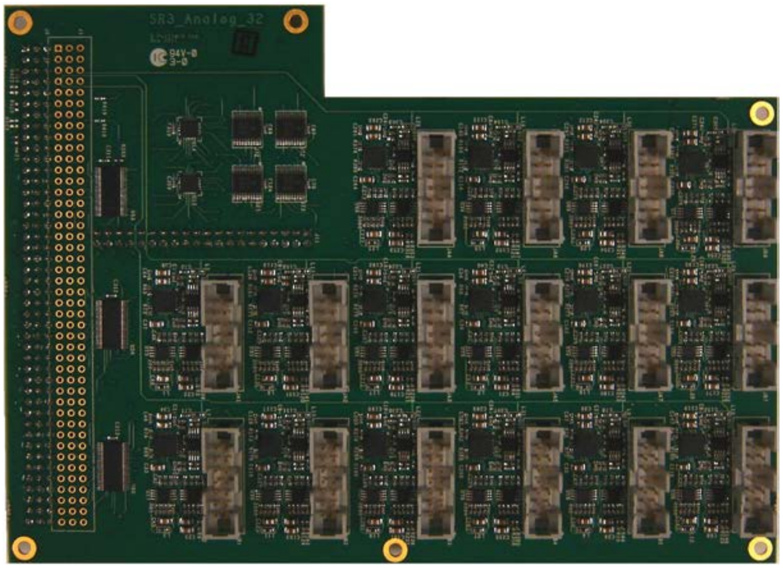
SR3_Analog_32

User's Manual



by

Soft dB

with the collaboration of

Convergence Instruments

March 2nd 2012

1040, avenue Belvédère, suite 215
Québec (Québec) G1S 3G3 Canada
Tél.: (418) 686-0993 Fax: (418) 686-2043

1 Introduction

SR3_Analog_32 is a 32-input/32-output analog board that is designed to work with the *SignalRanger_mk3_Pro* DSP board, in applications such as acoustics and vibration beamforming, and multi-channel control.

It features Sigma-Delta Analog Interface Circuits (1 AIC = 1 ADC + 1 DAC), with bit-depth up to 32 bits and sampling frequencies up to 96 kHz. It has one separate ADC and DAC per channel. Each input path includes differential inputs, as well as electret-microphone inputs

SR3_Analog_32 carries over the FPGA IOs of *SignalRanger_mk3_Pro*, through bus switches for improved reliability and 5V-tolerance.

2 Technical Data

2.1 Analog Inputs

Parameter	Value
Number of channels	32
Sampling Rate	6 kHz, 12 kHz, 24 kHz, 32 kHz, 48 kHz, 96 kHz
Resolution	16, 20, 24 or 32 bits
Gain	0 dB to 47.5 dB (0.5dB resolution)
Noise	65 μV_{RMS} (typ.) @ 0 dB gain 18 μV_{RMS} (typ.) @ +18 dB gain
Signal to Noise Ratio	90.4 dB (typ.) @ 48 kHz, differential input, 0 dB gain, A-filter
Bandwidth	Up to 48 kHz
Input types	• Differential • Electret-Microphone
Coupling	DC Differential Inputs AC (1Hz Cutoff) Electret Microphone Inputs
Dynamic Range (differential inputs)	+-2.8V Single Ended +-5.6V Differential @ 0dB gain
Dynamic Range (electret-microphone inputs)	+-0.9V @ 0 dB gain
Input Impedance	20 kΩ - Single-Ended 40 kΩ - Differential
Anti-Aliasing Filter	Integral: Filter A: best noise 22-sample group-delay Filter B: 17-sample group-delay

Table 1

Note: Not all combinations of *Sampling Frequency*, *Bit-Width* and *Channel Count* are possible.

2.2 Analog Outputs

Parameter	Value
Number of channels	32
Sampling Rate	6 kHz, 12 kHz, 24 kHz, 32 kHz, 48 kHz, 96 kHz
Resolution	16, 20, 24 or 32 bits
Noise (24 kHz bandwidth)	30 μV_{RMS} (typ.)
Signal to Noise Ratio	97 dB typ @ 48 kHz, A-filter
Bandwidth	Up to 48 kHz
Input types	Single Ended
Coupling	DC
Dynamic Range	+-2.8V
Output Impedance	50 Ω
Source/Sink Ability	25 mA
Anti-Aliasing Filter	Integral: Filter A: best noise 22-sample group-delay Filter B: 17-sample group-delay

Table 2

Note: Not all combinations of *Sampling Frequency*, *Bit-Width* and *Channel Count* are possible.

2.3 FPGA IOs

Parameter	Value
Number of IOs	59
Protection	All IOs are seen through switches which isolate the FPGA from external logic if either *SignalRanger_mk3_Pro* is unpowered or *Presence* signal is not active
IO Level	3.3V CMOS (5V-tolerant inputs)

Table 3

3 Connector Maps

3.1 Channel Locations

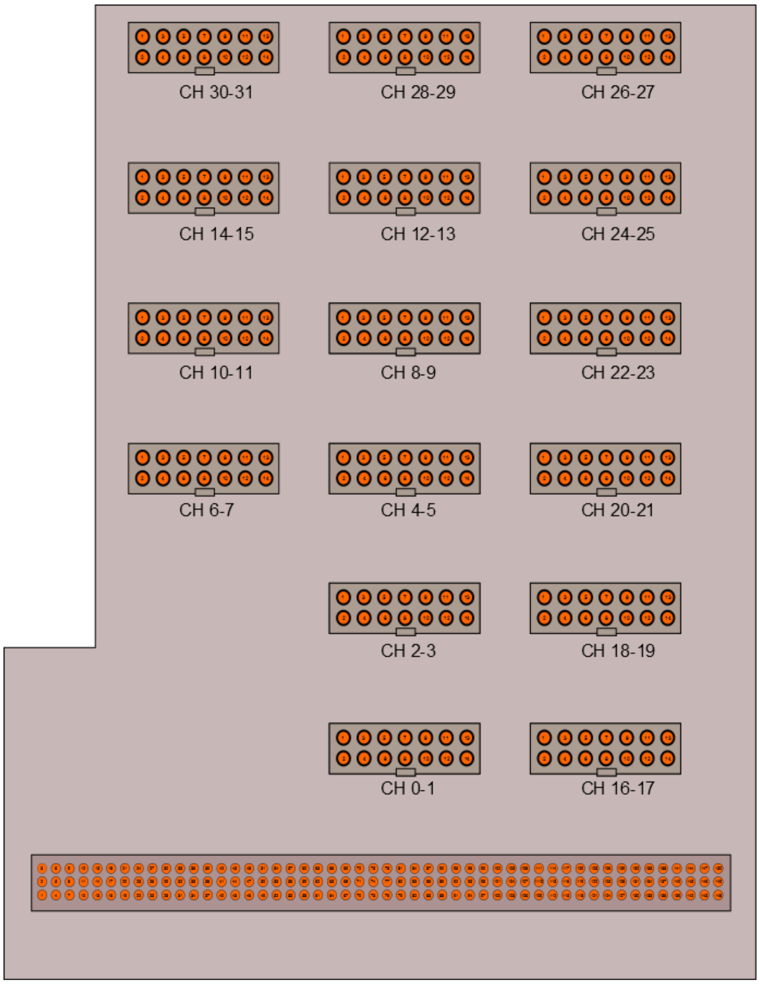


Figure 1 Channel Locations

3.2 Analog Connector Pinouts

Each analog connector (JA*i*, JB*i*) supports a pair of channels. The even channel is on the even pin row. The following odd channel is on the odd pin row.

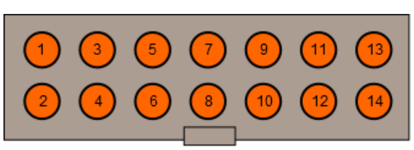


Figure 2 JA*i* – JB*i* connector pinouts

No	Function	No	Function
1	Out_1	2	Out_0

3	Gnd	4	Gnd
5	+In_1	6	+In_0
7	-In_1	8	-In_0
9	Gnd	10	Gnd
11	Electret_Mic_1	12	Electret_Mic_0
13	Electret_Polarization Voltage	14	Electret_Polarization Voltage

Table 4 Connector J1

Note: The table above shows channels 0 and 1. The pinout is the same for channels 2 and 3, 4 and 5…etc.

3.3 Expansion Connector Pinout

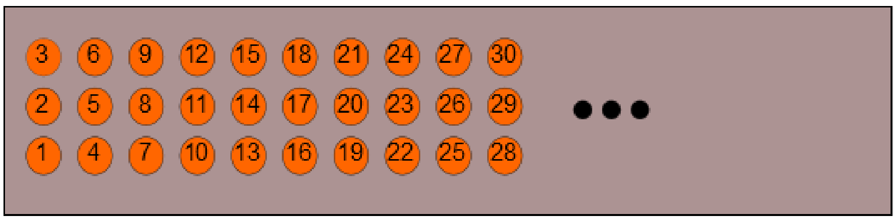


Figure 3 Connector J1

No	Function	No	Function	No	Function
1	+5V	2	Gnd	3	+2.5V
4	+1.8V	5	Gnd	6	+1.2V
7	NC	8	Gnd	9	+3.3V
10	-3.3V	11	Gnd	12	DSP_Reset
13	NC	14	Gnd	15	NC
16	NC	17	Gnd	18	NC
19	NC	20	Gnd	21	NC
22	NC	23	Gnd	24	NC
25	NC	26	Gnd	27	NC
28	NC	29	Gnd	30	NC
31	UART_Rx	32	Gnd	33	UART_Tx
34	NC	35	Gnd	36	Presence
37	FPGA_17	38	Gnd	39	NC
40	FPGA_14	41	Gnd	42	NC
43	FPGA_12	44	Gnd	45	NC
46	FPGA_10	47	Gnd	48	FPGA_8
49	FPGA_7	50	Gnd	51	FPGA_6
52	FPGA_5	53	Gnd	54	FPGA_4
55	FPGA_2	56	Gnd	57	FPGA_1
58	FPGA_141	59	Gnd	60	FPGA_140
61	FPGA_137	62	Gnd	63	FPGA_135
64	FPGA_68	65	Gnd	66	FPGA_69
67	FPGA_70	68	Gnd	69	FPGA_73
70	FPGA_74	71	Gnd	72	FPGA_76
73	FPGA_77	74	Gnd	75	FPGA_78
76	FPGA_79	77	Gnd	78	FPGA_80
79	FPGA_82	80	Gnd	81	FPGA_83
82	FPGA_84	83	Gnd	84	FPGA_85

85	FPGA_86	86	Gnd	87	FPGA_87
88	FPGA_89	89	Gnd	90	FPGA_90
91	FPGA_92	92	Gnd	93	FPGA_93
94	FPGA_95	95	Gnd	96	FPGA_96
97	FPGA_97	98	Gnd	99	FPGA_98
100	FPGA_99	101	Gnd	102	FPGA_100
103	FPGA_102	104	Gnd	105	FPGA_103
106	FPGA_104	107	Gnd	108	FPGA_105
109	FPGA_107	110	Gnd	111	FPGA_108
112	FPGA_112	113	Gnd	114	FPGA_113
115	FPGA_116	116	Gnd	117	FPGA_118
118	FPGA_119	119	Gnd	120	FPGA_122
121	FPGA_123	122	Gnd	123	FPGA_124
124	FPGA_125	125	Gnd	126	FPGA_129
127	FPGA_130	128	Gnd	129	FPGA_131
130	FPGA_132	131	Gnd	132	FPGA_HS_EN

Table 5 Connector J1

3.3.1 Bus Switches

The expansion signals on J1 are connected to the corresponding FPGA and DSP pins on *Signal_Ranger_mk3_Pro* through bus switches.

A *Presence* input is provided on J1 pin 36. This input is used to activate the bus switches that otherwise isolate all the signals of J1 from the DSP board.

The bus switches provide two functions:

- They isolate the DSP board from a user board connected on J1 when one board is powered but not the other. This is essential because in the absence of the switches, line drivers on one board could be driving unpowered input stages on the other, which could damage them.
- They provide level translation between the user-board, which can drive levels up to 5V and inputs on the DSP board that are not 5V-tolerant. In short, the switches make the DSP board inputs 5V-tolerant.

To provide the first function, the switches should only be activated (placed in low-impedance) when the user daughter board is powered. The *Presence* input serves that purpose. This input is normally pulled-up by a 10 kΩ resistor on *SR3_Analog_32*, which keeps the switches open. The *Presence* pin should be pulled low by an active open-drain driver on the user's daughter board. That driver should only pull low and close the switches, when the daughter board is properly powered. Pulling *Presence* low when the user's daughter board is not properly powered (before it is powered or after it is un-powered) exposes input stages on the user's board to damage inflicted by FPGA pins that may be configured as outputs. Depending on how the FPGA is configured at the time it may not be a problem. For instance if the FPGA is not configured, its IOs are floating and will not drive any current. In this case, *Presence* may be permanently tied to ground. Pulling *Presence* low when the user's daughter board is properly powered but the DSP board is not DOES NOT EXPOSE the FPGA inputs to damage that may be inflicted by line drivers on the user's daughter board. This is because the switches are open whenever the DSP board is unpowered.

The level-translation function is provided automatically whenever the switches are activated (whenever *Presence* is low).

3.3.2 Power Supply Pins

3.3.2.1 +5V
This is the same supply that is brought to the 5V power connector J3. The maximum current that may be drawn from this pin is 200 mA.

3.3.2.2 Other Supply Pins
It is not recommended to draw power from any of the other supply pins.

4 Installation

4.1 Software Installation

Unzip the *SR3_Applications_Installer_Vxxx.zip* file and run setup.exe. This installs the following:

- A directory *SR3_Applications* in *C:\Program Files* (if it did not exist already) containing documentation, LabVIEW libraries, Test applications, DSP code examples…etc.
- The driver for the *SignalRanger_mk3_Pro* DSP board
- Shortcuts to compiled test applications and documentation.

4.2 Development Resources

LabVIEW developers can find all the relevant resources in the following distribution: *SR3_DDCI_Library_Distribution_Vxxx.zip*. Those development resources include Vis and DSP code that are specific to the *SR3_Analog_32* IO board. See the *SignalRanger_mk3_Pro_UsersManual* for more information about those development resources.

4.3 FPGA Configuration

The FPGA on the DSP board is not used to manage the ADCs and DACs on *SR3_Analog_32*. The user has complete freedom in the logic that is implemented on the FPGA

4.4 Mating *SR3_Analog_32* to the DSP board

When *SR3_Analog_32* is purchased with a DSP board the two boards are already mated. If not *SR3_Analog_32* must be mated to the DSP board prior to powering-up. Never attempt to mate the two boards while the DSP board is powered.

5 Software

5.1 Self-Test Application

Two applications are provided to perform diagnostics on the *SR3_Analog_32* board:

- *SR3_PRO_A32_SelfTest_Simple* This application tests all the digital aspects of the board, including the AICs. That application runs quickly and does not require any loopback on the analog connectors. It can be performed at any time.
- *SR3_PRO_A32_SelfTest_LpBk* This application additionally tests the analog performance of the input and output channels. It requires special loopback connectors to be inserted into the analog connectors before the test. The loopback connectors connect the

output of each channel to both the positive terminal of the differential input, and the electret microphone input. It also connects the negative terminal of the differential input to ground.

5.1.1 SR3_PRO_A32_SelfTest_Simple

To perform the simple test:

- Power-up the board and connect it to the USB port of the PC.
- Run the *SR3_PRO_A32_SelfTest_Simple* application (or run the corresponding VI from the LabVIEW development environment).

The application performs all the tests in sequence and presents the results in the form of a green (good) or red (not-good) indicator. When the board is operating properly all indicators should be green.

The application presents a dialog just before running the FPGA test. That test configures the FPGA pins as outputs and drives patterns of zeros and ones on the pins. If the board is connected to external user logic that test could damage such logic. In such a situation we recommend to skip the FPGA test.

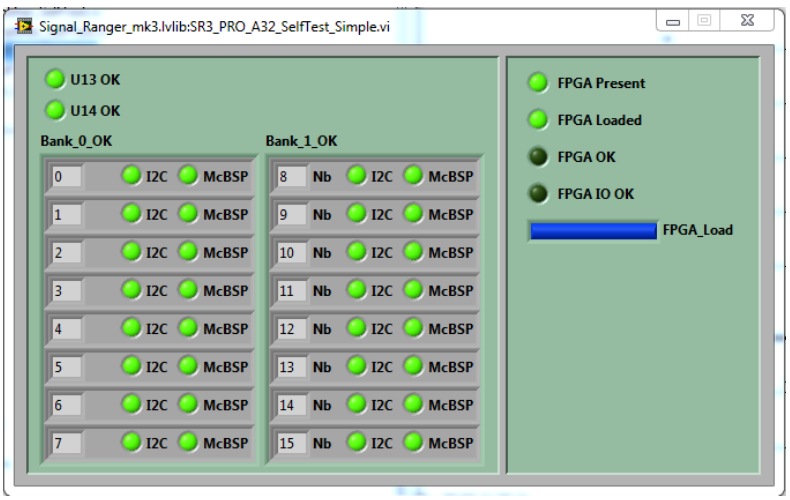


Figure 4

5.1.2 SR3_PRO_A32_SelfTest_LpBk

To perform this more thorough loopback test:

- Insert the loopback adapter boards into all the analog connectors.
- Power-up the board and connect it to the USB port of the PC.
- Run the *SR3_PRO_A32_SelfTest_LpBk* application (or run the corresponding VI from the LabVIEW development environment).

The application performs all the tests in sequence and presents the results in the form of a green (good) or red (not-good) indicator. When the board is functional all indicators should be green.

The application presents a dialog just before running the FPGA test. That test configures the FPGA pins as outputs and drives patterns of zeros and ones on the pins. If the board is connected to external user logic that test could damage such logic. In such a situation we recommend to skip the FPGA test.

Note: The channel indicators are only green on the channels that have the loopback board (channel 0 and 1 below).

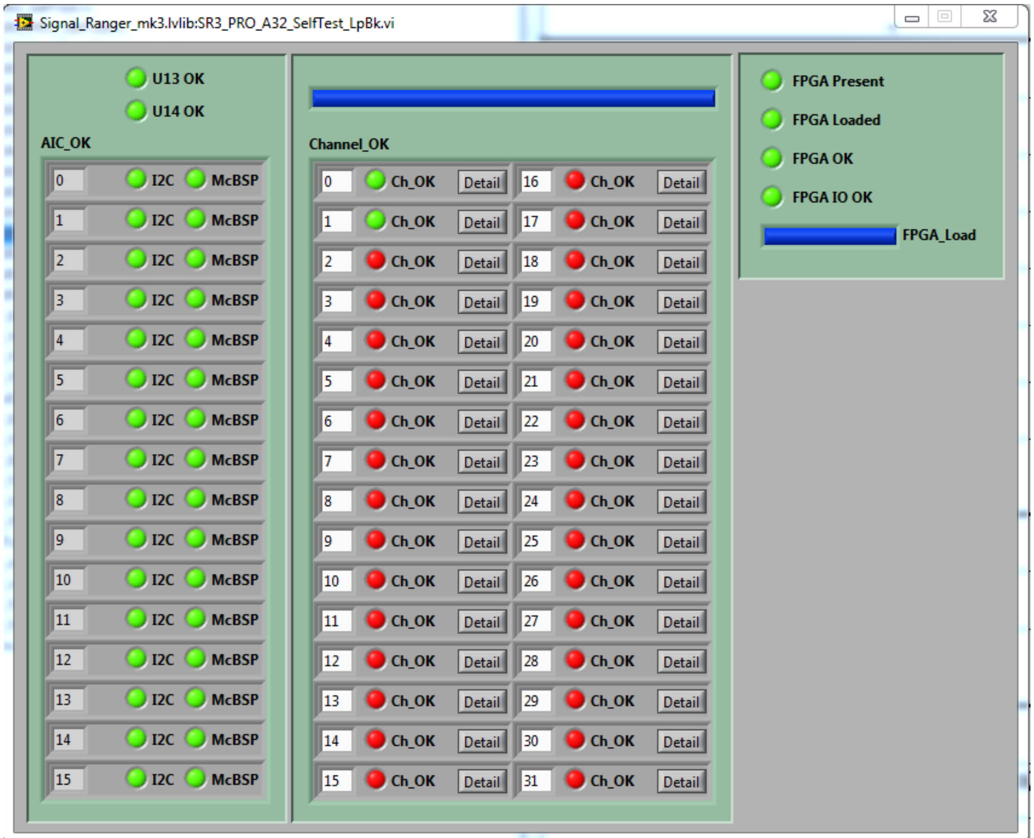


Figure 5

When the *Detail* button of a channel is pressed, a detail window is presented (see Figure 6).

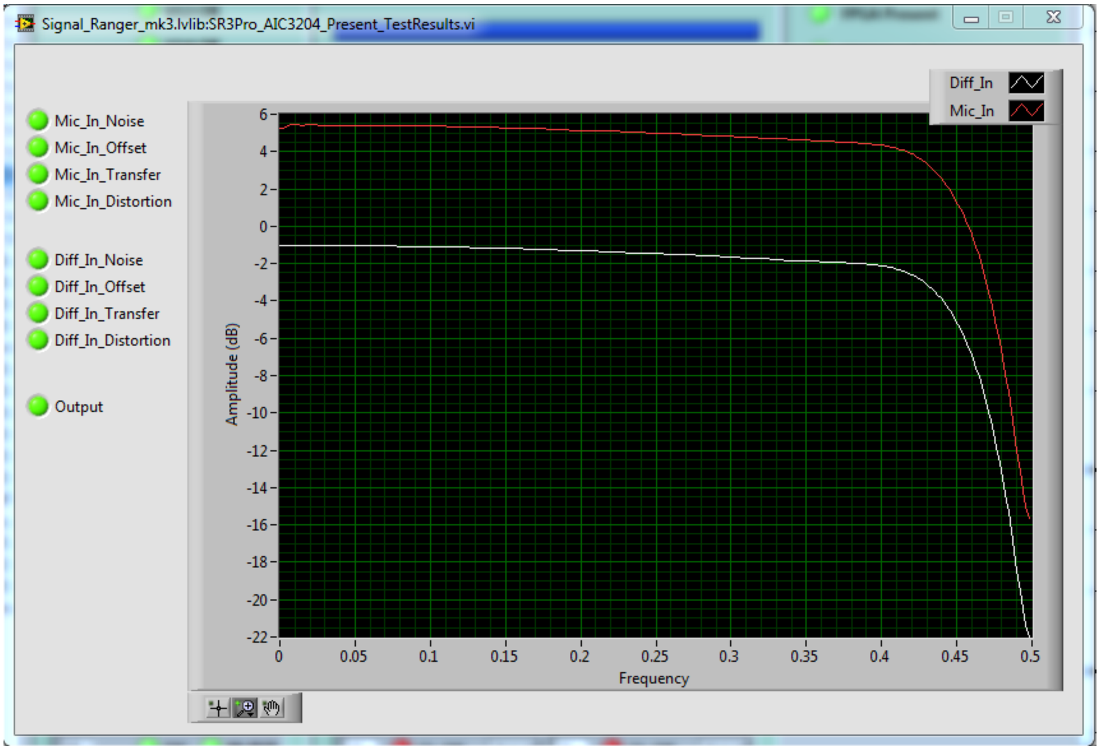


Figure 6

5.2 Demo Application

SR3_PRO_SignalTracker_A32 demonstrates the features and performance of the *SR3_Analog_32* IO board.

The application allows the user to send test signals to a selected output, and monitor the sampled signal on a selected input. Analog inputs are displayed both in terms of time signals, as well as instantaneous or averaged energy spectra. Averaged energy spectra are useful to assess the input noise.

The front-panel of the application is divided into several tabs, one for each functional group.

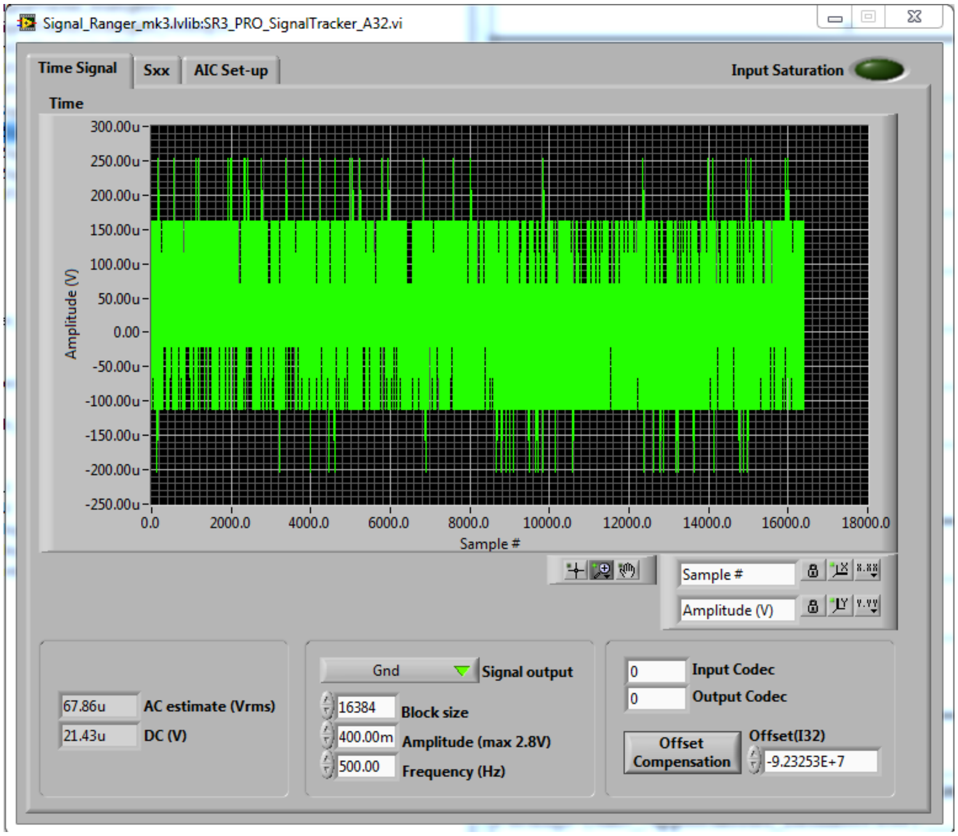


Figure 7 SR3_Pro_SignalTracker_A32 - Time Signal Tab

To start the application, simply click on the white arrow at the top-left of the window.

The application sends blocks of samples of the specified length and waveform to the selected output, and records blocks of samples of the same length on the selected input. The recorded input samples are synchronous to the output samples, with a fixed and known time relationship between input and output.

5.2.1 Time Signal Tab

5.2.1.1 Time Indicator

The *Time-Signal* tab presents a time plot of the signal sampled on the selected input. The amplitude scale takes into account the range of the ADC, so that the signal amplitude is represented in Volts at the connector.

5.2.1.2 AC estimate (Vrms) Indicator

This indicator presents the RMS value of the input signal (any DC offset is removed before the RMS calculation).

5.2.1.3 DC (V) Indicator

This indicator presents the average DC value of the recorded time signal.

5.2.2 Sxx Tab

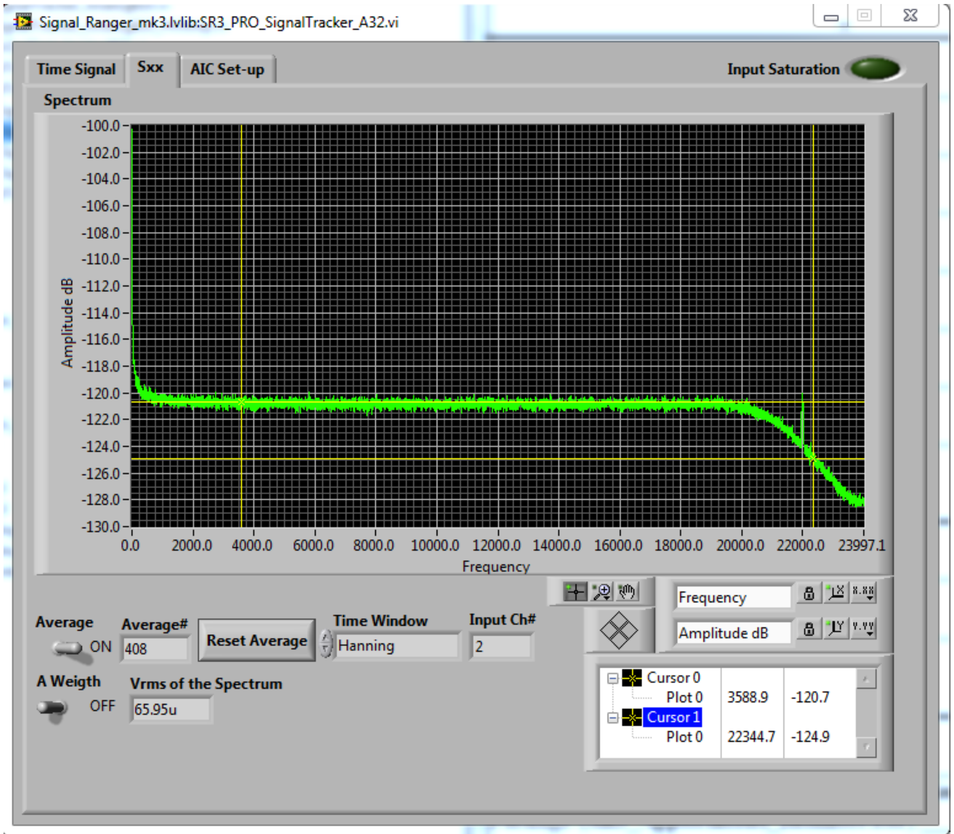


Figure 8 SR3_Pro_SignalTracker_A32 - Sxx Tab

5.2.2.1 Spectrum Indicator

The *Spectrum* indicator presents the instantaneous or averaged power spectrum of the input sampled block.

5.2.2.2 Average Control

To average the power-spectrum, simply place the *Average* control in the ON position.

5.2.2.3 Reset Average Button

The *Reset Average* button resets the averaging process.

5.2.2.4 Time Window selector

An optional weighting window can be chosen from the *Time-Window* list.

5.2.2.5 Graph and Zoom Controls

Graph controls can be used to change the zoom factor. By default the plot is auto-scaled in X and Y, which is indicated by the closed locks beside each scale name. To disable auto-scale, simply press the lock button.

5.2.3 Acquisition Set-Up Tab

The *Acquisition Set-Up* tab presents the various controls for the acquisition set-up.

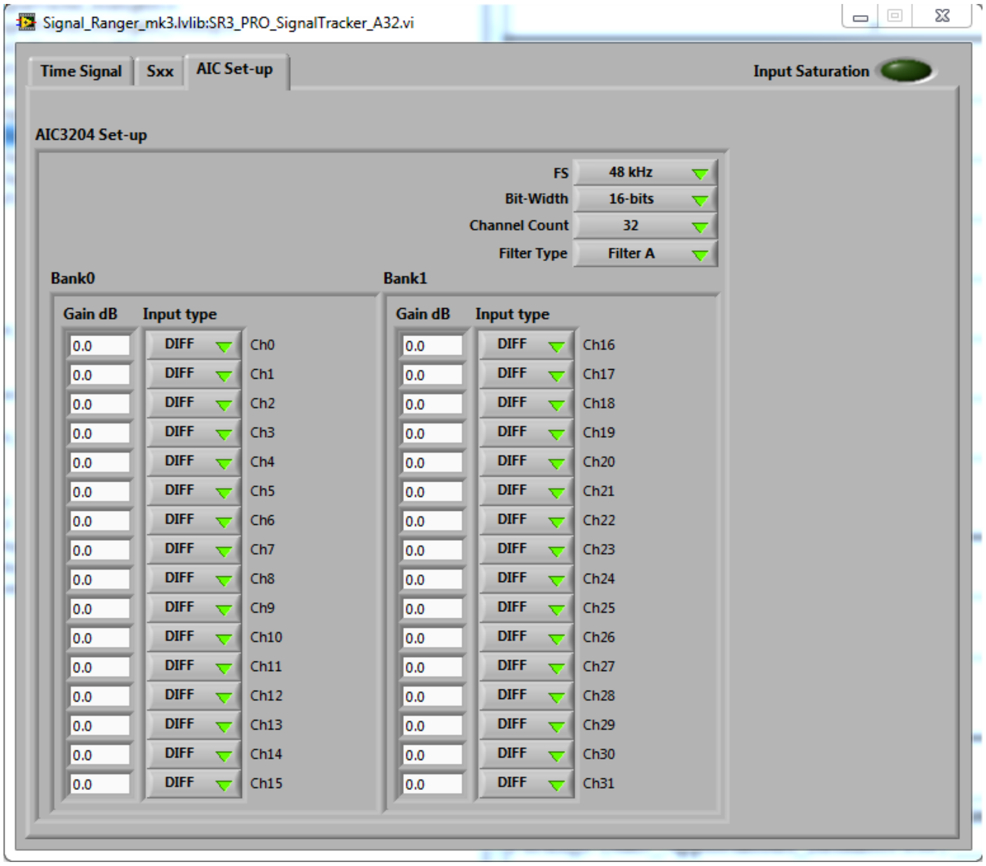


Figure 9 SR3_Pro_SignalTracker_A32 – AIC Setup Tab

The AIC Setup tab presents two types of information:

- Global Controls, such as Sampling Frequency or Bit-Width
- Per-Channel Controls, such as Input Type or Gain

Note: Not all combinations of *Sampling Frequency*, *Bit-Width* and *Channel Count* are possible. When a combination is not allowed, a dialog is presented to prompt the user to change the configuration.

5.2.3.1 FS

The sampling frequency can be set from 6 kHz to 96 kHz. Be aware that sampling frequencies below 48 kHz introduce additional output noise in the 24 kHz bandwidth.

5.2.3.2 Bit-Width

The sample bit-width is selectable between 16-bit, 20-bit, 24-bit and 32-bit. Bit-widths above 16-bit require more bandwidth and are incompatible with large channel counts.

5.2.3.3 Channel Count

The channel count is selectable between 8, 12 16, 20, 24 and 32 channels. When using channel counts below 32 the channels in operation are the first channels of each bank. For instance, when using 12 channels, the channel numbers are: 0, 1, 2, 3, 4, 5, 16, 17, 18, 19, 20 and 21.

5.2.3.4 Filter Type

The selection provides two reconstruction filters:

- **Filter A** has a longer group-delay (22 samples on inputs and outputs), but provides better rejection of out-of-band components, and better noise figures.

- **Filter B** provides a shorter group-delay (17 samples on inputs and outputs).

5.2.3.5 Gain dB
The input gain can be adjusted individually on each input from 0 dB to +47.5 dB. A higher gain provides a lower noise level, but also reduces the dynamic range correspondingly.

5.2.3.6 Input Type
Each channel can be set for a differential input or an electret microphone input.

5.3 *SR3_Analog_32* Drivers and Example-Code

5.3.1 Overview

A DSP driver for the analog I/Os is provided, together with the DSP code of the *SR3_PRO_SignalTracker_A32* demo application that uses that driver. An empty shell project is also provided for users who want to develop their own DSP application.

Source code for the analog IO driver resides in the folder named *SR3PRO_A32Driver*.

The source code for the *SR3_PRO_SignalTracker_A32* demo application resides in the folder named *SR3_PRO_A32_SignalTracker*

The source code for the empty shell project resides in the folders named *SR3_PRO_A32_IOShell*

The shell project constitutes an excellent starting point for developing DSP code that uses the *SR3_Analog_32* board.

The driver has been optimized in assembly language, but can be used in either C, or assembly language. It takes the form of a DSP object library *SR3PRO_AIC3204Driver.lib.* The driver contains C-callable functions to configure and use the analog IOs. A function named *Dataprocess()* is provided in C, where developers can conveniently place their own analog I/O processing code.

The driver uses the DMA to communicate with the analog IOs for maximum efficiency.

5.3.2 User-Accessible Structures and Functions

The *SR3PRO_AIC3204Driver.lib* library defines and allocates the following user-accessible structures, functions and variables (all defined in *SR3PRO_AIC3204_AICDriver.h*):

5.3.2.1 McBSP_Regs[4]
This vector contains the configuration data for the McBSP. We suggest using the VI *SR3Pro_AIC3204_Generate_AICRegistersApp.vi* or the .exe version of this VI to generate the content of this vector.

5.3.2.2 AIC3204_Regs[1344]
This vector contains the configuration data for the AICs. We suggest using the VI *SR3Pro_AIC3204_Generate_AICRegistersApp.vi* or the .exe version of this VI to generate the content of this vector.

5.3.2.3 Input-Output AIC Structure
The structure is allocated by the driver library as follows, and it is designed to contain the input and output samples to/from the AICs. The user DSP code has one complete sampling period to execute the

Dataprocess() function. If the function is not completed within a sampling period input samples are overwritten by the new samples, and the previous output samples are sent to the AICs.

BIN0 and BOUT0 represents the first ADC/DAC bank and BIN1 and BOUT1 represents the second ADC/DAC bank. Following the data format used, the user can select the I32, I24, I20 and I16 member of the union for input and output data.

Note: The structure is allocated automatically as a consequence of including the driver library within the user DSP code. There is no need for the user DSP code to allocate this structure.

5.3.2.4 StartAIC3204();

This function configures the analog IOs and starts the conversion process. It has no arguments. It uses the configuration values found in the *McBSP_Regs[4]* and *AIC3204_Regs[1344]* variables and configures the analog IOs accordingly. These variables must be initialized prior to calling *StartAIC3204()* (use the dedicated VI or application to do that). After the execution of this function, the user-defined processing function called *Dataprocess()* starts being triggered at each sampling period.

5.3.2.5 StopAIC3204();

This function stops the AIC conversion process. After the execution of this function, the user-defined *Dataprocess()* function is no longer triggered.

5.3.2.6 Dataprocess()

This function is declared by the AIC driver library, but must be provided by the user. It usually contains the DSP code that reads the input samples from the AIC structure, performs the signal processing between the inputs and outputs and writes the output samples to the AIC structure. The *_Dataprocess* symbol must be declared using the **.global** directive when the code is written in assembler

Note: The *Dataprocess()* function is a standard function and no special protection is required since the acquisition driver itself protects the context.

Just before the entry into *Dataprocess()* the global interrupt mask (CSR) is set. Therefore all interrupts are disabled. The local mask registers are saved on the stack and restored after the execution of *Dataprocess()*. If the user needs to unmask another interrupt within *Dataprocess()* they must be mindful of timing issues. The execution of *Dataprocess()* must complete within one sampling period. Also all the interrupt masks are restored to their state prior to the execution of *Dataprocess()* at the function's exit. Therefore any interrupt explicitly enabled by user-code in *Dataprocess(),* that was not enabled before entry, is disabled at exit.

5.3.3 Used Resources

The AIC driver uses the following hardware resources:

- DMA channels 2 to 5 are used by the driver.
- Reload DMA parameters 66 to 73 are used by the driver.
- INT5, linked to DMA channel 3 (region 0) is used by the driver.
- GPIO22 and 23 are used to control the reset of the AIC and I2C switches.
- McBSP0 and McBSP1 are used by the AIC driver.
- The I2C controller is set at 372 kHz and used by the AIC driver.

- Clkout0 is connected to the AIC Master-Clocks and set to 24 MHz.

- The driver uses 4576 bytes of code and 2122 bytes of data.

5.3.4 Restrictions

When developing C or assembly code using the AIC driver, the following restrictions apply:

- The user-defined I/O processing function *Dataprocess()* must be present in the user code. The AIC driver does the full context save before calling *Dataprocess()*. If the function is written in assembler, the *_Dataprocess* symbol must be defined using the **.global** directive.

- All C-accessible symbols and labels defined in the *SR3PRO_AIC3204Driver.lib* library must have a _ prefix when used in assembly language.

- The INT5, linked to the DMA channel 3 (region 0), the GPIO22 and 23, the DMA channels 2 to 5 and 66 to 73 are used by the AIC driver. The vectors for these interrupts must be properly declared. See the *SR3_PRO_SignalTracker_A32* or *SR3_PRO_A32_IOShell* DSP code projects for examples of a correct implementation of *vectors.asm*.