USER MANUAL

ProDAQ Data Acquisition Function Cards

ProDAQ 3430 4-Channel, Sigma-Delta ADC Function Card

PUBLICATION NUMBER: 3430-XX-UM-0100

Bustec Production, Ltd.
Bustec House, Shannon Business Park, Shannon, Co. Clare, Ireland
Tel: +353 (0) 61 707100, FAX: +353 (0) 61 707106

PROPRIETARY NOTICE

This document and the technical data herein disclosed, are proprietary to Bustec Production Ltd., and shall not, without express written permission of Bustec Production Ltd, be used, in whole or in part to solicit quotations from a competitive source or used for manufacture by anyone other than Bustec Production Ltd. The information herein has been developed at private expense, and may only be used for operation and maintenance reference purposes or for purposes of engineering evaluation and incorporation into technical specifications and other documents, which specify procurement of products from Bustec Production Ltd. This document is subject to change without further notification. Bustec Production Ltd. reserves the right to change both the hardware and software described herein.

TABLE OF CONTENTS

Glossary

SD-ADC	:	**S**igma **D**elta **ADC** function card
DA	:	Data Acquisition
DE	:	Direct Error line
DI	:	Direct Interrupt line
FC	:	Function Card
FP	:	Front Panel
FS	:	Full Scale
MB	:	Mother Board
RW	:	Read Write operations
RO	:	Read Only
RC	:	Read only with Clear of status information after access
RCW	:	Read with Clear on status information, Write
RWC	:	Read, Write with Clear on status information after access or after end of issued action
SM	:	Switch Matrix
WO	:	Write Only
Rec.	:	Recommended
Req.	:	Required
Res.	:	Reserved
opt.	:	Optional

1. Functional description

1.1 General information

The **3430,** a 4-channel Sigma-Delta ADC function card, is an add-on card to use together with the Mother Board.

It provides the following functions:

- Four analog channels of simultaneous sampling
- Programmable gain of 1, 2, 5, 10, 20, 50, 100 or 1, 2, 5, 10, 100, 200, 500, 1000
- Max. Input Range of ±10V
- DC/AC coupling
- Differential or Single-Ended Input
- On-board 1MΩ or 50Ω termination
- DAC offset for Single-Ended Input
- Output word rate of 625 kHz
- External synchronisation and clock inputs
- On-board FIR decimation (10 stages)
- On-board FIFO of the size of 32 ksamples

1.2 Block diagram

The block diagram of the 3430 is shown on the Figure 1.

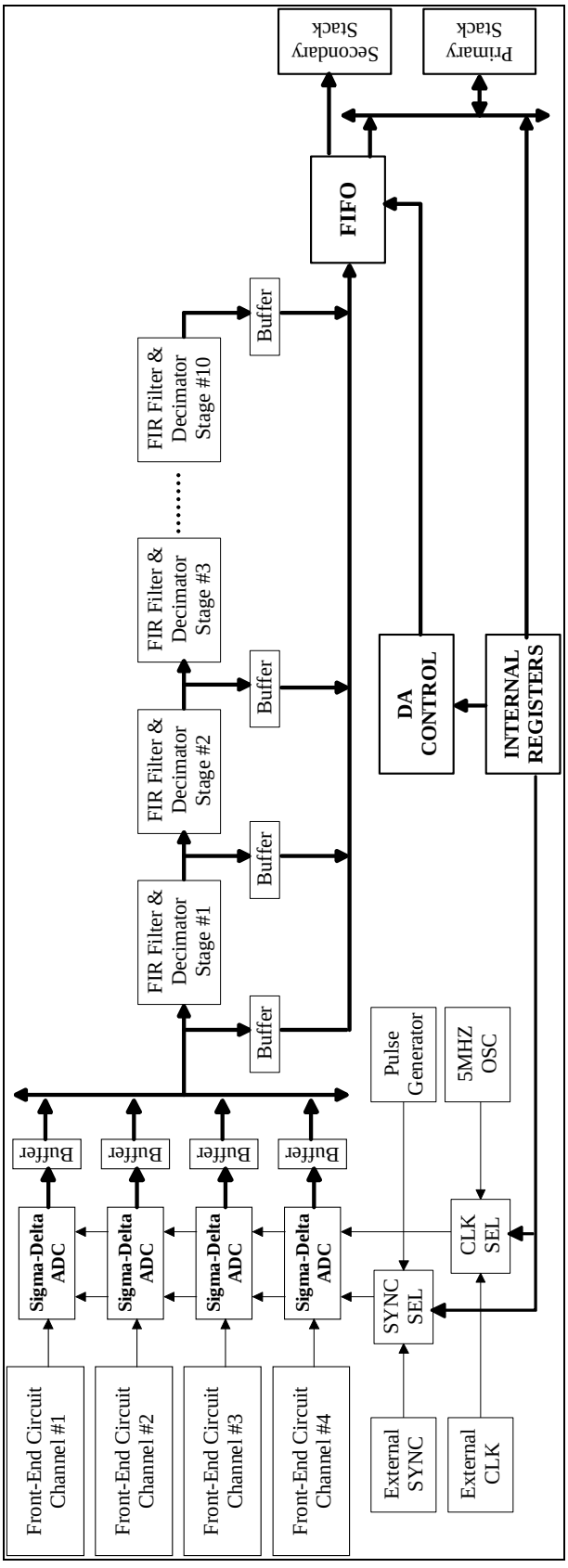

Figure 1: 3430 block diagram

1.3 Description

The 3430 is a four-channel sigma-delta ADC board. Every channel contains front-end conditioning circuitry followed by a sigma-delta ADC. The bit stream from every ADC is then directed to FIR decimation filters. There are up to 10 stages of the FIR filters, which are connected serially. The user has the possibility to select the stage of the FIR filters. The filtered data from the selected stage is then stored in FIFO. The data from FIFO can be moved on-the-fly to the Mother Board or to the host. Because of the double-width of the board the data transfer from the 3430 can be either 16-bit or 32-bit wide and to move the data it can use new functionality called Direct Interrupt, when the function card is assembled onto the 3150.

The 3430 can be configured to work either independently or it can be synchronised with the other 3430 boards. The multi-board synchronisation requires that the same sampling clock and sync signals are distributed to all 3430 boards.

1.3.1 Front-end conditioning circuitry

The front-end conditioning of the 3430 gives the user the flexible solution for the wide range of the applications.

The input can be configured as either AC or DC coupled. AC coupling has a high-pass filter characteristic with a 3dB cut-off frequency of 0.03Hz when using 1MΩ termination and 677Hz when using 50Ω termination.

A very useful feature is the possibility of selecting between either differential or single-ended input. In both cases the overvoltage protection of up to ±35V (with 1MΩ termination) increases the reliability of the hardware. For the single-ended input the offset voltage from the on-board generated DAC can be used to remove the DC voltage from the input signal and thus adjust the input range to the needs of the application.

Another feature is software selectable termination of 50Ω or 1MΩ for both differential and single-ended input.

For calibration purposes the input of every channel can be grounded or it can be connected to the voltage source from the voltage reference board (3201 board).

The input signal goes to the programmable gain amplifier (the possible gain values are 1, 2, 5, 10, 20, 50, 100 or 1, 2, 5, 10, 100, 200, 500, 1000) and then to the anti-aliasing filter. The 6-pole Bessel filter is to ensure proper suppressing of out of the band frequencies. With the fixed corner frequency of 312.5kHz this filter limits the bandwidth of the input signal to meet the requirements of the Nyquist theorem.

After this the conditioned input signal is ready to enter the ADC.

The front-end circuitry is shown on the Figure 2.

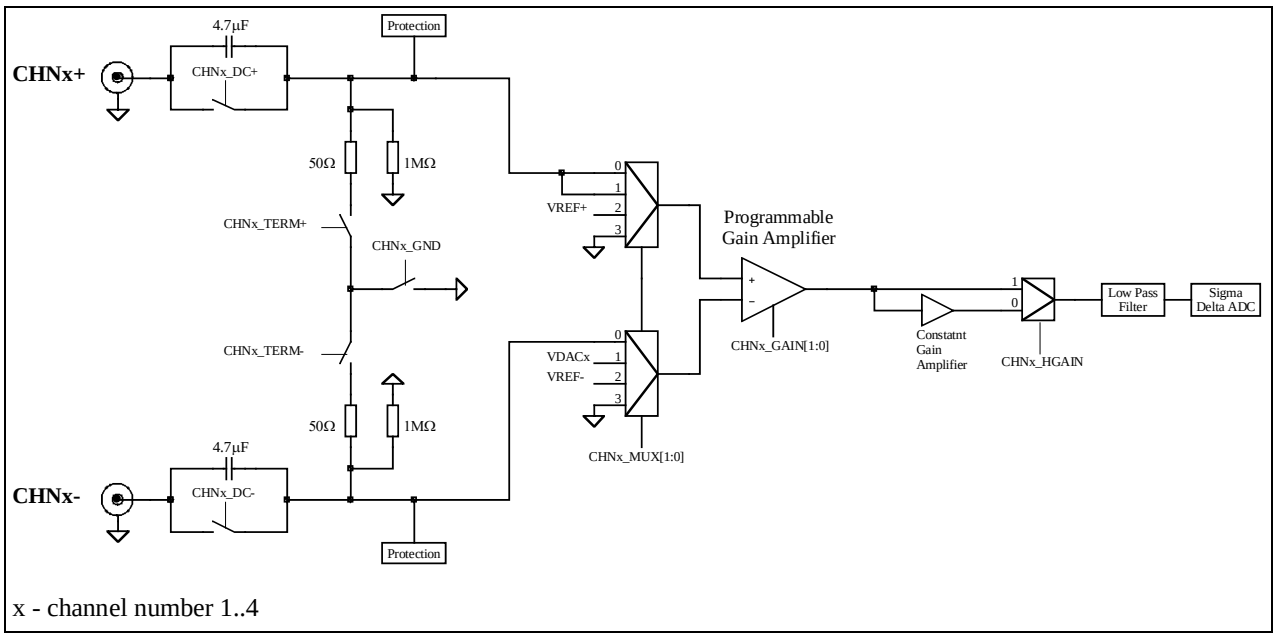

Figure 2: Front-end configuration

1.3.2 Data Acquisition

The data acquisition on the 3430 is controlled by the state machine. This state machine can be in one of the following states, which determines the status of the data acquisition:

STEP	STATE	DESCRIPTION
1	**IDLE**	All needed settings (front-end configuration, DA modes and clk/sync selection) should be done in this step
2	**ARMED**	Arming makes the card ready for the sync pulse. Before arming the card doesn't react on sync. If internal sync is selected then synchronisation pulse is generated immediately after arming command.
3	**SYNC**	The state machine stays in this state as long as synchronisation is on. The duration of the synchronisation depends on the selected stage number of the FIR filter: the higher stage number selected the longer synchronisation. The synchronisation is needed to settle the output of the ADC's modulator and filter and to settle the decimation filters in the FPGA. With the internal sync selected all four ADCs on the board are synchronised and they output the data from the same time. With the external sync selected many 3430 boards can be synchronised together.
4	**READY4DA**	After synchronisation is done the hardware is ready to start data acquisition. It can start either immediately or wait for the trigger. When start on trigger is selected the state machine stays in this state as long as the active edge of the trigger does not happen. The data is not stored in FIFO when the board is in this state but they are ready to be stored any time the data acquisition starts.

5	**DAON**	When the data acquisition is on the samples are stored in FIFO as long as the end of the DA does not happen. The DA can be ended either after a set number of the scans is collected or after trigger becomes inactive. Up to 65536 scans can be set in the first mode or an unlimited number of the scans can be collected assuming emptying FIFO on-the-fly.

The data acquisition ends when the DA stop condition is met. The DA can be stopped any time by the software. The third way to stop the DA is to enable stopping it when the error happens. There are three kinds of error:

- Data overflow – caused by the signal being out of the range or an overflow in the FIR filters
- FIFO overload – occurs when trying to write to a full FIFO
- Scan error – occurs when a sampling clock greater than 5MHz is applied to the ADCs

1.3.3 Sampling clock and sync signals

Sigma-delta converters operate in continuous mode and they need a continuous sampling clock. Once applied it cannot be stopped otherwise the data needs to be settled again. The ADC used on the 3430 (AD9260) uses 8x oversampling and the output word rate is 8 times lower than the sampling clock. The board has been designed to work with a sampling clock of 5MHz. In this case the output word rate from the ADC is 625ksamples/s so the anti-aliasing filter has been set to a corner frequency of 312.5kHz. The user can apply a different clock (lower than 5MHz) for example 3.2768MHz to get 409,6kHz as an output word rate. But in this case it is up to the user to limit the bandwidth of the input signal to meet the requirements of the sampling theorem.

In addition to the sampling clock a sync signal may be applied to the ADCs. The sync signal resets the internal modulators of the converters allowing the synchronisation of many converters: on the board and on many boards.

The table below summarises the selection of the sampling clock and sync signal.

Selection	SAMPLING CLOCK	SYNC	Notes
INTERNAL	From the on-board oscillator, 5MHz	Generated by internal logic	Allowed for the stand-alone mode only
EXTERNAL	From the front panel connector, max.5MHz	From the front panel connector	External cables are needed to distribute the signals

The selection of the sampling clock and sync signal are independent of each other. It means that the external clock source can be selected with the internal sync source or vice versa.

After a sync pulse is generated the decimation filters in ADC and/or FPGA have to be allowed to settle. The filters inside the ADC need 64.2μs to settle and the maximum settling time for the FIR decimation filters (stage number 10 selected) is equal to 97ms (assuming 5MHz sampling clock). The hardware takes care of the settling time of the filters and DA can start only after filters have settled.

1.3.4 Start/Stop mode of data acquisition

Data acquisition is the process of storing the samples from the ADCs in FIFO. When the filters have settled the board is ready to start DA. It can be done in two ways:

- Immediately; when the filters are settled or
- From the trigger; after the trigger goes active.

The samples from the ADCs are then stored in FIFO and it continues until the stop condition is met. The DA can be stopped in two ways:

- From the scans counter; when the preset number of scans has been collected
- From the trigger; after trigger goes inactive

In all modes software can start and stop data acquisition at any time.

In any mode after sync was done the direct data from ADCs is available in the monitoring registers. The monitoring registers are updated with the new data every time the data is ready.

1.3.5 FIR decimation filters

The 3430 is a sigma-delta ADC board. These kind of A/D converters use an oversampling technique. In our case 8 times oversampling is used. It means that the ADC samples the input signal 8 times more often than the output word rate. The ADC internally filters and down samples the data to get the output word rate. The sigma-delta ADC has a big advantage related to the oversampling: the performance of the anti-aliasing filter located in front of the ADC and required by the Nyquist sampling theorem is not as critical. The roll-off rate can be slower (the required attenuation to be reached for higher frequencies) so the number of the poles can be smaller. In addition the anti-aliasing filter can be designed for the minimum attenuation in the pass-band.

Because the anti-aliasing filter is designed for a fixed corner frequency the sampling clock has to be fixed as well. In this case the only way to sample the signal with the lower frequency is to reduce the output data rate. Before the output data rate can be down sampled the bandwidth of the signal has to be limited (filtered). The filtering and down sampling is done in the FIR decimation filters built into the onboard FPGA.

There are 10 stages of the FIR filter. Every stage is composed of the Half-Band FIR filter followed by the decimator reducing the sampling frequency by 2. The filter stages are connected in series so every stage reduces the output word rate by 2. After the last stage the output word rate of 625kHz is reduced by 1024 (2 to the power of 10).

The table below shows the effective output word rate after every filter stage (5MHz sampling clock):

STAGE #	OUTPUT WORD RATE [kHz]
0	625,00
1	312,50
2	156,25
3	78,13
4	39,06
5	19,53
6	9,77
7	4,88
8	2,44
9	1,22
10	0,61

The bandwidth of the signal is always half of the output word rate.

The user can apply an external sampling clock. For the values lower than 5MHz the bandwidth of the signal has to be limited by the user. In the case of 3.2768MHz as a sampling clock the output word rate would be as follow:

STAGE #	OUTPUT WORD RATE [kHz]

0	409,60
1	204,80
2	102,40
3	51,20
4	25,60
5	12,80
6	6,40
7	3,20
8	1,60
9	0,80
10	0,40

In order to minimise quantization error, which could accumulate after every filter stage the precision of the filter has been artificially increased by adding 4 more digits than the ADC data width.

1.3.6 Input Trigger

Input trigger is used to start or to stop collecting the data coming from the ADCs, depending on the DA start/stop mode.

The following sources of the trigger are available:

MB	Trigger from the switch matrix on the MB
EXTERNAL	Trigger through the FP connector
SOFTWARE	Trigger generated by software

More than one source can be selected at the time.

The input trigger configuration scheme is shown on the Figure 3.

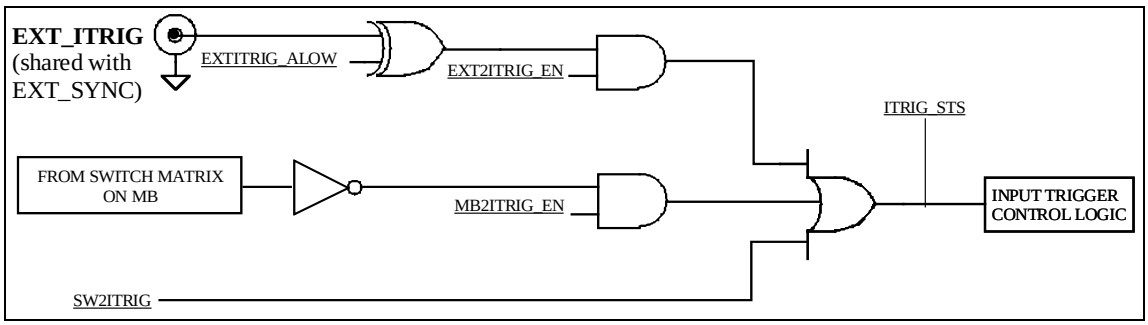

Figure 3: The input trigger configuration

1.3.7 Output Trigger

The output trigger is generated to signal the event, which happened on the 3430. The output trigger is located on the FC-MB connector and it goes to the MB switch matrix. Then it can be directed to any trigger line connected to the switch matrix.

These are the following sources of the output trigger event:

- FIFO flags – generated when the selected flag goes active
- Error event – generated when the selected error happens

- DA ON – generated when DA is on
- DA END – generated when the DA ends
- SW – generated by the software

More than one source of the trigger can be selected at the time. The trigger can generate the pulse or the level. If the level has been selected then the output trigger stays active as long as the source of the trigger is active. If the pulse is selected the pulse on the trigger is generated when the event becomes active. The width of the pulse is 400 μs.

The output trigger configuration scheme is shown on the Figure 4.

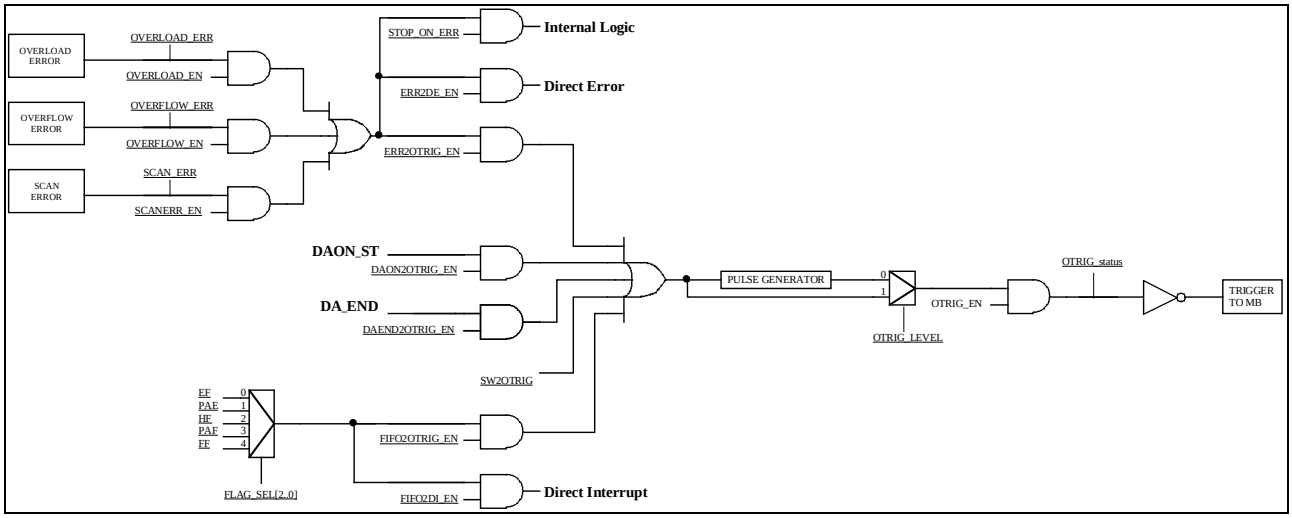

Figure 4: The output trigger configuration

1.3.8 Direct Interrupt and Direct Error

The 3430 is the first board in our range of function cards where the direct interrupt and direct error lines have been implemented. These features can be used only when working with the 3150 motherboard.

Direct Interrupt is generated when the selected FIFO flag goes active. It is used to send the interrupt to the LIST processor without using the output trigger line. The LIST processor reaction to the direct interrupt is much faster than the reaction to the trigger lines so this feature is very useful when the emptying on-the-fly of FIFO is needed. It ensures that moving the data from FIFO will be done on time and no data will be overwritten.

Direct Error is generated when one of the selected error sources happens. There are three possible errors:

- Data overflow – can happen when the input signal is out of range and this error can be generated either by ADC itself or FIR decimation filters
- FIFO overload – happens when FIFO is full and there is new data to write to FIFO
- Scan error – happens when the data from the new scan is ready to write and filtering of the data from the previous scan is not finished yet. It can only happen if a sampling clock frequency greater than 5MHz has been applied to the ADCs. If the sampling clock is not greater than 5MHz this error will not happen.

Lower and upper stack share the same direct error line so after direct error was set the software has to detect which board generated direct error by polling the proper register.

1.3.9 FIFO

The FIFO used on the 3430 board (IDT72V273L15) can be configured for 16-bit transfer or 32-bit transfer. For the 32-bit transfer the size of FIFO is doubled in the comparison to the 16-bit transfer. During 32-bit transfer two samples are sent, one through the first stack and the other through the second stack. It improves throughput when emptying FIFO on-the-fly.

FIFO flags are used to detect how many samples are stored in this memory. Five flags are available from FIFO:

- Empty flag
- Programmable almost empty
- Half flag
- Programmable almost full
- Full flag

Range	FF	PAF	HF	PAE	EF
0	0	0	0	1	1
1 to (n+1)	0	0	0	1	0
(n+2) to 8193	0	0	0	0	0
8194 to (16385-(m+1))	0	0	1	0	0
16385-m to 16384	0	1	1	0	0
16385	1	1	1	0	0

n – offset written to PAE register

m – offset written to PAF register

The values shown in the table reflects the number of words (for 16-bit transfer) or the number of double words (for 32-bit transfer).

All these flags can be selected to be the source of the output trigger and/or direct interrupt.

Two programmable flags can be programmed to any value from the available address range giving thus the flexible control over FIFO.

2. Input and output signals

The table below describes the signal on the front-panel connectors.

CONNECTOR	SIGNAL	DESCRIPTION	INPUT RANGE
J100	CHN 1+	Channel 1 positive input	±10V
J101	CHN 1-	Channel 1 negative input	±10V
J200	CHN 2+	Channel 2 positive input	±10V
J201	CHN 2-	Channel 2 negative input	±10V
J300	CHN 3+	Channel 3 positive input	±10V
J301	CHN 3-	Channel 3 negative input	±10V
J400	CHN 4+	Channel 4 positive input	±10V
J401	CHN 4-	Channel 4 negative input	±10V
J500	EXT_CLK	External sampling clock input	-2..+5V
J501	EXT_SYNC (and TRIG)	External sync input shared with external trigger input	-2..+5V

The 3430 board has the dimensions of double wide FC, i.e. 230 x 106.6 mm.
Figure 5 shows the location of the connectors.

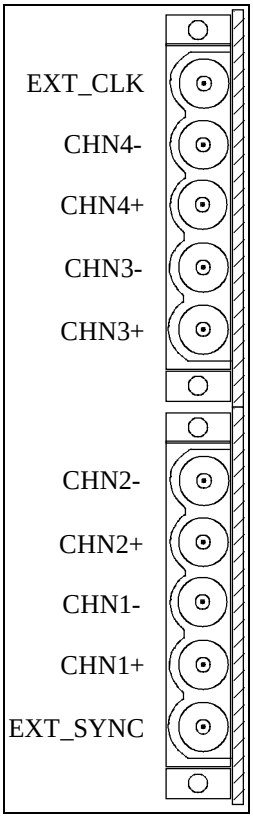

Figure 5: The connectors on the front panel (front view)

3. Programming considerations

3.1 Remarks about configuration

Some settings are common for all channels and some are independent for every channel.

Independent settings:

- Input range: gain and offset
- Coupling mode: AC / DC
- Input type: differential / single-ended
- Termination: 1MΩ / 50Ω

Common settings for all channels:

- Sampling rate / decimation selection
- Data Acquisition start/stop mode
- Clock/Sync selection
- Error handling
- In/Out Trigger selection

3.2 Programming of FIFO programmable flags

The programming of the programmable flags is described in the data sheet of the IDT72V273L15. Here are described the details related to the specific implementation.

WRITE

Write to FIFOWR_REG with the OFFREG_EN bit set to 1

- 16-bit transfer: use 16-bit programming sequence (2 writes)
- 32-bit transfer: use 8-bit programming sequence (4 writes)

READOUT

Readout from FIFORD_REG with the OFFREG_EN bit set to 1

Perform always 4 readouts and take second and third data as a valid data

For the 32-bit transfer compose the data word by taking lower byte from the first stack and lower byte from second stack.

Programming can be done in the IDLE state only.

4. Technical specification

ITEM	SPECIFICATION
Input characteristics	
• Number of Input Channels	4, simultaneous sampling
• Input Ranges (there are two versions of the board depending on the set of the gains)	±10V @ PGA gain = 1 ±5V @ PGA gain = 2 ±2V @ PGA gain = 5 ±1V @ PGA gain = 10 ±0.5V @ PGA gain = 20 ±0.2V @ PGA gain = 50 ±0.1V @ PGA gain = 100 or ±10V @ PGA gain = 1 ±5V @ PGA gain = 2 ±2V @ PGA gain = 5 ±1V @ PGA gain = 10 ±0.1V @ PGA gain = 100 ±0.05V @ PGA gain = 200 ±0.02V @ PGA gain = 500 ±0.01V @ PGA gain = 1000
• Input Type	Differential or Single-ended
• Coupling	DC, AC
• Input Impedance	50Ω or 1MΩ for both DC and AC coupling
• AC Coupling	4.7μF / 50V in series with input signal
• Max. Input Voltage	±10VDC @ 50Ω termination ±35VDC @ 1MΩ termination
• Offset range (single-ended input)	±5VDC, 10-bit resolution
• Analog bandwidth	312.5kHz for all PGA gains
• Offset Error	< 1% FSR @ gain = 1..10 (typ) 3% FSR @ gain = 1000
• Gain Error	1.5% FSR (typ)
• Noise	<1LSB RMS Noise @ gain = 1..10 (Source = Gnd)
Sampling	
• ADC resolution	16 bits
• ADC type	Sigma-delta, 8x oversampling
• Sampling rate range	500ksamples (625ksamples maybe, 409.6ksamples user specific)625ksamples/s to 610.3samples/s
• **Sampling clock, external**	
Input range	-2V to +5V
• Max Input Voltage	±6V
• LevelTermination	ECL to ground50Ω to ground

• FrequencyRange	512kHz to 32MHz (maybe 40MHz), for frequencies lower than 32MHz external antialiasing filter has to be applied5MHz, 45 to 55% duty cycle, low jitter
• **Sync/Trigger Input, external**	
Input range	-2V to +5V
• Max Input Voltage	±6V
• Termination	50Ω to ground
• Min. pulse width - SYNC	400 ns
• Min. pulse width - TRIGGER	TBD100 ns
• Active level - SYNC	High level
• Active level - TRIGGER	Software selectableSoftware selectable
• **FIFO size**	3216Kx16 or 2 x 16Kx16 ifor the 32-bit readout is requiredtransfer
16-bit transfer	16 x 16Ksamples
• 32-bit transfer	2 x 16 x 16Ksamples
Front Panel Connectors analog inputs • digital inputs	LEMO type EPL.00.250.DTN (gold plated) EPL.00.250.NTN (nickel plated)
Current Consumption	+15V 70mA -15V 60mA +12V 8mA -12V 1mA +5V 720mA -5.2V 35mA -2V 9mA (Note ±15V are derived from ±24V by a regulator on the motherboard)
Power Consumption	<6 W
Operating Temperature	0 .. +50°C
Storage Temperature	-40 .. +70°C
Humidity	0-90%, non-condensing
Warm-up time	< 15min.
Dimensions	Double width board104mm x 230mm, double width function card
Weight	195 g

5. Register Description

5.1 Address Map and Registers

All addresses are given in hexadecimal notation. FC_ADR is address in FC address space.

VXI_ADR is address in VXI address space. The appropriate address offset depends on the function card position on the motherboard (refer to the motherboard manual).

FC_ ADR	VXI_ ADR	Register Name	Access	Function
FPGA internal registers				
0	0	**FCID_REG**	RO	FC ID register
1	4	**FCVER_REG**	RO	FC version register
2	8	**FCCTRL_REG**	RWC	General control and status register
3	C	Reserved		
4	10	**COMMAND_REG**	RW	Command register
5	14	**OTRI_REG**	RW	Output trigger control register
6	18	**ITRI_REG**	RW	Input trigger control register
7	1C	**DAC_REG**	RW	DAC setting register
8	20	**MODE_REG**	RW	Mode register
9	24	**FIFOCTRL_REG**	RW	FIFO control/status register
A	28	**FIFOWR_REG**	WO	FIFO write test / offset register
B	2C	**NOS_REG**	RW	Number of samples register
C	30	**CHN1CFG_REG**	RW	Channel #1 configuration register
D	34	**CHN2CFG_REG**	RW	Channel #2 configuration register
E	38	**CHN3CFG_REG**	RW	Channel #3 configuration register
F	3C	**CHN4CFG_REG**	RW	Channel #4 configuration register
10	40	**MONIT1_REG**	RW	Monitoring register of channel #1
11	44	**MONIT2_REG**	RW	Monitoring register of channel #2
12	48	**MONIT3_REG**	RW	Monitoring register of channel #3
13	4C	**MONIT4_REG**	RW	Monitoring register of channel #4
Memory space				
8000	20000	**FIFORD_REG**	RO	Readout of FIFO memory

5.2 Register Description

5.2.1 FCID_REG

FC_ADR=0H, *VXI_ADR=0H*

FCID_REG contains identification number of function card type.
Readout should give a value of **3430H**.

5.2.2 FCVER_REG

FC_ADR=1H, *VXI_ADR=4H*

This is FC version register.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Operation	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Initial	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Content	VER[15:0]															

5.2.3 FCCTRL_REG

FC_ADR=2H, *VXI_ADR=8H*

Function card Control and Status Register.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Operation	RO			RO	RO	RO	RO	RO		RO	RO	RO				RWC
Initial	0			0	0	0	0	1		0	0	0				0
Content	DA_END	Not used		DAON_ST	RAEDY4DA_ST	SYNC_ST	ARMED_ST	IDLE_ST	Not used	SCAN_ERR	OVERFLOW_ERR	OVERLOAD_ERR	Not used			FSM_RESET

FSM_RESET Resets internal (in FPGA) state machines. The reset doesn't change contents of registers.

Reset is started by writing "1" to that bit. After the reset is done, the hardware clears the bit. Software should poll the bit until it is cleared.

It is recommended to perform reset during FC initialisation.

Write

0: no effect

1: starts reset of internal state machine

Read

0: reset finished (if reset previously started)

1: reset in progress

USAGE

- During initialisation process as first step
- To force state machine to known (IDLE_ST) state
- To abnormally stop data acquisition
- FSM_RESET takes approximately 0.4μs

OVERLOAD_ ERR The bit is read only and is set by hardware after FIFO overload event (occurs when trying to write to a full FIFO).

This bit is cleared on the arming command or clearing command.

Write

No effect

Read

0: no overload errors

1: overload error has occurred

USAGE

- This bit is used to detect if overload error happened

OVERFLOW_ERR The bit is read only and is set by hardware after data overflow event occurs.

This bit is cleared on the arming command or clearing command.

Write

No effect

Read

0: no overflow errors

1: overflow error has occurred

USAGE

- This bit is used to detect if overflow error happened

SCAN_ERR The bit is read only and is set by hardware after scan error occurs.

This bit is cleared on the arming command or clearing command.

Write

No effect

Read

0: no scan errors

1: scan error has occurred

USAGE

- This bit is used to detect if scan error happened

IDLE_ST The bit indicates IDLE state of internal state machine.

Write

No effect

Read

0: internal SM in state other than IDLE state

1: internal SM in IDLE state

USAGE

- The IDLE state is the initial state of internal SM. Card configuration is allowed only in this state.

ARMED_ST The bit indicates ARMED state of internal state machine.

Write

No effect

Read

0: internal SM in state other than ARMED state

1: internal SM in ARMED state

USAGE

- The ARMED state means that arming was done and card is waiting for synchronisation pulse
- The card waits for the synchronisation signal if external synchronisation selected or synchronisation pulse is generated internally if internal synchronisation selected.

SYNC_ST The bit indicates SYNC state of internal state machine.

Write

No effect

Read

0: internal SM in state other than SYNC state

1: internal SM in SYNC state

USAGE

- The SYNC state indicates that synchronisation pulse came and the synchronisation is in progress
- The SM stays in SYNC state until completion of synchronisation (settling the ADC and FIR filters).

READY4DA_ST The bit indicates READY4DA state of internal state machine.

Write

No effect

Read

0: internal SM in state other than READY4DA state

1: internal SM in READY4DA state

USAGE

- The READY4DA state indicates that synchronisation was completed and the card waits for event to start DA. The ADCs sample and FIR filters output the valid data but they are not stored in FIFO
- The SM stays in READY4DA state until DA start condition is met. The DA start condition depends on the card configuration

DAON_ST The bit indicates DAON (Data Acquisition ON) state of internal state machine.

Write

No effect

Read

0: internal SM in state other than DAON state

1: internal SM in DAON state

USAGE

- The DAON state means that data acquisition (storing data in FIFO) is in progress
- The collecting of data can take place in this state only. It leaves this state when DA end condition is met or when FSM reset is performed

DA_END

The bit is read only and is set by hardware after the normal end of data acquisition.

This bit is cleared on arming command or clearing command.

Write

No effect

Read

0: DA in progress (if previously started)

1: DA ended

USAGE

- This bit can be used to detect the end of the data acquisition. There are two possible ways: polling the bit or waiting for the interrupt generated by this bit when either output trigger or direct interrupt was enabled. The DA stop condition depends on the card configuration
- This bit is not set if DA is ended by FSM reset

5.2.4 COMMAND_REG

FC_ADR=4H, *VXI_ADR=10H*

This is a command register. Writing to this register generates ARMING_COMMAND and reading from this register generates CLEAR_COMMAND.

CLEAR_COMMAND clears errors (OVERLOAD, OVERFLOW, SCAN) and DA_END bit.

ARMING_COMMAND clears errors, DA_END bit and launches DA.

5.2.5 OTRI_REG

FC_ADR=5H, *VXI_ADR=14H*

Output Trigger register allows to select the source of the output trigger sent to the motherboard and further to the VXI controller.

The following are the trigger sources:

- Errors (OVERFLOW_ERR, OVERLOAD_ERR, SCAN_ERR)
- DA on
- End of DA
- Reaching the pre-set number of samples in FIFO
- FIFO flag

The hardware allows the use of more than one trigger source at the time.

The Output Trigger can be used to generate an interrupt at the end of the measurement process.

The other two signals, which are enabled using the bits of this register, are Direct Interrupt and Direct Error. They go directly to the LIST processor on the 3150 motherboard. The source of the Direct Interrupt is the FIFO flag and there are three errors, which might cause Direct Error.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Operation	RO	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Initial	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Content	OTRIG_ status	ERR2DE_EN	ERR2OTRIG_EN	SCANERR_EN	OVERFLOW_EN	OVERLOAD_EN	OTRIG_EN	OTRIG_LEVEL	SW2OTRIG	DAEND2OTRIG_EN	DAON2OTRIG_EN	FIFO2DI_EN	FIFO2OTRIG_EN	FLAG_SEL[2:0]		

FLAG_SEL[2:0] These bits select the FIFO flag that will be used to generate the trigger from FIFO.

Write

0: EF flag selected to generate the trigger

1: PAE flag selected to generate the trigger

2: HF flag selected to generate the trigger

3: PAF flag selected to generate the trigger

4: FF flag selected to generate the trigger

5: reserved

6: reserved

7: reserved

Read

Gives the last written value

USAGE

- In order to enable the trigger generated by FIFO the FIFO2OTRIG_EN and the OTRIG_EN bits should be set
- PAE and PAF flags are programmable and can be set to any value within the range

FIFO2OTRIG_EN This bit enables generating an output trigger when the selected flag (FLAG_SEL[1:0]) goes active.

Write

0: generating trigger from FIFO flag disabled

1: generating trigger from FIFO flag enabled

Read

Gives the last written value

USAGE

- In addition to this bit the OTRIG_EN bit should be set to enable output trigger
- The flag that generates the trigger is selected by FLAG_SEL[1:0] bits
- The flag is cleared if the number of samples in FIFO goes under the flag's threshold

FIFO2DI_EN This bit enables generating a direct interrupt to the List processor on the 3150 MB when the selected flag (FLAG_SEL[1:0]) goes active.

Write

0: generating direct interrupt from FIFO flag disabled

1: generating direct interrupt from FIFO flag enabled

Read

Gives the last written value

USAGE

- The flag that generates the direct interrupt is selected by FLAG_SEL[1:0] bits
- The flag is cleared if the number of samples in FIFO goes under the flag's threshold

DAON2OTRIG_EN This bit enables generating an output trigger when data acquisition is on. As long as DA is on the trigger is set.

Write

0: DA ON trigger disabled

1: DA ON trigger enabled

Read

Gives the last written value

USAGE

- In addition to this bit the OTRIG_EN bit should be set to enable the output trigger

DAEND2OTRIG_EN This bit enables generating an output trigger when the data acquisition is ended. At the end of DA the DA END trigger source is set.

Write

0: DA END trigger disabled

1: DA END trigger enabled

Read

Gives the last written value

USAGE

- In addition to this bit the OTRIG_EN bit should be set to enable output trigger

SW2OTRIG This bit sets the software-generated output trigger.

Write

0: inactive state of the software-generated output trigger

1: active state of the software-generated output trigger

Read

Gives the status of software-generated output trigger

USAGE

- In addition to this bit the OTRIG_EN bit should be set to enable output trigger

OTRIG_LEVEL This bit is for selection of the output trigger generating mode:

Pulse – after a rising edge a trigger source pulse of 400ns width will be generated independently of trigger source high level duration

Level – after a rising edge a trigger source output trigger level will follow the level of trigger source.

Write

0: output trigger generating mode set to pulse

1: output trigger generating mode set to level

Read

Gives the last written value

USAGE

- When working with interrupts to the host the level mode should be set

OTRIG_EN This bit is the main output trigger enable bit. If this bit is cleared no output trigger will be sent to MB independently of trigger source enable bits. If this bit is set output trigger will be sent if any trigger source is enabled and its condition is met.

Write

0: output trigger disabled

1: output trigger enabled

Read

Gives the last written value

USAGE

- Main output trigger enabling bit

OVERLOAD_EN This bit enables OVERLOAD error as an error trigger source.

Write

0: OVERLOAD error disabled

1: OVERLOAD error enabled

Read

Gives the last written value

USAGE

- This bit can be used to enable DA to be stopped on overload error (if STOP_ON_ERR bit was set) or to send the trigger to the MB when the error happens (when ERR2OTRIG_EN and OTRIG_EN bits were set) or to send the interrupt to the List processor on the 3150 MB through Direct Error line (when ERR2DE_EN bit was set)
- This error is cleared using CLEAR_CMD

OVERFLOW_EN This bit enables OVERFLOW error as an error trigger source.

Write

0: OVERFLOW error disabled

1: OVERFLOW error enabled

Read

Gives the last written value

USAGE

- This bit can be used to enable DA to be stopped on overload error (if STOP_ON_ERR bit was set) or to send the trigger to the MB when the error happens (when ERR2OTRIG_EN and OTRIG_EN bits were set) or to send the interrupt to the List processor on the 3150 MB through Direct Error line (when ERR2DE_EN bit was set)
- This error is cleared using CLEAR_CMD

SCANERR_EN This bit enables SCAN error as an error trigger source.

Write

0: SCAN error disabled

1: SCAN error enabled

Read

Gives the last written value

USAGE

- This bit can be used either to enable DA to be stopped on overload error (if STOP_ON_ERR bit was set) or to send the trigger to the MB when the error happens (when ERR2OTRIG_EN and OTRIG_EN bits were set) or to send the interrupt to the List processor on the 3150 MB through Direct Error line (when ERR2DI_EN bit was set)
- This error is cleared using CLEAR_CMD

ERR2OTRIG_EN This bit enables generating an output trigger when an error occurred.

Write

0: generating trigger on error disabled

1: generating trigger on error enabled

Read

Gives the last written value

USAGE

- In addition to this bit the OTRIG_EN bit should be set to enable the output trigger and the particular error source to be enabled

ERR2DE_EN The bit enables generating an interrupt to the DE line when an error occurs.

Write

0: generating DE interrupt disabled

1: generating DE interrupt enabled

Read

Gives the last written value

USAGE

- The source of the error has to be selected (OVERLOAD_EN and OVERFLOW_EN bits)

OTRIG_status The state of the output trigger line.

Write

No effect

Read

0: output trigger inactive

1: output trigger active

USAGE

- When working with interrupts to the host this bit should be used by the interrupt routing to determine the interrupt source

5.2.6 ITRI_REG

FC_ADR=6H, *VXI_ADR=18H*

The register allows selection of the common trigger source used optionally to start TI counters.
These are the possible input trigger sources:

- Software trigger
- MB input trigger
- External trigger through FP

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Operation									RO				RW	RW	RW	RW
Initial									X				0	0	0	0
Content	Not used								ITRIG_STS	Not used			SW2ITRIG	MB2ITRIG_EN	EXT2ITRIG_EN	EXTITRIG_ ALOW

EXTITRIG_ ALOW This bit sets the active level of the external trigger coming from the FP (ALOW stands for **A**ctive **LOW**).

Write

0: active level high

1: active level low

Read

Gives the last written value

EXT2ITRIG_EN This bit enables the external trigger as a source of input trigger.

Write

0: external trigger disabled

1: external trigger enabled

Read

Gives the last written value

USAGE

- The active level of external trigger can be set using EXTITRIG_ALOW bit.
- The external trigger shares the FP connector with synchronisation signal

MB2ITRIG_EN This bit enables the MB trigger as a source of input trigger.

Write

0: MB trigger disabled

1: MB trigger enabled

Read

Gives the last written value

SW2ITRIG This bit sets the software-generated input trigger.

Write

0: inactive state of the software-generated input trigger

1: active state of the software-generated input trigger

Read

Gives the status of software-generated input trigger

ITRIG_STS The state of input trigger. The status shows OR function of all sources of input trigger.

Write

Has no effect

Read

Gives the current state of the input trigger

5.2.7 DAC_REG

FC_ADR=7H, *VXI_ADR=1CH*

The DAC_REG sets the output value of the on-board DAC. The 8-output (only first four are used on this board) DAC is used to control the offset of each input channel.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Operation	RWC		WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
Initial	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
Content	DAC trans	Not used	DAC_ADDR[3:0]				DAC_DATA[9..0]									

DAC_DATA[9..0] These bits are data to write to 8-channel DAC. These bits set DAC outputs.

Write

DD[9..0] sets the output value of DAC

Read

These are write only bits

USAGE

- The data should be in binary format
- The DAC outputs range is –5.00V..+4.99V
- Output voltage $V_{OUT} = \frac{5*(DD[9..0]-512)}{512}$ and is expressed in volts

DAC_ADDR[3..0] These bits address the DAC channel to be set.

Write

DAC_ADDR[3..0] sets the address of DAC channel

Read

These are write only bits

USAGE

- DAC_ADDR[3..0] = DAC channel number.
- DAC channel number range is from 1 to 6 and this corresponds respectively to input channel 1 to 4, EXT_CLK, EXT_SYNC

DAC_TRANS This bit starts data shifting to the DAC. Writing “1” to this bit starts data (D[12..0]) shifting out to the DAC. This bit is cleared to “0” after shifting is finished.

Write

0: no effect

1: starts DAC_DATA[9..0] shifting out

Read

0 : shifting out to DAC finished (if previously started)

1 : shifting out to DAC in progress

USAGE

- To start and detect the end of shifting out
- Shifting out takes approximately 8μs
- During shifting out DACtrans bit is set

5.2.8 MODE_REG

FC_ADR=8H, *VXI_ADR=20H*

This is mode register. Each bit of this register is write and read able.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Operation		RO	RO	RO	RW	RW	RW	RW		RW	RW	RW	RW	RW	RW	RW
Initial		0	0	0	0	0	0	0		0	0	0	1	1	0	0
Content		CFG[2:0]			DECIM_SEL[3:0]					STOP_ON_ERR	DA_STOP_SEL	DA_START_SEL	EXTSYNC_TERM	EXTCLK_TERM	SYNC_SEL	CLK_SEL

CLK_SEL This bit selects the source of the clock signal for the AD converters.

Write

0: internally generated 5MHz clock selected

1: external 5MHz clock selected

Read

Gives the last written value

USAGE

- AD converters require a continuous clock. Internal clock is applied to the converters immediately after selection. The external clock when selected should be applied before a SYNC pulse is generated and should remain on.

SYNC_SEL This bit selects the source of a synchronisation pulse for the AD converters.

Write

0: internally generated pulse

1: external pulse applied to FP connector

Read

Gives the last written value

EXTCLK_TERM This bit sets 50Ω termination for the external clock input

Write

0: 50Ω termination connected

1: 50Ω termination not connected

Read

Gives the last written value

EXTSYNC_TERM This bit sets 50Ω termination for the external sync input

Write

0: 50Ω termination connected

1: 50Ω termination not connected

Read

Gives the last written value

DA_START_SEL This bit selects the way the data acquisition is started

Write

0: DA starts immediately after synchronisation is done

1: DA starts when input trigger goes active

Read

Gives the last written value

USAGE

- This bit together with DA_STOP_SEL bit define the mode of Data Acquisition

DA_STOP_SEL

This bit selects the way the data acquisition is stopped

Write

0: DA stops on completion of collecting set number of samples

1: DA stops when input trigger goes inactive

Read

Gives the last written value

USAGE

- This bit together with DA_START_SEL bit define the mode of Data Acquisition

STOP_ON_ERR

When set this bit enables stopping the data acquisition if an error happens.

Write

0: errors don't stop measurement

1: errors stop measurement

Read

Gives the last written value

USAGE

- In addition to this bit the error source has to be selected and enabled (bits OVERFLOWERR_EN, OVERLOADERR_EN in TRIG_REG)

DECIM_SEL[3:0]

This bits set the decimation factor of the chain of FIR filters.

Write

[3:0]: decimation factor is equal to $2^{DECIM\_SEL[3:0]}$

Read

Gives the last written value

USAGE

- Decimation equal 1 (DECIM_SEL[3:0]=0) means no decimation

CFG[2:0]

Hardware configuration bits are used to detect the version of the board.

CFG(2:1) are reserved.

Write

No effect

Read

CFG(0) = 0, the version of the board with the gain of 1, 2 ,5 10, 20, 50, 100

CFG(0) = 1, the version of the board with the gain of 1, 2, 5, 10, 100, 200, 500, 1000

CFG(2:1): reserved,

USAGE

- To distinct between two versions of the board: with lower and higher

set of gains

5.2.9 FIFOCTRL_REG

FC_ADR=9H, *VXI_ADR=24H*

This register is a control/status register of FIFO memory.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Operation				RO	RO	RO	RO	RO					WR	WR	RWC	RWC
Initial				0	0	0	0	0					0	0	0	0
Content				FF	PAF	HF	PAE	EF					OFFREG_EN	TRANS32	PRS	MRS

MRS This is FIFO master reset. Reset is done by writing “1” to that bit and waiting for “0”. The master reset of FIFO means clearing read and write pointers, configuring FIFO and assigning default programmable settings.

Write

0: no effect

1: starts master reset of FIFO

Read

0: resetting finished (if previously started)

1: resetting in progress

USAGE

- To initialise FIFO
- To switch between FIFO in/out bus width (in connection with TRANS32 bit)
- After master reset the programmable flags are set to their default values

PRS This is FIFO partial reset. Reset is done by writing “1” to that bit and waiting for “0”. The partial reset of FIFO means clearing read and write pointers.

Write

0: no effect

1: starts partial reset of FIFO

Read

0: resetting finished (if previously started)

1: resetting in progress

USAGE

- For emptying FIFO
- The FSMreset bit launches partial reset of FIFO as well

TRANS32 This bit selects the in/out FIFO bus width and in fact it switches between 16-bit or 32-bit transfer from FIFO. This bit can be changed only during master reset. If MRS is not started TRANS32 stays unchanged.

Write

0: 16-bit transfer from FIFO selected

1: 32-bit transfer from FIFO selected

Read

Gives the last value written during the master reset

USAGE

- To switch between 16-bit and 32-bit transfer from FIFO
- This bit can be changed only if the MRS bit is set during the same write to the register

OFFREG_EN This bit enables writing and reading to/from programmable flag offset registers.

Write

0: access to offset register disabled

1: access to offset register enabled

Read

Gives the current value

USAGE

- Setting the bit to a value of 1 is enabled in IDLE_ST only
- Master Reset and states other than IDLE_ST resets the OFFREG_EN bit
- This bit should be set only for the time of accessing offset registers
- Depending on the selection of the transfer width (16 or 32 bit) the access to the offset registers is respectively 16-bit or 8-bit wide

EF The Empty Flag bit indicates that FIFO memory is empty.

Write

No effect

Read

0: FIFO not empty

1: FIFO empty

USAGE

- To detect if FIFO is empty when moving data from the FIFO

PAE The Programmable Almost Empty flag indicates that the number of samples stored in FIFO is equal or lower than the programmed value.

Write

No effect

Read

0: number of samples in FIFO greater than programmed value

1: number of samples in FIFO equal or lower than programmed value

USAGE

- To detect if the number of samples in FIFO reached the set threshold
- After master reset the PAE flag acquires the default value

HF The Half Flag indicates that FIFO memory is half full.

Write

No effect

Read

0: number of samples in FIFO less than half size of the memory

1: number of samples in FIFO at least half size of the memory

USAGE

- To detect if FIFO is half full when moving data from FIFO

PAF The Programmable Almost Full flag indicates that the number of samples stored in FIFO reached the programmed value.

Write

No effect

Read

0: number of samples in FIFO lower than programmed value

1: number of samples in FIFO equal or greater than programmed value

USAGE

- To detect if the number of samples in FIFO reached the set threshold
- After master reset the PAF flag acquires the default value

FF The Full Flag indicates that FIFO memory is full.

Write

No effect

Read

0: FIFO not full

1: FIFO full

USAGE

- To detect if FIFO is full when moving data from FIFO

5.2.10 FIFOWR_REG

FC_ADR=AH, *VXI_ADR=28H*

This register is used to write the data to FIFO. Writing to FIFO is allowed in IDLE_ST only.

The data can be written to FIFO or to offset register in FIFO, depending on the state of OFFREG_EN bit.

If 32-bit transfer mode is set, lower byte is written to first stack memory and upper byte is written to second stack memory.

5.2.11 NOS_REG

FC_ADR=BH, *VXI_ADR=2CH*

Number_**O**f_**S**cans register.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Operation	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Initial	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Content	NOS[15:0]															

This register is used to set the number of scans that have to be collected to finish Data Acquisition, assuming that DA_STOP_SEL bit was set to “0”. Number of scans is in the range from 0 to 65536. When 0 is written it means that no scan will be taken and DA will finish immediately after start.

The content of this register doesn’t change with the progress of samples acquisition.

5.2.12 CHNxCFG_REG

FC_ADR=CH, DH, EH, FH *VXI_ADR=30H, 34H, 38H, 3CH*

The register allows the configuration of the channel number x, where x is in the range from 1 to 4.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Operation	RW	RW	RW	RW	RW				RW	RW	RW	RW	RW			RW
Initial	1	1	1	1	1				1	1	1	1	1			0
Content	CHNx_HGAIN	CHNx_GAIN[1:0]		CHNx_MUX[1:0]		Not Used			CHNx _GND	CHNx _TERM-	CHNx _TERM+	CHNx_DC-	CHNx_DC+	Not Used		CHNx_EN

CHNx_EN When set the bit enables writing the samples from the channel number x to FIFO.

Write

0: samples from the channel #x are not stored in FIFO

1: samples from the channel #x are stored in FIFO

Read

Gives the last written value

USAGE

- The bits don't disable the sampling in the channels but only storing the data in FIFO. If channel is disabled the samples are still available through monitoring registers
- Default value is channel disabled (after FC reset)
- The bit can be modified in IDLE_STATE only

CHNx_DC+ The bit sets DC coupling for the positive input of the channel #x.

Write

0: positive input DC coupling on

1: positive input AC coupling on

Read

Gives the last written value

CHNx_DC- The bit sets DC coupling for the negative input of the channel #x.

Write

0: negative input DC coupling on

1: negative input AC coupling on

Read

Gives the last written value

CHNx_TERM+ The bit sets 50Ω termination for the positive input of the channel #x.

Write

0: positive input 50Ω termination

1: positive input 1MΩ termination

Read

Gives the last written value

CHNx_TERM- The bit sets 50Ω termination for the negative input of the channel #x.

Write

0: negative input 50Ω termination

1: negative input 1MΩ termination

Read

Gives the last written value

CHNx_GND The bit connects the ground to 50Ω termination resistor for positive and negative input of the channel #x.

Write

0: 50Ω terminators grounded

1: 50Ω terminators not grounded

Read

Gives the last written value

CHNx_MUX[1:0] The bits select the input of the multiplexer in the channel #x.

Write

0: differential input

1: single-ended input with offset adjusting

2: voltage reference signal

3: input of the multiplexer grounded

Read

Gives the last written value

CHNx_GAIN[1:0] The bits select the gain of the Programmable Gain Amplifier in the channel #x.

Write

0: gain of 1

1: gain of 2

2: gain of 5

3: gain of 10

Read

Gives the last written value

USAGE

The final gain value is a multiplication result of gain selected here and HGAIN selected using bit CHNx_HGAIN

CHNx_HGAIN The bit selects the gain of 10 (or 100) in the channel #x.

Write

0: high gain of 10 (or 100) on. Gain set on the PGA is multiplied by 10 (or 100).

1: high gain of 10 (or 100) off. Gain set on the PGA is multiplied by 1.

Read

Gives the last written value

USAGE

CHNx_HGAIN is used to select higher gains. The multiplier 10 or 100 depends on the version of the board

5.2.13 MONITx_REG

FC_ADR=10H, 11H, 12H, 13H *VXI_ADR=40H, 44H, 48H, 4CH*

Monitoring registers. They contain the direct value from the ADC in two's compliment format.

6. VXI*plug&play* Driver

6.1 Introduction

Plug and play demonstration software was developed in compliance with the ProDAQ software line. The VXI*plug&play* soft front panel is a graphical user interface application developed for the instrument (Sigma-Delta ADC function card). It is used to verify instrument communications and functionality when the instrument is first integrated into a system. It provides instrument control in a user-friendly environment, being both Windows 95 and NT compatible. The user interface uses the installed driver to control and process instrument information. The test software can be divided into two distinct parts, firstly the use of the test software and secondly a discussion on the driver functions developed and their use in an application environment.

6.2 Software Installation

With the function card a VXI*plug&play* Disk is delivered. It contains the software required to operate the function card in the ProDAQ environment. After the Counter Timer Function Card has been installed into the ProDAQ motherboard, the VXI*plug&play* software may be used to communicate with the ProDAQ motherboard. To install the software, first power on the mainframe, then perform the following operations:

1. Start Windows (95 or NT) on your computer if it is not already running.
2. Ensure no ProDAQ software is currently running on your computer.
3. Insert the ProDAQ installation disk into CD drive.
4. Follow the instructions presented by the SETUP program.
5. Select Full Installation to the default directory.

After the SETUP program has completed, the executable Soft Front Panel program may be run. The drivers are available for WIN 95 or WIN/NT. In the following table win**xx** stands for the particular version being used. If the system is a Windows NT then the VXI*plug&play* path is \VXIpnp\WinNT

Description	File	Hard Disk Destination
Instrument Driver		
Driver Source	Bu3430.c	\Vxipnp\win**xx**\bu3430
Header File	Bu3430.h	\Vxipnp\win**xx**\include\
Function Panel	Bu3430.fp	\Vxipnp\win**xx**\ bu3430\
Microsoft Windows DLL	Bu3430_32.dll	\Vxipnp\win**xx**\bin\
Microsoft Windows import Library	Bu3430.lib	\Vxipnp\win**xx**\lib\msc\ \Vxipnp\win**xx**\lib\Bc\
Microsoft Visual Basic function declaration file	Bu3430.bas	\Vxipnp\win**xx**\include\
Driver documentation	Bu3430.doc	\Vxipnp\win**xx**\ bu3430\
Driver help	Bu3430.hlp	\Vxipnp\win**xx**\ bu3430\
ProDAQ motherboard access library bu3100		
Header File	Bu3100.h	\Vxipnp\win**xx**\include\
Microsoft Windows DLL	Bu3100_32.dll	\Vxipnp\win**xx**\bin\
Microsoft Windows import Library	Bu3100.lib	\Vxipnp\win**xx**\lib\msc\ \Vxipnp\win**xx**\lib\Bc\
Soft Front Panel		
SFP help	Bu3430fp.hlp	\vxipnp\win**xx**\bu3430\
Executable file	Bu3430fp_32.exe	\vxipnp\win**xx**\ bu3430\
Uninstall program	Uninst_bu3430.exe	\vxipnp\win**xx**\ bu3430\

6.2.1 Software Utilisation

After the start of the application, the ProDAQ browser will find all available SD-ADC function cards in the system, and if there will be more than one found the dialog window will appear to select one of the SD-ADC function cards (see Figure 6).

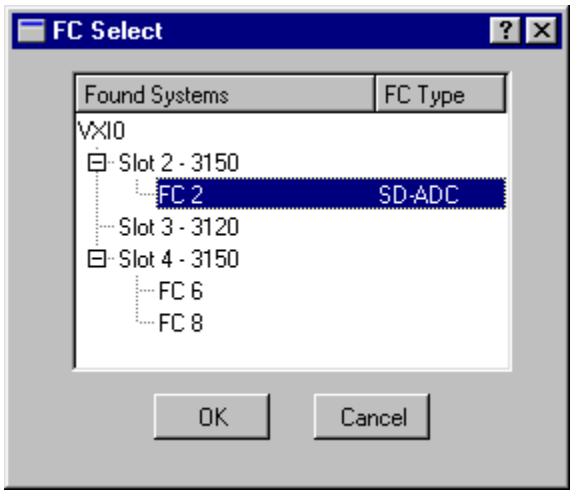

Figure 6: SD-ADC Select Dialog window

After SD-ADC Function card is selected, it will be initialised and the main window is shown (see Figure 7).

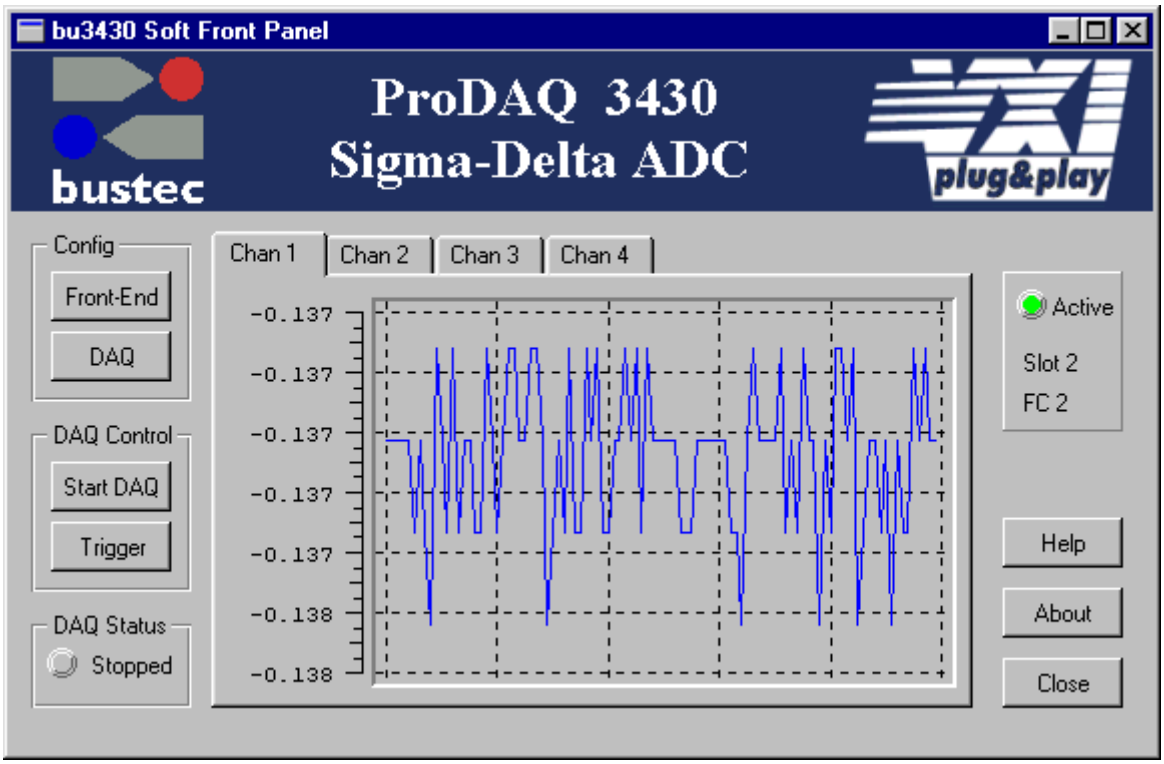

Figure 7: SD-ADC Main Window

The main window is used to start/stop data acquisition, display status of the board and plot data.

To configure front-end circuitry of all channels, external Synch/Trigger signal line, external Sampling Clock line, the Front-end configuration window is used (see Figure 8).

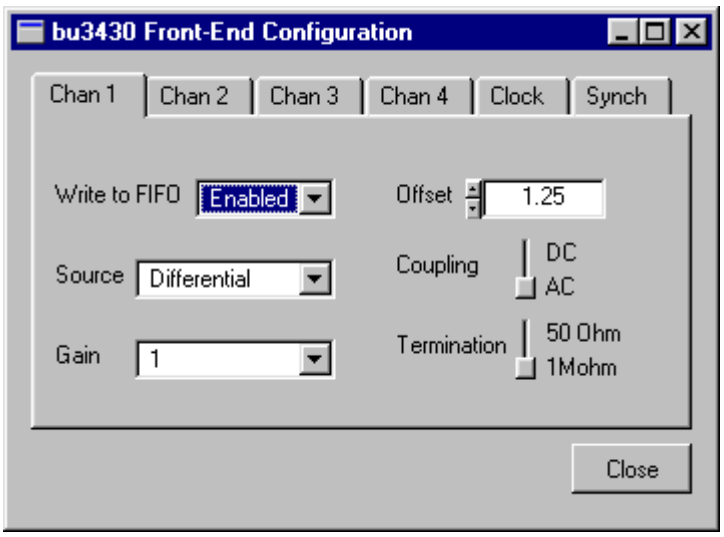

Figure 8: Front-End Configuration Window

To configure the Data Acquisition parameters i.e. number of samples to collect, decimation factor, start mode, trigger, clock and synch signal sources, the DAQ Configuration Window is used (see Figure 9).

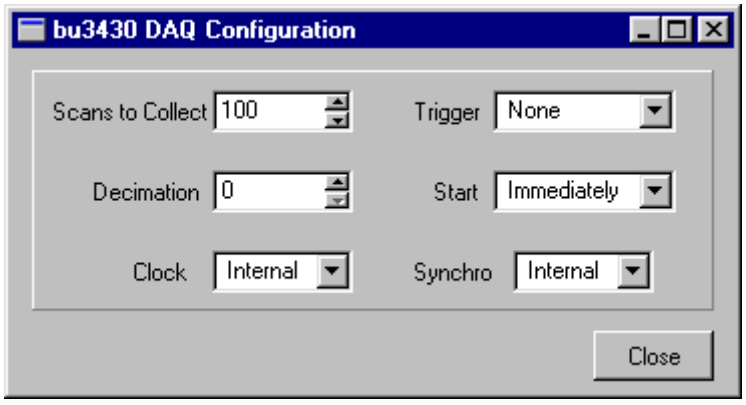

Figure 9: Data Acquisition Configuration Window

6.3 Programming Concepts

6.3.1 Instrument Driver Overview

The Instrument Driver for Sigma-Delta ADC function card can operate with SD-ADC function card installed on any of ProDAQ motherboard – 3120 or 3150 modules. It uses bu3100_32.dll to access the motherboard's hardware. The bu3100_32.dll is distributed and installed together with Instrument Driver for any of ProDAQ module. The Instrument Driver function tree (Figure 10) is divided in two main parts.

- First part contains high-level access functions and called "Measurement Functions class". Those functions give easy and comprehensive access to the main functionality of the board and allow collecting waveforms form the input channels of the function card. The hardware configuration of the board can be customized using functions from the "Hardware Configuration" subclass of functions.
- Second part of functions ("Low-level Access" class) offers detailed register-level access to the hardware of the board and can be used to build very specific, customized Data Acquisition process, which cannot be performed by high-level functions. Using those functions assumes detailed knowledge of hardware of the board and software of the Instrument Driver.

It is strongly recommended not using Low-level functions of the Instrument Driver together with High-level functions. In most cases the parameters set by Low-level functions will be just overwritten by High-level functions, but in the worse case the High-level functions will not work properly with particular low-level configuration.

A full description of the instrument driver functions can be referenced in Appendix C.

Bustec ProDAQ System SD-ADC Module	Function Name:
Initialisation	bu3430_init
Select Function Card	bu3430_fcSelect
Initialisation With Parameters	bu3430_paramInit
Measurement functions	
Acquire Waveform	bu3430_acquireWaveform
Acquire Waveforms	bu3430_acquireWaveforms
Acquire Triggered Waveforms	bu3430_acquireTriggeredWaveforms
Sample Channel	bu3430_sampleChannel
Sample Channels	bu3430_sampleChannels
Asynchronous Acquisition	
Start Acquisition	bu3430_startAcqisition
Check Acquisition	bu3430_checkAcquisition
Read Acquisition	bu3430_readAcquisition
Stop Acquisition	bu3430_stopAcquisition
Install Acquisition Callback	bu3430_installAcquisitionCallback
Hardware Configuration	
Set Channel Configuration	bu3430_setChanConfig
Get Channel Configuration	bu3430_getChanConfig
Set FE Configuration	bu3430_setFEconf
Get FE Configuration	bu3430_getFEconf
Set Acquisition Attribute	bu3430_setAcquisitionAttribute
Get Acquisition Attribute	bu3430_getAcquisitionAttribute
Low-Level Access	
DAQ configuration	
Set Channel Pattern	bu3430_setChanPattern
Get Channel Pattern	bu3430_getChanPattern
Set Decimation Factor	bu3430_setDecimFactor
Get Decimation Factor	bu3430_getDecimFactor
Set Synch Source	bu3430_setSynchSrc
Get Synch Source	bu3430_getSynchSrc
Set Clock Source	bu3430_setClockSrc
Get Clock Source	bu3430_getClockSrc
Set DAQ Mode	bu3430_setDAQmode
Get DAQ Mode	bu3430_getDAQmode
Set Number of Scans	bu3430_setNScans
Get Number of Scans	bu3430_getNScans
Set Output Trigger Sources	bu3430_setOTRIsrc
Get Output Trigger Sources	bu3430_getOTRIsrc
Set Input Trigger Configuration	bu3430_setITRI
Get Input Trigger Configuration	bu3430_getITRI
DAQ Control	
Start DAQ	bu3430_startDAQ
Wait for DAQ	bu3430_waitDAQ
Stop DAQ	bu3430_stopDAQ
Pulse Input Trigger	bu3430_pulseITRI
FIFO Readout / Control	
Reset FIFO	bu3430_resetFIFO
Get FIFO Status	bu3430_getFIFOstatus
Read FIFO	bu3430_readFIFO
Utility Functions	
Reset	bu3430_reset
Error Message	bu3430_error_message
Self Test	bu3430_self_test
Revision Query	bu3430_revision_query
Close	bu3430_close

Figure 10: Instrument Driver function tree.

6.3.2 Instrument Control

Every instrument driver function has the same return type format. Returning either a completion code or an error code.

```
ViStatus _VI_FUNC bu3430_functionName ( Parameters… );
```

In order to identify the successful operation of any function these codes can be used. The following example illustrates this principle.

```
ViSession vi ;
ViStatus error;
ViChar msg[512];
   :
   :
if ((error = bu3430_reset(vi)) < VI_SUCCESS)
{
   bu3430_error_message (vi, error, msg);
}
```

If an error occurs, a value less than VI_SUCCESS is returned. The function bu3430_error_message converts the error code into a readable string. All driver functions operate along the same principles, so any errors in hardware access are easily determined.

6.3.3 Instrument Initialisation

The Instrument initialisation should be done in two steps. First, the VISA session should be opened to the ProDAQ motherboard using **bu3430_init()** function. And then **bu3430_fcSelect()** function should be used to attach the Instrument Driver to the particular function card fitted to the ProDAQ motherboard. There is another function called **bu3430_paramInit()**, which makes both at the same time; it opens session to the ProDAQ motherboard and attaches the driver to the function card. Upon successive completion the program should release system resources by calling **bu3430_close()** function. This function detaches the driver from the function card and closes the VISA session to the ProDAQ motherboard.

Important notice: the ProDAQ motherboard communicates to the Instrument Driver in point-to-point mode. It means that it is not allowed to run two or more applications concurrently working with the same motherboard, even if they work with different function cards. From the other side, one program can work with many function cards, even if they are fitted into the different VXI modules (ProDAQ motherboards). For **every function card** the program works with, it must obtain unique VISA session using **bu3430_init()** function and then attach obtained VISA session to the function card using **bu3430_fcSelect()** function (or use **bu3430_paramInit()** function instead of pair **bu3430_init()** and **bu3430_fcSelect()**).

The program cannot initialise one particular function card twice. Any subsequent initialisations will fail. To reinitialise the function card, the program must close previously opened session to this function card using **bu3430_close()** function.

6.3.4 Data Acquisition

Prior to call any data acquisition function the function card must be initialised and the front-end circuitry (source, coupling, termination, gain, offset) of all used channels should be properly configured.

6.3.4.1 Get immediate value

The following example shows the simplest way to get data form the selected channel – just read immediate value.

```
#define MB_INSTR_DESCR "VXI0::11::INSTR"
#define SD_ADC_FC   2
#define CHANNEL     3

int main(int argc, char *argv[])
{
  ViSession vi;
  ViReal64 value;

  bu3430_paramInit(MB_INSTR_DESCR, SD_ADC_FC, VI_TRUE, VI_TRUE, &vi);
  bu3430_setChanConfig (vi, CHANNEL, bu3430_CHX_MODE_DIFF,
                            bu3430_CHX_COUP_DC, 1,
                            0.0, bu3430_TERM_1M);
  bu3430_sampleChannel(vi, CHANNEL, &value);
  bu3430_close(vi);
}
```

6.3.4.2 Acquire waveforms

The following example shows how to get the waveform from two channels at the same time.

```
#define MB_INSTR_DESCR "VXI0::11::INSTR"
#define SD_ADC_FC     2
#define CHANNEL_MASK  5 /* waveforms from channels 1 and 3 */
#define N_SCANS       100

int main(int argc, char *argv[])
{
  ViSession vi;
  ViReal64 waveforms[N_SCANS * 2]; /*waveforms from two channels*/

  bu3430_paramInit(MB_INSTR_DESCR, SD_ADC_FC, VI_TRUE, VI_TRUE, &vi);
  bu3430_setChanConfig (vi, 1, bu3430_CHX_MODE_DIFF,
                            bu3430_CHX_COUP_DC, 1,
                            0.0, bu3430_TERM_1M);
```

```
  bu3430_setChanConfig (vi, 3, bu3430_CHX_MODE_DIFF,
```



```
                                bu3430_CHX_COUP_DC, 1,
                                0.0, bu3430_TERM_1M);
    bu3430_acquireWaveforms (vi, CHANNEL_MASK, N_SCANS, 0,
                               bu3430_GROUP_BY_CHANNEL,NULL,
                                waveforms);
    bu3430_close(vi);
  }
```

In this example after **bu3430_acquireWaveforms()** function call the first 100 elements of waveforms[] array will contain the data from channel 1 and next 100 elements will contain the data from the channel 3.

6.3.4.3 Customized asynchronous acquisition

The most complicated way to operate with the function card is to build the customized asynchronous data acquisition process. The flowchart in Figure 11 gives an illustration of the steps involved in configuring the instrument and run the data acquisition.

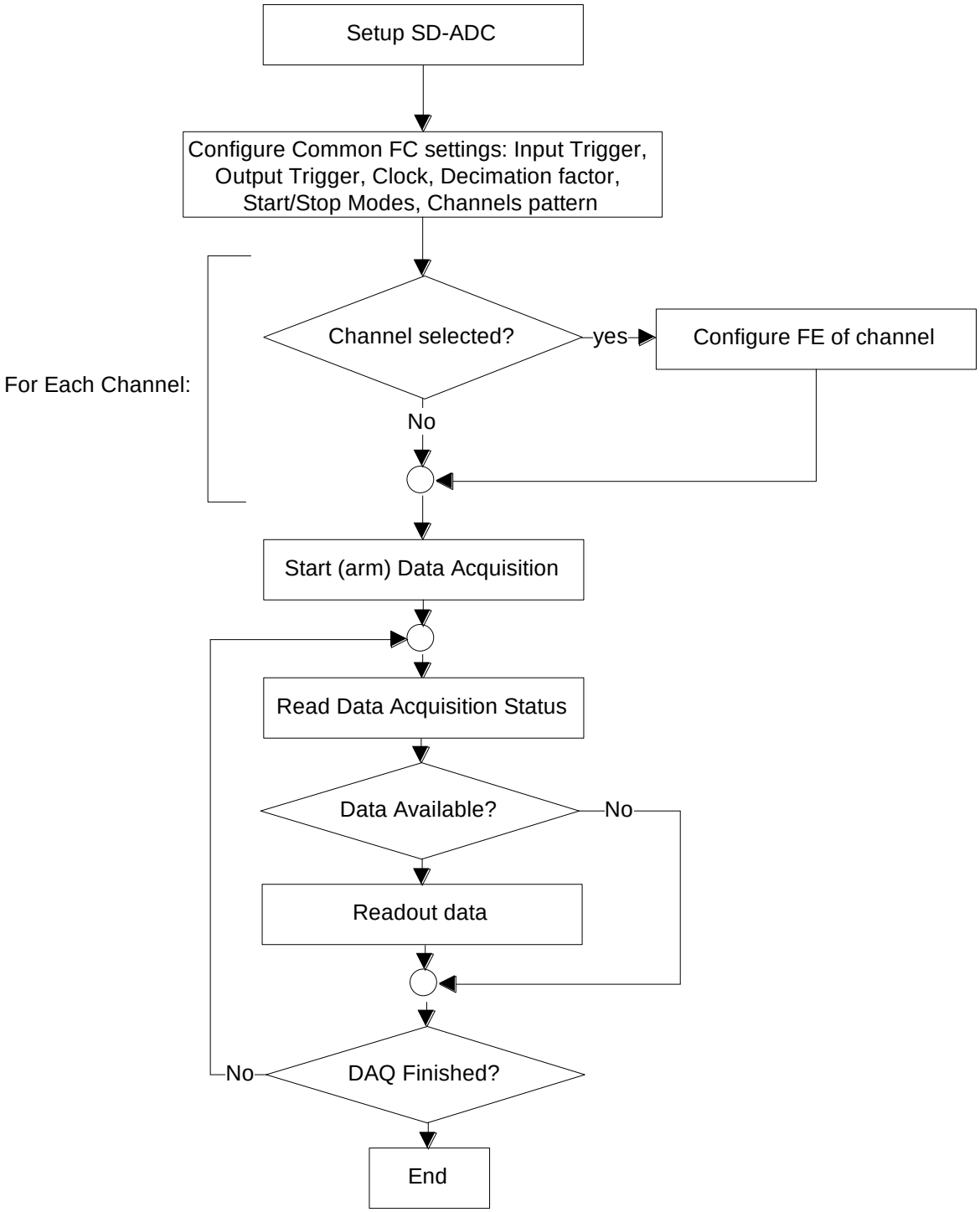

Figure 11: Asynchronous Data Acquisition flowchart

7. Appendix A

Driver Function Reference:

```
bu3430_acquireTriggeredWaveforms

    ViStatus bu3430_acquireTriggeredWaveforms (ViSession instrumentHandle,
                                               ViInt16 channelMask,
                                               ViInt32 numberofScans,
                                               ViInt32 decimationFactor,
                                               ViInt16 triggerSource,
                                               ViReal64 triggerLevel,
                                               ViInt16 fillMode,
                                               ViReal64 *actualScanRate,
                                               ViReal64 waveforms[]);

Purpose

    Acquires the triggered waveforms from the selected channels.

Parameter List

    instrumentHandle

        Variable Type       ViSession

        The Instrument Handle is used to identify the unique session or
        communication channel between the driver and the instrument.

        If more than one instrument of the same model type is used, this
        Handle will be used to differentiate between them.


    channelMask

        Variable Type       ViInt16

        Selects the channels from which the data will be taken.

        bit 0 corresponds to the Channel 1
        ...
        bit 3 corresponds to the Channel 4

        "1" written in the appropriate bit means that the channel will be
        included in the data acquisition.

    numberofScans

        Variable Type       ViInt32

        Specifies the number of samples to be taken from each selected
        channel.

    decimationFactor

        Variable Type       ViInt32

        Sets the value for the decimation factor.
        Possible values are from 0 to 10.
        If the decimation factor equals N
        the decimation will be N power of 2.
        Default value is 0.
```


triggerSource

Variable Type ViInt16

Specifies the source of the Input Trigger line. Possible values are:

bu3430_NONE	0x0000	disabled (default) The data acquisitio will start immediately regardless on the trigger line state.
bu3430_ITRI_FP_HI	0x0002	Input Trigger will be taken from front panel, acvtive level high
bu3430_ITRI_FP_LOW	0x0003	Input Trigger will be taken from front panel, acvtive level low
bu3430_ITRI_MB	0x0004	Input Trigger will be taken from the Motherboard line

Motherboard input can be ORed together with FP input of selected polarity.

triggerLevel

Variable Type ViReal64

Specifies the threshold level for the External Trigger signal. The range of the values is from -5.0V to +5.0V.
Trigger Level is valid only when the Trigger Source is taken from the Front Panel Input of the Function Card.

fillMode

Variable Type ViInt16

The parameter specifies whether the Waveform array will be grouped by channels or grouped by scans.

For example:

If you scan channels A through C and Number of Scans is 5, then the possible fill modes are:

Grouped by channel:

```
   A1 A2 A3 A4 A5 B1 B2 B3 B4 B5 C1 C2 C3 C4 C5
    \----------/   \----------/   \----------/
```

or

Grouped by scan:

```
  A1 B1 C1 A2 B2 C2 A3 B3 C3 A4 B4 C4 A5 B5 C5
   \----/   \----/   \----/   \----/   \----/
```

If you are to pass the array to a graph, you should acquire the data grouped by channel.

If you are to pass the array to a strip chart, you should acquire the data grouped by scan.

actualScanRate

Variable Type ViReal64 (passed by reference)

The actual scan rate (number of scans per second) depends on the

sampling clock frequency and the decimation factor selected.
This value will be calculated properly only if internal sampling clock was selected.

waveforms

Variable Type ViReal64[]

The output buffer containing the samples from the specified channel. This buffer should be allocated by application before the function call with appropriate size to hold all data.

Return Value

Displays the return status of the function call. If the function was successful, it will return a status of VI_SUCCESS, otherwise it will return an error code. Passing the error code into the function "bu3430_error_message", will return a string describing the error.

VISA Errors:
See section 3.3 of the VPP 4.2.2 document for a complete list of the WIN Framework error codes and their values. Appendix B of VPP 4.2 contains descriptions and a breakdown of the error codes returned by each of the VISA functions.

BU3430 Driver Errors:
Errors returned from the BU3430 drivers will be between 0xBFFC0800 and 0xBFFC0FFF.

---

bu3430_acquireWaveform

```
ViStatus bu3430_acquireWaveform (ViSession instrumentHandle, ViInt16 channel,
                                 ViInt32 numberofSamples,
                                 ViInt32 decimationFactor,
                                 ViReal64 *actualSampleRate,
                                 ViReal64 waveform[]);
```

Purpose

Collects the waveform from the selected channel.

Parameter List

instrumentHandle

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

channel

Variable Type ViInt16

Selects the channel from which the waveform will be acquired.

numberofSamples

Variable Type ViInt32


```
        Specifies the number of samples to be taken from the selected
        channel.

    decimationFactor

        Variable Type       ViInt32

        Sets the value for the decimation factor.
        Possible values are from 0 to 10.
        If the decimation factor equals N
        the decimation will be N power of 2.
        Default value is 0.

    actualSampleRate

        Variable Type       ViReal64 (passed by reference)

        The actual sample rate (number of samples per second) depends on the
        sampling clock frequency and the decimation factor selected.
        This value will be calculated properly only if internal sampling
        clock was selected.

    waveform

        Variable Type       ViReal64[]

        The output buffer containing the samples from the specified channel.
        This buffer should be allocated by application before the function
        call with appropriate size to hold all data.

Return Value

        Displays the return status of the function call.  If the function was
        successful, it will return a status of VI_SUCCESS, otherwise it will
        return an error code.  Passing the error code into the function
        "bu3430_error_message", will return a string describing the error.

        VISA Errors:
        See section 3.3 of the VPP 4.2.2 document for a complete list of the
        WIN Framework error codes and their values.  Appendix B of VPP 4.2
        contains descriptions and a breakdown of the error codes returned by
        each of the VISA functions.

        BU3430 Driver Errors:
        Errors returned from the BU3430 drivers will be between 0xBFFC0800
        and 0xBFFC0FFF.
```

```
bu3430_acquireWaveforms

    ViStatus bu3430_acquireWaveforms (ViSession instrumentHandle,
                                      ViInt16 channelMask, ViInt32 numberofScans,
                                      ViInt32 decimationFactor, ViInt16 fillMode,
                                      ViReal64 *actualScanRate,
                                      ViReal64 waveforms[]);

Purpose

    Acquires the waveforms from the selected channels.

Parameter List

    instrumentHandle
```

```
    Variable Type       ViSession

    The Instrument Handle is used to identify the unique session or
    communication channel between the driver and the instrument.

    If more than one instrument of the same model type is used, this
    Handle will be used to differentiate between them.


channelMask

    Variable Type       ViInt16

    Selects the channels from which the data will be taken.

    bit 0 corresponds to the Channel 1
    ...
    bit 3 corresponds to the Channel 4

    "1" written in the appropriate bit means that the channel will be
    included in the data acquisition.

numberofScans

    Variable Type       ViInt32

    Specifies the number of samples to be taken from each selected
    channel.

decimationFactor

    Variable Type       ViInt32

    Sets the value for the decimation factor.
    Possible values are from 0 to 10.
    If the decimation factor equals N
    the decimation will be N power of 2.
    Default value is 0.

fillMode

    Variable Type       ViInt16

    The parameter specifies whether the Waveform array will be grouped by
    channels or grouped by scans.

    For example:

    If you scan channels A through C and Number of Scans is 5, then the
    possible fill modes are:

    Grouped by channel:
       A1 A2 A3 A4 A5 B1 B2 B3 B4 B5 C1 C2 C3 C4 C5
        \----------/   \----------/   \----------/
    or


    Grouped by scan:
      A1 B1 C1 A2 B2 C2 A3 B3 C3 A4 B4 C4 A5 B5 C5
       \----/   \----/   \----/   \----/   \----/


    If you are to pass the array to a graph, you should acquire the data
    grouped by channel.
```


If you are to pass the array to a strip chart, you should acquire the data grouped by scan.

actualScanRate

Variable Type ViReal64 (passed by reference)

The actual scan rate (number of scans per second) depends on the sampling clock frequency and the decimation factor selected. This value will be calculated properly only if internal sampling clock was selected.

waveforms

Variable Type ViReal64[]

The output buffer containing the samples from the specified channel. This buffer should be allocated by application before the function call with appropriate size to hold all data.

Return Value

Displays the return status of the function call. If the function was successful, it will return a status of VI_SUCCESS, otherwise it will return an error code. Passing the error code into the function "bu3430_error_message", will return a string describing the error.

VISA Errors:
See section 3.3 of the VPP 4.2.2 document for a complete list of the WIN Framework error codes and their values. Appendix B of VPP 4.2 contains descriptions and a breakdown of the error codes returned by each of the VISA functions.

BU3430 Driver Errors:
Errors returned from the BU3430 drivers will be between 0xBFFC0800 and 0xBFFC0FFF.

---

bu3430_checkAcquisition

```
ViStatus bu3430_checkAcquisition (ViSession instrumentHandle, ViInt16 *state,
                                  ViInt16 *err, ViInt32 *scanbacklog);
```

Purpose

Returns the state of the last or current Data Acquisition.

Parameter List

instrumentHandle

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

state

Variable Type ViInt16 (passed by reference)


```
        Returns the status of the Data Acquisition process. Possible values
        are:
        bu3430_DAQ_IDLE    0  IDLE state
        bu3430_DAQ_ARMED   1  ARMED state - DAQ ready to start, waiting
                              for the synchronization pulse
        bu3430_DAQ_SYNCH   2  synchronization pulse came and
                              synchronization in progress
        bu3430_DAQ_READY   3  synchronization completed and board waits
                              for event (trigger) to start
        bu3430_DAQ_RUNNING 4  board is collecting data
        bu3430_DAQ_END  5  DAQ is completed or aborted

    err

        Variable Type       ViInt16 (passed by reference)

        Returns the error code if any happened during the Data Acquisition
        process. This value might be bitwise-OR of the following values:
                              0  No error
           bu3430_OW_ERROR    1  Overwrite error
           bu3430_OF_ERROR    2  Overflow error
           bu3430_SCAN_ERROR  2  Scan error



    scanbacklog

        Variable Type       ViInt32 (passed by reference)

        Returns the backlog of scans that have been acquired into the buffer
        but have not been read using bu3430_readAcquisition.

        If bu3430_readAcquisition() is called in "latest" read mode, the scan
        backlog is reset to zero.

Return Value

        Displays the return status of the function call.  If the function was
        successful, it will return a status of VI_SUCCESS, otherwise it will
        return an error code.  Passing the error code into the function
        "bu3430_error_message", will return a string describing the error.

        VISA Errors:
        See section 3.3 of the VPP 4.2.2 document for a complete list of the
        WIN Framework error codes and their values.  Appendix B of VPP 4.2
        contains descriptions and a breakdown of the error codes returned by
        each of the VISA functions.

        BU3430 Driver Errors:
        Errors returned from the BU3430 drivers will be between 0xBFFC0800
        and 0xBFFC0FFF.
```

```
bu3430_close

    ViStatus bu3430_close (ViSession instrumentHandle);

Purpose

    Closes the instrument and deallocates the resources allocated by the call
    to the initialisation function bu3430_init().

    This should be called once for every instrument handle returned by the
    initialise functions prior to terminating the application program.
```


Parameter List

instrumentHandle

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

Return Value

Displays the return status of the function call. If the function was successful, it will return a status of VI_SUCCESS, otherwise it will return an error code. Passing the error code into the function "bu3430_error_message" will return a string describing the error.

VISA Errors:
See section 3.3 of the VPP 4.2.2 document for a complete list of the WIN Framework error codes and their values. Appendix B of VPP 4.2 contains descriptions and a breakdown of the error codes returned by each of the VISA functions.

BU3430 Driver Errors:
Errors returned from the BU3430 drivers will be between 0xBFFC0800 and 0xBFFC0FFF.

---

bu3430_error_message

```
ViStatus bu3430_error_message (ViSession instrumentHandle,
                               ViStatus errorReturnValue, ViChar
errorMessage[]);
```

Purpose

Converts a numeric error code returned by one of the functions of this driver into a descriptive error message string.

Parameter List

instrumentHandle

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

errorReturnValue

Variable Type ViStatus

Accepts the error code returned by one of the functions in this instrument driver.

errorMessage

Variable Type ViChar[]

Upon return from the function, holds a text error message which corresponds to the error code.

The VISA Warnings and VISA Errors are described in section 3.3 of the VPP 4.2.2 document and Appendix B of VPP 4.2

Return Value

Displays the return status of the function call. If the function was successful, it will return a status of VI_SUCCESS, otherwise it will return an error code. Passing the error code into this function, i.e. "bu3430_error_message", will return a string describing the error.

VISA Errors:
See section 3.3 of the VPP 4.2.2 document for a complete list of the WIN Framework error codes and their values. Appendix B of VPP 4.2 contains descriptions and a breakdown of the error codes returned by each of the VISA functions.

bu3430 Driver Errors:
Errors returned from the bu3430 drivers will be between 0xBFFC0900 and 0xBFFC0FFF.

---

bu3430_fcSelect

```
ViStatus bu3430_fcSelect (ViSession instrumentHandle, ViInt16 functionCard,
                          ViBoolean resetFC);
```

Purpose

Selects the Function Card to be accessed further bu the driver's functions.

Parameter List

instrumentHandle

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

functionCard

Variable Type ViInt16

The function card to which the instrument handler will be binded.

resetFC

Variable Type ViBoolean

Specifies if the Function Card is to be reset to its power-on settings during the initialisation procedure.


```
          Valid Range:   1 = Yes
                         0 = No

          Default Value: 1 - Yes

         NOTE: If you do not want the instrument reset set this control to No
         while initialising the instrument.

  Return Value

         Displays the return status of the function call.  If the function was
         successful, it will return a status of VI_SUCCESS, otherwise it will
         return an error code.  Passing the error code into the function
         "bu3430_error_message", will return a string describing the error.

         VISA Errors:
         See section 3.3 of the VPP 4.2.2 document for a complete list of the
         WIN Framework error codes and their values.  Appendix B of VPP 4.2
         contains descriptions and a breakdown of the error codes returned by
         each of the VISA functions.

         BU3430 Driver Errors:
         Errors returned from the BU3430 drivers will be between 0xBFFC0800
         and 0xBFFC0FFF.

_____________________________________________________________________________

  bu3430_getAcquisitionAttribute

      ViStatus bu3430_getAcquisitionAttribute (ViSession instrumentHandle,
                                               ViInt32 attribute, ViInt32 *value);

  Purpose

      Returns the value for the specified attribute for the Data Acquisition
      Process.

  Parameter List

      instrumentHandle

          Variable Type       ViSession

          The Instrument Handle is used to identify the unique session or
          communication channel between the driver and the instrument.

          If more than one instrument of the same model type is used, this
          Handle will be used to differentiate between them.


      attribute

          Variable Type       ViInt32

          Specifies the DAQ Attribute for which the value should be retrieved.

          Possible Attributes:

          bu3430_DAQ_ATTR_STOP_ON_ERR   0

            Possible Values:
             bu3430_OFF    0   (Default) DAQ will continue regardless on
                                data error happened during DAQ
             bu3430_ON     1    DAQ will abort immediately if any data
                                error happened.
```

```
        bu3430_DAQ_ATTR_STOP_ON_TRIG  1

          Possible Values:
            bu3430_OFF   0   (Default) DAQ will continue regardless on
                              the state of the Input Trigger Line.
            bu3430_ON    1    DAQ will stop when the Input Trigger will
                              go from active to inactive state (edge
                              sensitive).

        bu3430_DAQ_ATTR_START_ON_TRIG 2

          Possible Values:
            bu3430_OFF   0   (Default) DAQ will start immediately after
                              synchronization completed.
            bu3430_ON    1   After synchronization completed DAQ will
                             wait until trigger will go to active state

        bu3430_DAQ_ATTR_BUF_SIZE      3

          This attribute is available only if the board is used with
          ProDAQ 3150 High Performance Motherboard and specifies
          the size for the internal circular buffer in 16-bit
          words. Default value is 64Kwords. If zero is specified, the
          internal circular buffer will be disabled and Read Acquisition
          function will read directly from the on-board FIFO.

        bu3430_DAQ_ATTR_BUF_START     4

          This attribute is available only if the board is used with
          ProDAQ 3150 High Performance Motherboard and specifies
          the starting address for the internal circular buffer. This
          address might be acquired by using bu3150_allocDram function.

        bu3430_DAQ_ATTR_BUF_THRESHOLD 5

          This attribute is available only if the board is used with
          ProDAQ 3150 High Performance Motherboard and specifies
          the threshold - number of 16-bit samples collected in the
          internal circular buffer when the registered interrupt service
          routine will be called. Default value is 0x10000

        bu3430_DAQ_ATTR_NO_LP         6

          This attribute is available only if the Function Card is
          installed on bu3150 ProDAQ High Performance Motherboard (HPM).

          Possible values are:
            VI_FALSE   0   (Default) The LIST Processor of HPM will
                           readout the data from the Function Card FIFO
                           to the HPM DRAM buffer during the Data
                           Acquisition process.
            VI_TRUE    1   The data will remain in the Function Card
                           FIFO until it will be explicitly readout by
                           the driver functions. LIST processor will
                           not be involved in the Data Acquisition
                           process.

    value

        Variable Type       ViInt32 (passed by reference)

        Returns the value for the selected attribute.

Return Value
```


Displays the return status of the function call. If the function was successful, it will return a status of VI_SUCCESS, otherwise it will return an error code. Passing the error code into the function "bu3430_error_message", will return a string describing the error.

VISA Errors:
See section 3.3 of the VPP 4.2.2 document for a complete list of the WIN Framework error codes and their values. Appendix B of VPP 4.2 contains descriptions and a breakdown of the error codes returned by each of the VISA functions.

BU3430 Driver Errors:
Errors returned from the BU3430 drivers will be between 0xBFFC0800 and 0xBFFC0FFF.

---

bu3430_getChanConfig

```
ViStatus bu3430_getChanConfig (ViSession instrumentHandle, ViInt16 channel,
                               ViInt16 *source, ViInt16 *coupling,
                               ViInt16 *gain, ViReal64 *offset,
                               ViInt16 *termination);
```

Purpose

Returns the configuration of the selected channel.

Parameter List

instrumentHandle

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

channel

Variable Type ViInt16

Selects the channel for which the FE configuration should be taken.

source

Variable Type ViInt16 (passed by reference)

Returns the connection mode of the given channel.
Possible values are:

bu3430_CHX_MODE_DIFF	0	- differential input
bu3430_CHX_MODE_SINGLE	1	- single-ended input with offset adjusting
bu3430_CHX_MODE_VREF	2	- voltage reference signal
bu3430_CHX_MODE_GND	3	- both inputs of the channel are grounded

coupling

Variable Type ViInt16 (passed by reference)

Returns the type of coupling of Channel to the input signal.


```
        Possible values are:

        bu3430_CHX_COUP_AC  1  AC coupling (Default)
        bu3430_CHX_COUP_DC  0  DC coupling

    gain

        Variable Type       ViInt16 (passed by reference)

        Returns the gain for the selected channel.
        Depending on the version of the hardware possible values are:
        1, 2, 5, 10, 100, 200, 500, 1000
        or
        1, 2, 5, 10, 20, 50, 100

    offset

        Variable Type       ViReal64 (passed by reference)

        Returns the offset voltage applied to the negative input of the
        channel, if the channel configured for single-ended mode.
        The range of the values is from -5.0V to +5.0V.

    termination

        Variable Type       ViInt16 (passed by reference)

        Returns the termination of input signal for the selected channel.
        Possible values are:
        bu3430_TERM_1M 0 - In single-ended mode input signal is terminated to
        ground with 1MOhm resistor. In differential mode both inputs
        (positive and negative) are terminated to ground with 1MOhm resistor.

        bu3430_TERM_50 1 - In single-ended mode input signal is terminated to
        ground with 50 Ohm resistor. In differential mode both inputs
        terminated to each other with 100Ohm resistors.

        bu3430_TERM_50_GND 2 - In single-ended mode input signal is
        terminated to ground with 50 Ohm resistor. In differential mode both
        inputs terminated to ground with 50 Ohm resistors.

Return Value

        Displays the return status of the function call.  If the function was
        successful, it will return a status of VI_SUCCESS, otherwise it will
        return an error code.  Passing the error code into the function
        "bu3430_error_message", will return a string describing the error.

        VISA Errors:
        See section 3.3 of the VPP 4.2.2 document for a complete list of the
        WIN Framework error codes and their values.  Appendix B of VPP 4.2
        contains descriptions and a breakdown of the error codes returned by
        each of the VISA functions.

        BU3430 Driver Errors:
        Errors returned from the BU3430 drivers will be between 0xBFFC0800
        and 0xBFFC0FFF.
```

```
bu3430_getChanPattern

    ViStatus bu3430_getChanPattern (ViSession instrumentHandle,
                                    ViInt16 *channelPattern);
```


Purpose

Returns the channels involved in the Data Acquisition.

Parameter List

instrumentHandle

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

channelPattern

Variable Type ViInt16 (passed by reference)

Selects the channels which are enabled for the DAQ.

bit 0 corresponds to the Channel 1
...
bit 3 corresponds to the Channel 4

"1" written in the appropriate bit means that the channel is included in the data acquisition.

Return Value

Displays the return status of the function call. If the function was successful, it will return a status of VI_SUCCESS, otherwise it will return an error code. Passing the error code into the function "bu3430_error_message", will return a string describing the error.

VISA Errors:
See section 3.3 of the VPP 4.2.2 document for a complete list of the WIN Framework error codes and their values. Appendix B of VPP 4.2 contains descriptions and a breakdown of the error codes returned by each of the VISA functions.

BU3430 Driver Errors:
Errors returned from the BU3430 drivers will be between 0xBFFC0800 and 0xBFFC0FFF.

---

bu3430_getClockSrc

ViStatus bu3430_getClockSrc (ViSession instrumentHandle, ViInt32 *clockSource);

Purpose

Returns the source for the sampling clock signal.

Parameter List

instrumentHandle

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

clockSource

Variable Type ViInt32 (passed by reference)

Returns the source for the sampling clock signal.
Possible values are:

bu3430_CLOCK_INTERNAL 0 Internally generated signal (Default)
bu3430_CLOCK_EXTERNAL 1 External signal applied to FP connector

Return Value

Displays the return status of the function call. If the function was successful, it will return a status of VI_SUCCESS, otherwise it will return an error code. Passing the error code into the function "bu3430_error_message", will return a string describing the error.

VISA Errors:
See section 3.3 of the VPP 4.2.2 document for a complete list of the WIN Framework error codes and their values. Appendix B of VPP 4.2 contains descriptions and a breakdown of the error codes returned by each of the VISA functions.

BU3430 Driver Errors:
Errors returned from the BU3430 drivers will be between 0xBFFC0800 and 0xBFFC0FFF.

---

bu3430_getDAQmode

```
ViStatus bu3430_getDAQmode (ViSession instrumentHandle, ViInt16 *startMode,
                            ViInt16 *stopMode, ViBoolean *breakonerror);
```

Purpose

Returns the operation mode of the data acquisition process.

Parameter List

instrumentHandle

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

startMode

Variable Type ViInt16 (passed by reference)

Selects the way how the Data Acquisition stops. Possible values are:

bu3430_DAQ_START_IMMEDIATE 0 Immediate Mode (default) - DAQ starts immediately after synchronisation completed.


```
        bu3430_DAQ_START_TRIG       1  Triggered mode - DAQ starts when
                                       input trigger goes active (edge
                                       sensitive).

    stopMode

        Variable Type       ViInt16 (passed by reference)

        Selects the way how the Data Acquisition stops. Possible values are:

        bu3430_DAQ_STOP_CONT  0  (Default) Continuous Mode - DAQ stops
                                 when
                                 specified number of samples is
                                 collected

        bu3430_DAQ_STOP_TRIG  1  Triggered mode - DAQ stops when input
                                 trigger goes inactive

    breakonerror

        Variable Type       ViBoolean (passed by reference)

        Specifies whether the Data Acquisition process should break if any
        error encountered.

        Possible values are:

           bu3430_OFF    0   (Default) DAQ will continue regardless on
                              data error happened during DAQ
           bu3430_ON     1    DAQ will abort immediately if any data
                              error happened.


Return Value

        Displays the return status of the function call.  If the function was
        successful, it will return a status of VI_SUCCESS, otherwise it will
        return an error code.  Passing the error code into the function
        "bu3430_error_message", will return a string describing the error.

        VISA Errors:
        See section 3.3 of the VPP 4.2.2 document for a complete list of the
        WIN Framework error codes and their values.  Appendix B of VPP 4.2
        contains descriptions and a breakdown of the error codes returned by
        each of the VISA functions.

        BU3430 Driver Errors:
        Errors returned from the BU3430 drivers will be between 0xBFFC0800
        and 0xBFFC0FFF.

_____________________________________________________________________________

bu3430_getDecimFactor

    ViStatus bu3430_getDecimFactor (ViSession instrumentHandle,
                                    ViInt32 *decimFactor);

Purpose

    Returns the decimation factor for the data acquisition.

Parameter List

    instrumentHandle
```

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

decimFactor

Variable Type ViInt32 (passed by reference)

Returns the value for the decimation factor.
Possible values are from 0 to 10.
If the decimation factor equals N
the decimation will be N power of 2.

Return Value

Displays the return status of the function call. If the function was successful, it will return a status of VI_SUCCESS, otherwise it will return an error code. Passing the error code into the function "bu3430_error_message", will return a string describing the error.

VISA Errors:
See section 3.3 of the VPP 4.2.2 document for a complete list of the WIN Framework error codes and their values. Appendix B of VPP 4.2 contains descriptions and a breakdown of the error codes returned by each of the VISA functions.

BU3430 Driver Errors:
Errors returned from the BU3430 drivers will be between 0xBFFC0800 and 0xBFFC0FFF.

---

bu3430_getFEconf

```
ViStatus bu3430_getFEconf (ViSession instrumentHandle, ViInt16
*terminationClk,
                           ViReal64 *thresholdClk, ViInt16 *terminationSynch,
                           ViReal64 *thresholdSynch);
```

Purpose

Configures the Front-End circuit of the selected channel.

Parameter List

instrumentHandle

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

terminationClk

Variable Type ViInt16 (passed by reference)

Returns the termination of input signal for the External Clock signal. Possible values are:

bu3430_CHX_TERM_1M 1 - External Clock signal terminated to ground with 1MOhm resistor.

bu3430_CHX_TERM_50 0 - External Clock signal is terminated to ground with 50 Ohm resistor.

thresholdClk

Variable Type ViReal64 (passed by reference)

Returnss the threshold level for the External Clock signal. The range of the values is from -5.0V to +5.0V.

terminationSynch

Variable Type ViInt16 (passed by reference)

Returnss the termination of input signal for the External Synch/Trigger signal. Possible values are:

bu3430_CHX_TERM_1M 1 - External Synch/Trigger signal is terminated to ground with 1MOhm resistor.

bu3430_CHX_TERM_50 0 - External Synch/Trigger signal is terminated to ground with 50 Ohm resistor.

thresholdSynch

Variable Type ViReal64 (passed by reference)

Returns the threshold level for the External Synch/Trigger signal. The range of the values is from -5.0V to +5.0V.

Return Value

Displays the return status of the function call. If the function was successful, it will return a status of VI_SUCCESS, otherwise it will return an error code. Passing the error code into the function "bu3430_error_message", will return a string describing the error.

VISA Errors:
See section 3.3 of the VPP 4.2.2 document for a complete list of the WIN Framework error codes and their values. Appendix B of VPP 4.2 contains descriptions and a breakdown of the error codes returned by each of the VISA functions.

BU3430 Driver Errors:
Errors returned from the BU3430 drivers will be between 0xBFFC0800 and 0xBFFC0FFF.

---

bu3430_getFIFOstatus

```
ViStatus bu3430_getFIFOstatus (ViSession instrumentHandle, ViInt32 *status,
                               ViInt32 *minsamples);
```

Purpose

Returns the status of the FIFO flags and estimate number of samples in

```
    the FIFO.

Parameter List

    instrumentHandle

        Variable Type       ViSession

        The Instrument Handle is used to identify the unique session or
        communication channel between the driver and the instrument.

        If more than one instrument of the same model type is used, this
        Handle will be used to differentiate between them.


    status

        Variable Type       ViInt32 (passed by reference)

        Returns the status of the FIFO flags.
        Possible values are:

        bu3430_FIFO_EMPTY          0  N Samples = 0
        bu3430_FIFO_ALMOST_EMPTY   1  N Samples <= PAE Threshold
        bu3430_FIFO_NOT_EMPTY      2  PAE > N Samples > FIFO Half Full
        bu3430_FIFO_HALF_FULL      3  N Samples >= FIFO Half Full
        bu3430_FIFO_ALMOST_FULL    4  N Samples >= PAF Threshold
        bu3430_FIFO_FULL           5  N Samples = Full FIFO

    minsamples

        Variable Type       ViInt32 (passed by reference)

        Returns the minimal estimated number of samples contained in the FIFO

Return Value

        Displays the return status of the function call.  If the function was
        successful, it will return a status of VI_SUCCESS, otherwise it will
        return an error code.  Passing the error code into the function
        "bu3430_error_message", will return a string describing the error.

        VISA Errors:
        See section 3.3 of the VPP 4.2.2 document for a complete list of the
        WIN Framework error codes and their values.  Appendix B of VPP 4.2
        contains descriptions and a breakdown of the error codes returned by
        each of the VISA functions.

        BU3430 Driver Errors:
        Errors returned from the BU3430 drivers will be between 0xBFFC0800
        and 0xBFFC0FFF.
```

```
bu3430_getITRI

    ViStatus bu3430_getITRI (ViSession instrumentHandle, ViInt16 *source,
                             ViInt16 *status);

Purpose

    Returns the configuration and current status for the input trigger.

Parameter List
```


instrumentHandle

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

source

Variable Type ViInt16 (passed by reference)

Returns the source of the Input Trigger line. Possible values are:

bu3430_NONE	0x0000	disabled (default).
bu3430_ITRI_FP_HI	0x0002	Input Trigger will be taken from front panel, active level high
bu3430_ITRI_FP_LOW	0x0003	Input Trigger will be taken from front panel, active level low
bu3430_ITRI_MB	0x0004	Input Trigger will be taken from the Motherboard line

Motherboard input can be ORed together with FP input of selected polarity.

status

Variable Type ViInt16 (passed by reference)

Returns the current status of the Input trigger line. Possible values are:

bu3430_OFF	0	Trigger line is inactive
bu3430_ON	1	Trigger line is active

Return Value

Displays the return status of the function call. If the function was successful, it will return a status of VI_SUCCESS, otherwise it will return an error code. Passing the error code into the function "bu3430_error_message", will return a string describing the error.

VISA Errors:
See section 3.3 of the VPP 4.2.2 document for a complete list of the WIN Framework error codes and their values. Appendix B of VPP 4.2 contains descriptions and a breakdown of the error codes returned by each of the VISA functions.

BU3430 Driver Errors:
Errors returned from the BU3430 drivers will be between 0xBFFC0800 and 0xBFFC0FFF.

---

bu3430_getNScans

```
ViStatus bu3430_getNScans (ViSession instrumentHandle, ViInt32 *scans);
```

Purpose

Returns the number of samples to be collected from each channel enabled for the Data Acquisition.

Parameter List

instrumentHandle

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

scans

Variable Type ViInt32 (passed by reference)

Returns the number of scans to be collected.

Return Value

Displays the return status of the function call. If the function was successful, it will return a status of VI_SUCCESS, otherwise it will return an error code. Passing the error code into the function "bu3430_error_message", will return a string describing the error.

VISA Errors:
See section 3.3 of the VPP 4.2.2 document for a complete list of the WIN Framework error codes and their values. Appendix B of VPP 4.2 contains descriptions and a breakdown of the error codes returned by each of the VISA functions.

BU3430 Driver Errors:
Errors returned from the BU3430 drivers will be between 0xBFFC0800 and 0xBFFC0FFF.

---

bu3430_getOTRIsrc

```
ViStatus bu3430_getOTRIsrc (ViSession instrumentHandle, ViInt16 *source,
                            ViInt16 *outMode, ViInt16 *status);
```

Purpose

Returnss the sources and the status of the output trigger.

Parameter List

instrumentHandle

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

source

Variable Type ViInt16 (passed by reference)

Returns the source of the Output Trigger line. Possible values are:


```
        bu3430_NONE          0x0000  disabled (default)
        bu3430_OTRI_DAQ_ON   0x0020  Trigger remains active as long as
                                     DAQ is in progress
        bu3430_OTRI_DAQ_END  0x0040  Trigger goes active on the DAQ
                                     completion
        bu3430_OTRI_ERR      0x2000  Trigger goes active if error
                                     happened during DAQ

        bu3430_OTRI_FIFO_EF  0x0008  Trigger goes active if FIFO is
                                     empty
        bu3430_OTRI_FIFO_AE  0x0009  Trigger goes active if FIFO is
                                     almost empty
        bu3430_OTRI_FIFO_HF  0x000A  Trigger goes active if FIFO is
                                     half full
        bu3430_OTRI_FIFO_AF  0x000B  Trigger goes active if FIFO is
                                     almost full
        bu3430_OTRI_FIFO_FF  0x000C  Trigger goes active if FIFO is
                                     full
        Any of first four sources can be ORed together or with one of last
        sources (FIFO flags)

    outMode

        Variable Type       ViInt16 (passed by reference)

        Returns the type of output signal of the trigger.
        Possible values are:
        bu3430_OTRI_OUT_PULSE       0 output mode set to pulse
        bu3430_OTRI_OUT_LEVEL       1 output mode set to level

    status

        Variable Type       ViInt16 (passed by reference)

        Returns the current status of the Output trigger line. Possible
        values are:
        bu3430_OFF   0   Trigger line is inactive
        bu3430_ON    1   Trigger line is active

  Return Value

        Displays the return status of the function call.  If the function was
        successful, it will return a status of VI_SUCCESS, otherwise it will
        return an error code.  Passing the error code into the function
        "bu3430_error_message", will return a string describing the error.

        VISA Errors:
        See section 3.3 of the VPP 4.2.2 document for a complete list of the
        WIN Framework error codes and their values.  Appendix B of VPP 4.2
        contains descriptions and a breakdown of the error codes returned by
        each of the VISA functions.

        BU3430 Driver Errors:
        Errors returned from the BU3430 drivers will be between 0xBFFC0800
        and 0xBFFC0FFF.
```

```
  bu3430_getSynchSrc

    ViStatus bu3430_getSynchSrc (ViSession instrumentHandle, ViInt32
*synchSource);

  Purpose
```

```
    Returns the source for the synchronisation pulse.

Parameter List

    instrumentHandle

        Variable Type       ViSession

        The Instrument Handle is used to identify the unique session or
        communication channel between the driver and the instrument.

        If more than one instrument of the same model type is used, this
        Handle will be used to differentiate between them.


    synchSource

        Variable Type       ViInt32 (passed by reference)

        Returns the source for the synchronisation pulse.
        Possible values are:

        bu3430_SYNCH_INTERNAL  0  Internally generated pulse (Default)
        bu3430_SYNCH_EXTERNAL  1  External pulse applied to FP connector

Return Value

        Displays the return status of the function call.  If the function was
        successful, it will return a status of VI_SUCCESS, otherwise it will
        return an error code.  Passing the error code into the function
        "bu3430_error_message", will return a string describing the error.

        VISA Errors:
        See section 3.3 of the VPP 4.2.2 document for a complete list of the
        WIN Framework error codes and their values.  Appendix B of VPP 4.2
        contains descriptions and a breakdown of the error codes returned by
        each of the VISA functions.

        BU3430 Driver Errors:
        Errors returned from the BU3430 drivers will be between 0xBFFC0800
        and 0xBFFC0FFF.

_____________________________________________________________________________

bu3430_init

    ViStatus bu3430_init (ViRsrc instrumentDescriptor, ViBoolean IDQuery,
                          ViBoolean resetDevice, ViSession *instrumentHandle);

Purpose

    Initializes the instrument and returns an "instrument handle".  The
    instrument handle must be used with all of the other functions of this
    driver.

    The initialise call allows the instrument to be queried to ensure that it
    is a Bustec Data Acquisition System.
    It also resets the Module to the power-up state if the "Reset" parameter
    is True (ON).

    This function interrogates the motherboard registers to ascertain in
    which locations there are function cards fitted and then checks those
    locations to identify the type of function card fitted.

    Note that for each "bu3430_init()" call, a new unique instrument handle
```


is returned. Thus, if four calls are made to the initialise call in succession, four unique instrument handles will be returned.

After call of "bu3430_init()", the function "bu3430_fcSelect" must be called to bind the acquired instrument handler to the specific Function Card.

For each instrument handle returned by the "bu3430_init()" function, the "bu3430_close()" function should be called to free up the resources allocated by "bu3430_init()". The call(s) to "bu3430_close()" should be made before the application program terminates.

Parameter List

instrumentDescriptor

Variable Type ViRsrc

Specifies which remote instrument to establish a communication session with. Based on the syntax of the Instr Descriptor, the Initialise function configures the I/O interface and generates an Instr Handle

The default value is for a VXI interface for logical address 7:-

Default Value: "VXI::7::INSTR"

Based on the Instrument Descriptor, this operation establishes a communication session with a device.

The grammar for the Instrument Descriptor is shown below. Optional parameters are shown in square brackets ([]).

For a GPIB-VXI interface with the instrument set to logical address 19, the value should be:

"GPIB-VXI::19::INSTR"

Interface Grammar

VXI	VXI[board]::VXI logical address[::INSTR]
GPIB-VXI	GPIB-VXI[board][::GPIB-VXI primary address] ::VXI logical address[::INSTR]

The VXI keyword is used for VXI instruments via either embedded or MXIbus controllers.
The GPIB-VXI keyword is used for a GPIB-VXI controller.

The default value for optional parameters are shown below.

Optional Parameter	Default Value
board	0
secondary address	none - 31
GPIB-VXI primary address	1

IDQuery

Variable Type ViBoolean

Specifies if an ID Query is sent to the instrument during the initialization procedure.


```
          Valid Range:   1 = Yes
                         0 = No

          Default Value: 1 - Yes

         NOTE:  Under normal circumstances the ID Query insures that the
         instrument initialised over the bus is the type supported by this
         driver. However, circumstances may arise where it is undesirable to
         send an ID Query to the instrument. In those cases set this control
         to Skip Query and this function will initialise the bus and the
         Command arrays in the driver, without doing an ID Query.

    resetDevice

         Variable Type       ViBoolean

         Specifies if the instrument is to be reset to its power-on settings
         during the initialisation procedure.

          Valid Range:   1 = Yes
                         0 = No

          Default Value: 1 - Yes

         NOTE: If you do not want the instrument reset set this control to No
         while initialising the instrument.

    instrumentHandle

         Variable Type       ViSession (passed by reference)

         The Instrument Handle is used to identify the unique session or
         communication channel between the driver and the instrument.

         If more than one instrument of the same model type is used, this
         Handle will be used to differentiate between them.

         NOTE:  A new (unique) handle will be returned EACH time the
         Initialise function is called.  The bu3150_close() call should be
         used for EVERY handle returned by the bu3150_init() function.


Return Value

         Displays the return status of the function call.  If the function was
         successful, it will return a status of VI_SUCCESS, otherwise it will
         return an error code.  Passing the error code into the function
         "bu3430_error_message", will return a string describing the error.

         VISA Errors:
         See section 3.3 of the VPP 4.2.2 document for a complete list of the
         WIN Framework error codes and their values.  Appendix B of VPP 4.2
         contains descriptions and a breakdown of the error codes returned by
         each of the VISA functions.

         BU3430 Driver Errors:
         Errors returned from the BU3430 drivers will be between 0xBFFC0800
         and 0xBFFC0FFF.

_____________________________________________________________________________

bu3430_installAcquisitionCallback

    ViStatus bu3430_installAcquisitionCallback (ViSession instrumentHandle,
                                                bu3100_irqHandler_t callback,
```



```
                                                    void *parameter);
```

Purpose

Installs the asynchronous callback routine to serve the asynchronous data acquisition

Parameter List

instrumentHandle

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

callback

Variable Type bu3100_irqHandler_t

This interrupt service routine should readout data from function card

parameter

Variable Type void *

Pointer to a parameter which will be transferred to the interrupt service routine.

Return Value

Displays the return status of the function call. If the function was successful, it will return a status of VI_SUCCESS, otherwise it will return an error code. Passing the error code into the function "bu3430_error_message", will return a string describing the error.

VISA Errors:
See section 3.3 of the VPP 4.2.2 document for a complete list of the WIN Framework error codes and their values. Appendix B of VPP 4.2 contains descriptions and a breakdown of the error codes returned by each of the VISA functions.

BU3430 Driver Errors:
Errors returned from the BU3430 drivers will be between 0xBFFC0800 and 0xBFFC0FFF.

---

bu3430_paramInit

```
ViStatus bu3430_paramInit (ViRsrc instrumentDescriptor, ViInt16 functionCard,
                           ViBoolean IDQuery, ViBoolean resetDevice,
                           ViSession *instrumentHandle);
```

Purpose

Initialises the specified Function Card on the specified ProDAQ module and returns an "instrument handle". The instrument handle must be used with all of the other functions of this driver.

The Initialise With Parameters call allows the VXI module to be queried to ensure that it is a Bustec Data Acquisition System and the selected

```
    Function Card is one of the appropriate type (Sigma-Delta Function Card).
    It also resets the Module to the power-up state if the "Reset" parameter
    is True (ON).


Parameter List

    instrumentDescriptor

        Variable Type       ViRsrc

        Specifies which remote instrument to establish a communication
        session with.  Based on the syntax of the Instr Descriptor, the
        Initialize function configures the I/O interface and generates an
        Instr Handle

        The default value is for a VXI interface for logical address 7:-

         Default Value: "VXI::7::INSTR"

        Based on the Instrument Descriptor, this operation establishes a
        communication session with a device.

        The grammar for the Instrument Descriptor is shown below.
        Optional parameters are shown in square brackets ([]).

        For a GPIB-VXI interface with the instrument set to logical address
        19, the value should be:

              "GPIB-VXI::19::INSTR"


        Interface Grammar
        ---------------------------------------------------
        VXI       VXI[board]::VXI logical address[::INSTR]
        GPIB-VXI  GPIB-VXI[board][::GPIB-VXI primary address]
                  ::VXI logical address[::INSTR]

        The VXI keyword is used for VXI instruments via either embedded or
        MXIbus controllers.
        The GPIB-VXI keyword is used for a  GPIB-VXI controller.

        The default value for optional parameters are shown below.

        Optional Parameter          Default Value
        ----------------------------------------
        board                       0
        secondary address           none - 31
        GPIB-VXI primary address    1


    functionCard

        Variable Type       ViInt16

        The function card to which the VISA session will be opened.

    IDQuery

        Variable Type       ViBoolean

        Specifies if an ID Query is sent to the instrument during the
        initialisation procedure.

         Valid Range:   1 = Yes
```



```
                        0 = No

         Default Value: 1 - Yes

        NOTE:  Under normal circumstances the ID Query insures that the
        instrument initialised over the bus is the type supported by this
        driver. However, circumstances may arise where it is undesirable to
        send an ID Query to the instrument. In those cases set this control
        to Skip Query and this function will initialise the bus and the
        Command arrays in the driver, without doing an ID Query.

    resetDevice

        Variable Type       ViBoolean

        Specifies if the instrument is to be reset to its power-on settings
        during the initialisation procedure.

         Valid Range:   1 = Yes
                        0 = No

         Default Value: 1 - Yes

        NOTE: If you do not want the instrument reset set this control to No
        while initialising the instrument.

    instrumentHandle

        Variable Type       ViSession (passed by reference)

        The Instrument Handle is used to identify the unique session or
        communication channel between the driver and the instrument.

        If more than one instrument of the same model type is used, this
        Handle will be used to differentiate between them.

        NOTE:  A new (unique) handle will be returned EACH time the
        Initialise function is called.  The bu3150_close() call should be
        used for EVERY handle returned by the bu3150_init() function.


Return Value

        Displays the return status of the function call.  If the function was
        successful, it will return a status of VI_SUCCESS, otherwise it will
        return an error code.  Passing the error code into the function
        "bu3430_error_message", will return a string describing the error.

        VISA Errors:
        See section 3.3 of the VPP 4.2.2 document for a complete list of the
        WIN Framework error codes and their values.  Appendix B of VPP 4.2
        contains descriptions and a breakdown of the error codes returned by
        each of the VISA functions.

        BU3430 Driver Errors:
        Errors returned from the BU3430 drivers will be between 0xBFFC0800
        and 0xBFFC0FFF.

_____________________________________________________________________________

bu3430_pulseITRI

    ViStatus bu3430_pulseITRI (ViSession instrumentHandle, ViInt16 operation);

Purpose
```

```
      Generates the pulse / sets the level on the Input Trigger Line.

  Parameter List

      instrumentHandle

          Variable Type       ViSession

          The Instrument Handle is used to identify the unique session or
          communication channel between the driver and the instrument.

          If more than one instrument of the same model type is used, this
          Handle will be used to differentiate between them.


      operation

          Variable Type       ViInt16

          Specifies the operation to be performed on the Input Trigger line.
          Possible values are:
          bu3430_OFF        0 Set Input Trigger to inactive state
          bu3430_ON         1 Set Input Trigger to active state
          bu3430_ITRI_PULSE 2 Generate a pulse on Input Trigger line

          Pulse will be generated despite of initial state of the Input Trigger
          Line. After pulse generation the Common Trigger line will be set into
          inactive state.

  Return Value

          Displays the return status of the function call.  If the function was
          successful, it will return a status of VI_SUCCESS, otherwise it will
          return an error code.  Passing the error code into the function
          "bu3430_error_message", will return a string describing the error.

          VISA Errors:
          See section 3.3 of the VPP 4.2.2 document for a complete list of the
          WIN Framework error codes and their values.  Appendix B of VPP 4.2
          contains descriptions and a breakdown of the error codes returned by
          each of the VISA functions.

          BU3430 Driver Errors:
          Errors returned from the BU3430 drivers will be between 0xBFFC0800
          and 0xBFFC0FFF.
```

```
  bu3430_readAcquisition

      ViStatus bu3430_readAcquisition (ViSession instrumentHandle,
                                       ViInt32 scanstoRead, ViInt16 readMode,
                                       ViInt16 fillMode, ViInt16 *scanbacklog,
                                       ViInt32 *actualScansRead, ViReal64
waveforms[]);

  Purpose

      Fetches the specified amount of data from the function card.

  Parameter List

      instrumentHandle

          Variable Type       ViSession
```


The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

scanstoRead

Variable Type ViInt32

Specifies the number of 16-bit words to be fetched from each channel of the function card.
-1 means to fetch all available data.

readMode

Variable Type ViInt16

Consecutive
==========
Scans will be read from the internal circular buffer starting from the last scan that was read. Using this mode, you are guaranteed that you will not lose data unless an error occurs.

Latest
======
The most recently acquired n scans are read from the internal circular buffer, where n is Scans to Read. Calling bu3430_readAcquisition() in this mode resets the Scan Backlog to zero.

fillMode

Variable Type ViInt16

The parameter specifies whether the Waveform array will be grouped by channels or grouped by scans.

For example:

If you scan channels A through C and Number of Scans is 5, then the possible fill modes are:

Grouped by channel:

```
   A1 A2 A3 A4 A5 B1 B2 B3 B4 B5 C1 C2 C3 C4 C5
    \----------/   \----------/   \----------/
```

or

Grouped by scan:

```
  A1 B1 C1 A2 B2 C2 A3 B3 C3 A4 B4 C4 A5 B5 C5
   \----/   \----/   \----/   \----/   \----/
```

If you are to pass the array to a graph, you should acquire the data grouped by channel.

If you are to pass the array to a strip chart, you should acquire the data grouped by scan.

scanbacklog

Variable Type ViInt16 (passed by reference)

Returns the backlog of scans that have been acquired into the buffer but have not been read using bu3430_readAcquisition.

If bu3430_readAcquisition() is called in "latest" read mode, the scan backlog is reset to zero.

actualScansRead

Variable Type ViInt32 (passed by reference)

Returns the number of elements fetched from the function card for each channel and stored in the output buffer.

waveforms

Variable Type ViReal64[]

The output buffer containing the samples from the specified channel. This buffer should be allocated by application before the function call with appropriate size to hold all data.

Return Value

Displays the return status of the function call. If the function was successful, it will return a status of VI_SUCCESS, otherwise it will return an error code. Passing the error code into the function "bu3430_error_message", will return a string describing the error.

VISA Errors:
See section 3.3 of the VPP 4.2.2 document for a complete list of the WIN Framework error codes and their values. Appendix B of VPP 4.2 contains descriptions and a breakdown of the error codes returned by each of the VISA functions.

BU3430 Driver Errors:
Errors returned from the BU3430 drivers will be between 0xBFFC0800 and 0xBFFC0FFF.

---

bu3430_readFIFO

```
ViStatus bu3430_readFIFO (ViSession instrumentHandle, ViInt32 n_elements,
                          ViInt16 flush, ViInt16 buffer[], ViInt32 *n_read);
```

Purpose

Readout the specified amount of samples from FIFO.

Parameter List

instrumentHandle

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

n_elements

Variable Type ViInt32

Specifies the number of 16-bit words to be fetched from the function card. Possible values are even numbers from 0 to 0x8000 if the board operates in 32-bit mode with ProDAQ 3150 High-Performance Motherboard and any numbers from 0 to 0x4000 if the board operates in 16-bit mode with ProDAQ 3120 Low-Performance Motherboard

flush

Variable Type ViInt16

If Yes, then this function will try to retrieve all requested data from FIFO even if "FIFO Almost Empty" flag will be set. If No, it will try to retrieve all requested data from FIFO but until "FIFO Almost Empty" flag will be set. When "FIFO Almost Empty" flag is set the function switches from block transfers to single word transfer until "FIFO Empty" flag will be set. Therefore the Flush operation is much more time consuming.

buffer

Variable Type ViInt16[]

Pointer to the buffer where the fetched data should be stored. This buffer should be allocated by application before the function call with appropriate size to hold all fetched data.

n_read

Variable Type ViInt32 (passed by reference)

Returns the number of elements fetched from the function card and stored in the output buffer.

Return Value

Displays the return status of the function call. If the function was successful, it will return a status of VI_SUCCESS, otherwise it will return an error code. Passing the error code into the function "bu3430_error_message", will return a string describing the error.

VISA Errors:
See section 3.3 of the VPP 4.2.2 document for a complete list of the WIN Framework error codes and their values. Appendix B of VPP 4.2 contains descriptions and a breakdown of the error codes returned by each of the VISA functions.

BU3430 Driver Errors:
Errors returned from the BU3430 drivers will be between 0xBFFC0800 and 0xBFFC0FFF.

---

bu3430_reset

ViStatus bu3430_reset (ViSession instrumentHandle);

Purpose

Resets the function card to its power-on state.

Parameter List

instrumentHandle

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

Return Value

Displays the return status of the function call. If the function was successful, it will return a status of VI_SUCCESS, otherwise it will return an error code. Passing the error code into the function "bu3430_error_message", will return a string describing the error.

VISA Errors:
See section 3.3 of the VPP 4.2.2 document for a complete list of the WIN Framework error codes and their values. Appendix B of VPP 4.2 contains descriptions and a breakdown of the error codes returned by each of the VISA functions.

BU3430 Driver Errors:
Errors returned from the BU3430 drivers will be between 0xBFFC0800 and 0xBFFC0FFF.

---

bu3430_resetFIFO

```
ViStatus bu3430_resetFIFO (ViSession instrumentHandle);
```

Purpose

Resets the FIFO pointers to 0.

Parameter List

instrumentHandle

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

Return Value

Displays the return status of the function call. If the function was successful, it will return a status of VI_SUCCESS, otherwise it will return an error code. Passing the error code into the function "bu3430_error_message", will return a string describing the error.

VISA Errors:
See section 3.3 of the VPP 4.2.2 document for a complete list of the WIN Framework error codes and their values. Appendix B of VPP 4.2 contains descriptions and a breakdown of the error codes returned by each of the VISA functions.

BU3430 Driver Errors:
Errors returned from the BU3430 drivers will be between 0xBFFC0800 and 0xBFFC0FFF.

---

bu3430_revision_query

```
ViStatus bu3430_revision_query (ViSession instrumentHandle,
                                ViChar driverRevision[],
                                ViChar instrumentFirmwareRevision[]);
```

Purpose

Returns the driver revision.

However, because the instrument revision query function is not supported this function always returns the VI_WARN_NSUP_REV_QUERY warning.

Parameter List

instrumentHandle

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

driverRevision

Variable Type ViChar []

Returns the Instrument Driver revision

instrumentFirmwareRevision

Variable Type ViChar[]

Because the instrument revision query function is not supported this control always returns the message "Not Available"

Return Value

Displays the return status of the function call. If the function was successful, it will return a status of VI_SUCCESS, otherwise it will return an error code. Passing the error code into the function "bu3430_error_message" will return a string describing the error.

VISA Errors:
See section 3.3 of the VPP 4.2.2 document for a complete list of the WIN Framework error codes and their values. Appendix B of VPP 4.2 contains descriptions and a breakdown of the error codes returned by each of the VISA functions.

BU3430 Driver Errors:
Errors returned from the BU3430 drivers will be between 0xBFFC0800 and 0xBFFC0FFF.

---

bu3430_sampleChannel

```
ViStatus bu3430_sampleChannel (ViSession instrumentHandle, ViInt16 channel,
                               ViReal64 *sample);
```

Purpose

Reads out the contents of the given channel.
If Data Acquisition is in idle state, this function will arm it, wait until synchronization completed and then read out selected channel.

Parameter List

instrumentHandle

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

channel

Variable Type ViInt16

Selects the channel to read.

sample

Variable Type ViReal64 (passed by reference)

After readout contains the value from the specified channel.

Return Value

Displays the return status of the function call. If the function was successful, it will return a status of VI_SUCCESS, otherwise it will return an error code. Passing the error code into the function "bu3430_error_message", will return a string describing the error.

VISA Errors:
See section 3.3 of the VPP 4.2.2 document for a complete list of the WIN Framework error codes and their values. Appendix B of VPP 4.2 contains descriptions and a breakdown of the error codes returned by each of the VISA functions.

BU3430 Driver Errors:
Errors returned from the BU3430 drivers will be between 0xBFFC0800 and 0xBFFC0FFF.

---

bu3430_sampleChannels

```
ViStatus bu3430_sampleChannels (ViSession instrumentHandle, ViInt16
channelMask,
                                ViReal64 samplesArray[]);
```

Purpose


```
    Reads out the samples from the given channels.
    The Data Acquisition must be running or in armed state.

Parameter List

    instrumentHandle

        Variable Type       ViSession

        The Instrument Handle is used to identify the unique session or
        communication channel between the driver and the instrument.

        If more than one instrument of the same model type is used, this
        Handle will be used to differentiate between them.


    channelMask

        Variable Type       ViInt16

        Selects the channels from which the data will be taken.

        bit 0 corresponds to the Channel 1
        ...
        bit 3 corresponds to the Channel 4

        "1" written in the appropriate bit means that the channel will be
        included in the data acquisition.

        If Data Acquisition is in idle state, this function will
        arm it, wait until synchronization completed and then
        read out selected channels.


    samplesArray

        Variable Type       ViReal64[]

        This array will contain the values acquired on the channels specified
        in the Channel Mask.

        This array must be declared with the size of 4 elements.

        Array[0] will contain data for the Channel 1
        ...
        Array[3] will contain data for the Channel 4

        If the channel is not selected with Channel Mask, zero value will bi
        written to the appropriate place.

Return Value

        Displays the return status of the function call.  If the function was
        successful, it will return a status of VI_SUCCESS, otherwise it will
        return an error code.  Passing the error code into the function
        "bu3430_error_message", will return a string describing the error.

        VISA Errors:
        See section 3.3 of the VPP 4.2.2 document for a complete list of the
        WIN Framework error codes and their values.  Appendix B of VPP 4.2
        contains descriptions and a breakdown of the error codes returned by
        each of the VISA functions.

        BU3430 Driver Errors:
        Errors returned from the BU3430 drivers will be between 0xBFFC0800
```

```
        and 0xBFFC0FFF.
```

```
bu3430_self_test

    ViStatus bu3430_self_test (ViSession instrumentHandle, ViInt16 *testResult,
                               ViChar testMessage[]);

Purpose

    Performs a self-test on the instrument
    ** Note Self_test is not supported by the hardware.

Parameter List

    instrumentHandle

        Variable Type       ViSession

        The Instrument Handle is used to identify the unique session or
        communication channel between the driver and the instrument.

        If more than one instrument of the same model type is used, this
        Handle will be used to differentiate between them.


    testResult

        Variable Type       ViInt16 (passed by reference)

        Returns the result of the self test:

                              0 = no error (test passed)
                              1 = test failed

    testMessage

        Variable Type       ViChar[]

        Returns description of result of self-test

Return Value

        Displays the return status of the function call.  If the function was
        successful, it will return a status of VI_SUCCESS, otherwise it will
        return an error code.  Passing the error code into the function
        "bu3430_error_message", will return a string describing the error.

        VISA Errors:
        See section 3.3 of the VPP 4.2.2 document for a complete list of the
        WIN Framework error codes and their values.  Appendix B of VPP 4.2
        contains descriptions and a breakdown of the error codes returned by
        each of the VISA functions.

        BU3430 Driver Errors:
        Errors returned from the BU3430 drivers will be between 0xBFFC0800
        and 0xBFFC0FFF.
```

```
bu3430_setAcquisitionAttribute

    ViStatus bu3430_setAcquisitionAttribute (ViSession instrumentHandle,
                                             ViInt32 attribute, ViInt32 value);
```



```
Purpose

    Sets the specified attribute for the Data Acquisition Process.

Parameter List

    instrumentHandle

        Variable Type       ViSession

        The Instrument Handle is used to identify the unique session or
        communication channel between the driver and the instrument.

        If more than one instrument of the same model type is used, this
        Handle will be used to differentiate between them.


    attribute

        Variable Type       ViInt32

        Specifies the DAQ Attribute for which the new value should be set.

        Possible Attributes:

        bu3430_DAQ_ATTR_STOP_ON_ERR   0

          Possible Values:
           bu3430_OFF    0   (Default) DAQ will continue regardless on
                              data error happened during DAQ
           bu3430_ON     1    DAQ will abort immediately if any data
                              error happened.

        bu3430_DAQ_ATTR_STOP_ON_TRIG  1

          Possible Values:
            bu3430_OFF   0   (Default) DAQ will continue regardless on
                              the state of the Input Trigger Line.
            bu3430_ON    1    DAQ will stop when the Input Trigger will
                              go from active to inactive state (edge
                              sensitive).

        bu3430_DAQ_ATTR_START_ON_TRIG 2

          Possible Values:
            bu3430_OFF   0   (Default) DAQ will start immediately after
                              synchronization completed.
            bu3430_ON    1   After synchronization completed DAQ will
                             wait until trigger will go to active state

        bu3430_DAQ_ATTR_BUF_SIZE      3

          This attribute is available only if the board is used with
          ProDAQ 3150 High Performance Motherboard and specifies
          the size for the internal circular buffer in 16-bit
          words. Default value is 64Kwords. If zero is specified, the
          internal circular buffer will be disabled and Read Acquisition
          function will read directly from the on-board FIFO.

        bu3430_DAQ_ATTR_BUF_START     4

          This attribute is available only if the board is used with
          ProDAQ 3150 High Performance Motherboard and specifies
          the starting address for the internal circular buffer. This
          address might be acquired by using bu3150_allocDram function.
```

```
        bu3430_DAQ_ATTR_BUF_THRESHOLD 5

          This attribute is available only if the board is used with
          ProDAQ 3150 High Performance Motherboard and specifies
          the threshold - number of 16-bit samples collected in the
          internal circular buffer when the registered interrupt service
          routine will be called. Default value is 0x10000

        bu3430_DAQ_ATTR_NO_LP         6

          This attribute is available only if the Function Card is
          installed on bu3150 ProDAQ High Performance Motherboard (HPM).

          Possible values are:
            VI_FALSE   0   (Default) The LIST Processor of HPM will
                           readout the data from the Function Card FIFO
                           to the HPM DRAM buffer during the Data
                           Acquisition process.
            VI_TRUE    1   The data will remain in the Function Card
                           FIFO until it will be explicitly readout by
                           the driver functions. LIST processor will
                           not be involved in the Data Acquisition
                           process.

    value

        Variable Type       ViInt32

        Specifies the new value for the selected attribute.

Return Value

        Displays the return status of the function call.  If the function was
        successful, it will return a status of VI_SUCCESS, otherwise it will
        return an error code.  Passing the error code into the function
        "bu3430_error_message", will return a string describing the error.

        VISA Errors:
        See section 3.3 of the VPP 4.2.2 document for a complete list of the
        WIN Framework error codes and their values.  Appendix B of VPP 4.2
        contains descriptions and a breakdown of the error codes returned by
        each of the VISA functions.

        BU3430 Driver Errors:
        Errors returned from the BU3430 drivers will be between 0xBFFC0800
        and 0xBFFC0FFF.

_____________________________________________________________________________

bu3430_setChanConfig

    ViStatus bu3430_setChanConfig (ViSession instrumentHandle, ViInt16 channel,
                                   ViInt16 source, ViInt16 coupling, ViInt16 gain,
                                   ViReal64 offset, ViInt16 termination);

Purpose

    Configures the selected channel.

Parameter List

    instrumentHandle

        Variable Type       ViSession
```



```
    The Instrument Handle is used to identify the unique session or
    communication channel between the driver and the instrument.

    If more than one instrument of the same model type is used, this
    Handle will be used to differentiate between them.


channel

    Variable Type       ViInt16

    Selects the channel to be configured.

source

    Variable Type       ViInt16

    Selects the connection source for the given channel.
    Possible values are:
    bu3430_CHX_MODE_DIFF    0 - differential input (default)
    bu3430_CHX_MODE_SINGLE  1 - single-ended input with offset
                                adjusting
    bu3430_CHX_MODE_VREF    2 - voltage reference signal
    bu3430_CHX_MODE_GND     3 - both inputs of the channel are
                                grounded

coupling

    Variable Type       ViInt16

    Specifies the type of coupling of Channel to the input signal.
    Possible values are:

    bu3430_CHX_COUP_AC  1  AC coupling (Default)
    bu3430_CHX_COUP_DC  0  DC coupling

gain

    Variable Type       ViInt16

    Sets the gain for the selected channel.
    Depending on the version of the hardware possible values are:
    1, 2, 5, 10, 100, 200, 500, 1000
    or
    1, 2, 5, 10, 20, 50, 100

offset

    Variable Type       ViReal64

    Specifies the offset voltage applied to the negative input of the
    channel, if the channel configured for single-ended mode.
    The range of the values is from -5.0V to +5.0V.

termination

    Variable Type       ViInt16

    Sets the termination of input signal for the selected channel.
    Possible values are:
    bu3430_CHX_TERM_1M 0 - In single-ended mode input signal is
    terminated to ground with 1MOhm resistor. In differential mode both
    inputs (positive and negative) are terminated to ground with 1MOhm
    resistor. (default)
```


bu3430_CHX_TERM_50 1 - In single-ended mode input signal is terminated to ground with 50 Ohm resistor. In differential mode both inputs terminated to each other with 100Ohm resistors.

bu3430_CHX_TERM_50_GND 2 - In single-ended mode input signal is terminated to ground with 50 Ohm resistor. In differential mode both inputs terminated to ground with 50 Ohm resistors.

Return Value

Displays the return status of the function call. If the function was successful, it will return a status of VI_SUCCESS, otherwise it will return an error code. Passing the error code into the function "bu3430_error_message", will return a string describing the error.

VISA Errors:
See section 3.3 of the VPP 4.2.2 document for a complete list of the WIN Framework error codes and their values. Appendix B of VPP 4.2 contains descriptions and a breakdown of the error codes returned by each of the VISA functions.

BU3430 Driver Errors:
Errors returned from the BU3430 drivers will be between 0xBFFC0800 and 0xBFFC0FFF.

---

bu3430_setChanPattern

```
ViStatus bu3430_setChanPattern (ViSession instrumentHandle,
                                ViInt16 channelPattern);
```

Purpose

Sets the channels to be involved in the Data Acquisition.

Parameter List

instrumentHandle

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

channelPattern

Variable Type ViInt16

Selects the channels which will be enabled for the DAQ.

bit 0 corresponds to the Channel 1
...
bit 3 corresponds to the Channel 4

"1" written in the appropriate bit means that the channel will be included in the data acquisition.

Return Value

Displays the return status of the function call. If the function was

successful, it will return a status of VI_SUCCESS, otherwise it will return an error code. Passing the error code into the function "bu3430_error_message", will return a string describing the error.

VISA Errors:
See section 3.3 of the VPP 4.2.2 document for a complete list of the WIN Framework error codes and their values. Appendix B of VPP 4.2 contains descriptions and a breakdown of the error codes returned by each of the VISA functions.

BU3430 Driver Errors:
Errors returned from the BU3430 drivers will be between 0xBFFC0800 and 0xBFFC0FFF.

---

bu3430_setClockSrc

```
ViStatus bu3430_setClockSrc (ViSession instrumentHandle, ViInt32 clockSource);
```

Purpose

Sets the source for the sampling clock.

Parameter List

instrumentHandle

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

clockSource

Variable Type ViInt32

Sets the source for the sampling clock signal.
Possible values are:

bu3430_CLOCK_INTERNAL 0 Internally generated clock (Default)
bu3430_CLOCK_EXTERNAL 1 External clock applied to FP connector

Return Value

Displays the return status of the function call. If the function was successful, it will return a status of VI_SUCCESS, otherwise it will return an error code. Passing the error code into the function "bu3430_error_message", will return a string describing the error.

VISA Errors:
See section 3.3 of the VPP 4.2.2 document for a complete list of the WIN Framework error codes and their values. Appendix B of VPP 4.2 contains descriptions and a breakdown of the error codes returned by each of the VISA functions.

BU3430 Driver Errors:
Errors returned from the BU3430 drivers will be between 0xBFFC0800 and 0xBFFC0FFF.

---

bu3430_setDAQmode

```
ViStatus bu3430_setDAQmode (ViSession instrumentHandle, ViInt16 startMode,
                            ViInt16 stopMode, ViBoolean breakonerror);
```

Purpose

Sets the operation mode for the data acquisition process.

Parameter List

instrumentHandle

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

startMode

Variable Type ViInt16

Selects the way how the Data Acquisition starts. Possible values are:

bu3430_DAQ_START_IMMEDIATE	0	Immediate Mode (default) - DAQ starts immediately after synchronisation completed.
bu3430_DAQ_START_TRIG	1	Triggered mode - DAQ starts when input trigger goes active (edge sensitive).

stopMode

Variable Type ViInt16

Selects the way how the Data Acquisition stops. Possible values are:

bu3430_DAQ_STOP_CONT	0	Continuous Mode - DAQ stops when specified number of samples is collected
bu3430_DAQ_STOP_TRIG	1	Triggered mode - DAQ stops when input trigger goes inactive

breakonerror

Variable Type ViBoolean

Specifies whether the Data Acquisition process should break if any error encountered.

Possible values are:

bu3430_OFF	0	(Default) DAQ will continue regardless on data error happened during DAQ
bu3430_ON	1	DAQ will abort immediately if any data error happened.

Return Value

Displays the return status of the function call. If the function was successful, it will return a status of VI_SUCCESS, otherwise it will return an error code. Passing the error code into the function "bu3430_error_message", will return a string describing the error.

VISA Errors:
See section 3.3 of the VPP 4.2.2 document for a complete list of the WIN Framework error codes and their values. Appendix B of VPP 4.2 contains descriptions and a breakdown of the error codes returned by each of the VISA functions.

BU3430 Driver Errors:
Errors returned from the BU3430 drivers will be between 0xBFFC0800 and 0xBFFC0FFF.

---

bu3430_setDecimFactor

```
ViStatus bu3430_setDecimFactor (ViSession instrumentHandle, ViInt32
decimFactor);
```

Purpose

Sets the decimation factor for the data acquisition.

Parameter List

instrumentHandle

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

decimFactor

Variable Type ViInt32

Sets the value for the decimation factor.
Possible values are from 0 to 10.
If the decimation factor equals N
the decimation will be N power of 2.
Default value is 0.

Return Value

Displays the return status of the function call. If the function was successful, it will return a status of VI_SUCCESS, otherwise it will return an error code. Passing the error code into the function "bu3430_error_message", will return a string describing the error.

VISA Errors:
See section 3.3 of the VPP 4.2.2 document for a complete list of the WIN Framework error codes and their values. Appendix B of VPP 4.2 contains descriptions and a breakdown of the error codes returned by each of the VISA functions.

BU3430 Driver Errors:
Errors returned from the BU3430 drivers will be between 0xBFFC0800 and 0xBFFC0FFF.

---

bu3430_setFEconf

```
ViStatus bu3430_setFEconf (ViSession instrumentHandle, ViInt16 terminationClk,
                           ViReal64 thresholdClk, ViInt16 terminationSynch,
                           ViReal64 thresholdSynch);
```

Purpose

Configures the Front-End circuit of the selected channel.

Parameter List

instrumentHandle

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

terminationClk

Variable Type ViInt16

Sets the termination of input signal for the External Clock signal. Posssible values are:

bu3430_CHX_TERM_1M 1 - (Default) External Clock signal terminated to ground with 1MOhm resistor.

bu3430_CHX_TERM_50 0 - External Clock signal terminated to ground with 50 Ohm resistor.

thresholdClk

Variable Type ViReal64

Specifies the threshold level for the External Clock signal. The range of the values is from -5.0V to +5.0V.

terminationSynch

Variable Type ViInt16

Sets the termination of input signal for the External Synch/Trigger signal. Possible values are:

bu3430_CHX_TERM_1M 1 - (Default) External Synch/Trigger signal enabled and terminated to ground with 1MOhm resistor.

bu3430_CHX_TERM_50 0 - External Synch/Trigger signal enabled and terminated to ground with 50 Ohm resistor.

thresholdSynch

Variable Type ViReal64

Specifies the threshold level for the External Synch/Trigger signal. The range of the values is from -5.0V to +5.0V.

Return Value

Displays the return status of the function call. If the function was successful, it will return a status of VI_SUCCESS, otherwise it will return an error code. Passing the error code into the function "bu3430_error_message", will return a string describing the error.

VISA Errors:
See section 3.3 of the VPP 4.2.2 document for a complete list of the WIN Framework error codes and their values. Appendix B of VPP 4.2 contains descriptions and a breakdown of the error codes returned by each of the VISA functions.

BU3430 Driver Errors:
Errors returned from the BU3430 drivers will be between 0xBFFC0800 and 0xBFFC0FFF.

---

bu3430_setITRI

```
ViStatus bu3430_setITRI (ViSession instrumentHandle, ViInt16 source);
```

Purpose

Sets the sources and polarity for the input trigger. It is possible to use more than one trigger source at the time.

Parameter List

instrumentHandle

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

source

Variable Type ViInt16

Specifies the source of the Input Trigger line. Possible values are:

bu3430_NONE	0x0000	disabled (default)
bu3430_ITRI_FP_HI	0x0002	Input Trigger will be taken from front panel, active level high
bu3430_ITRI_FP_LOW	0x0003	Input Trigger will be taken from front panel, active level low
bu3430_ITRI_MB	0x0004	Input Trigger will be taken from the Motherboard line

Motherboard input can be ORed together with FP input of selected polarity.

Return Value

Displays the return status of the function call. If the function was

successful, it will return a status of VI_SUCCESS, otherwise it will return an error code. Passing the error code into the function "bu3430_error_message", will return a string describing the error.

VISA Errors:
See section 3.3 of the VPP 4.2.2 document for a complete list of the WIN Framework error codes and their values. Appendix B of VPP 4.2 contains descriptions and a breakdown of the error codes returned by each of the VISA functions.

BU3430 Driver Errors:
Errors returned from the BU3430 drivers will be between 0xBFFC0800 and 0xBFFC0FFF.

---

bu3430_setNScans

```
ViStatus bu3430_setNScans (ViSession instrumentHandle, ViInt32 scans);
```

Purpose

Specifies the number of scans to be collected.

Parameter List

instrumentHandle

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

scans

Variable Type ViInt32

Specifies the number of scans to be collected.
Possible values are even numbers from 0 to 0xFFFE.

Return Value

Displays the return status of the function call. If the function was successful, it will return a status of VI_SUCCESS, otherwise it will return an error code. Passing the error code into the function "bu3430_error_message", will return a string describing the error.

VISA Errors:
See section 3.3 of the VPP 4.2.2 document for a complete list of the WIN Framework error codes and their values. Appendix B of VPP 4.2 contains descriptions and a breakdown of the error codes returned by each of the VISA functions.

BU3430 Driver Errors:
Errors returned from the BU3430 drivers will be between 0xBFFC0800 and 0xBFFC0FFF.

---

bu3430_setOTRIsrc


```
ViStatus bu3430_setOTRIsrc (ViSession instrumentHandle, ViInt16 source,
                            ViInt16 outMode);
```

Purpose

Sets the sources for the output trigger. It is possible to use more than one trigger source at the time.

Parameter List

instrumentHandle

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

source

Variable Type ViInt16

Specifies the source of the Output Trigger line. Possible values are:

bu3430_NONE	0x0000	disabled
bu3430_OTRI_DAQ_ON	0x0020	Trigger remains active as long as DAQ is in progress
bu3430_OTRI_DAQ_END	0x0040	Trigger goes active on the DAQ completion
bu3430_OTRI_ERR	0x2000	Trigger goes active if error happened during DAQ
------		
bu3430_OTRI_FIFO_EF	0x0008	Trigger goes active if FIFO is empty
bu3430_OTRI_FIFO_AE	0x0009	Trigger goes active if FIFO is almost empty
bu3430_OTRI_FIFO_HF	0x000A	Trigger goes active if FIFO is half full
bu3430_OTRI_FIFO_AF	0x000B	Trigger goes active if FIFO is almost full
bu3430_OTRI_FIFO_FF	0x000C	Trigger goes active if FIFO is full

Any of first four sources can be ORed together or with one of last five sources (FIFO flags)

outMode

Variable Type ViInt16

Specifies the type of output signal of the trigger.
Possible values are:

bu3430_OTRI_OUT_PULSE	0 output mode set to pulse
bu3430_OTRI_OUT_LEVEL	1 output mode set to level

Return Value

Displays the return status of the function call. If the function was successful, it will return a status of VI_SUCCESS, otherwise it will return an error code. Passing the error code into the function "bu3430_error_message", will return a string describing the error.

VISA Errors:
See section 3.3 of the VPP 4.2.2 document for a complete list of the

WIN Framework error codes and their values. Appendix B of VPP 4.2 contains descriptions and a breakdown of the error codes returned by each of the VISA functions.

BU3430 Driver Errors:
Errors returned from the BU3430 drivers will be between 0xBFFC0800 and 0xBFFC0FFF.

---

bu3430_setSynchSrc

```
ViStatus bu3430_setSynchSrc (ViSession instrumentHandle, ViInt32 synchSource);
```

Purpose

Sets the source for the synchronisation pulse.

Parameter List

instrumentHandle

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

synchSource

Variable Type ViInt32

Sets the source for the synchronisation pulse.
Possible values are:

bu3430_SYNCH_INTERNAL 0 Internally generated pulse (Default)
bu3430_SYNCH_EXTERNAL 1 External pulse applied to FP connector

Return Value

Displays the return status of the function call. If the function was successful, it will return a status of VI_SUCCESS, otherwise it will return an error code. Passing the error code into the function "bu3430_error_message", will return a string describing the error.

VISA Errors:
See section 3.3 of the VPP 4.2.2 document for a complete list of the WIN Framework error codes and their values. Appendix B of VPP 4.2 contains descriptions and a breakdown of the error codes returned by each of the VISA functions.

BU3430 Driver Errors:
Errors returned from the BU3430 drivers will be between 0xBFFC0800 and 0xBFFC0FFF.

---

bu3430_startAcqisition

```
ViStatus bu3430_startAcqisition (ViSession instrumentHandle,
                                 ViInt16 channelMask, ViInt32
decimationFactor,
```



```
                                        ViInt16 triggerSource, ViInt32 synchSource,
                                        ViInt32 clockSource, ViReal64
*actualScanRate);
```

Purpose

Starts the Data Acquisition process on the Function Card.

Parameter List

instrumentHandle

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

channelMask

Variable Type ViInt16

Selects the channels from which the data will be taken.

bit 0 corresponds to the Channel 1
...
bit 3 corresponds to the Channel 4

"1" written in the appropriate bit means that the channel will be included in the data acquisition.

decimationFactor

Variable Type ViInt32

Sets the value for the decimation factor.
Possible values are from 0 to 10.
If the decimation factor equals N
the decimation will be N power of 2.
Default value is 0.

triggerSource

Variable Type ViInt16

Specifies the source of the Input Trigger line. Possible values are:

bu3430_NONE	0x0000	disabled (default) The data acquisition will start immediately regardless on the trigger line state.
bu3430_ITRI_FP_HI	0x0002	Input Trigger will be taken from front panel, active level high
bu3430_ITRI_FP_LOW	0x0003	Input Trigger will be taken from front panel, active level low
bu3430_ITRI_MB	0x0004	Input Trigger will be taken from the Motherboard line

Motherboard input can be ORed together with FP input of selected polarity. If the input trigger is enabled, it works in the gate mode, i.e. Data Acquisition will start when trigger goes into active state and will stop when trigger goes back into inactive state


```
    synchSource

        Variable Type       ViInt32

        Sets the source for the synchronisation pulse.
        Possible values are:

        bu3430_SYNCH_INTERNAL  0  Internally generated pulse (Default)
        bu3430_SYNCH_EXTERNAL  1  External pulse applied to FP connector

    clockSource

        Variable Type       ViInt32

        Sets the source for the clock signal.
        Possible values are:

        bu3430_CLOCK_INTERNAL  0 Internally generated signal (Default)
        bu3430_CLOCK_EXTERNAL  1 External signal applied to FP connector

    actualScanRate

        Variable Type       ViReal64 (passed by reference)

        The actual scan rate (scans per second) depends on the sampling clock
        frequency and the decimation factor selected.

Return Value

        Displays the return status of the function call.  If the function was
        successful, it will return a status of VI_SUCCESS, otherwise it will
        return an error code.  Passing the error code into the function
        "bu3430_error_message", will return a string describing the error.

        VISA Errors:
        See section 3.3 of the VPP 4.2.2 document for a complete list of the
        WIN Framework error codes and their values.  Appendix B of VPP 4.2
        contains descriptions and a breakdown of the error codes returned by
        each of the VISA functions.

        BU3430 Driver Errors:
        Errors returned from the BU3430 drivers will be between 0xBFFC0800
        and 0xBFFC0FFF.

------------------------------------------------------------------------------

bu3430_startDAQ

    ViStatus bu3430_startDAQ (ViSession instrumentHandle);

Purpose

    Starts the Data Acquisition process on the Function Card.

Parameter List

    instrumentHandle

        Variable Type       ViSession

        The Instrument Handle is used to identify the unique session or
        communication channel between the driver and the instrument.

        If more than one instrument of the same model type is used, this
```


Handle will be used to differentiate between them.

Return Value

Displays the return status of the function call. If the function was successful, it will return a status of VI_SUCCESS, otherwise it will return an error code. Passing the error code into the function "bu3430_error_message", will return a string describing the error.

VISA Errors:
See section 3.3 of the VPP 4.2.2 document for a complete list of the WIN Framework error codes and their values. Appendix B of VPP 4.2 contains descriptions and a breakdown of the error codes returned by each of the VISA functions.

BU3430 Driver Errors:
Errors returned from the BU3430 drivers will be between 0xBFFC0800 and 0xBFFC0FFF.

---

bu3430_stopAcquisition

```
ViStatus bu3430_stopAcquisition (ViSession instrumentHandle);
```

Purpose

Stops any DAQ activity and flushes the buffers.

Parameter List

instrumentHandle

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

Return Value

Displays the return status of the function call. If the function was successful, it will return a status of VI_SUCCESS, otherwise it will return an error code. Passing the error code into the function "bu3430_error_message", will return a string describing the error.

VISA Errors:
See section 3.3 of the VPP 4.2.2 document for a complete list of the WIN Framework error codes and their values. Appendix B of VPP 4.2 contains descriptions and a breakdown of the error codes returned by each of the VISA functions.

BU3430 Driver Errors:
Errors returned from the BU3430 drivers will be between 0xBFFC0800 and 0xBFFC0FFF.

---

bu3430_stopDAQ

```
ViStatus bu3430_stopDAQ (ViSession instrumentHandle);
```


Purpose

Puts the DAQ FSM into the IDLE state. Aborts from any DAQ activity.

Parameter List

instrumentHandle

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

Return Value

Displays the return status of the function call. If the function was successful, it will return a status of VI_SUCCESS, otherwise it will return an error code. Passing the error code into the function "bu3430_error_message", will return a string describing the error.

VISA Errors:
See section 3.3 of the VPP 4.2.2 document for a complete list of the WIN Framework error codes and their values. Appendix B of VPP 4.2 contains descriptions and a breakdown of the error codes returned by each of the VISA functions.

BU3430 Driver Errors:
Errors returned from the BU3430 drivers will be between 0xBFFC0800 and 0xBFFC0FFF.

---

bu3430_waitDAQ

```
ViStatus bu3430_waitDAQ (ViSession instrumentHandle, ViInt32 timeout,
                         ViInt16 *state, ViInt16 *err);
```

Purpose

Waits until the Data Acquisition process is terminated or timeout happened and returns the state of the DAQ.

Parameter List

instrumentHandle

Variable Type ViSession

The Instrument Handle is used to identify the unique session or communication channel between the driver and the instrument.

If more than one instrument of the same model type is used, this Handle will be used to differentiate between them.

timeout

Variable Type ViInt32

Specifies the timeout in milliseconds. If the timeout is 0 the function will return current state of DAQ process immediately. If the

timeout value is not 0, the function will wait specified number of milliseconds or until the DAQ process is terminated (if it was started before).

state

Variable Type ViInt16 (passed by reference)

Returns the status of the Data Acquisition process. Possible values are:

bu3430_DAQ_IDLE	0	IDLE state
bu3430_DAQ_ARMED	1	ARMED state - DAQ is armed and waiting for the external synchronization pulse
bu3430_DAQ_SYNCH	2	SYNCH state - DAQ is performing the synchronization cycle
bu3430_DAQ_READY	3	READY state - DAQ ready to start, waiting for the Input Trigger
bu3430_DAQ_RUNNING	4	DAQ is in the progress
bu3430_DAQ_STOPED	5	DAQ is completed or aborted

err

Variable Type ViInt16 (passed by reference)

Returns the error code if any happened during the Data Acquisition process. This value might be bitwise-OR of the following values:

	0	No error
bu3430_OW_ERROR	1	Overwrite error
bu3430_OF_ERROR	2	Overflow error
bu3430_SCAN_ERROR	4	Scan error

Return Value

Displays the return status of the function call. If the function was successful, it will return a status of VI_SUCCESS, otherwise it will return an error code. Passing the error code into the function "bu3430_error_message", will return a string describing the error.

VISA Errors:
See section 3.3 of the VPP 4.2.2 document for a complete list of the WIN Framework error codes and their values. Appendix B of VPP 4.2 contains descriptions and a breakdown of the error codes returned by each of the VISA functions.

BU3430 Driver Errors:
Errors returned from the BU3430 drivers will be between 0xBFFC0800 and 0xBFFC0FFF.

