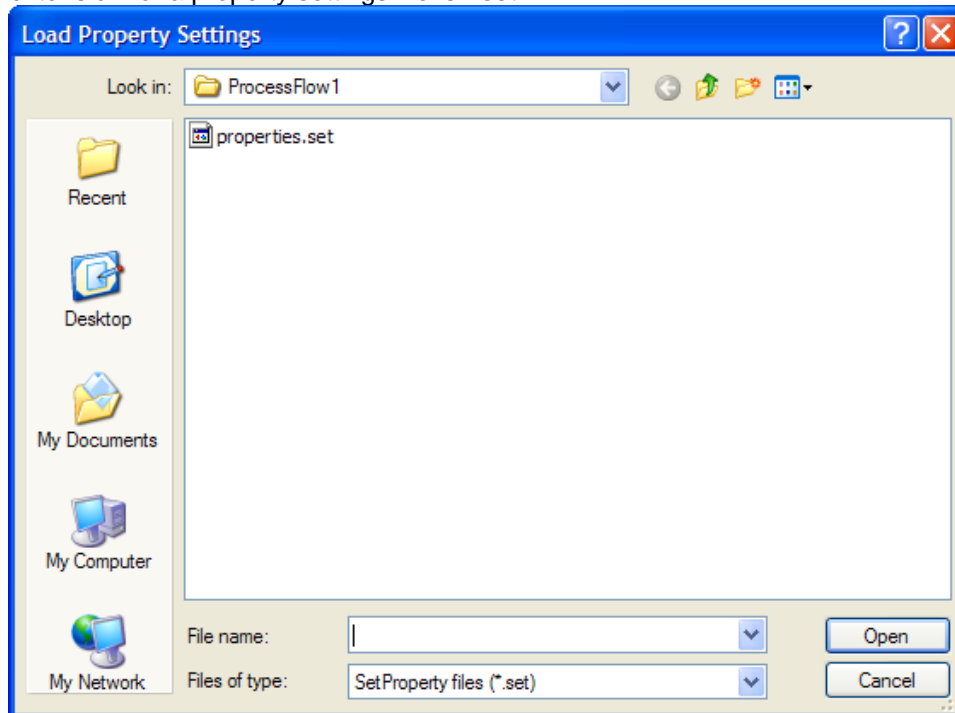
The command name (SetProperty) is case-insensitive. The following are examples of valid command lines:

```
Setproperty Mixer_1 Ch1_Gain 0.75
setproperty Mixer_1 Ch1_Gain 0x00800000
SetProperty StereoMux_1 MuxSelect StereoCh1
```

3.2.10.1 Loading the Property Settings

The property setting can be loaded by selecting the File->Load Property Settings menu item.

This will open a Load Property Settings dialog to select the file you wish to load. The default extension for a property settings file is “.set”.



The GDE will parse each command line and set the property value. Any error information will be displayed in the Output Window.

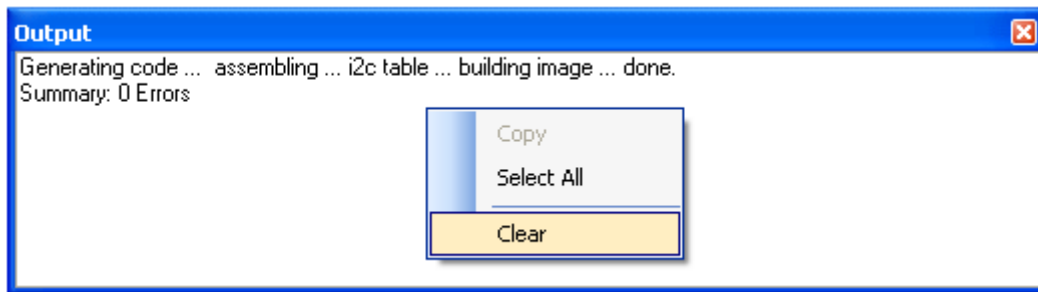
3.2.10.2 Saving the Property Settings

The property settings can be saved by selecting File->Save Property Settings. This will open a “Save Property Settings” dialog to specify a file. All of the runtime properties that belong to the current process flow will be saved to the file in the proper format.

This will display the “Save Property Settings” dialog.

3.2.11 Output Window

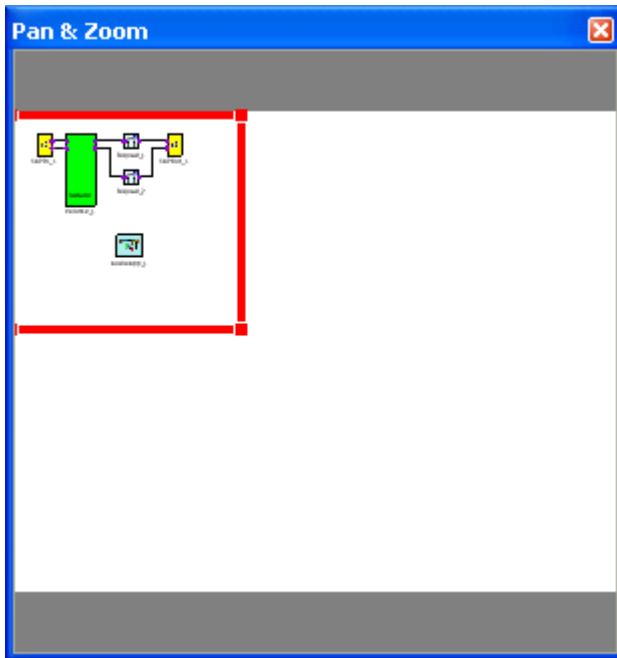
The Output Window will display any error, warning or informational messages that occur during the operation of the GDE.



The Output Window can be cleared by right-clicking in the Output Window and selecting clear.

3.2.12 Pan / Zoom Window

The Pan / Zoom Window allows the user to easily move around and zoom in or out on a large process flow diagram.



Moving the red square with the mouse will scroll the diagram window correspondingly. Making the red square larger or smaller will zoom in or out on the diagram window.

3.2.13 Resource Window

The Resource Window displays the resource usage of the current diagram. Resource usage is usually updated after code generation; however some resource items are updated in other ways, such as from process flow controllers when properties are modified.

Resources	
Resource	Usage
miniDSP_A_coeff (words)	70 (27.34%)
miniDSP_A_cycles	194 (21.46%)
miniDSP_A_cycles_alloc	904 (100.00%)
miniDSP_A_data (words)	158 (17.63%)
miniDSP_A_instr (words)	194 (18.95%)
miniDSP_A_instr_alloc (words)	904 (88.28%)
miniDSP_D_coeff (words)	45 (17.58%)
miniDSP_D_cycles	289 (31.97%)
miniDSP_D_cycles_alloc	904 (100.00%)
miniDSP_D_data (words)	145 (16.22%)
miniDSP_D_instr (words)	319 (31.15%)
miniDSP_D_instr_alloc (words)	934 (91.21%)

Values that go over 100% will display in red.

The resources shown in the resource window include:

- **miniDSP_A_coeff (words)** - amount of miniDSP_A coeff memory in use.
- **miniDSP_A_cycles** - number of miniDSP_A cycles used (not counting NOP instructions used for spacing in placement algorithm).
- **miniDSP_A_cycles_alloc** - number of miniDSP_A cycles allocated and used by the framework.
- **miniDSP_A_data (words)** - amount of miniDSP_A data memory in use.
- **miniDSP_A_instr (words)** - miniDSP_A instructions in use (not counting NOP instructions used for spacing in placement algorithm).
- **miniDSP_A_instr_alloc (words)** - miniDSP_A instruction RAM allocated and used by the framework.
- **miniDSP_D_coeff (words)** - amount of miniDSP_D coeff memory usage.
- **miniDSP_D_cycles** - number of miniDSP_D cycles used (not counting NOP instructions used for spacing in placement algorithm).
- **miniDSP_D_cycles_alloc** - number of miniDSP_D cycles allocated and used by the framework.
- **miniDSP_D_data (words)** - amount of miniDSP_D data memory in use.
- **miniDSP_D_instr (words)** - miniDSP_D instructions in use (not counting NOP instructions used for spacing in placement algorithm).

- **miniDSP_D_instr_alloc (words)** - miniDSP_D instruction RAM allocated and used by the framework.

Resource names ending with **_instr** indicate the memory used by the application code alone (as a percentage of the total memory). While **_instr_alloc** indicates the memory allocated and consumed by the given framework (as a percentage of the total memory). The difference gives an indication of how much more application code can be added in the given framework.

Typically **_instr_alloc** will remain constant (except for cases where components have control code) and **_instr** will increase and move closer to **_instr_alloc**. If control intensive code is added in the application, it may so happen that **_instr_alloc** will allocate more than available in the device even if **_instr** is small enough.

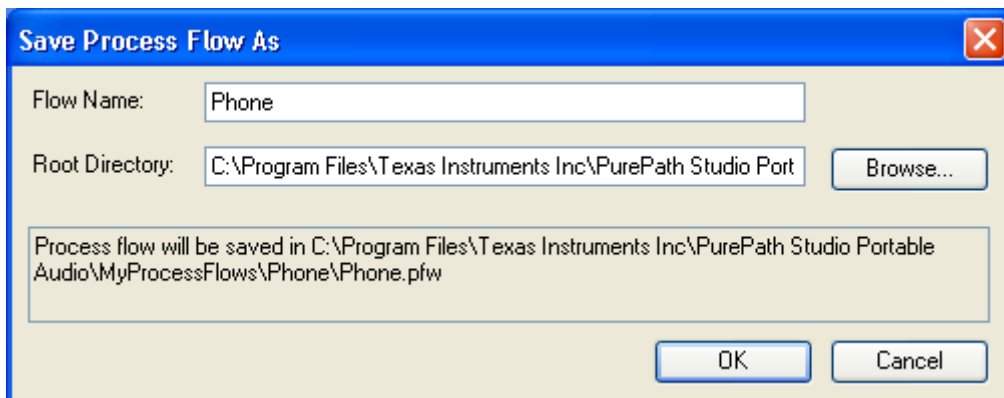
Resources ending with **_cycles** and **_cycles_alloc** have a similar concept to **_instr** and **_instr_alloc** in the sense that **_cycles_alloc** represents the cycles allocated and consumed by the given framework whereas **_cycles** represent the cycles consumed only by the application. Even if **_cycles** is small compared to **_cycles_alloc** it may so happen that **_cycles_alloc** will allocate more cycles than available in the given framework, if a component is too large to be placed within the cycle gaps that are provided for the application.

3.2.14 Source Code Window

The GDE can display the generated assembly source code in a window named Source Code Window.

3.2.15 Process Flow File Management

Process flow files are managed by the GDE in the following manner. A process flow is stored in a .pfl file by the GDE, in a directory named the same as the .pfl file root name. The Save and Save As dialogs in the GDE will enforce this naming as shown below.



All generated files from the process flow will be placed in the process flow directory. Only one process flow diagram can be saved per directory.

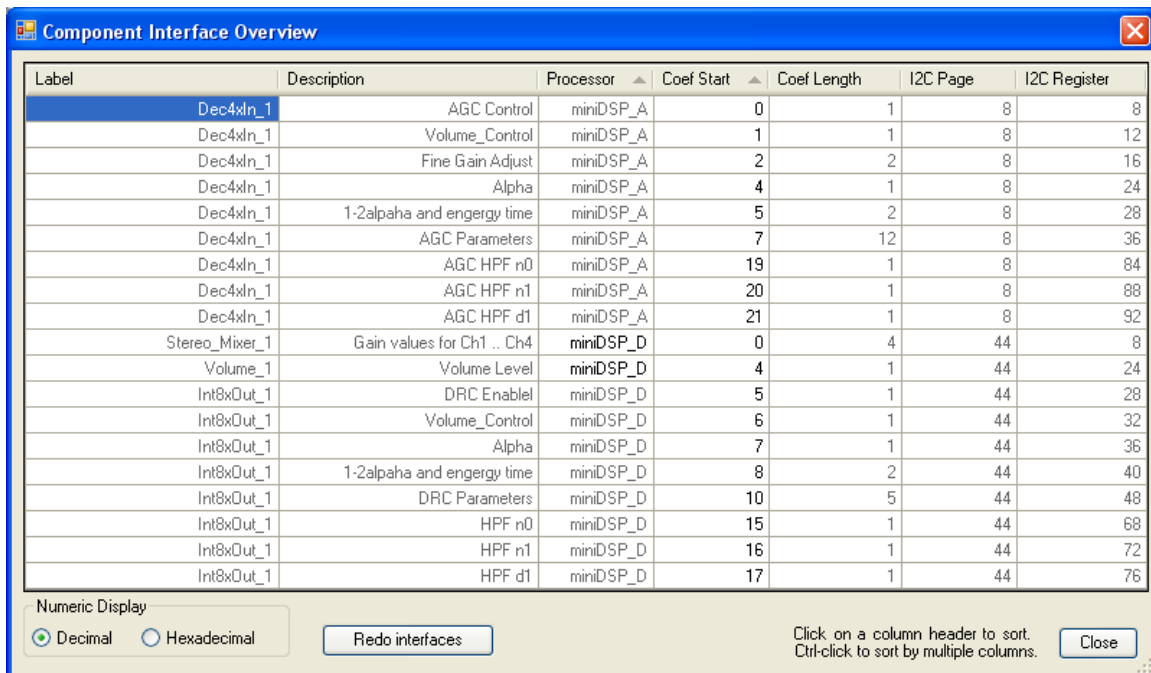
3.2.16 Component Interface Overview

Previously called the I²C overview, this window will be renamed Component Interface Overview for the miniDSP devices.

3.2.16.1 miniDSP Component Interface Overview – showing coefficient addresses and the corresponding starting I2C page and register.

The interface to miniDSP components is coefficient memory blocks, not specific I²C addresses. Although I²C is used to write the memory, the component is not assigned a specific I²C address as the interface.

The window will display component name, description and the coefficient block information for each interface. In addition, the display will show the page and register of the first I2C register used for the interface.



Label	Description	Processor	Coef Start	Coef Length	I2C Page	I2C Register
Dec4xIn_1	AGC Control	miniDSP_A	0	1	8	8
Dec4xIn_1	Volume_Control	miniDSP_A	1	1	8	12
Dec4xIn_1	Fine Gain Adjust	miniDSP_A	2	2	8	16
Dec4xIn_1	Alpha	miniDSP_A	4	1	8	24
Dec4xIn_1	1-2alpaha and energy time	miniDSP_A	5	2	8	28
Dec4xIn_1	AGC Parameters	miniDSP_A	7	12	8	36
Dec4xIn_1	AGC HPF n0	miniDSP_A	19	1	8	84
Dec4xIn_1	AGC HPF n1	miniDSP_A	20	1	8	88
Dec4xIn_1	AGC HPF d1	miniDSP_A	21	1	8	92
Stereo_Mixer_1	Gain values for Ch1 .. Ch4	miniDSP_D	0	4	44	8
Volume_1	Volume Level	miniDSP_D	4	1	44	24
Int8xOut_1	DRC Enablel	miniDSP_D	5	1	44	28
Int8xOut_1	Volume_Control	miniDSP_D	6	1	44	32
Int8xOut_1	Alpha	miniDSP_D	7	1	44	36
Int8xOut_1	1-2alpaha and energy time	miniDSP_D	8	2	44	40
Int8xOut_1	DRC Parameters	miniDSP_D	10	5	44	48
Int8xOut_1	HPF n0	miniDSP_D	15	1	44	68
Int8xOut_1	HPF n1	miniDSP_D	16	1	44	72
Int8xOut_1	HPF d1	miniDSP_D	17	1	44	76

Numeric Display
☒ Decimal ☐ Hexadecimal

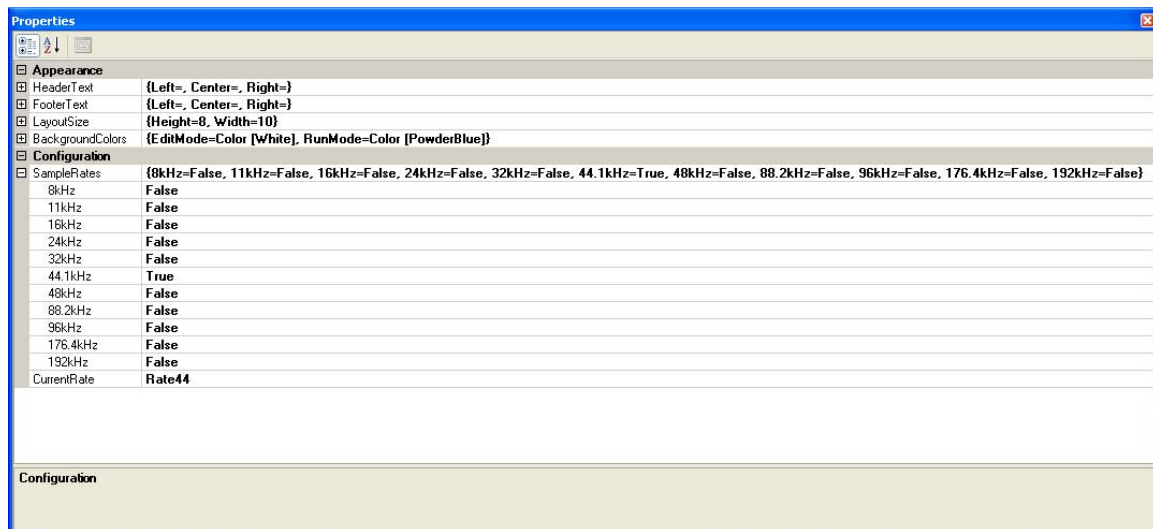
Click on a column header to sort.
Ctrl-click to sort by multiple columns.

3.2.17 Sample Rate Support

The GDE allows the user to choose one sample rate from the following set of supported sample rates:

- 8, 11, 16, 24, 32, 44.1, 48, 88.2, 96, 176.4, 192 kHz

The sample rate selection is made in the Property Window when the diagram (background) is selected. The CurrentRate property holds the set of supported rates.



The current rate is available as the property CurrentRate in any component.

3.3 PRINTING

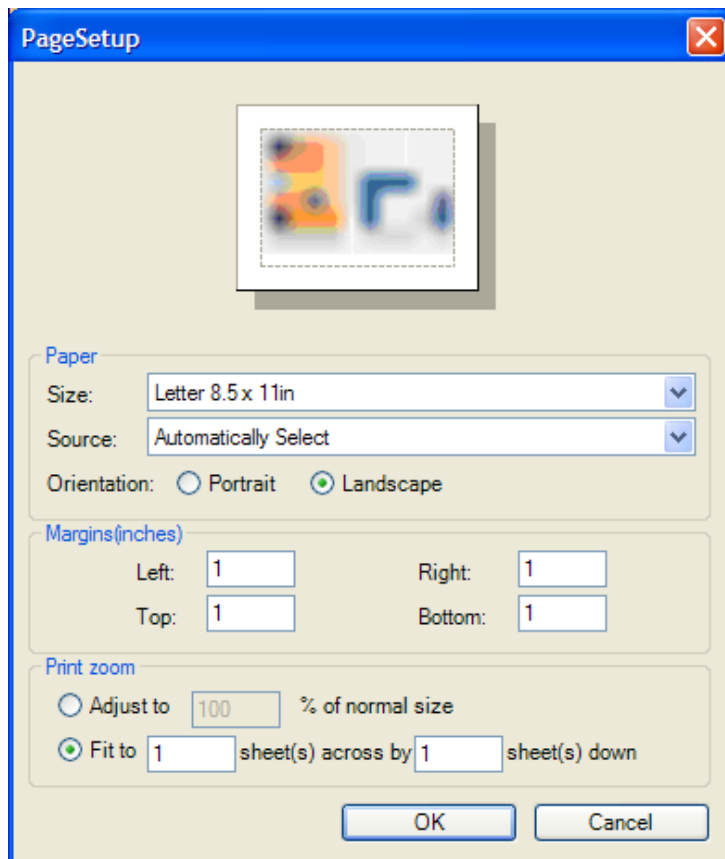
3.3.1 Printing

The File / Print menu item will open a standard Windows print dialog box. This will permit the printing of a diagram to any Windows compatible printer, including Adobe PDF generators.



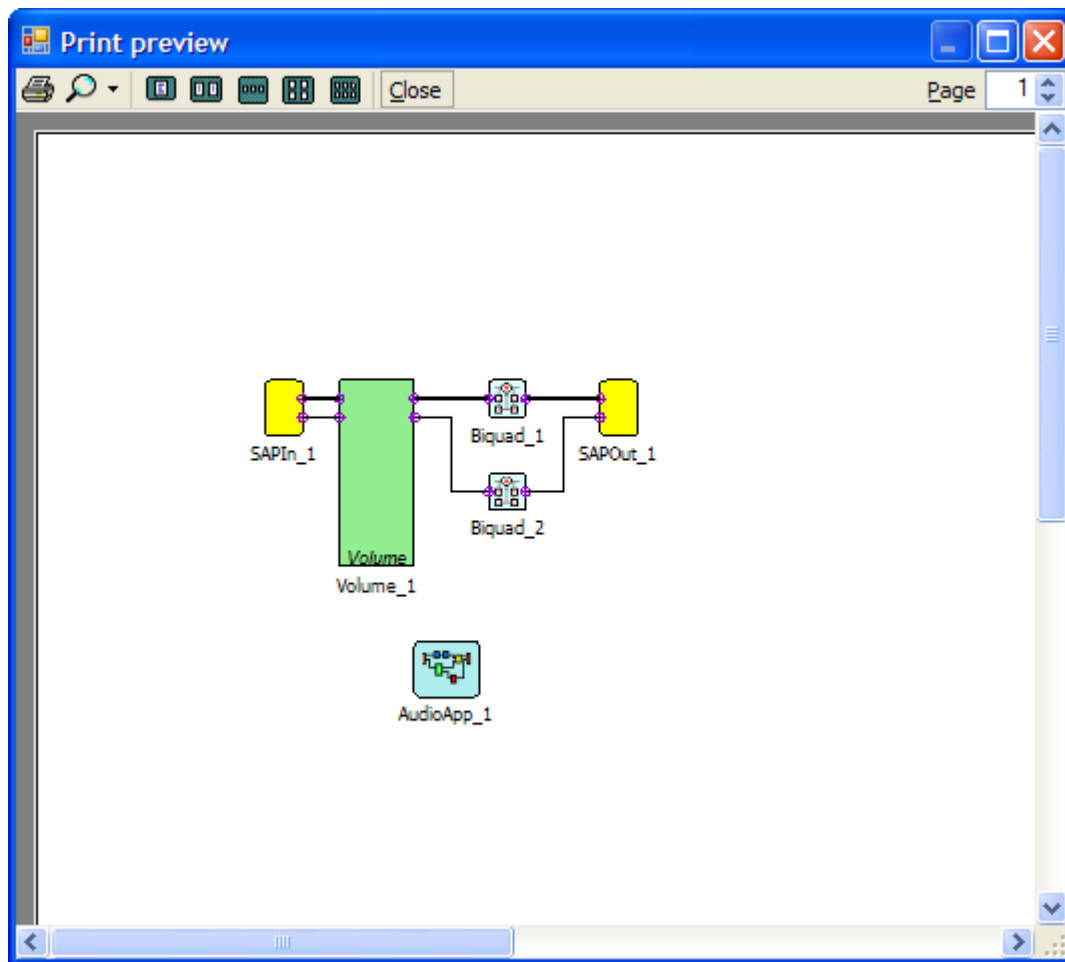
3.3.2 Page Setup

The File / Page Setup menu item will open the Page Setup dialog, which will allow the user to control the printing of the diagram, including fitting the diagram to specific number of pages.



3.3.3 Print Preview

The File / Print Preview menu item will allow the user to view the diagram as it would print on the page, including headers and footers.



3.4 GENERATING APPLICATION CODE

Generating application code from a diagram is done with the Build menu. Build / Generate code will generate application code for the current process flow.

The following actions will be performed when Build / Generate Code is selected:

- If the current diagram has never been saved, the user will be presented with a Save As dialog.
- If the current diagram has been modified since it was last saved, **it will be saved** without requesting confirmation by the user.
Application code for the current diagram will be generated. All errors and warnings will be echoed to the Output Window.
- If no errors are detected during the build, a .cfg file will exist in the process flow directory that is named with the same root name as the process flow .pflow file. If errors are

detected during the build, no .cfg is produced, and any existing .cfg file in the process flow directory named the same as the process flow file will have been removed.

3.4.1 Process Listing File

The Code Generator produces a process listing file (ProcessListingFile.txt) that includes the following information about the process flow:

- Sample Rate information.
- Component Information – the following information is provided for each component:
 - Version information
 - Component version
 - Alias information
 - Design properties
 - Runtime properties

3.5 EDIT MODE AND RUN MODE

The GDE has a concept of **Edit Mode** and **Run Mode**.

In **Edit Mode**, the process flow may be modified as follows:

- Components may be added, removed and moved
- Links between components may be added, removed and moved
- Design properties may be modified
- Runtime properties may be modified

In **Run Mode**, the structure of the process flow (components and links) may not be modified. If AdaptiveMode (see below) is enabled, the following may be done:

- Runtime properties may be modified

3.5.1 miniDSP Adaptive Mode

The miniDSP framework has two properties `ADC_miniDSPAdaptiveMode` and `DAC_miniDSPAdaptiveMode` that each have two settings, Enabled and Disabled.

The default setting for this mode is the Adaptive Mode

When Enabled, the following actions are made:

- Code is generated to place each processor, ADC miniDSP and DAC miniDSP, into adaptive mode when the processor is started. The coefficient areas for both parts of adaptive memory are initialized.
- The GDE allows runtime properties to be modified in run mode. Changes are immediately written to the coefficient bank that is connected to the control interface. The Build/Write State menu item and toolbar button swap coefficient banks to effect any current changes.

When AdaptiveMode is Disabled, the following actions are made.

- Run mode does not allow properties to be modified in the GDE. The Write State menu item and toolbar button are not enabled.

3.5.2 Codec Mode Programming

When PurePath programs the AIC3254, the AIC3254 is in a user program mode as opposed to a user selected ROM based mode that is selected by specifying an ADC or a DAC processing block. We accomplish much of the same functionality as we would in selecting a PRB configuration by selecting a framework, the interpolator, decimator, and any other processing blocks.

PurePath Studio GDE programs all of coefficient and instruction memory and sets a default configuration in pages 0 and 1 remain. You can create a script to include your custom Page 0 and 1 register settings. PurePath will load these using Tools/ I2C Command File.

3.6 COMPONENT INTERFACES

The GDE interacts with components at runtime through *component interfaces*. These interfaces are normally the same style of interface that will be used in the end application to control the same components.

3.6.1 Component Interfaces

The miniDSP uses a component interface based directly on coefficient memory writes. This is due to the I²C interface from the miniDSP can directly address all of coefficient memory while the processor is running (adaptive mode only).

In order for component interfaces to operate, the application framework must have Adaptive Mode enabled, which allows for modifications of coefficient memory during execution. If Adaptive mode is not enabled, the GDE will not allow any changes to the target while running, and the component interfaces, even if defined, will not be used.

3.6.1.1 Component Definition of the Direct-Memory Component Interfaces

Components define a direct-memory by defining a set of properties. A component may define up to 16 independent component interfaces.

Each component interface has the following properties, which are defined when the component is created.

In the diagram below, *N* is 1 thru 16.

Name	Description
DspCoefBlockStart N	Initially set to -1. The starting address of a block of coefficient memory to be declared to hold the contents of the values transferred. This property is initially set by the GDE and may be edited by the user, but the user must be careful to choose memory that is available for use and does not conflict with other components.
DspCoefBlockLength N	The length (in words) of the coefficient block of memory for the transfer. Or 0 if no coefficient memory is used. This property is read-only.

3.6.1.2 Component Interface assignment by the GDE

The GDE will manage the component interfaces for all components, including the allocation of coefficient memory. The components will be assigned a component interface when they are first placed on the process flow diagram. The user will have the option to edit the interface placement by hand later.

3.6.1.2.1 Initialization

When a framework component is dragged and dropped onto the diagram, the GDE reads the following property from the framework to initialize the component interface information.

Name	Type	Description
ComponentInterfaceStartingCoefBlock	integer	The first address to be used for DSPCoefBlockStart use

3.6.1.2.2 Coefficient Memory Allocation

The GDE will manage the allocation of coefficient memory to components. The GDE will begin allocating coefficient blocks for interfaces starting at the address specified by the framework property ComponentInterfaceStartingCoefBlock.

As component instances are created and removed, this block may become fragmented. The GDE will use a *first-fit* allocation scheme to attempt to allocate component memory blocks, as described in Section 3.6.1.2.3 below. The user may also modify the GDE's allocation as described in Section 3.6.1.2.5 below.

3.6.1.2.3 Assignment

When a component is dragged and dropped onto the diagram, the GDE will use the following steps to assign the component interface to the component.

- Starting at the coefficient address specified by ComponentInterfaceStartingCoefBlock, the first available coefficient memory block that is large enough will be assigned the component. If no coefficient memory block is available, an error will be given.

3.6.1.2.4 Component Removal

When a component on a diagram is removed or aliased to another component, the following occurs:

- The component's coefficient memory is marked as free in the GDE's memory manager and will be immediately available for use in new components.

3.6.1.2.5 Component Interface Modification by the user

After a component is dragged and dropped and has been assigned a component interface, the user may change the location of the component interface's coefficient memory block by modifying the DspCoefBlockStart property for that component interface.

Making these changes should only be done by expert users, and care must be taken when making the changes, as it is possible to inadvertently double-allocate memory blocks, causing

unpredictable results at runtime. The Component Interface Overview Window (below) should be used to view the end result of any user modification to ensure that it yields the expected results.

3.6.1.2.6 Component Interface Overview Window

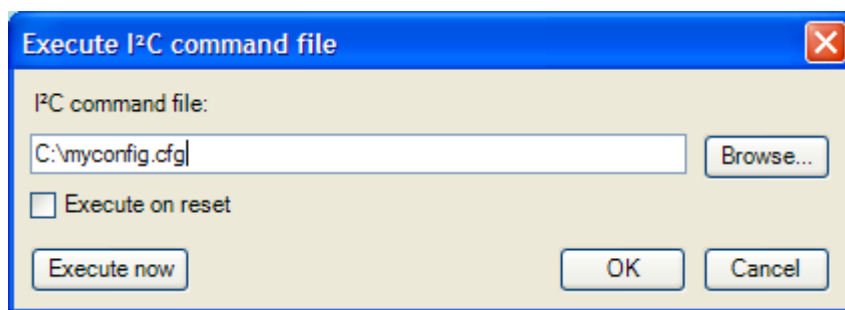
The GDE can display the overall usage of coefficient memory for component interfaces in a Component Interface Window. For help with the Component Interface Window see section 3.2.14.

3.7 CONFIG FILE EXECUTION

The GDE can execute .cfg config files. The format of the .cfg files is described below in Section 3.8.4 below.

3.7.1 Command File Execution

The following simple dialog handles config file execution.



The GDE can execute a .cfg file immediately, or set the .cfg file to be executed after every EVM Reset.

3.8 I²C MEMORY TOOL

The I²C Memory Tool is available for directly reading and writing I2C registers and memory locations.

General rules for using the MemoryTool:

1. GDE in EDIT MODE
 - a. Changing runtime properties will not result in any writes to the Coeff memory.
2. GDE in RUN MODE
 - a. If miniDSP processor is running (Start/Stop is GREEN),
 - i. miniDSP in Adaptive Mode.
 1. Any changes to the runtime properties in GDE will get written to the Coeff buffer that is not used by the miniDSP (e.g. BufferB)
 2. If the Coeff buffer is swapped (either via the "Adaptive Mode" buffer selection above, or the "Write State" Button in GDE)
 - a. BufferB will now be used by miniDSP, and BufferA will be accessible by external Host.

- b. A peek on BufferA or BufferB will yield the same value (= new runtime value). It seemed that GDE will automatically modify BufferA to the new value as well. And this only happens under the condition where:
 - i. GDE is in RUN MODE.
 - ii. Adaptive Mode is enabled.
 - iii. Most importantly, the miniDSP processor is 'running' i.e. Start/Stop button is GREEN.
- c. If the miniDSP processor is not running (Start/Stop Button is RED), then this auto modification of BufferA does not happen.


3.8.1 Peek/Poke Support

The Peek/Poke tab supports memory read and write to the miniDSP processor as follows:

- **Connect** – Attempt to establish a connection with a board. The MemoryTool will automatically attempt to connect to the board on startup. If the board is not connected, the Connect button will be shown so the user can connect the MemoryTool after connecting the USB to the board. Allow a few seconds for Connect to process.
- **Address** – The address to peek or poke. This may be entered in decimal or in hex (with 0x).

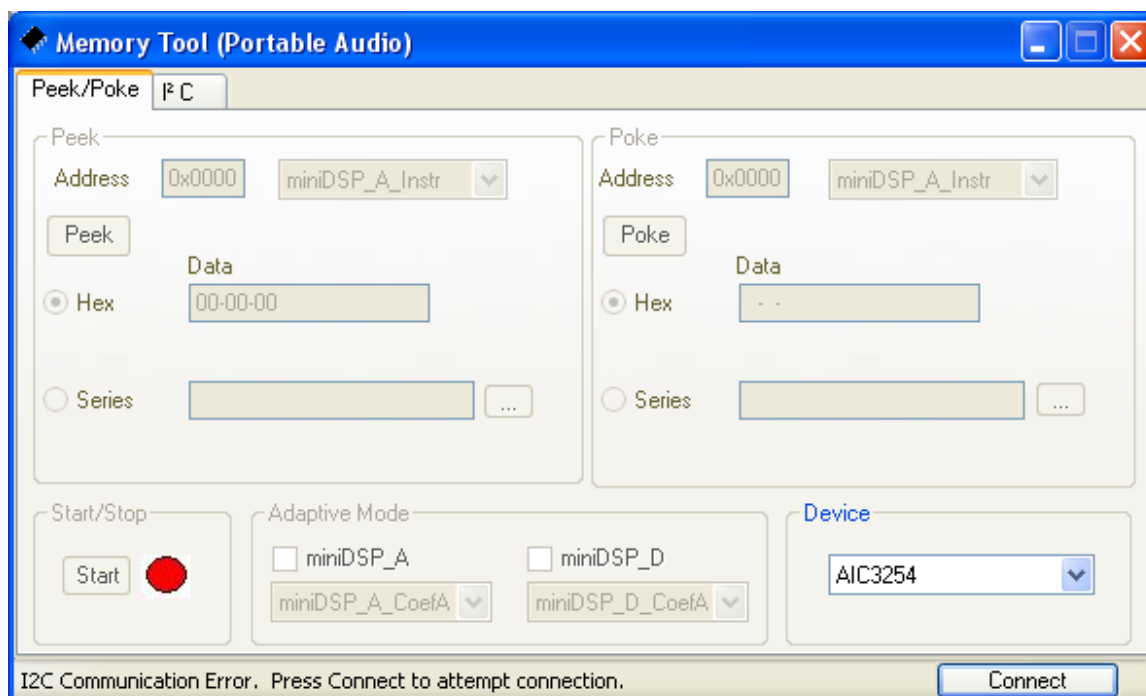
miniDSP_A_CoefA, miniDSP_A_CoefB, miniDSP_A_Instr, miniDSP_D_CoefA, miniDSP_D_CoefB, miniDSP_D_Instr – Choose the memory to peek or poke. **miniDSP_A_Instr** and **miniDSP_D_Instr** are displayed as hex only. Therefore, the Fixed field will not be visible for that selections.

If the miniDSP_A or miniDSP_D processor is stopped, all three memories (CoefA, CoefB, Instr) will be available for that processor. If the processor is running (see below), only the coefficient buffer that is not connected to the processor at this time will be available for peek/poke. No other memory will be available.

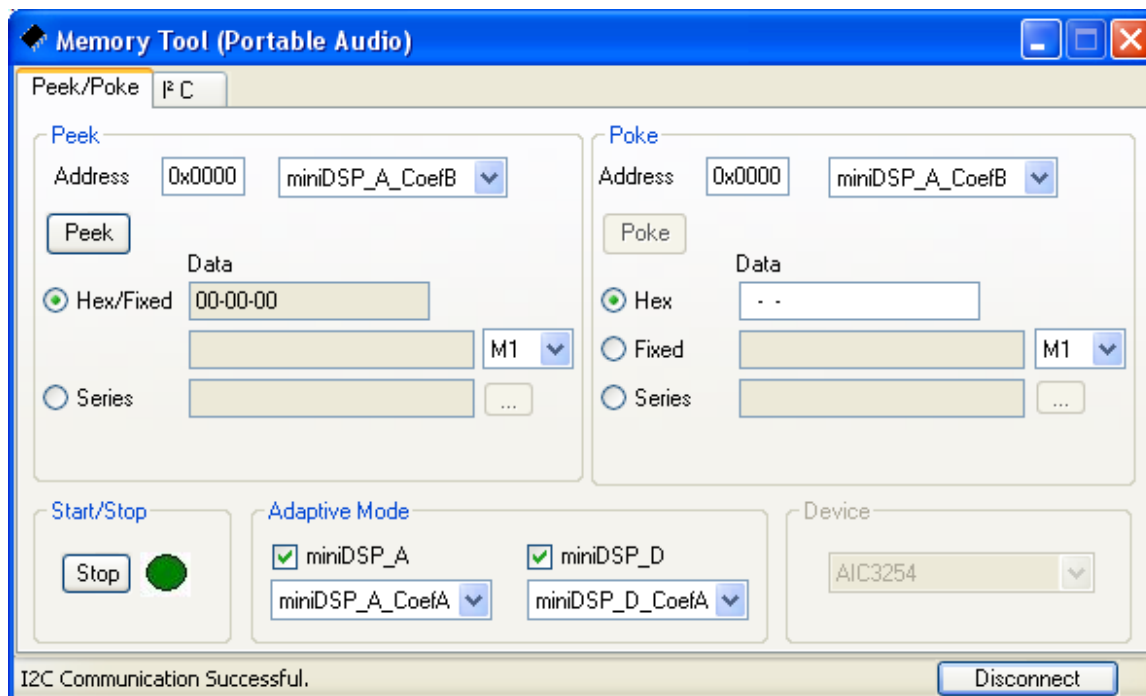
- **Peek/Poke** – Button corresponding to Peek or Poke.
- - **Peek**
 - **Hex/Fixed** – Peeks single address designated by Address box, returns address value in both Hex and Fixed Point numbers.
 - **Series** – Designate .txt file via  to save values. Saves values from peek beginning at address designated by Address box and by the amount of times designated by Addr count spin control. Must not be a blank space in order to Peek.
 - **Poke**
 - **Hex** – Poke address based upon hex number designated by user. Disabled unless values are filled.
 - **Fixed** – Poke address based upon fixed point number designated by user. Disabled when null.
 - **Series** – Pokes series of addresses based upon file designated by user. Must contain a valid file path in order to Poke.
- **Addr Count** – (Displayed only when Peek Series is chosen) Number of sequential addresses to peek for a Peek series. This is a spin control with valid values 1 ... 999.



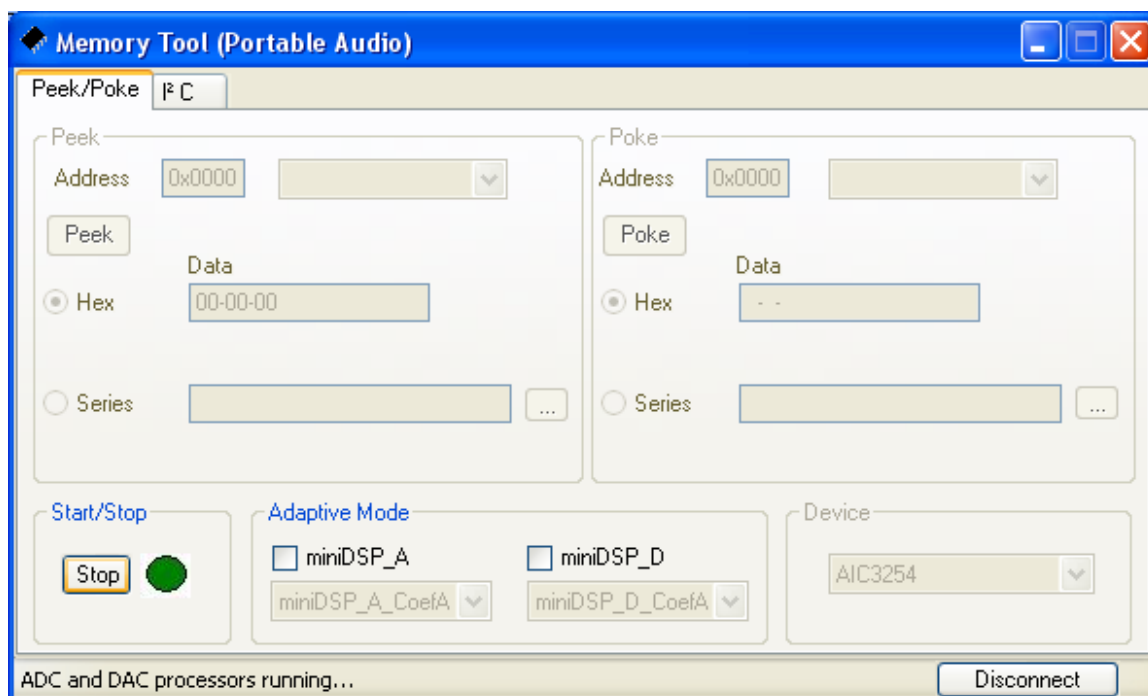
Start/Stop – Starts and stops the ADC miniDSP and DAC miniDSP processor.



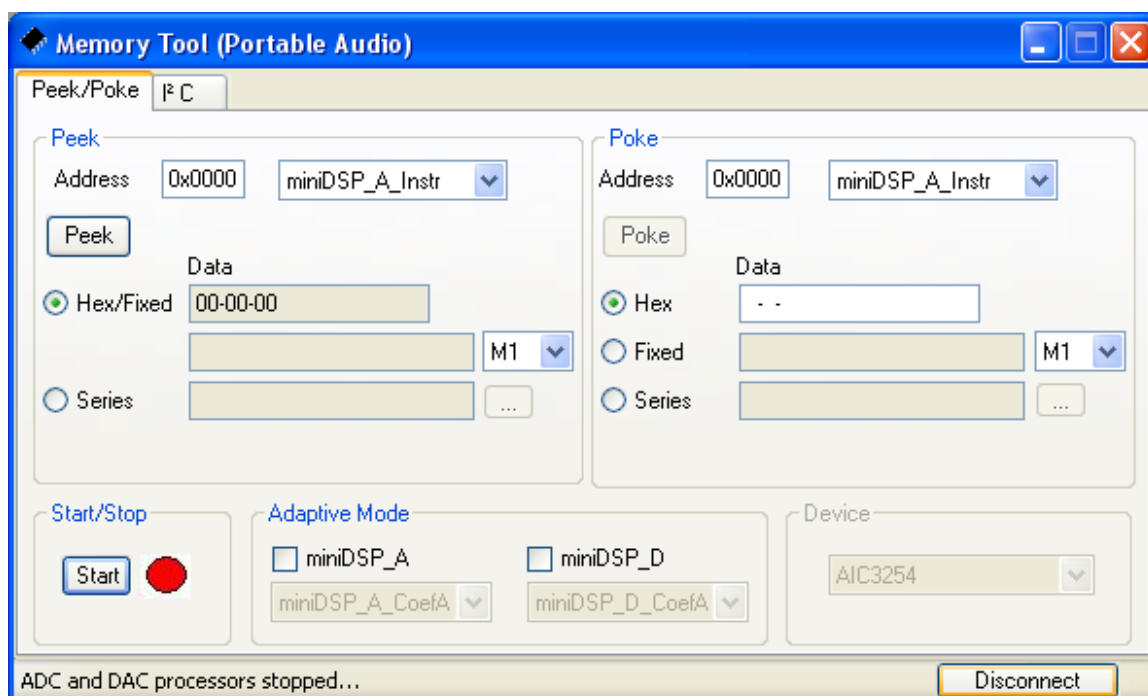
MemoryTool Peek/Poke window, when no hardware is present.



MemoryTool Peek/Poke window after Connecting, when hardware is present.



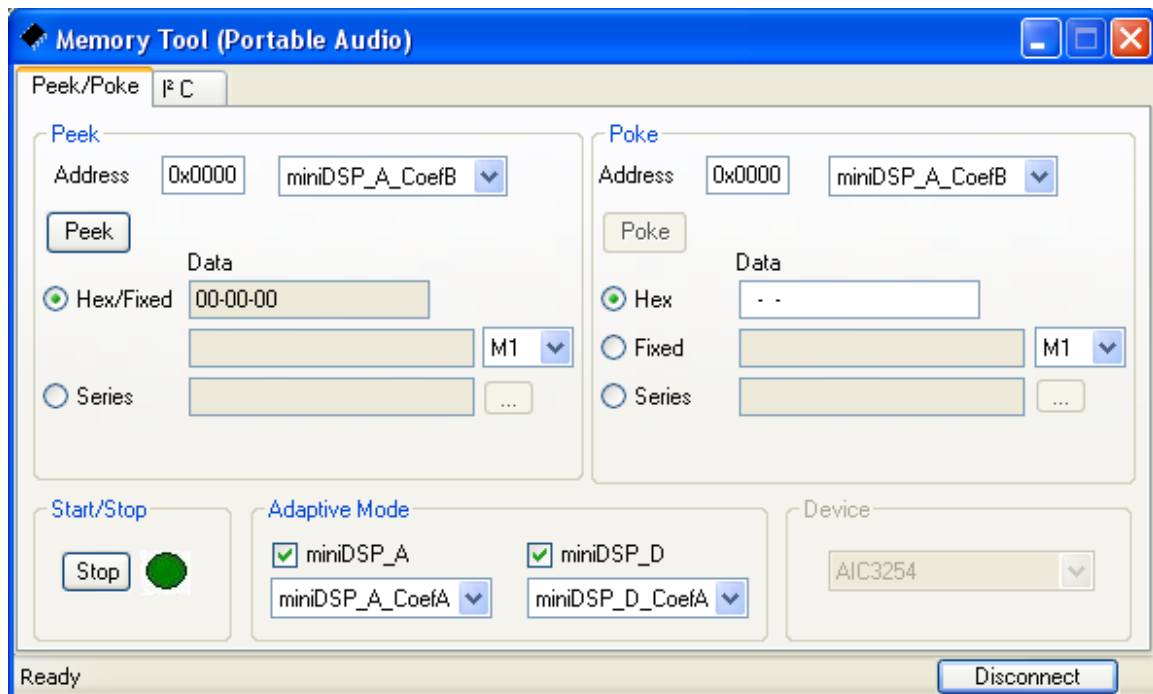
MemoryTool Peek/Poke window in Connected state, when miniDSP processors are not in Adaptive mode and are running.



MemoryTool Peek/Poke window in Connected state, when miniDSP processors are not in Adaptive mode and are not running.

3.8.2 Peek/Poke Support in Adaptive Mode

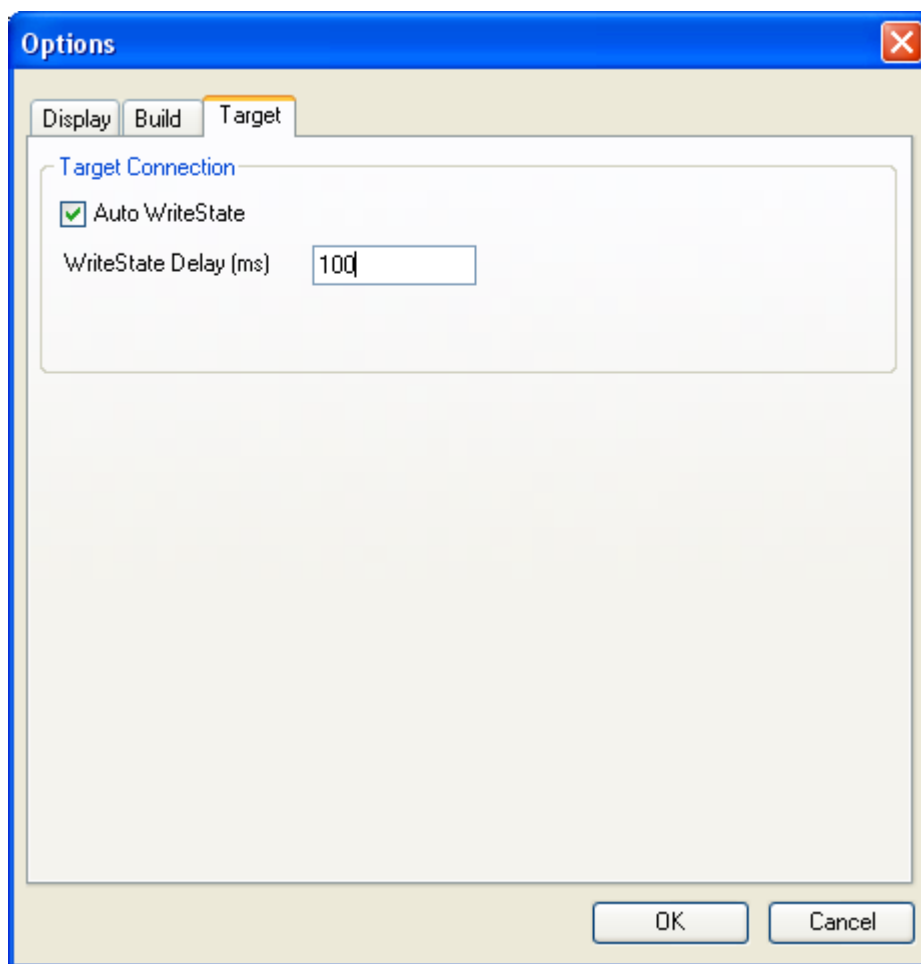
Adaptive mode – Place the miniDSP_A or miniDSP_D processor into adaptive mode. There are two identical coefficient memory buffers, A and B. When the miniDSP processors are running only one is accessible to the I2C interface while the other is accessible by the miniDSP. Once in adaptive mode, the drop-box may be used to change the current coefficient bank that is in use by the processor. The bank not in use by the processor will be shown in the Peek and Poke section and addressable by the user. Adaptive mode is the default configuration for the AIC3254.



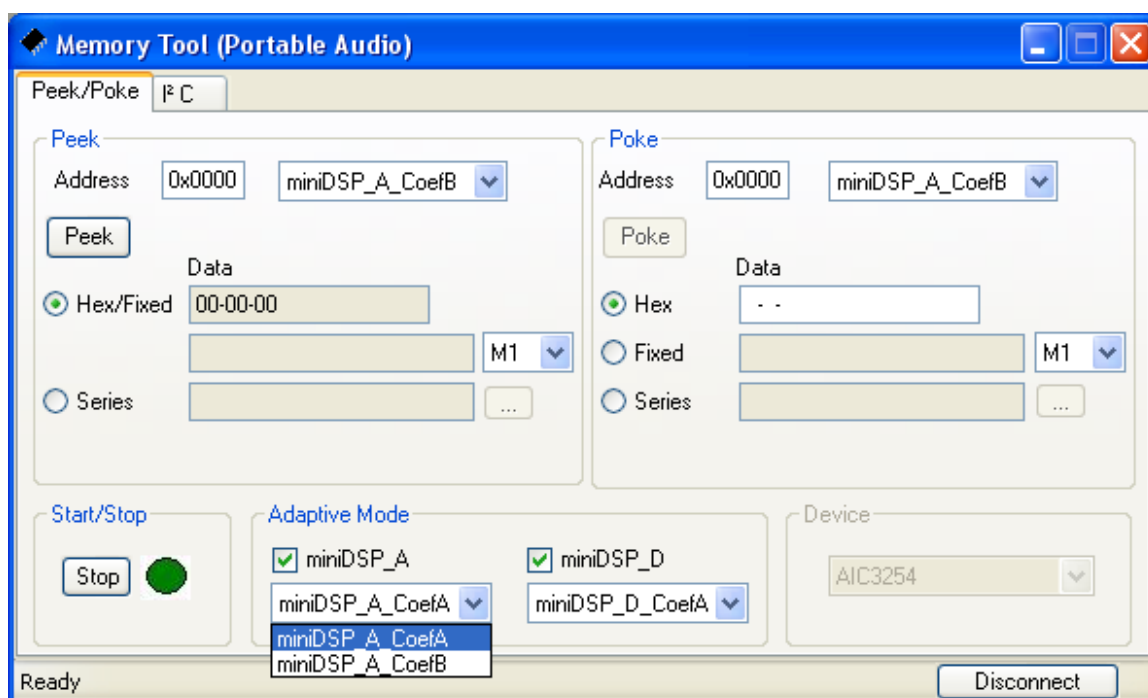
MemoryTool Peek/Poke window when miniDSP_A and miniDSP_D are in Adaptive mode.

The Adaptive mode control at the bottom middle of the dialog shows that the miniDSP processors are both running out of the CoeffA buffers. Therefore the CoeffB buffers are available for Peek/Poke operations. Any data from a Peek operation is from the currently unused CoeffB buffer. Likewise, any data written to by a Poke operation will go to the CoeffB buffer. This data will not become active for the miniDSP processors until the buffers are swapped.

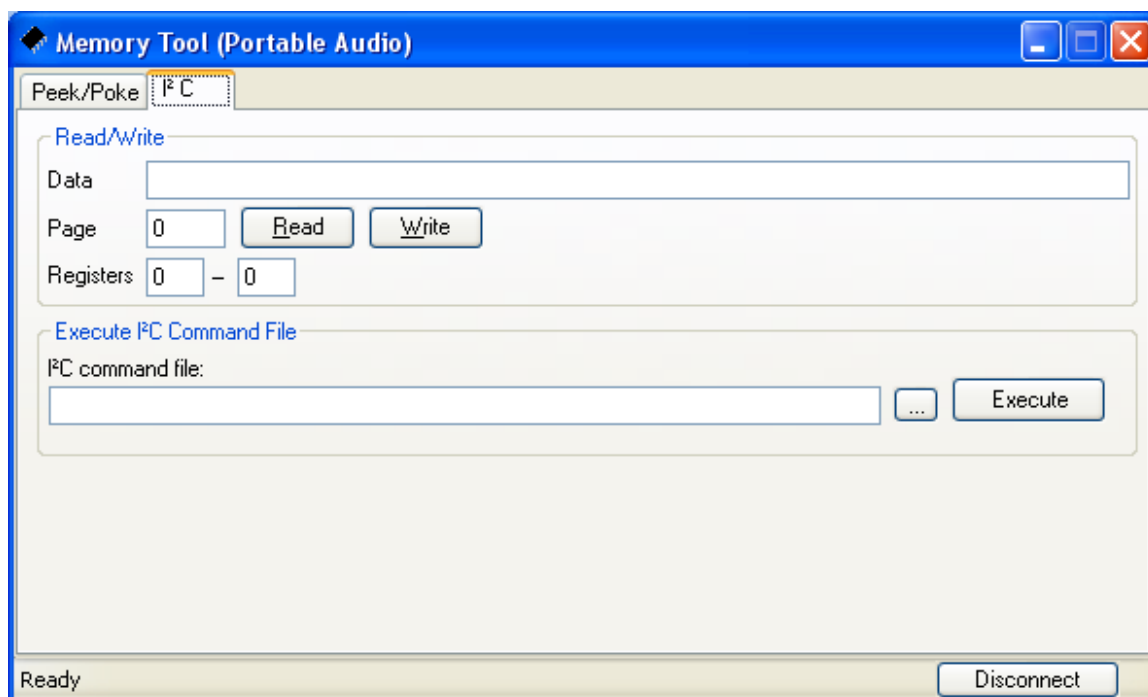
Adaptive mode buffer swapping occurs automatically in the default GDE configuration. This action is controlled by the Auto WriteState option found on the GDE Tools menu, Options dialog, Target tab. This option is enabled by default and will swap the Adaptive buffers every 100ms, as shown below. If this option is disabled, the buffers may be swapped manually by using the Write State command on the Build menu of the GDE.



We can toggle which Coefficient buffer (A or B) is displayed by selecting it in the drop-down box, as shown below:



3.8.3 I2C Read/Write Support



The I²C tab supports general I²C reading and writing. The interface is as follows:

- **Connect** – Attempt to establish a connection with a board. Allow a few seconds after reconnecting device before using Connect.
- 0x).
- **Page** – **The I2C register page (0 – 255), may be entered as decimal or hex (with 0x)**
- **Registers** – **User designated register start and end. If user enters 0 – 0, 1 byte will be read. The MemoryTool reads consecutive I2C registers.**
- **Data** – Hex data, separated by dashes. The number of bytes displayed is set by values in Registers miniDSP supports up to 24 bytes.
- **Read** – Perform an I²C read.
- **Write** – Perform an I²C write.

- **Execute I2C Command** – Executes an I²C Command File.

3.8.4 I2C Command File Support for miniDSP

A script is simply a text file that contains data to send to the serial control buses. The scripting language is quite simple.

Each line in a script file is one command. There is no provision for extending lines beyond one line. A line is terminated by a carriage return.

The first character of a line is the command. Commands are:

i	Set interface bus to use
r	Read from the serial control bus (optionally compare with a given value)
w	Write to the serial control bus
#	Comment
b	Break
d	Delay

The first command, **i**, sets the interface to use for the commands to follow. This command must be followed by one of the following parameters:

- i2cstd** Standard mode I²C Bus
- i2cfast** Fast mode I²C bus (default)
- spi8** SPI bus with 8-bit register addressing
- spi16** SPI bus with 16-bit register addressing
- gpio** Use the USB-MODEVM GPIO capability

For example, if a fast mode I²C bus is to be used, the script would begin with:

i i2cfast

At this time, only the i2cfast interface mode is be accepted and used. Any other interface will be an error. If no 'i' command is given, i2cfast is assumed.

No data follows the break command. Anything following a comment command is ignored by the parser, provided that it is on the same line. The delay command allows the user to specify a time, in milliseconds, that the script will pause before proceeding.

Note: *UNLIKE ALL OTHER NUMBERS USED IN THE SCRIPT COMMANDS, THE DELAY TIME IS ENTERED IN A DECIMAL FORMAT. Also, note that because of latency in the USB bus as well as the time it takes the processor on the USB-MODEVM to handle requests, the delay time may not be precise.*

A series of byte values follows either a read or write command. Each byte value is expressed in hexadecimal, and each byte must be separated by a space.

The first byte following a read or write command is the I²C slave address of the device (if I²C is used) or the first data byte to write (if SPI is used—note that SPI interfaces are not standardized on protocols, so the meaning of this byte will vary with the device being addressed on the SPI bus). **At this time, the I²C slave address will be read and ignored by the scripting. The I²C address is defined by the interface software.**

The second byte is the starting register address that data will be written to. Following these two bytes are data, if writing; if reading, the third byte value is the number of bytes to read, (expressed in hexadecimal).

For example, to write the values 0xAA 0x55 to an I²C device with a slave address of 0x90, starting at a register address of 0x03, one would write:

```
#example script
i i2cfast
w 90 03 AA 55
r 90 03 2 AA 55
```

The read command can be given in two forms. Given just an address and length, the read occurs and nothing is done with the result. Given an address, length and data, the read occurs and the data is compared with the given data that follows the read. If more data is given than the count, the extra is ignored, if less data is given, only the given data is used to compare. If the comparison fails, the script is halted with an error.

This script begins with a comment, specifies that a fast I²C bus will be used, then writes 0xAA 0x55 to the I²C slave device at address 0x90, writing the values into registers 0x03 and 0x04. The script then reads back two bytes from the same device starting at register address 0x03 and compares the values with AA 55.

Here is an example of using an SPI device that requires 16-bit register addresses:

```
# setup TSC2101 for input and output
# uses SPI16 interface
# this script sets up DAC and ADC at full volume, input from onboard mic
#
# Page 2: Audio control registers
w 10 00 00 00 80 00 00 00 45 31 44 FD 40 00 31 C4
w 13 60 11 20 00 00 00 80 7F 00 C5 FE 31 40 7C 00 02 00 C4 00 00 00 23 10 FE 00 FE 00
```

If there are breakpoints in the script, the script will execute to that point, and the user will be presented with a dialog box with a button to press to continue executing the script. When ready to proceed, push that button and the script will continue.

Here an example of a (partial) script with breakpoints:

```
# setup AIC3xxx for input and output
# uses I2C interface
i i2cfast
# reg 07 - codec datapath
w 30 07 8A
r 30 07 1
d 1000
# regs 15/16 - ADC volume, unmute and set to 0dB
w 30 0F 00 00
```



```
r 30 0F 2  
b
```

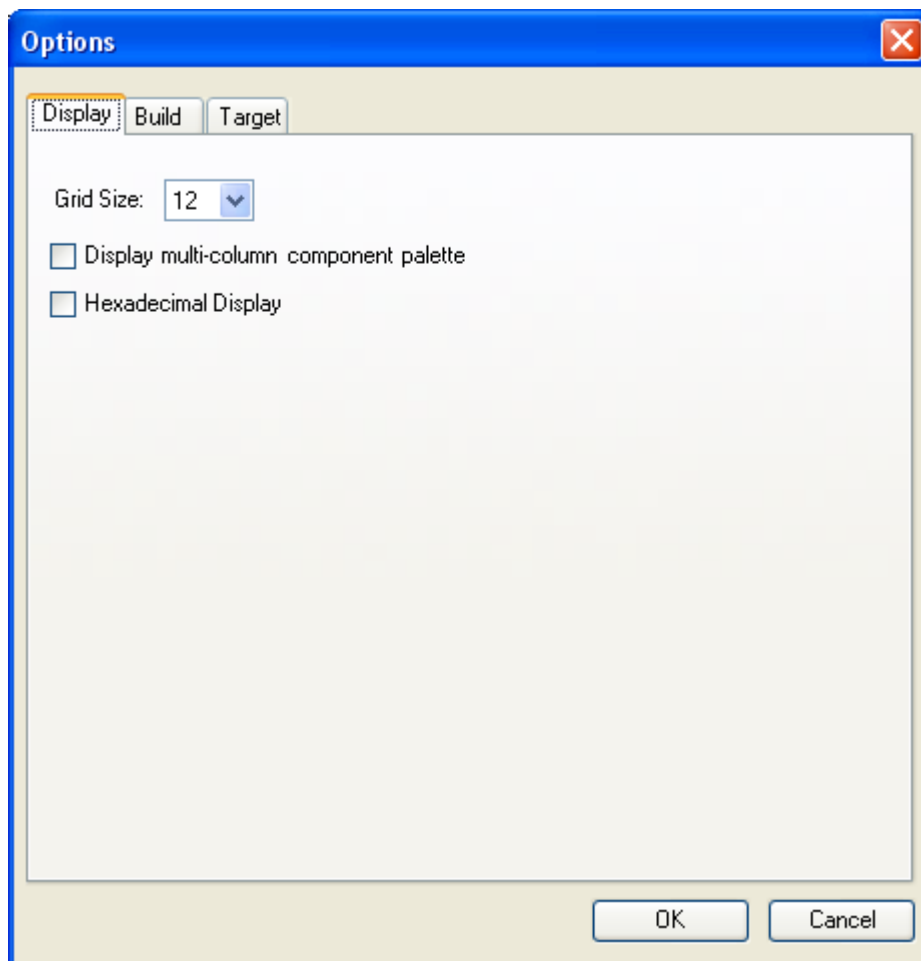
This script writes the value 8A at register 7, then reads it back to verify that the write was good. A delay of 1000ms (one second) is placed after the read to pause the script operation. When the script continues, the values 00 00 will be written starting at register 0F. This output is verified by reading two bytes, and pausing the script again, this time with a break. The script would not continue until the user allows it to by pressing OK in the dialog box that will be displayed due to the break.

3.9 GDE OPTIONS

The Options dialog is used to set GDE options. All GDE options are persisted on the user's PC. Persisted settings may be cleared by running the GDE with a /clean switch.

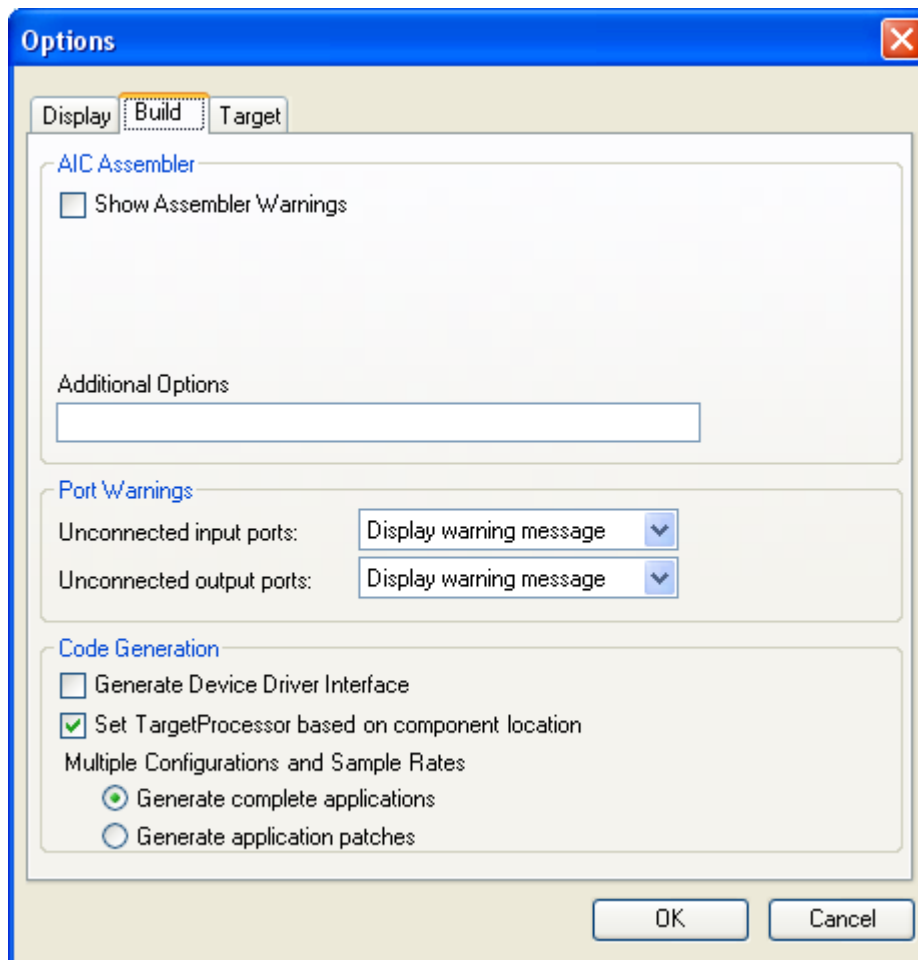
3.9.1 Display Tab

The display tab supports setting the grid size, displaying components in the palette in one column or multiple columns and turning the hexadecimal display of Fixed Point and Integer properties on and off.



3.9.2 Build Tab

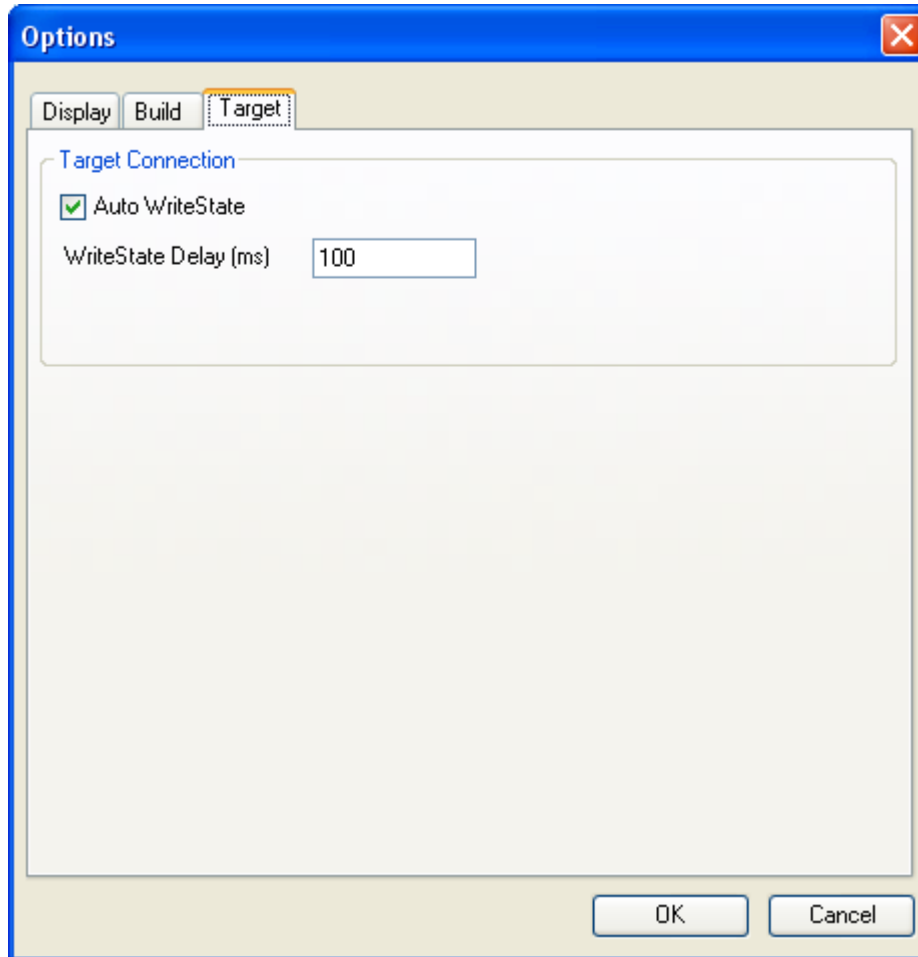
The Build tab sets the options for building applications.



- **DSP Assembler Options** – Choose whether assembler warnings are shown in the Output Window. Additional options to send to the DSP Assembler. This may be used to specify /Target options to support additional chipsets in the GDE.
- **Port Warnings** – Set the desired behavior of the GDE when an unconnected input port or output port is found in the diagram during code generation.

3.9.3 Target Tab

The Target tab sets the options for target connections.



3.10 ONLINE HELP

The GDE has online Help available from the Help / Index menu item or from the F1 key at any time.

Each component has a help File that is accessed by right clicking on the component and selecting help.

3.11 SUPPORTING MULTIPLE TARGET CHIPSETS

Support for multiple target chipsets in the GDE is driven by the components and the application framework. Each application framework defines a TargetType attribute.

Each component also defines a list of TargetType attributes that list all of the target types for which the component is valid. The special target type 'Any' will match with any framework type.

3.12 COMPONENT LIBRARY

Components in the GDE are stored in a Component Library.

3.12.1 System Component Library

The system component library is located in the GDE installation area in a directory named ComponentLibrary. This area holds components that are shipped with the GDE and third party components that are added later to the GDE.

3.12.2 User Component Library

A user component library is located in the GDE installation area in a directory named UserComponentLibrary.

3.12.3 Component Cache

Prior to using components, the GDE unpacks components from the System Component Library and User Component Library to a Component Cache. On GDE startup, any component whose .zip file in the component library is newer than the component in the component cache is refreshed in the component cache.

Starting the GDE with the /clean switch clears the component cache.

If the same named component exists in the System Component Library and the User Component Library, the component from the User Component Library is used.

If the same component exists with different version numbers, both components are unpacked into the Component Cache, however, only the 'latest' component, as defined by a lexical comparison of version number, is loaded into the Component Palette.

3.13 PROCESS FLOW CONTROLLERS

The GDE communicates to components on the EVM during run mode via the I²C interface. The GDE exports an IProcessFlowController interface which may be implemented by a DLL to provide component-specific I²C control.

3.14 POPUP GUIs

The GDE provides an interface for specialized popup GUIs to be built to edit the properties of a component. One example is the BiQuad filter designer GUI. Popup GUIs implement the IPopUp interface defined in the GDE.