

## Using Hardware Controls with SigmaDSP GPIO Pins

by Brett Gildersleeve

### INTRODUCTION

Several devices in the SigmaDSP® family include general-purpose input/ output (GPIO) pins. These pins can be connected to external hardware elements, such as LEDs, push-buttons, or rotary encoders.

Some of the GPIO pins may be multiplexed to an auxiliary ADC, allowing direct connection of a potentiometer or an analog control voltage to the IC. Each pin can operate in a variety of modes, including open-collector output, inputs with debounce, and outputs driven directly by the control port.

Within the SigmaStudio™ graphical programming software, GPIO pins can be assigned to control or be controlled by various parts of the audio signal processing program. The functionality of the pins can be changed during run-time operation of the device. Utilization of the GPIO pins for user interface functionality can reduce or eliminate the need for a microcontroller. This can greatly reduce the cost of a simple audio system.

This application note begins by describing the hardware interfacing necessary to effectively use the GPIOs. In addition, this application note discusses several cases that could greatly reduce the complexity of a SigmaDSP audio system.

## TABLE OF CONTENTS

Introduction .....	1	Using the Evaluation Board with the GPIO Board.....	7
Hardware Interfacing .....	3	Example Implementations of GPIO in Software.....	8
Momentary Push-Button Input.....	3	Push-Button Volume Control—Up/Down/Mute .....	8
Single-Pole Switch Input.....	3	Rotary Encoder Volume Control .....	10
Rotary Encoder Input .....	3	Push-Button-Controlled Demultiplexer .....	11
Potentiometer Input .....	3	Push-Button-Controlled Filter .....	12
LED Output.....	3	Auxiliary ADC Volume Control .....	13
Using GPIO in a SigmaStudio Project .....	4	Auxiliary ADC Voltage-Controlled Oscillator.....	13
Setting Up GPIO Pins in the SigmaStudio Register		Auxiliary ADC Slew Multiplexer .....	14
Window.....	4	Blinking LED .....	14

## HARDWARE INTERFACING

GPIO pins can be interfaced to several types of hardware controls. This section details several examples of these controls and their associated circuits. Unless otherwise mentioned, IOVDD = 3.3 V.

### MOMENTARY PUSH-BUTTON INPUT

One of the simplest forms of a control input to the SigmaDSP is a push-button. In Figure 1, a momentary switch is used in an active-low configuration with a 10 kΩ pull-up resistor. More information about pin current limits can be found in the relevant device data sheet.

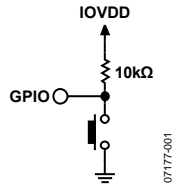


Figure 1. Active-Low Push-Button Input Circuit Example

SigmaDSP GPIO pins feature a debounce circuit, which should be activated in SigmaStudio to avoid errors from contact chatter on switching. For active-low operation, as in this example circuit, the corresponding GPIO register should be set as an inverting input. For active-high operation, the register can be set as noninverting, and the circuit switch and resistor should exchange places. For devices without hardware invert bits, a software logic inverter is available in SigmaStudio.

### SINGLE-POLE SWITCH INPUT

Switches can be used in a similar fashion to momentary push-buttons. In the example shown in Figure 2, the switch is shown in the active-low configuration with a pull-up resistor. This switch is not momentary, in contrast to the example shown in Figure 1.

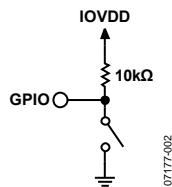


Figure 2. Active-Low Switch Input Circuit Example

### ROTARY ENCODER INPUT

A GPIO conditioning cell is available in SigmaStudio for incremental rotary encoders, also called relative rotary encoders. An incremental rotary encoder is essentially a knob that produces output pulses depending upon which direction it is turned. This should not be confused with an absolute rotary encoder, which has a set of binary codes corresponding to all possible angular positions. To use an incremental rotary encoder, connect the output pins to two GPIO pins. The corresponding block in SigmaStudio calculates the phase difference between pulses

on the two pins in order to recognize clockwise and counter-clockwise turns.

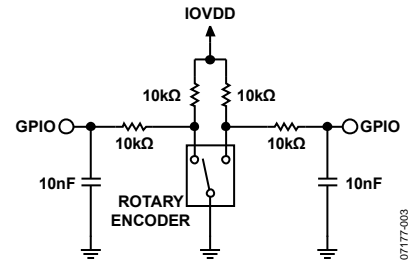


Figure 3. Rotary Encoder Input Circuit Example

### POTENTIOMETER INPUT

A potentiometer can be used as an analog control, most often to adjust volume. Because the auxiliary ADC has linearly spaced steps, a linear potentiometer should be used for best results. A logarithmic look-up table can be implemented within software if a logarithmic control is desired. For some SigmaDSPs, the full-scale input voltage for the ADC is less than IOVDD. Because of this, a voltage divider should be used to appropriately scale the signal, as in the example circuit in Figure 4.

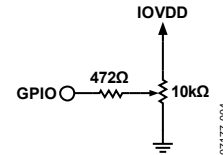


Figure 4. Potentiometer Input Circuit Example

### LED OUTPUT

An LED is the simplest form of GPIO output. A transistor is necessary to avoid excessive loading of the SigmaDSP. An example transistor is the MMBT2222A. The SigmaDSP can source or sink the current, as long as it does not exceed the current limitations as specified in the device data sheet. See the relevant data sheet for more details. For an active-high, current-sourcing implementation, the transistor should be NPN in a common-emitter (or N-channel in a common-source) configuration. For an active-low, current-sinking implementation, the transistor should be PNP in a common-collector (or P-channel in a common-drain) configuration.

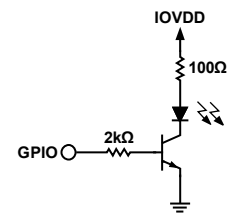


Figure 5. Current Sourcing LED Output Circuit Example

## USING GPIO IN A SIGMASTUDIO PROJECT

GPIO pins are accessible in SigmaStudio via the General Purpose Input and General Purpose Output cells in the IO section of the ToolBox.

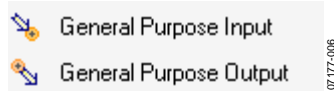


Figure 6. GPIO ToolBox Cells



Figure 7. GPIO Schematic Cells

Click the drop-down menu for access to all available GPIO pins. These cells may be wired just as any other cell in Sigma-Studio, with control signal inputs and outputs shown in red.

Auxiliary ADC input cells work in the same way.



Figure 8. Auxiliary ADC Input Cell

## SETTING UP GPIO PINS IN THE SIGMASTUDIO REGISTER WINDOW

GPIO pins must be independently configured using the Register Control window in SigmaStudio. Examples of GPIO register controls are shown in Figure 9, Figure 10, and Figure 11.

Pin	Value	Direction	Inv
MP0	Low	Input GPIO Debounce	<input checked="" type="checkbox"/>
MP1	Low	Input GPIO No Debounce	<input type="checkbox"/>
MP2	Low	Output GPIO	<input type="checkbox"/>
MP3	Low	Output GPIO Open Collec	<input type="checkbox"/>
MP4	Low	Input Lrclk_in	<input type="checkbox"/>
MP5	Low	Output GPIO	<input type="checkbox"/>
MP6	Low	Input GPIO Debounce	<input checked="" type="checkbox"/>
MP7	Low	Output Sdata_out1	<input type="checkbox"/>
MP8	Low	ADC3	<input type="checkbox"/>
MP9	Low	Output GPIO Open Collec	<input type="checkbox"/>
MP10	Low	In Lrclk_out	<input type="checkbox"/>
MP11	Low	In Bclk_out	<input type="checkbox"/>

Figure 9. GPIO Section of ADAU170x Register Control Window

Register	Addr	Reserved [15:4]	MP Value [3:0]
MP 0	57860	b 000000000000	Input Aux ADC
MP 1	57861	b 000000000000	Input with debounce 10ms
MP 2	57862	b 000000000000	Output driven by control port with pullup
MP 3	57863	b 000000000000	Output driven by core without pullup
MP 4	57864	b 000000000000	Output driven by core without pullup
MP 5	57865	b 000000000000	Input without debounce
MP 6	57866	b 000000000000	Output driven by core with pullup
MP 7	57867	b 000000000000	Input with debounce 20ms
MP 8	57868	b 000000000000	Output CRC error sticky
MP 9	57869	b 000000000000	Output watchdog error sticky
MP 10	57870	b 000000000000	Input with debounce 0.3ms
MP 11	57871	b 000000000000	Input with debounce 0.6ms

Figure 10. GPIO Section of ADAU144x Register Control Window

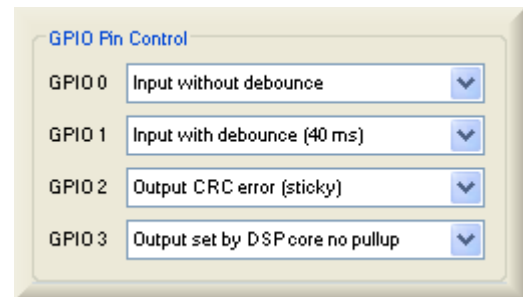


Figure 11. GPIO Section of ADAU176x Register Control Window

Here, the GPIO pins can be set appropriately depending on interface circuitry. Possible settings include Input GPIO Debounce, Input GPIO No Debounce, Output GPIO, Output GPIO Open Collector, and ADC. In addition, some devices can input or output digital audio data or clocks on these pins. On some devices, GPIO pins are labeled MP (for multipurpose), but they are used in the same way.

Consult the relevant device and evaluation board data sheets, available from Analog Devices, Inc., for more details about GPIO register settings.

**Input GPIO Debounce**

When connecting a switch or push-button to a GPIO pin, a common problem that arises is contact bounce (also called chatter). Due to various mechanical and electrical factors, a series of random oscillations may appear during switching. To mitigate these effects, a timing-based debounce circuit is integrated into the GPIO circuits of certain SigmaDSPs. The debounce time can generally be set in the GPIO section of the register control window. The core reads the input value from its associated register once per audio frame.

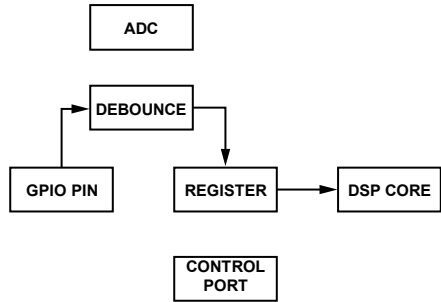


Figure 12. Input GPIO With Debounce Data Flow

**Input GPIO No Debounce**

For inputs not subject to contact bounce effects, such as outputs from external logic ICs, the debounce circuit may be bypassed with this setting. The core reads the input value from its associated register once per audio frame.

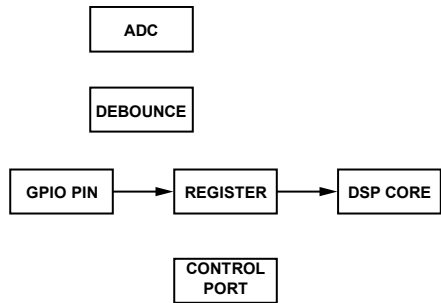


Figure 13. Input GPIO No Debounce Data Flow

**Output GPIO**

This setting allows the pin to be used as a digital output. Typically, each pin can drive a maximum of a few milliamps. See the associated SigmaDSP data sheet for more information. The GPIO pin reads the output value from its associated register once per audio frame.

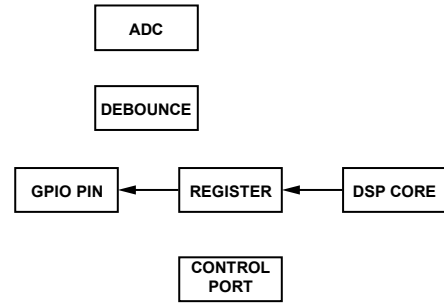


Figure 14. Output GPIO Data Flow

**Output GPIO Open Collector**

This setting puts the pin in an open-collector or open-drain output mode (depending on the device's internal circuitry) and requires an external pull-up resistor. The pull-up resistor can connect to a different IOVDD supply than that of the DSP, thus this mode is useful when interfacing with ICs at different logic levels. The GPIO pin reads the output value from its associated register once per audio frame.

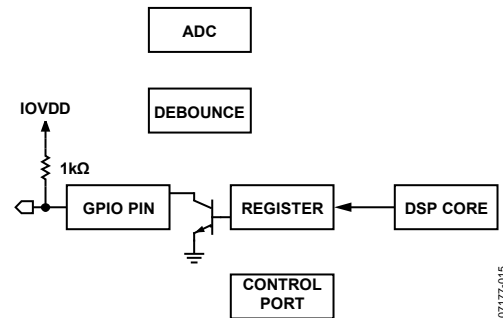


Figure 15. Output GPIO Open Collector Data Flow

**Input Driven by Control Port**

In this mode, the GPIO pin is bypassed, and the core reads its value from the associated register. This register's value can be written or read via the control port. This mode is useful for controlling elements of the signal flow with an external host controller. The core reads the input value from its associated register once per audio frame.

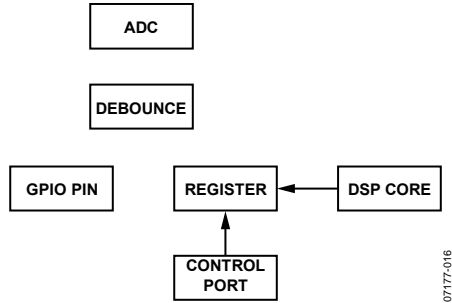


Figure 16. Input Driven by Control Port Data Flow

**Output Driven by Control Port**

In this mode, the signal flow in the core does not affect the output of the associated GPIO pin and the pin reads its output value from the associated register. This register's value can be written or read via the control port.

This mode is useful for directly controlling circuitry, such as an LED, connected to the GPIO pins with an external host controller. The GPIO pin reads the output value from its associated register once per audio frame.

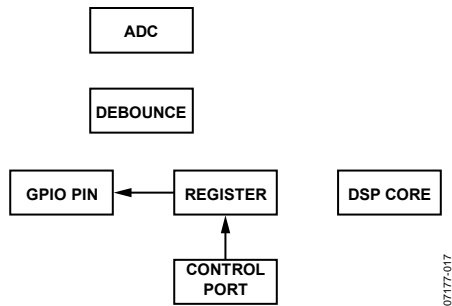


Figure 17. Output Driven By Control Port Data Flow

**ADC**

By setting the pin in ADC mode, it is used as one of the multiplexed inputs to the auxiliary ADC. On the ADAU170x, the invert bit should be activated for proper ADC function. The core reads the input value from its associated register once per audio frame, although the ADC sampling rate is dependent on the particular SigmaDSP being used.

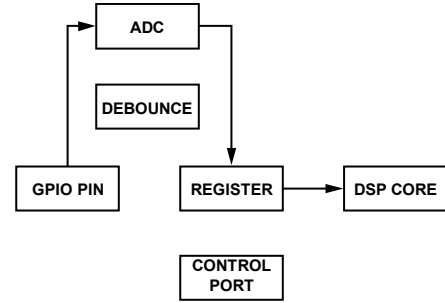


Figure 18. ADC Data Flow

## USING THE EVALUATION BOARD WITH THE GPIO BOARD

Some SigmaDSP evaluation packages include a daughter board with reference GPIO interface circuits. It may be helpful to evaluate GPIO applications or algorithms by using this board.

This evaluation packages include:

- Four potentiometers
- Four LEDs with transistor drivers
- Eight momentary push-buttons
- Six slide switches
- One incremental rotary encoder
- Header connections for serial data I/O
- Prototyping area for user-created interfacing circuitry

Documentation for the daughter board is included in the associated evaluation board kits.

## EXAMPLE IMPLEMENTATIONS OF GPIO IN SOFTWARE

This section provides examples of some commonly used GPIO conditioning signal flows in the context of a SigmaStudio project.

The SigmaStudio software is continually updated with new algorithms, thus the appearance and function of some functional blocks may change over time. However, the ideas presented in this application note should be applicable in all future versions of the software.

The information in this application note corresponds to the Version 3.1 release of SigmaStudio.

### PUSH-BUTTON VOLUME CONTROL—UP/DOWN/MUTE

This example uses two push-buttons to control the pushhold GPIO cell, which in turn controls volume with a look-up table. Pressing both buttons simultaneously mutes the audio output. This example is shown in Figure 20.

Pressing each push-button once increments or decrements the index. If a push-button is depressed for a duration defined by hold (ms), the index increments repeatedly at an interval defined by repeat (ms).

Two GPIO pins, GPIO\_0 and GPIO\_1, are used as inputs to the pushhold U1 cell, running the push/hold with a two-button mute algorithm. The three output pins of this cell, Up Pulse, Down Pulse, and Mute Pulse, are used as inputs to the look-up table, UpDownLUT1. Click **Table** to display the table values.

The table in Figure 19 shows an example of a volume curve ranging from 0 to 1. Press the Up push-button for the cell to output the next number in the table. Press the Down push-button for the cell to output the previous number in the table. Note that the Up and Down push-buttons are not visible in Figure 19.

In addition to control inputs and outputs (shown in red in Figure 20), the look-up table cell has a yellow interface register input and output, which are linked to the Interface Read and

Interface Write cells. These cells enable writeback of parameter values from the ADAU1701 to an external EEPROM on power-down. For more information, see the ADAU1701 data sheet.

The look-up table output is sent as a control input to the SW volume cell. This cell also has two audio inputs (green) and two audio outputs (blue). The control input takes a value from the table and scales the audio accordingly. When a push-button is pressed, the table outputs a new value, and the volume cell slews to this new value based on the SW slew rate entered in the cell.

Index	Value
1	0
2	0
3	0.002
4	0.008
5	0.01
6	0.02
7	0.03
8	0.04
9	0.05
10	0.06
11	0.07
12	0.08
13	0.09
14	0.1
15	0.15
16	0.2
17	0.25
18	0.3
19	0.35
20	0.4
21	0.45
22	0.5
23	0.55
24	0.6
25	0.65
26	0.7
27	0.75
28	0.8
29	0.85
30	0.9
31	0.95
32	0.99
33	1

Figure 19. Push-Button Volume Control—Up/Down/Mute Index Table

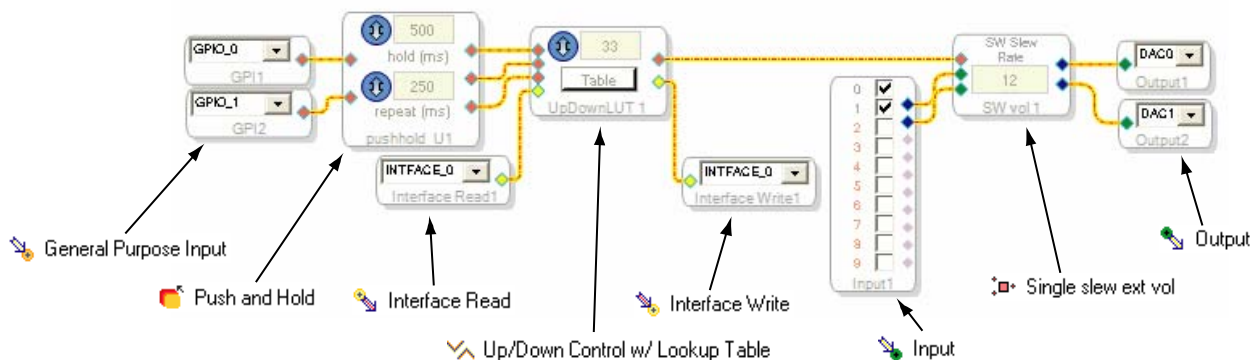


Figure 20. Push-Button Volume Control—Up/Down/Mute Signal Flow

Table 1 shows several 33-point index lists of ascending and descending values for linear and logarithmic volume controls. A logarithmic index table is recommended for audio volume control applications.

**Table 1. Common 33-Point Index Table Values**

Logarithmic (Exponential) Curve		Linear Curve	
Ascending, -96 dB to 0 dB (3 dB per Step)	Descending, 0 dB to -96 dB (3 dB per Step)	Ascending, 0 to 1	Descending, 1 to 0
1.5849E-05	1.0000	0.0000	1.0000
2.2387E-05	0.7079	0.0313	0.9688
3.1623E-05	0.5012	0.0625	0.9375
4.4668E-05	0.3548	0.0938	0.9063
6.3096E-05	0.2512	0.1250	0.8750
8.9125E-05	0.1778	0.1563	0.8438
1.2589E-04	0.1259	0.1875	0.8125
1.7783E-04	0.0891	0.2188	0.7813
2.5119E-04	0.0631	0.2500	0.7500
3.5481E-04	0.0447	0.2813	0.7188
5.0119E-04	0.0316	0.3125	0.6875
7.0795E-04	0.0224	0.3438	0.6563
0.0010	0.0158	0.3750	0.6250
0.0014	0.0112	0.4063	0.5938
0.0020	0.0079	0.4375	0.5625
0.0028	0.0056	0.4688	0.5313
0.0040	0.0040	0.5000	0.5000
0.0056	0.0028	0.5313	0.4688
0.0079	0.0020	0.5625	0.4375
0.0112	0.0014	0.5938	0.4063
0.0158	0.0010	0.6250	0.3750
0.0224	7.0795E-04	0.6563	0.3438
0.0316	5.0119E-04	0.6875	0.3125
0.0447	3.5481E-04	0.7188	0.2813
0.0631	2.5119E-04	0.7500	0.2500
0.0891	1.7783E-04	0.7813	0.2188
0.1259	1.2589E-04	0.8125	0.1875
0.1778	8.9125E-05	0.8438	0.1563
0.2512	6.3096E-05	0.8750	0.1250
0.3548	4.4668E-05	0.9063	0.0938
0.5012	3.1623E-05	0.9375	0.0625
0.7079	2.2387E-05	0.9688	0.0313
1.0000	1.5849E-05	1.0000	0.0000

The number of points in a volume curve is not fixed at 33. It can be changed as required in a given application.

Although values can be entered in the table in floating-point format, they are stored in the SigmaDSP in 5.23 decimal format.

The calculation of gain values for logarithmic and linear curves is shown in Equation 1 and Equation 2.

For an ascending, x-point linear curve, the gain value g for index n can be calculated with Equation 1. The descending curve can be derived by reversing the indices.

$$g_n = \frac{n}{x} \tag{1}$$

where n = 0 to x.

For a descending, x-point exponential curve with gain steps of s dB, the gain value (g) for index n can be calculated with Equation 2. The ascending curve can be derived by reversing the indices.

$$g_n = 10^{\frac{-n \times s}{20}} \tag{2}$$

where n = 0 to x.

The cell names as they appear in the software and the number of each used in this example are as follows: GPIO Input (2), Interface Read (1), Push and Hold (1), Interface Write (1), Input (1), Up/Down Control with Look-up Table (1), Single Slew Ext Vol (1), Output (2).

In SigmaStudio version 3.1 and later, a Push Button Volume cell is available. This cell combines the functionality of the Push and Hold, Up/Down Control, Index Lookup Table, and Single Slew Ext Vol cells. A simplified implementation using this block is shown in Figure 21.

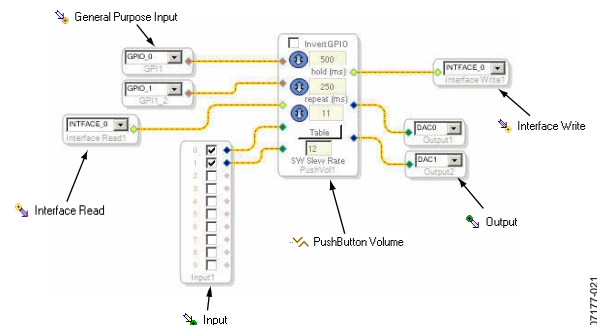


Figure 21. Push Button Volume Simplified Signal Flow

071177-021

**ROTARY ENCODER VOLUME CONTROL**

This example project utilizes the rotary encoder cell as a volume control. The PushHold cell is replaced by the RotEnc cell, with the remainder of the schematic equivalent to the previous example.

The top control input (red) to the RotEnc cell is the up pulse input (see Figure 23). The bottom control input is the down pulse input. The text entry block in the center of the cell sets the time constant in samples, which is 20 in this example.

Note that the back end of this signal flow (everything after the rotary encoder block) is the same as in the push-button volume control example described in the Push-Button Volume Control—Up/Down/Mute section.

The cells used in this example are as follows: GPIO Input (2), Interface Read (1), Rotary Encoder (1), Interface Write (1), Input (1), Single Slew Ext Vol (1), and Output (2).

In SigmaStudio version 3.1 and later, a cell is available that combines the functionality of the Rotary Encoder, Up/Down Control, Index Lookup Table, and Single Slew Ext Vol cells. It is called the Rotary Volume cell. A simplified implementation using this block is shown in Figure 22.

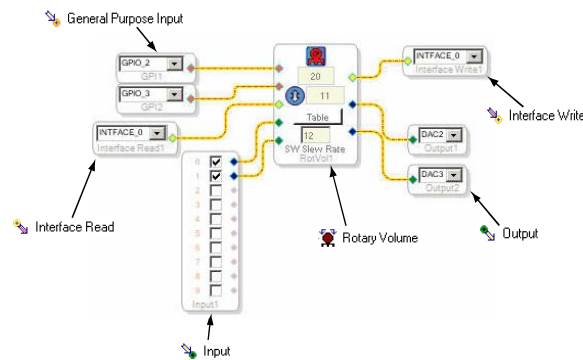


Figure 22. Rotary Volume Simplified Signal Flow

07177-023

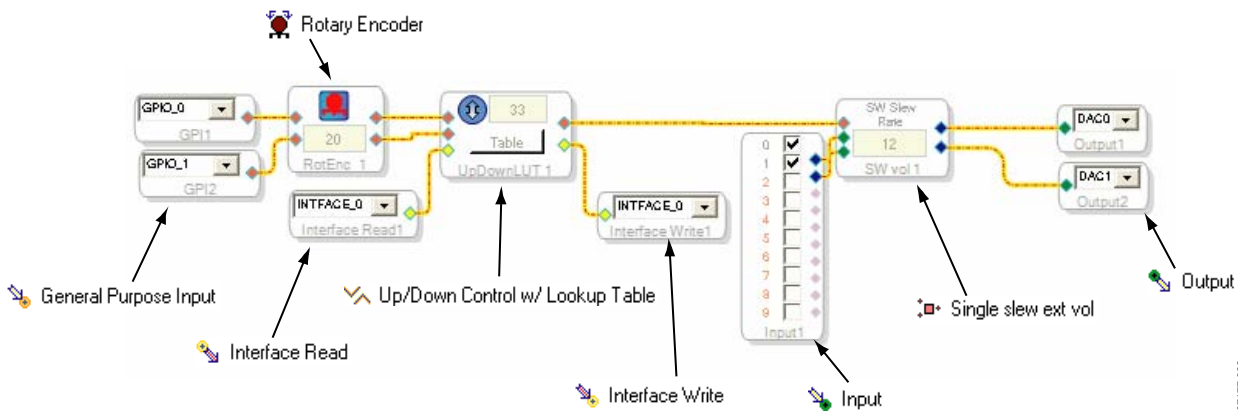


Figure 23. Rotary Encoder Volume Control Signal Flow

07177-022

**PUSH-BUTTON-CONTROLLED DEMULTIPLEXER**

This example utilizes a set of two push-buttons to control a demultiplexer, which allows a 1 kHz sine wave to be output on any of eight possible outputs.

The table entries in this case are slightly different because of the number formats used in the DSP. There are two main formats used by SigmaDSP: integer format (28.0) and decimal format (5.23). All audio data and most parameters are represented in 5.23 decimal format within the DSP. Most GPIO-related signals are represented in 28.0 integer format because it increases the range of allowable values they can take.

In SigmaStudio index tables, the index value is represented in 28.0 integer format, whereas the output is in 5.23 decimal format. However, because this 5.23 output signal is being used to control the demultiplexer (which accepts a 28.0 input format), number formatting is required within the index table.

To convert between 28.0 integer format and 5.23 decimal format, use Equation 3 and Equation 4. Equation 3 converts 28.0 to 5.23 while Equation 4 converts 5.23 to 28.0.

$$n_{5.23} = n_{28.0} \times 2^{-23} \tag{3}$$

$$n_{28.0} = n_{5.23} \times 2^{23} \tag{4}$$

The first eight indices, converted to 5.23 number format, are shown in Figure 24.

Index	Value
1	0
2	1.19209289550781E-07
3	2.38418579101562E-07
4	3.57627868652343E-07
5	4.76837158203125E-07
6	5.96046447753906E-07
7	7.15255737304687E-07
8	8.34465026855468E-07

Figure 24. Push-Button-Controlled Demultiplexer Index Table

The cell names as they appear in the software and the number of each used in this example are as follows: GPIO Input (2), Interface Read (1), Up/Down Control with Lookup Table (1), Interface Write (1), Tone (lookup/sine) (1), Index Selectable Demultiplexer (1), and Output (8).

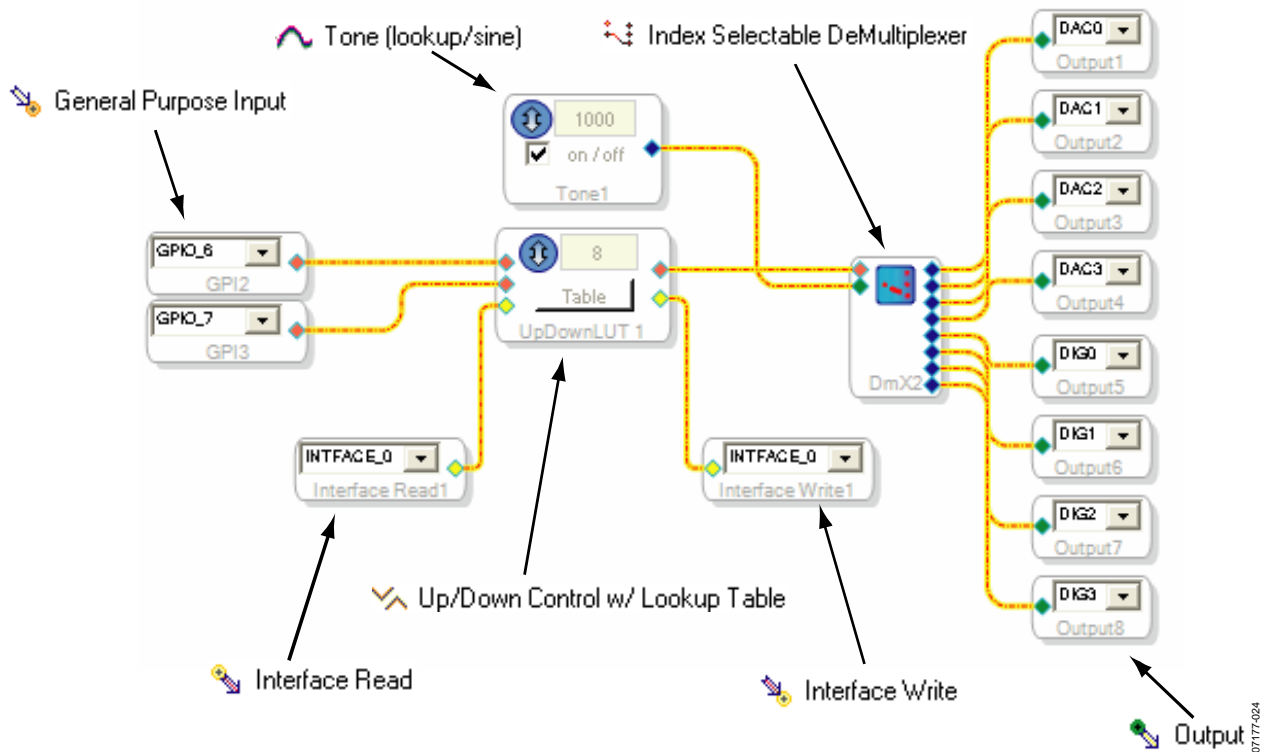


Figure 25. Push-Button-Controlled Demultiplexer Signal Flow

**PUSH-BUTTON-CONTROLLED FILTER**

This project utilizes the push-buttons as described in the previous examples, although it now uses the index value to select from one of four equalizer curves.

The cell names as they appear in the software and the number of each used in this example are as follows: GPIO Input (2), Interface Read (1), Up/Down Control w/ Lookup Table (1), Interface Write (1), Input (1), General (2nd Order/Lookup) (1), and Output (2).

Index	Value
1	0
2	1.19209289550781E-07
3	2.38418579101562E-07
4	3.57627868652343E-07

Figure 26. Push-Button-Controlled Filter Index Table

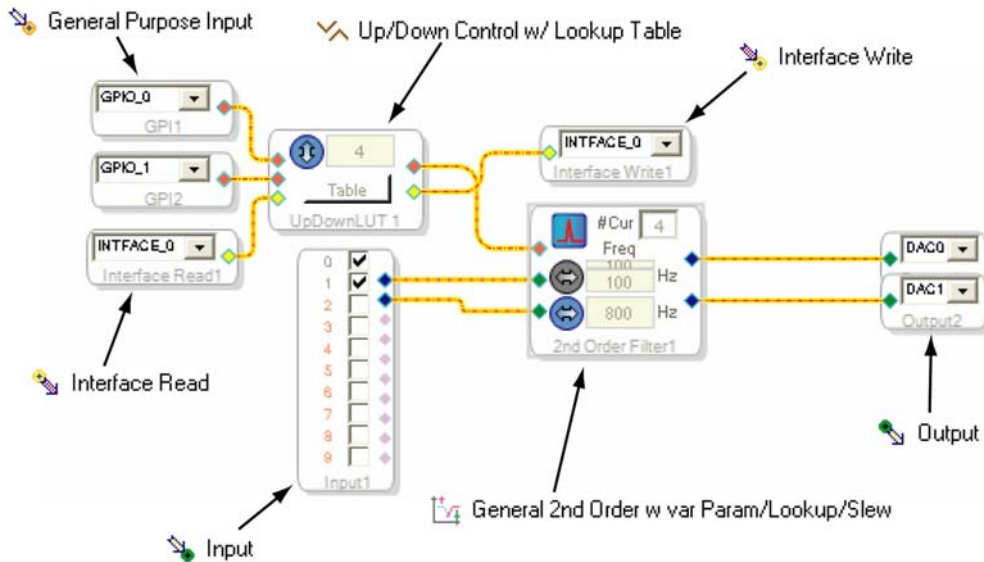


Figure 27. Push-Button-Controlled Filter Signal Flow

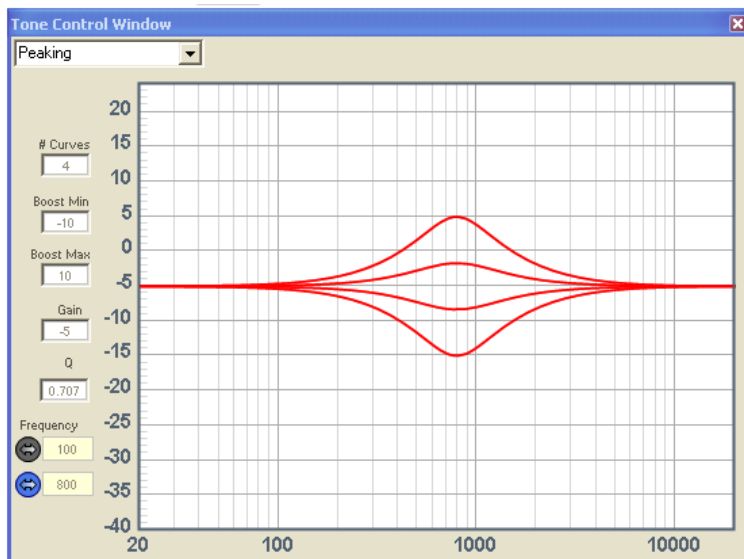


Figure 28. Push-Button-Controlled Filter Tone Control Window

**AUXILIARY ADC VOLUME CONTROL**

Sometimes it is desirable to have analog control of one or more parameters in a system. The most common case is an analog master volume control.

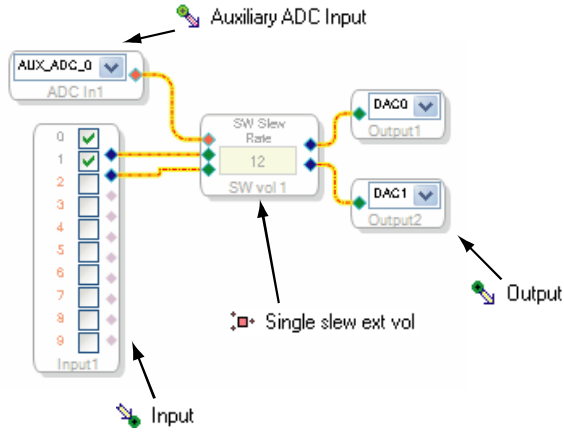


Figure 29. Auxiliary ADC Volume Control Signal Flow

In this example, a SW Slew Volume Control is inserted between the input and output of the audio signal flow. The auxiliary ADC input is connected directly to the control pin on the volume control.

The cell names as they appear in the software and the number of each used in this example are as follows: Auxiliary ADC Input (1), Input (1), Single Slew Ext Vol (1), and Output (1).

**AUXILIARY ADC VOLTAGE-CONTROLLED OSCILLATOR**

In this example, an analog input voltage is used to control the frequency of an oscillator.

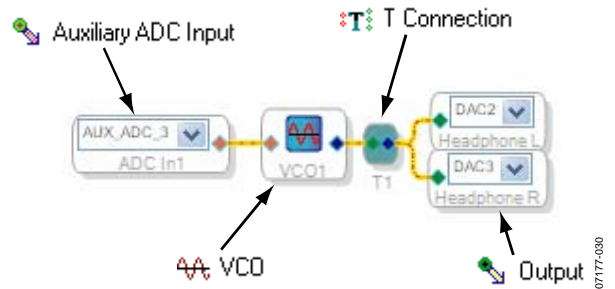


Figure 30. Auxiliary ADC Voltage-Controlled Oscillator Signal Flow

As the input voltage increases, the frequency of the oscillator increases accordingly.

The cell names as they appear in the software and the number of each used in this example are as follows: Auxiliary ADC Input (1), Voltage Controlled Oscillator (1), T Connection (1), and Output (2).

**AUXILIARY ADC SLEW MULTIPLEXER**

An analog input voltage can also be used to select between sources (see Figure 31). In this example, an input voltage is broken down into five equally-sized zones, each associated with a different tone generator.

By multiplying the auxiliary ADC input signal by 5 in 28.0 format (implemented here as a DC input entry cell), the full scale of the input is mapped to five index values: 0, 1, 2, 3, and 4 in 28.0 format, which are appropriate input values for the multiplexer. The multiplexer cell uses this index to output the appropriate sine tone.

The cell names as they appear in the software and the number of each used in this example are as follows: ADC Input (1), DC Input Entry (1), Multiply (1), Tone (lookup/sine) (5), Index Selectable Slewing Mux (1), T Connection (1), and Output (2).

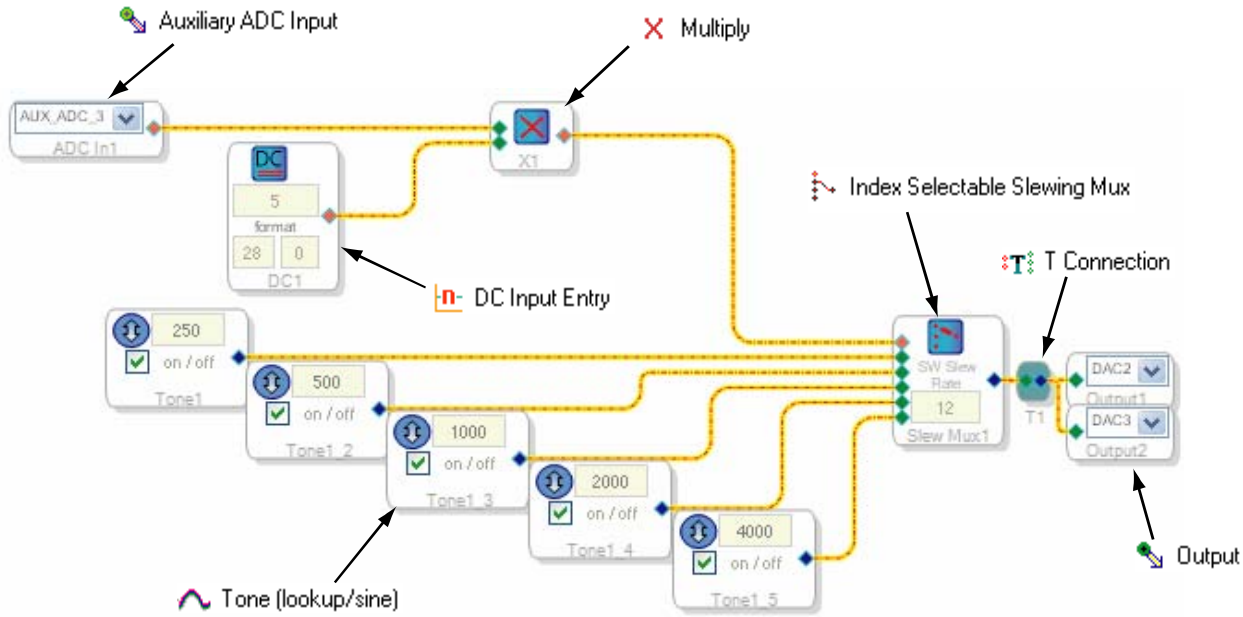


Figure 31. Auxiliary ADC Slew Multiplexer Signal Flow

07177-031

**BLINKING LED**

In the example shown in Figure 32, a square wave generator is used to drive an output LED. Output GPIO pins turn off when their associated register is equal to zero, and turn on when equal to any other value. By taking a square wave with a minimum value of  $-1_{5,23}$  and a maximum of  $1_{5,23}$ , then adding  $1_{5,23}$ , the result is a waveform with a minimum value of  $0_{5,23}$  and maximum value of  $2_{5,23}$ . This causes the LED to blink at a frequency equal to that of the square wave generator (3 Hz in this example).

The cell names as they appear in the software and the number of each used in this example are as follows: DC Input Entry (1), Signal Add (1), Square Wave (1), and GPIO Output (1).

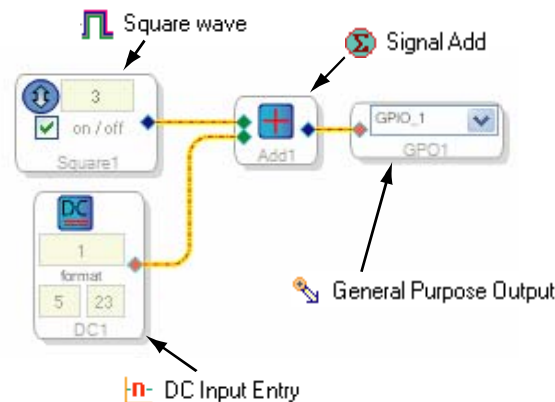


Figure 32. Square Wave LED Driver Signal Flow

07177-032

**NOTES**

**NOTES**