

IOtech, Inc.

25971 Cannon Road

Cleveland, OH 44146-1833

Phone: (440) 439-4091

Fax: (440) 439-4093

E-mail (sales): sales@iotech.com

E-mail (post-sales): productsupport@iotech.com

Internet: www.iotech.com

ADAC-LVi User's Manual

Data Acquisition VIs for LabVIEW™

p/n 1107-0901 Rev 1.0

ADAC-LVi, V3.10

LabVIEW™, NI-DAQ™ and National Instruments™ are trademarks of National Instruments Corp.

Ch 1 - Introduction	1
Using the ADAC VIs with LabVIEW.....	1
ADAC-LVi System Requirements	2
Getting Started	3
Installation	3
Hardware Installation.....	7
Configuring ADAC-LVi to Recognize your Hardware.....	7
Verification	10
Existing LabVIEW Applications – Special Note	10
Ch 2 - ADAC LabVIEW Data Acquisition VIs.....	12
The Analog Input VIs	13
Easy Analog Input VIs.....	14
Analog Input Examples	20
Intermediate Analog Input VIs.....	23
Utility Analog Input VIs	32
Advanced Analog Input VIs	49
The Analog Output VIs.....	51
Analog Output Examples	56
The Digital I/O VIs.....	57
Digital I/O Examples	77
The Miscellaneous VIs.....	78
The Signal Conditioning VIs.....	82
The Advanced System Function VIs - Program Flow	86
The Advanced System Function VIs	98
ADAC Environment Config	99
ADAC Board Func Library	102
ADAC Buffer Config Library.....	106
ADAC Burst Config Library	119
ADAC Channel Config Library.....	124
ADAC Clock Config Library.....	128
ADAC Data Convert Library.....	133
ADAC Analog Ouput Library.....	136
ADAC Digital I/O Library.....	137
ADAC DMA Config Library.....	151
ADAC Filter Config Library	153
ADAC Gate Config Library.....	156

ADAC Input Config Library.....	161
ADAC IRQ (Interrupt) Config Library	163
ADAC Device Function Library.....	166
ADAC Panel Config Library.....	176
ADAC Range Config Library	181
ADAC CJ (Cold Junction) Config Library.....	187
ADAC Trigger Config Library	191
Ch 3 - Supported Boards & Subsystems Listed by VI Functions..	198
Easy Analog Input VIs.....	198
Intermediate Analog Input VIs.....	198
Analog Input Utility VIs	199
Advanced Analog Input VIs	199
Easy Analog Output VIs.....	199
Intermediate Analog Output VIs	199
Easy Digital I/O VIs.....	200
Intermediate Digital I/O VIs	200
Advanced Digital I/O VIs	200
Miscellaneous VIs	201
Signal Conditioning VIs	201
Advanced Function Environment VIs:	201
Advanced Function Board VIs:	201
Advanced Function Device Configuration VIs:	202
Advanced Function Analog Output VIs:	204
Advanced Function Digital I/O VIs:	205
Advanced Function Triggering VIs:	206
Advanced Function Clocking VIs:	208
Advanced Function Gating VIs:	210
Advanced Function Burst VIs:	210
Advanced Function Channels VIs:	211
Advanced Function Expansion Panels VIs:	212
Advanced Function Thermocouple Configuration VIs:	213
Advanced Function DMA Configuration VIs:	213
Advanced Function IRQ Configuration VIs:	213
Advanced Function Input Configuration VIs:	214
Advanced Function Data Codes VIs:	214
Advanced Function Filters VIs:	215

Advanced Function Buffers VIs:	216
Data Conversion:	218
Ch 4 - ADAC-LVi Error Codes.....	220

1

Introduction

The ADAC-LVi software package consists of LabVIEW™ Data Acquisition VIs designed to interface to the ADAC line of DAQ boards. These VIs provide the LabVIEW user with complete access to all of the board functions (A/D, D/A, DIO, etc.) on ADAC type boards within the Windows 95/98/ME/2000/XP or NT4.0 environment.

We use the ADLIB libraries to access all data acquisition I/O within LabVIEW in the same manner that NI uses NI-DAQ™. As with NI-DAQ, the ADLIB library is transparent to the LabVIEW user. This provides ADAC-LVi with a seamless integration into National Instruments™ LabVIEW product line.

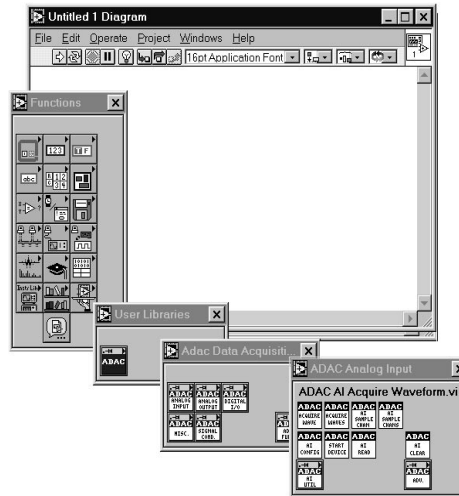
It is important to note that we do not “clone” other manufacturer’s boards; the ADAC designs are unique, however, the user will find many similarities (number of I/O, Gain, DMA, etc) when comparing ADAC DAQ boards to those from NI. See our web-site for detailed information on our complete line of data acquisition boards.

Using the ADAC VIs with LabVIEW

To begin using the ADAC VIs, simply run the ADAC LVi installation program; the following folders will be created:

...\LabView\ADAC Examples	;DAQ Board Demos
...\LabView\User.lib\AdacDaq	;User VI's
...\LabView\ADAC-LVi	;Board Driver Files
...\LabView\User.lib\AdacDaq\Advanced	;Driver Level VI's

When you run LabVIEW, the ADAC VIs will appear under the User Library Icon located in the Functions Palette as shown in the following diagram. The LabVIEW programmer selects and uses the ADAC Data Acquisition VIs just as they would those from NI.



User Library Palette Showing ADAC Vis

ADAC-LVi System Requirements

To Install and use the ADAC-LVi drivers with LabVIEW, you will need a PC with LabVIEW fully installed per National Instruments' specifications, and the following additional ADAC-LVi requirements:

- One 3.5" 1.44MB floppy disk drive for installation for ADAC-Lvi-NT / ADAC-Lvi-95 or one CD driver for ADAC-Lvi-WDM.
- 10MB available hard disk space.
- Microsoft Window 95/98/ME/200/XP or Windows NT 4.0 or higher.
- National Instruments' LabVIEW Version 5.0 or higher,
- An ADAC data acquisition board.

Getting Started

The steps for using and running the ADAC VIs are as follows:

- 1) Install LabVIEW 5.0 or higher on PC
- 2) Run the ADAC LVi installation program
- 3) Install ADAC DAQ board(s) in PC
- 4) Configure the ADAC-LVi.con file
- 5) Configure the ADAC-LVi.ini file
- 6) Create Application in LabVIEW
- 7) Launch LabVIEW executable

Installation

After LabVIEW (5.0 or higher) has been successfully installed on your computer, exit LabVIEW and insert the ADAC-LVi disk # 1 or CD and select "Setup.exe" from the ADAC-Lvi directory if installing from a CD. The Setup will step you through the complete installation process. It is important to note that ADAC-LVi **must** be installed in your LabVIEW or LabView 6 directory to function properly. The Setup program attempts to locate your ... \LabVIEW (6) installation directory during installation. If Setup does not find your LabVIEW (6) directory, use the browse button provided by Setup to locate the directory manually.

After you have successfully installed the ADAC-LVi the following folders and files of interest will have been added to your hard drive:

... \LabView (6) \ADAC Examples	;ADAC DAQ Board Demo Programs.
... \LabView (6) \ADAC-LVi	;Board driver files and configuration text files.
... \LabView (6) \User.lib \AdacDaq	;User VI's available from the LabVIEW Functions toolbox.
... \LabView (6) \User.lib \AdacDaq \Advanced	;Driver level VI's available from the LabVIEW Functions toolbox.
C:\Windows\ADAC-LVi.con	;ADAC-LVi Board configuration File.
C:\Windows\ADAC-LVi.ini	; ADAC-LVi Board subsystem default initialization setup.

C:\Windows\ADAC-LVi.log ;Un-install log File for ADAC-Lvi-95 or ADAC-Lvi-NT only.

In addition to the ADAC VIs, the installation will add the “low level” software that serves as an interface between the ADAC VIs and the DAQ boards (similar to the NI DAQ software from NI).

Check the our web site for possible updates to the ADAC-LVi.

After setup has completed:

ADAC PCI Bus boards with Windows 98/ME/2000/XP:

For **ADAC-LVi WDM** installations, the following files have been added to your Windows directory:

Windows\system32\drivers	Low Level board system (.SYS) drivers
Windows\adlcore.dll	ADAC-LVi interface DLL
Windows\adlgrm.dll	ADAC-LVi global resource manager

After ADAC-LVi installation, the ADAC VIs will automatically appear under the User Library Icon the next time you start LabVIEW.

ADAC PCI Bus boards with Windows NT 4.0:

For **ADAC-LVi Windows NT** PCI installations, the following files have been added to your WINNT directory:

Win NT \system32\drivers	Low Level board system (.SYS) drivers
win NT \system32\adlcore.dll	ADAC-LVi interface DLL
win NT \system32\adlgrm.dll	ADAC-LVi global resource manager

When using ADAC-LVi with ADAC PCI boards and Win NT 4.0, you must do the following after installation:

1. Manually copy the “aPcixx.sys” file from the ADAC CD’s NT4 directory to your \WinNT\system32\drivers directory.
2. From the ADAC CD’s NT4 directory, run the “aPci55xx.reg” file by double clicking on it, this will update your registry to look for ADAC PCI boards.

3. Reboot your PC to load the ADAC drivers.

ADAC ISA Bus boards with Windows 95/98:

For **ADAC-LVi Windows 95** installations, the following files have been added to your WIN95 or WIN98 directory:

win95/98 \system\adacdm32.VXD	DMA and Interrupt support VXD services
win95/98 \adlcore.dll	ADAC-LVi interface DLL
win95/98 \adlgrm.dll	ADAC-LVi global resource manager

The following changes have been made to your system.ini file in the [386Enh] section:

```
device = *vdmad (If not already present.)
device = adacdm32.vxd
ADACDMABUFFERSIZE=256
```

The ADACDMABUFFERSIZE entry specifies the amount of DMA accessible memory available to the ADAC-LVi drivers. The ADACDMABUFFERSIZE entry is specified in kilobytes.

Once the system.ini file has been updated, restart Windows to load the ADAC virtual DMA device driver.

ADAC ISA Bus boards with Windows NT 4.0:

For **ADAC-LVi Windows NT** ISA installations, the following files have been added to your WINNT directory:

Win NT \system32\drivers	Low Level board system (.SYS) drivers
win NT \system32\adlcore.dll	ADAC-LVi interface DLL
win NT \system32\adlgrm.dll	ADAC-LVi global resource manager

Once the installation has completed, use Explorer to edit the newly installed boards registry installation file (*boardname.reg*) located in the ... \LabView\ADAC-LVi\BoardName directory.

Edit the following fields and save the file.

```
IOPortRange 0_low = dword:00000120 (Board Address)
DMA0 = dword:00000005 (DMA Channel)
```

IRQ0 = dword:00000005 (IRQ Level)

Once the *boardname.reg* has been completed, double clicking on the file's icon will install the boards registry settings, and the computer must be restarted to load the proper settings.

After ADAC-LVi installation, the ADAC VIs will automatically appear under the User Library Icon the next time you start LabVIEW.

Hardware Installation

Before attempting to use your ADAC hardware with LabVIEW, it is highly recommended you know and understand your hardware capabilities and configuration. To most effectively use ADAC hardware with LabVIEW all legacy devices capable of supporting a **DMA channel and Interrupt level** should be setup in hardware. Although polling in LabVIEW is fully supported, the polling method can only support limited acquisition operations. ADAC PCI devices automatically acquire an interrupt level on system start-up and provide their own on-board DMA engines.

Refer to your ADAC hardware manual for instructions on proper installation and configuration of your ADAC DAQ board.

Configuring ADAC-LVi to Recognize your Hardware

ADAC-Lvi WDM provides, a configuration utility program ADACCONFIG.EXE to setup and configure your ADAC boards within the PC. The utility provides all the necessary information to create both the Configuration (.CON) and Initialization (.INI) files. The following sections provide the necessary detailed information on the .CON and .INI files for manual configuration of these files and are not required reading when using the ADACCONFIG utility. The ADACCONFIG.EXE is located in your main ADAC\AdacConfig directory or from the programs menu group chosen when ADAC-LVi was installed.

An example ADAC-Lvi.INI file is provided in each \LabView (6)\Adac-Lvi\Board Type directory. For older legacy devices the DMA channel and Interrupt level settings should be changed to match the actual hardware jumper or software settings. All other setting must be left at their factory defaults in order for the ADAC example VI's to function properly.

For ADAC-Lvi-WDM both ADAC-Lvi.INI and Adac-Lvi.CON file examples are provided in each \LabView (6)\Adac-Lvi\Board Type directory. All setting must be left at their defaults in order for the ADAC example VI's to function properly. For single board setups simply copy the .CON and .INI files to your Windows directory. For multiple board first copy the appropriate .CON and .INI file to your Windows directory then use the AdacConfig.exe utility modify ADAC-Lvi.CON file in your Windows directory. The default .INI settings created by the utility are appropriate to run with LabView and must not be changed.

The ADAC-LVi.CON File

The ADAC-LVi.con board configuration file contains settings that apply to all ADAC boards currently installed in your system. There is only one ADAC-LVi.CON file per system and it contains “static” system-wide configuration information about all the boards and drivers installed. This file is read during the allocation of an ADAC device through a LabVIEW application program. When the first device is allocated by an ADAC VI, the ADAC-LVi.CON file is loaded.

Contents of the Configuration file includes:

- The user defined environment specific string name.
- The path and filename of the ADAC-LVi.INI located in your Windows top level directory.
- An assigned DAQ board designator number for each installed board (this is the BoardID number use by ADAC VIs).
- The user defined board string name.
- The actual board model number per the capabilities file.
- Path and filename of the capabilities file for each DAQ board located in your ...\\LabVIEW (6)\\ADAC-LVi \\BoardName directory. From the capabilities file, ADAC-LVi knows the type of the board, and all its available features.
- Path to directory containing board driver, for each unique type of board located in your ...\\LabVIEW (6)\\ADAC-LVi directory.
- The name of the ADAC-LVi system VXD or SYS driver.
- I/O or memory base address of each board.

To configure the ADAC-LVi.CON file open it up in NOTEPAD.EXE. The file consists of two active sections called [Environment] and [Board0] and a third inactive section [Board1] which has been commented out with semi-colons. Text within the file describes changing the settings and adding additional board(s).

The Capabilities File

A Capabilities file contains all the possible programmable options for each ADAC function call for a particular DAQ board. There is one Capabilities file per unique DAQ board, ADAC-LVi uses this information to validate all application program VI call parameters, and when initializing a logical device from the initialization file described below. The Capabilities file has a .CAP filename extension. CAP files are for internal driver use only and should not be changed by the user.

Contents of the Capabilities File includes:

- Device capabilities: The features and options the DAQ board supports
- How the board's subsections are divided up into logical hardware devices (such as ADC0, DAC0, DAC1, DOT0, DIN3, CTR0, etc.)

The ADAC-LVi.INI File

The initialization file specifies how the programmable features of boards and logical devices are to be initialized during the ADAC-LVi device allocation VI call. There is only one .INI file per LabVIEW program application, and the name and location is specified in the ADAC-LVi.CON file.

The initialization file reduces the amount of user programming required by initially setting the ADAC-LVi logical device subsystem database (LSD) to the desired user application defaults. This feature allows only the bare minimum amount of ADAC-LVi calls to start a logical device.

Contents of the INI file includes:

- One or more named sections, each corresponding to a Logical Device, including information for data transfer method, number of channels enabled, channel gain settings, sampling mode, sampling rate, notification method, and logical device option information.

If an .INI parameter is not specified, but the capabilities file indicated support for the option, the parameter will default to the first option for that entry in the capabilities file.

To configure the ADAC-LVi.INI file open it up in NOTEPAD.EXE. The file consists of one active section called [LogicalDevice0] and an inactive section [LogicalDevice1], which has been commented out with semi-colons. Text within the file describes changing the settings and adding additional Logical Devices sections.

Verification

Once the hardware and software have been configured properly, select an ADAC example program from the ...\\LabVIEW\\ADAC Examples directory that closely matches your ultimate acquisition mode of your application within LabVIEW. Running the example will verify your hardware jumpers, ADAC-LVi.INI and ADAC-LVi.CON software settings.

If any error codes are returned, a comprehensive list of numeric error codes is located at the end of the manual. Most errors are simple miss configurations within the ADAC-LVi.INI or ADAC-LVi.CON files, and the error code will lead you to it.

Existing LabVIEW Applications – Special Note

If you are switching an existing LabVIEW application over to ADAC DAQ boards, you must be certain that the ADAC card(s) supports all of the features that your LabVIEW software program uses on the NI card (# of I/O, DMA, Interrupts, Triggering Methods, etc.).

Once you have confirmed that the ADAC board contains the features used by your program, you must then remove all of the NI VIs and replace them with the corresponding VI from ADAC. ADAC made great effort to keep our VIs as close as possible to those from NI, however, you will notice some differences in the connections to certain VIs. This is because ADAC type boards are not NI clones, and some of the features are used differently. Consult the VI section of this manual for complete details.

2

The ADAC LabVIEW Data Acquisition VIs

This chapter contains the basic information about ADAC data acquisition (DAQ) VIs for use with LabVIEW. The ADAC LVI is a library of VIs to use with ADAC type DAQ hardware for developing applications with LabVIEW.

Access to the ADAC DAQ VIs in the **Functions** palette in the block diagram view in LabVIEW. To access the ADAC Data Acquisition palette, choose **Functions >> User Libraries >> ADAC**.

ADAC data acquisition VIs are grouped as follows:

- Analog Input VIs
- Analog Output VIs
- Digital I/O VIs
- Miscellaneous VIs
- Signal Conditioning VIs
- Advanced System Function VIs

The Analog Input VIs

The Analog Input VIs are used to perform analog input functions.

To access the ADAC Analog Input VIs, choose **Functions >> User Libraries >> ADAC >> Analog Input**.

ADAC has four types of analog input VIs:

- **Easy Analog Input VIs**
Located in the top row of the Analog Input palette, the Analog Input Easy VIs are created using the ADAC Analog Input Intermediate and the ADAC Analog Input Utilities VIs. The Easy VIs are used to perform simple analog input functions; they are the easiest VIs to use, however, they provide the least amount of user flexibility.
- **Intermediate Analog Input VIs**
Located in the second row of the Analog Input palette, the Analog Input Intermediate VIs were built using the ADAC Analog Input Advanced VIs and the ADAC Analog Input Utilities VIs. The Intermediate VIs group the features of the Analog Input Advance VIs to provide the user with an easier programming interface than the Advanced VIs.
- **Utilities Analog Input VIs**
Accessed by clicking on the “AI UTIL” icon in the Analog Input palette, the Analog Input Utilities VIs were built using the ADAC Analog Input Advanced VIs. The Utility VIs group the common features of the Analog Input Advance VIs to provide the user with an easier programming interface than the Advanced VIs.
- **Advanced Analog Input VIs**
Accessed by clicking on the “AI ADV” icon in the Analog Input palette, the Analog Input Advanced VIs were built using the ADAC Analog Input Advanced Function VIs. The Advanced VIs are used to build the Intermediate VIs functions; they are the most complicated to use, however, the Advanced VIs provide the greatest amount of user programming flexibility.

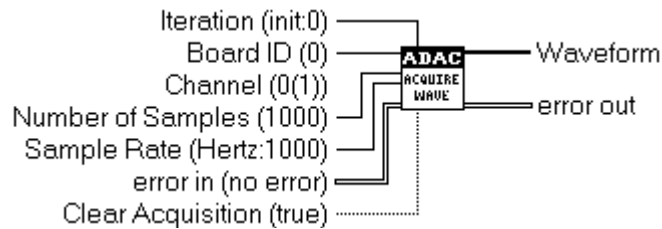
Easy Analog Input VIs

This section describes the Easy Analog Input VIs that are used to perform simple analog input functions. Easy VIs can be used by themselves, or as sub VIs in a user application.

To access the ADAC Analog Input Easy VIs, choose **Functions >> User Libraries >> ADAC >> Analog Input**. The Easy VIs are listed on the top line of the analog input palette:

- ADAC AI Acquire Wave Form
- ADAC AI Acquire Wave Forms
- ADAC AI Sample Channel
- ADAC AI Sample Channels

ADAC AI Acquire Wave Form

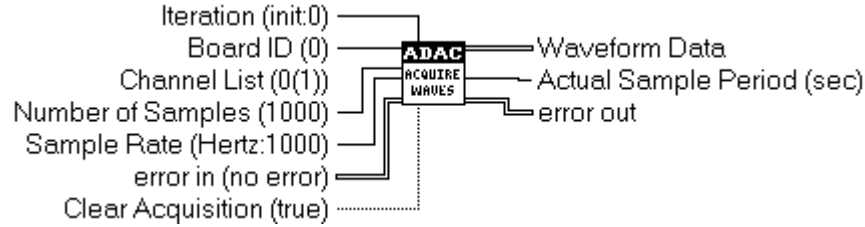


The ADAC AI Acquire Waveform VI acquires the specified number of samples at the specified sample rate and returns all the data acquired in scaled data units (volts) in a one dimensional array. This VI typically acquires voltage readings on a single channel only, but may be used to collect multiple channels into its 1 dimensional array. See the ADAC AI Acquire Waveforms VI for multi channel acquisition with a 2 dimensional array.

ADAC AI Acquire Wave Form (con't.)

- I32** **Board ID (0)** Specifies the ADAC-LVi board ID number to be used in this acquisition. The ID numbers are defined in the configuration file ADAC-LVi.con located in the Windows main directory.
- abc** **Channel (0(1))** Channel is a string that specifies the analog input channel number to acquire data from. Channel is typically set to acquire only a single channel of data but may be used to collect multiple channels into its 1 dimensional array. See the ADAC AI Acquire Waveforms VI for multi channel acquisition with a two dimensional array.
- For boards that support input gain the channel list also specifies a separate gain value for each channel. Gains are specified in parentheses after the channel number. For example, "0(1),1(8),2(4)" is interpreted as channel 0 at a gain of 1, channel 1 at a gain of 8 and channel 2 at a gain of 4.
- U32** **Number of Samples (1000)** Number of samples is the total number of data samples the VI acquires before the acquisition completes. The default value is 1000 samples.
- DBL** **Sample Rate (Hertz:1000)** Sample Rate is the total number of samples per second the VI acquires from the specified channel(s). The default value is 1000.00 samples per second.
- I32** **Iteration (init:0)** Iteration controls when the ADAC-LVi device subsystem allocation and device initialization is performed. If iteration is 0, the VI allocates the device and initializes it with the hardware specifications connected to the VI. The VI then starts the acquisition and reads the data. If iteration is greater than zero, then VI assumes that the current configuration is valid and starts the acquisition, then reads the data.
- TF** **Clear Acquisition (true)** Clear acquisition determines whether the VI releases the device after reading the specified number of samples. The VI should pass a TRUE to this parameter when reading the last set of samples for a given acquisition. The default is TRUE, which means that the VI reads data only once if this input remains unwired. You typically wire this input to the terminating condition of a loop, so that when the loop finishes the device is released.
- ERR** **error in (no error)** error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
- [SGL]** **Waveform** Waveform is a one-dimensional array containing the scaled analog input data for the specified channels in volts.
- ERR** **error out** error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC AI Acquire Wave Forms



The ADAC AI Acquire Waveforms VI acquires the specified number of samples at the specified sample rate and returns all the data acquired in scaled data units (volts) in a two dimensional array. This VI typically acquires voltage readings on multiple channels, but may be used to collect single channel points into its 2 dimensional array.

I32 **Board ID (0)** Specifies the ADAC-LVi board ID number to be used in this acquisition. The ID numbers are defined in the configuration file ADAC-LVi.con located in the Windows main directory.

abc **Channel List (0(1))** Channel List is a string that specifies the analog input channel numbers to acquire data from. Channel List is typically set to acquire multiple channels, but may be used to collect single channels into its two dimensional array.

For boards that support input gain the channel list also specifies a separate gain value for each channel. Gains are specified in parentheses after the channel number. For example, "0(1),1(8),2(4)" is interpreted as channel 0 at a gain of 1, channel 1 at a gain of 8 and channel 2 at a gain of 4.

U32 **Number of Samples (1000)** Number of samples is the total number of data samples the VI acquires before the acquisition completes. The default value is 1000 samples.

I32 **Iteration (init:0)** Iteration controls when the ADAC-LVi device subsystem allocation and device initialization is performed. If iteration is 0, the VI allocates the device and initializes it with the hardware specifications connected to the VI. The VI then starts the acquisition and reads the data. If iteration is greater than zero, then VI assumes that the current configuration is valid and starts the acquisition, then reads the data.

DBL **Sample Rate (Hertz:1000)** Sample Rate is the total number of samples per second the VI acquires from the specified channel(s). The default value is 1000.00 samples per second.

ADAC AI Acquire Wave Form (con't.)

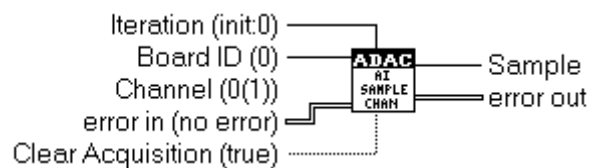
[TF] **Clear Acquisition (true)** Clear acquisition determines whether the VI releases the device after reading the specified number of samples. The VI should pass a TRUE to this parameter when reading the last set of samples for a given acquisition. The default is TRUE, which means that the VI reads data only once if this input remains unwired. You typically wire this input to the terminating condition of a loop, so that when the loop finishes the device is released.

[SGL] **error in (no error)** error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

[SGL] **Waveform Data** Waveform Data is a two-dimensional array containing analog input data in volts. The data appears in columns, where each column contains the data for a single channel. The second (bottom) dimension selects the channel. The first (top) dimension selects a single data point for that channel.

[DBL] **Actual Sample Period (sec)** Actual Sample Period is the actual interval between samples in seconds, which is the inverse of the actual sample rate in hertz. The actual sample period can differ from the requested sample rate, depending on the clocking capabilities of your selected device.

[SGL] **error out** error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC AI Sample Channel

The ADAC AI Sample Channel VI performs an immediate, **untimed** acquisition on a single channel and returns the data acquired in scaled data units (volts).

[I32] **Board ID (0)** Specifies the ADAC-LVi board ID number to be used in this acquisition. The ID numbers are defined in the configuration file ADAC-LVi.con located in the Windows main directory.

ADAC AI Sample Channel (con't.)



Channel (0 (1)) Channel is a string that specifies the analog input channel number to acquire data from.

For boards that support input gain the channel string also specifies a separate. Gains are specified in parentheses after the channel number. For example, "0(1)" is interpreted as channel 0 at a gain of 1.



Iteration (init:0) Iteration controls when the ADAC-LVi device subsystem allocation and device initialization is performed. If iteration is 0, the VI allocates the device and initializes it with the hardware specifications connected to the VI. The VI then starts the acquisition and reads the data. If iteration is greater than zero, then VI assumes that the current configuration is valid and starts the acquisition, then reads the data.



Clear Acquisition (true) Clear acquisition determines whether the VI releases the device after reading the specified number of samples. The VI should pass a TRUE to this parameter when reading the last set of samples for a given acquisition. The default is TRUE, which means that the VI reads data only once if this input remains unwired. You typically wire this input to the terminating condition of a loop, so that when the loop finishes the device is released.



error in (no error) error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

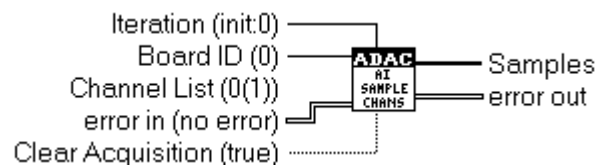


Sample Sample contains the acquired channel data in volts.



error out error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC AI Sample Channels



The ADAC AI Sample Channels VI performs an immediate, **untimed** acquisition on the specified channel(s) and returns all the data acquired in scaled data units (volts) in a one dimensional array.

ADAC AI Sample Channels (con't)

[I32] **Board ID (0)** Specifies the ADAC-LVi board ID number to be used in this acquisition. The ID numbers are defined in the configuration file ADAC-LVi.con located in the Windows main directory.

[abc] **Channel List (0(1))** Channel List is a string that specifies the analog input channel numbers to acquire data from. Channel List is typically set to acquire data from multiple channels, but may be used to collect a single channel into its one-dimensional array.

For boards that support input gain the channel list also specifies a separate gain value for each channel. Gains are specified in parentheses after the channel number. For example, "0(1),1(8),2(4)" is interpreted as channel 0 at a gain of 1, channel 1 at a gain of 8 and channel 2 at a gain of 4.

[I32] **Iteration (init:0)** Iteration controls when the ADAC-LVi device subsystem allocation and device initialization is performed. If iteration is 0, the VI allocates the device and initializes it with the hardware specifications connected to the VI. The VI then starts the acquisition and reads the data. If iteration is greater than zero, then VI assumes that the current configuration is valid and starts the acquisition, then reads the data.

[TF] **Clear Acquisition (true)** Clear acquisition determines whether the VI releases the device after reading the specified number of samples. The VI should pass a TRUE to this parameter when reading the last set of samples for a given acquisition. The default is TRUE, which means that the VI reads data only once if this input remains unwired. You typically wire this input to the terminating condition of a loop, so that when the loop finishes the device is released.

[Err] **error in (no error)** error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

[SGL] **Samples** Samples is a one-dimensional array containing the scaled analog input data for the specified channels in volts.

[Err] **error out** error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

Analog Input Examples

Examples for each Easy Analog Input VIs are provided, and located in the **ADAC Examples** in your LabVIEW directory. Example programs all provide a dialog box allowing the selection of any ADAC board installed in ADAC-LVi. If a board incapable of performing a particular example is selected, an error will occur indicating non-support.

The **Easy AI Acq Wave** example demonstrates the use of ADAC AI Acquire Waveform VI. The example performs a single channel acquisition for the number of samples specified at the specified clocking frequency. Once all the samples have been collected the data collection stops and the samples are plotted to a graph. The whole process is placed in a loop that repeats until the EXIT button is depressed, exiting the application. The DAQ hardware must be configured for DMA and INTERRUPT operation to use this VI.

The **Easy AI Acq Waves** example demonstrates the use of ADAC AI Acquire Waveforms VI. The example performs a multi channel acquisition for the number of samples specified at the specified clocking frequency. Once all the samples have been collected the data collection stops and the samples are plotted to a graph. The whole process is placed in a loop that repeats until the EXIT button is depressed, exiting the application. The DAQ hardware must be configured for DMA and INTERRUPT operation to use this VI.

The **Easy AI Sample Channel** example demonstrates the use of ADAC AI Sample Channel VI. The example performs an immediate, **untimed** acquisition on a single channel and returns the sample acquired in volts.

The **Easy AI Sample Channels** example demonstrates the use of ADAC AI Sample Channel VI. The example performs an immediate, **untimed** acquisition on the specified channel(s) and returns the sample(s) acquired in volts in a one-dimensional array.

The **Easy AI Continuous Acq** example demonstrates the use of ADAC AI Continuous Scan VI. The example continuously performs a multi channel acquisition for the number of samples specified at the specified clocking frequency, and plots the samples to a graph. The acquisition is gap free if the hardware is capable of gap free data collection. The acquisition continues until the EXIT button is pressed, exiting the application. The DAQ hardware must be configured for DMA and INTERRUPT operation to use this VI.

The **Easy TC Sample Channels** example demonstrates the use of ADAC AI Sample Channels and ADAC Convert Thermocouple Buffer VIs. This example continuously performs an immediate, untimed acquisition on the specified channel(s) and returns all the data acquired in the specified temperature units. The desired thermocouple type(s) must be set in the ADAC-LCi.INI file before running this example.

The **Advanced AI Continuous Async Occurrence** example demonstrates the use of ADAC OccurrenceConfig VI. The example shows how to use multiple occurrences for a continuous asynchronous acquisition. This is a timed acquisition, meaning that a hardware clock is used to control the acquisition for fast and accurate timing. It is also a continuous circular buffered acquisition. This means that a software buffer is used between your DAQ board and LabVIEW. While data is being transferred from your board into one of the input buffers, LabVIEW is reading data from another buffer. The occurrence VI makes the acquisition asynchronous (allowing processor time for other things to run) by causing the loop to sleep until notified a buffer is available which is then read into LabVIEW.

The **Advanced AI AboutTrigger Async Occurrence** example demonstrates the use of ADAC OccurrenceConfig VI with ABOUT TRIGGER mode enabled. Currently the 5400 and 5800 series boards are the only ADAC boards that support ABOUT TRIGGER mode. The example VI demonstrates how to use the Intermediate Analog Input VIs to capture data before and after a trigger. Once started data is continuously collected into a buffer until a trigger is received. The number of samples obtained after a trigger is "Post Trigger Samples" and the number of samples obtained before the trigger is "BufferSize (minus) Post Trigger Samples". The starting point for pre-trigger data is indicated in the BUFFER STATUS cluster as TriggerPointPre and the post-trigger data as TriggerPoint. If a buffer has not been filled before the trigger occurs the Buffer Status flag is set to 2 and the BufferWrite is set to the number of pre-trigger samples actually obtained.

This VI also shows how to use multiple occurrences for a continuous asynchronous acquisition. This is a timed acquisition, meaning that a hardware clock is used to control the acquisition for fast and accurate timing. It is also a continuous circular buffered acquisition. This means that a software buffer is used between your DAQ board and LabVIEW. While data is being transferred from your board into one of the input buffers, LabVIEW is reading data from another buffer. The occurrence VI makes the acquisition asynchronous (allowing processor time for other things to run) by causing the loop to sleep until notified a buffer is available which is then read into LabVIEW.

The **Advanced AI PreTrigger Async Occurrence** example demonstrates the use of ADAC OccurrenceConfig VI with PRE TRIGGER mode enabled. Currently the 5400 and 5800 series boards are the only ADAC boards that support PRE TRIGGER mode. The example VI demonstrates how to use the Intermediate Analog Input VIs to capture data before a trigger. Once started data is continuously collected into a buffer until a trigger is received. The number of samples obtained before the trigger is the actual Buffer Size. If a buffer has not been filled before the trigger occurs the Buffer Status flag is set to 2 and the BufferWrite is set to the number of pre-trigger samples actually obtained.

This VI also shows how to use multiple occurrences for a continuous asynchronous acquisition. This is a timed acquisition, meaning that a hardware clock is used to control the acquisition for fast and accurate timing. It is also a continuous circular buffered acquisition. This means that a software buffer is used between your DAQ board and

LabVIEW. While data is being transferred from your board into one of the input buffers, LabVIEW is reading data from another buffer. The occurrence VI makes the acquisition asynchronous (allowing processor time for other things to run) by causing the loop to sleep until notified a buffer is available which is then read into LabVIEW.

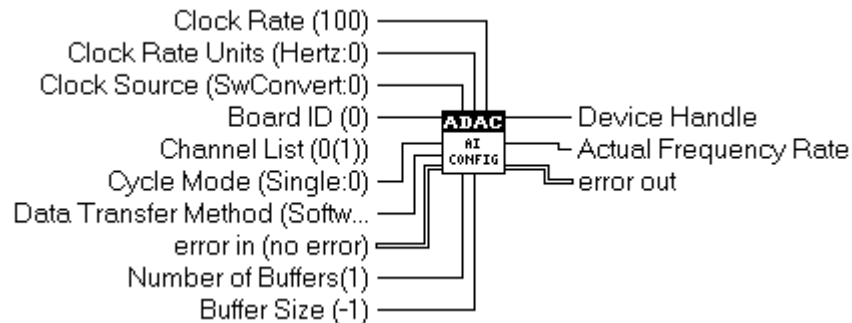
Intermediate Analog Input VIs

This section describes the Intermediate Analog Input VIs that are used to perform analog input functions. Intermediate Analog Input VIs provide greater flexibility and user configuration than the Easy Analog Input VIs.

To access the ADAC Analog Input Easy VIs, choose **Functions >> User Libraries >> ADAC >> Analog Input**. The Intermediate VIs are listed on the second line from the top of the analog input palette:

- ADAC AI Config
- ADAC AI Start Device
- ADAC AI Read
- ADAC AI Clear

ADAC AI Config



The ADAC AI Config VI allocates and configures a device's A/D subsystem, returning a Device Handle to be used in subsequent VI calls.

If the error in cluster contains an error from a previous VI call this VI does nothing, and returns the error in cluster unmodified in the error out. In the event of a previous error the Device Handle is set to -1.

ADAC AI Config (con't)

I32 **Board ID (0)** Specifies the ADAC-LVi board ID number to be used in this acquisition. The ID numbers are defined in the configuration file ADAC-LVi.con located in the Windows main directory.

abc **Channel List (0(1))** Channel List is a string that specifies the analog input channel number(s) to acquire data from.

For boards that support input gain the channel list also specifies a separate gain value for each channel. Gains are specified in parentheses after the channel number. For example, "0(1),1(8),2(4)" is interpreted as channel 0 at a gain of 1, channel 1 at a gain of 8 and channel 2 at a gain of 4.

I32 **Cycle Mode (Single:0)** Cycle Mode is a numeric value that sets the software cycle operation mode. Data can be collected in a single scan (SINGLE_CYCLE) filling buffer 0-n and then stopping acquisition or collected continuously (CONTINUOUS_CYCLE) filling buffers 0-n repeatedly. The available options are:

0:SINGLE_CYCLE Complete acquisition scan of 0-n buffers and stop. Supported by all A/D's.

1:CONTINUOUS_CYCLE Restart acquisition, repeatedly loop on 0-n buffers. Supported by A/Ds with DMA, Interrupt or FIFO capabilities.

For CONTINUOUS_CYCLE the Clock Source can not be set to SOFTWARE_CONVERT or an error will be generated immediately.

The default Cycle Mode is 0:SINGLE_CYCLE.

ADAC AI Config (con't)

Clock Source (SwConvert:0) Clock Source is a numeric value that sets the hardware clocking source. The available options are:

- 0:SOFTWARE_CONVERT Software clocked acquisition, supported by all A/D's.
- 1:INTERNAL On-board clocking acquisition, supported by A/D's with DMA, Interrupt or FIFO capabilities.
- 2:EXT_RISING_EDGE External clocking rising edge, supported by A/D's with DMA, Interrupt or FIFO capabilities.
- 3:EXT_FALLING_EDGE External clocking falling edge, supported by A/D's with DMA, Interrupt or FIFO capabilities.
- 4:MMTIMER Multi-Media system timer, supported by the 4012AD and 4112AD series in the Windows 95/98 environment only.

The default Clock Source is 0:SOFTWARE_CONVERT



Clock Rate Units (Hertz:0) Units is a numeric value that specifies base units in which the Clock Rate parameter is specified. The available options are:

- 0:HERTZ Rate is specified in units of hertz (samples per second)
- 1:TICS Rate is the divisor programmed to the on-board clocking source.
- 2:MSECONDS Miliseconds, supported by the 4012AD and 4112AD series in the Windows 95/98 environment only. The 4012 or 4112 series can be set to either HERTZ or MSECONDS only.

If TICS is chosen, the Clock Rate parameter is used to specify the divisor to the on-board CLK input source. The on-board clock source is typically an 8254 device that has a 1,000,000 CLK input source. The 1,000,000 source is divided from 2 to 65525 to set actual A/D clocking rate.

For boards that use 2 on-board clocks concatenated together, the divisor broken into two 16bit WORDS. The high word divides the first CLK that feeds the second CLK that is further divided by the low word.

The default Unit is 0:HERTZ

ADAC AI Config (con't)

- DBL **Clock Rate (100)** Clock Rate is a double value that specifies rate at which clocks occur. The units can be either Hertz (samples per second) or TICS. See the Clock Rate Units setting for details.
- I32 **Number of Buffers(1)** Specifies the number of buffers the VI allocates. This parameter defaults to 1, and should only be changed for multi-buffered acquisitions.
- I32 **Buffer Size (-1)** Buffer Size is the number of samples you want each buffer to hold. If Buffer Size is set to -1 the buffer size will be will automatically set to number of channels specified. Buffer Size should always be set to a multiple of the number of channels specified on the Channel List. For example a channel list of "0,1,2" three channels should have its Buffer size set to 3, 6, 9, 36, 300, ..., or any multiple of three. This requirement is needed to align buffer starting channels when the Number of Buffers exceeds 1.
- E6 **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
- I32 **Device Handle** Device Handle is a numeric value that identifies the board's A/D subsystem to be acted upon in subsequent VI subsystem calls.
- DBL **Actual Frequency Rate** Actual Frequency Rate is the actual interval between samples in hertz. The actual sample frequency can differ from the requested sample rate, depending on the clocking capabilities of your selected device.
- E6 **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC AI Start Device



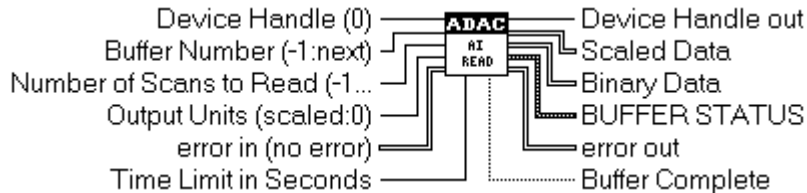
This VI starts analog input acquisition on the specified Device Handle. All configuration of the device must be completed before calling this VI.

Depending on the current ADAC-LVi transfer mode this call may not return until completed. See ADAC AI Config VI for detailed information on asynchronous operations.

AI Start Device (con't)

- I32** **Device Handle (0)** Device Handle is a numeric value that identifies the board's A/D subsystem to be acted upon. The default Device Handle is 0.
- Err** **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
- I32** **Device Handle out** Device Handle out contains the value of Device Handle in.
- Err** **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC AI Read



This VI reads data from a queued A/D buffer and converts the data to voltage upon request in a two dimensional array. This VI typically acquires voltage readings on multiple channels, but may be used to collect single channel points into its two dimensional array.

The BUFFER STATUS cluster is useful in determining the current status of the acquisition by presenting the Buffer DoneFlag, StatusFlag and ErrorFlag conditions as described below. The BUFFER STATUS is especially useful in multi-buffered/continuous acquisition where no interaction takes place except from incoming buffers.

Before reading any data, AI Read checks to see if the input cluster error in indicates that an error has already occurred. If so, this VI does not read any data, but passes the error information unmodified through error out.

- I32** **Device Handle (0)** Device Handle is a numeric value that identifies the board's A/D subsystem to be acted upon. The default Device Handle is 0.

ADAC AI Read (con't)

[I32] **Buffer Number (-1:next)** Buffer Number is the ID number of the buffer to be obtained. The Buffers are numbered from 0-n as specified in the ADAC AI Config VI. The default input is -1, which tells ADAC-LVi to return the next available buffer in the done queue. For multi-buffered or continuous acquisitions setting Buffer Number to -1 provides the easiest method of obtaining the next buffer available in the done queue. Queued buffers are returned in a first done first available fashion.

[I32] **Number of Scans to Read (-1:all)** Number of Scans to Read specifies the number of scans the VI retrieves from the acquisition buffer. The default input is -1, which tells LabVIEW to set number of scans to read equal to the value of the DAQ buffer size. Once a buffer is successfully read into LabVIEW the buffer is placed back into the device's available buffer queue. If the Number of Scans to Read is less than the actual buffer size, all unread data will be lost.

[U16] **Output Units (scaled:0)** Output Units specifies whether the VI returns unscaled binary data or scaled voltage data.

0:Scaled Return scaled voltage data only (default setting). The binary data array appears empty.

1:Binary Return binary data only. The scaled data array appears empty. The VI executes faster if you select this value, because the VI does not perform scaling.

[DBL] **Time Limit in Seconds** Time Limit in Seconds specifies the maximum length of time this VI waits for a done buffer. You express time limit in seconds. If this VI does not receive a completed buffer status prior to the timeout period, the VI returns an error.

[ERR] **error in (no error)** The error in cluster describes error conditions occurring before this VI executes. This cluster defaults to no error.

[I32] **Device Handle out** Device Handle out contains the value of Device Handle in.

[SGL] **Scaled Data** Scaled Data is a two-dimensional array containing analog input data in volts. The data appears in columns, where each column contains the data for a single channel. The second (bottom) dimension selects the channel. The first (top) dimension selects a single data point for that channel. This array is empty if Output Units is set to binary.

[I16] **Binary Data** Binary Data is a two-dimensional array containing unscaled analog input data. The data appears in columns, where each column contains the data for a single channel. The second (or bottom) dimension selects the channel. The first (or top) dimension selects a single data point for that channel. This array is empty if Output Units is set to scaled.

ADAC AI Read (con't)

U32 **BUFFER STATUS** ADAC-LVi buffers are not just arrays of data to a DAQ board's subsystem data, but are structures that contains the buffer type, size, data array, current sample count, hardware trigger points, completion flags, status flags, and error flags. Some of these fields remain constant for the duration of the acquisition. Others contain relevant information about the current condition of the buffer, and error conditions that can occurred during the current acquisition.

The three structure variables that provide state/status/error information are the DoneFlag, StatusFlag and ErrorFlags.

U32 **BufferType** [U32] BufferType
Data Buffer type, DMA, Interrupt or Polling.

U32 **BufferSize** [U32] BufferSize
Size of the buffer in bytes.

U32 **BufferWrite** [U32] BufferWrite
Number of data samples in the buffer.

U32 **BufferRead** [U32] BufferRead
Number of data samples obtained from a buffer.

U32 **TriggerPoint** [U32] TriggerPoint
Buffer position number of the trigger point.

U32 **TiggerPointStart** [U32] TriggerPointStart
Buffer position number of the pre-trigger data.

U32 **DoneFlag** [U32] DoneFlag:
The Done Flag specifies the current state of a buffer. This flag can be set to any of the three following conditions.

BUFFER_IDLE = 0 The buffer is available for use, but acquisition has not started on this buffer.

BUFFER_DONE = 1 The buffer has been filled with samples and is available for use in the user application.

BUFFER_INUSE = 2 The buffer is currently being filled with samples

ADAC AI Read (con't.)

U32

StatusFlag [U32] StatusFlag:

The Status Flag specifies the completion status of a buffer. This flag can be set to any of the four following conditions.

BUFFER_EMPTY = 1 No samples are available in the buffer.

BUFFER_INCOMPLETE = 2 The output buffer has not read completely or and input has not filled to capacity.

BUFFER_COMPLETE = 4 The output buffer has been read completely.

BUFFER_FULL = 8 The input buffer has filled to capacity.

I32

ErrorFlag [i32] ErrorFlags:

The Error Flags specify the condition which has stopped a buffer from successfully completing. Although an error may be reported, this does not indicate the BUFFER_FULL flag is not set. This would be the case if an error was detected and enough samples were available to fill the buffer to capacity. One or more of these flags

may also be set together. If any of these flags have set, checking the current hardware error conditions will usually reveal the actual condition that caused the acquisition runtime error. This flag can be set to any of the four following conditions.

BUFFER_STOPPED = 1 An error occurred and the buffer(s) are no longer being serviced.

BUFFER_OVERRUN = 2 An error occurred when attempting to place the next sample beyond the bounds of the buffer.

BUFFER_UNDERRUN = 4 An error occurred and the buffer is incomplete.

BUFFER_NEXTBUSY = 8 When attempting to access the next buffer, the current state of the DoneFlag indicated the buffer was not yet available for use.

U32

BufferNum [U32] BuffNum

Buffer Number ID (0-n)

TF

Buffer Complete Buffer Complete is TRUE if this VI was able to successfully obtain a done buffer. A completed buffer may only indicate the buffer is no longer being serviced and the status is available, always check the StatusFlag and ErrorFlag.

ADAC AI Clear



This VI stops, frees all allocated resources and releases an A/D subsystem associated with the specified Device Handle. Once called, the Device Handle should be considered invalid, and not use in any subsequent subsystem calls.

This VI calls AL_StopDevice and AL_ReleaseDevice to stop and release the subsystem even if the error in cluster contains an error, or if any VIs called result in an error.

Before another acquisition can begin, you must call ADAC AI Config again to reallocate the device subsystem.

Note

This VI will automatically release the ADAC-LVi environment if the specified Device Handle is the last valid ADAC-LVi Device Handle in use. This behavior is controlled by the AutoLoad=YES setting in the ADAC-LVi.con file located in the main Windows directory.



Device Handle (0) Device Handle is a numeric value that identifies the board's A/D subsystem to be acted upon. The default Device Handle is 0.



error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.



error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

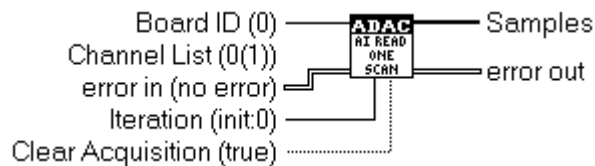
Utility Analog Input VIs

This section describes the Utility Analog Input VIs that are used to perform analog input functions. Utility Analog Input VIs provide support for the Easy and Intermediate Analog Input VIs. Utility Analog Input VIs may also be used as sub VIs in a user application.

To access the ADAC Analog Input Easy VIs, choose **Functions >> User Libraries >> ADAC >> Analog Input >> AI Util.** The Utility Analog Input VIs are listed in a pop-up window as follows:

- ADAC AI Read One Scan
- ADAC AI Wave Scan
- ADAC AI Cont Scan
- ADAC AI Set Trigger Config

ADAC AI Read One Scan



The ADAC AI Read One Scan VI performs an immediate, **untimed** acquisition on the specified channel(s) and returns all the data acquired in scaled data units (volts). This VI typically acquires voltage readings on a single channel only, but may be used to collect multiple channels into its 1 dimensional array.

ADAC AI Read One Scan (con't)

[I32] **Board ID (0)** Specifies the ADAC-LVi board ID number to be used in this acquisition. The ID numbers are defined in the configuration file ADAC-LVi.con located in the Windows main directory.

[abc] **Channel List (0 (1))** Channel List is a string that specifies the analog input channel number to acquire data from. Channel is typically set to acquire only a single channel of data but may be used to collect multiple channels into its 1 dimensional array.

For boards that support input gain the channel list also specifies a separate gain value for each channel. Gains are specified in parentheses after the channel number. For example, "0(1),1(8),2(4)" is interpreted as channel 0 at a gain of 1, channel 1 at a gain of 8 and channel 2 at a gain of 4.

[I32] **Iteration (init:0)** Iteration controls when the ADAC-LVi device subsystem allocation and device initialization is performed. If iteration is 0, the VI allocates the device and initializes it with the hardware specifications connected to the VI. The VI then starts the acquisition and reads the data. If iteration is greater than zero, then VI assumes that the current configuration is valid and starts the acquisition, then reads the data.

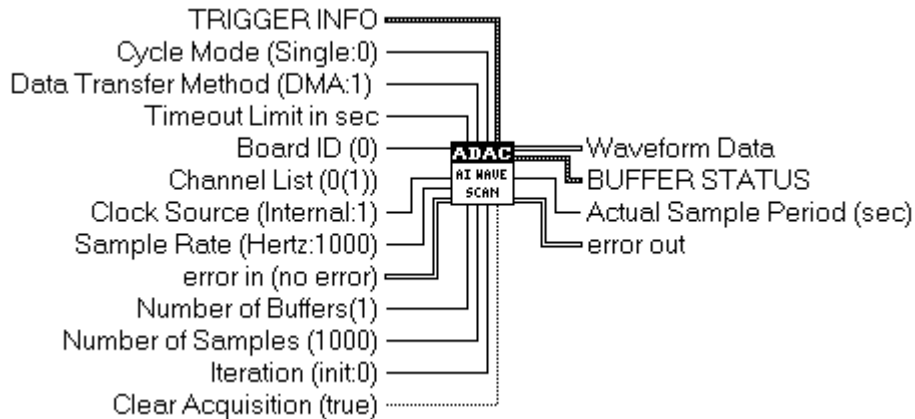
[TF] **Clear Acquisition (true)** Clear acquisition determines whether the VI releases the device after reading the specified number of samples. The VI should pass a TRUE to this parameter when reading the last set of samples for a given acquisition. The default is TRUE, which means that the VI reads data only once if this input remains unwired. You typically wire this input to the terminating condition of a loop, so that when the loop finishes the device is released.

[Err] **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

[SGL] **Samples** Samples is a one-dimensional array containing the scaled analog input data for the specified channels in volts.

[Err] **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC AI Wave Scan



The ADAC AI Waveform Scan VI collects the specified number of samples at the specified sample rate and returns the waveform data in a two dimensional array. This VI typically acquires voltage readings on multiple channels, but may be used to collect single channel points into its two dimensional array.

If the error in cluster contains an error from a previous VI call, this VI does nothing and returns the error in cluster unmodified in the error out.

132 **Board ID (0)** Specifies the ADAC-LVi board ID number to be used in this acquisition. The ID numbers are defined in the configuration file ADAC-LVi.con located in the Windows main directory.

abc **Channel List (0 (1))** Channel List is a string that specifies the analog input channel numbers to acquire data from. Channel List is typically set to acquire multiple channels, but may be used to collect single channels into its two dimensional array.

For boards that support input gain the channel list also specifies a separate gain value for each channel. Gains are specified in parentheses after the channel number. For example, "0(1),1(8),2(4)" is interpreted as channel 0 at a gain of 1, channel 1 at a gain of 8 and channel 2 at a gain of 4.

ADAC AI Wave Scan (con't.)

- I32** **Clock Source (Internal:1)** Clock Source is a numeric value that sets the hardware clocking source. The available options are:
- 0:SOFTWARE_CONVERT Software clocked acquisition, supported by all A/D's.
 - 1:INTERNAL On-board clocking acquisition, supported by A/D's with DMA, Interrupt or FIFO capabilities.
 - 2:EXT_RISING_EDGE External clocking rising edge, supported by A/D's with DMA, Interrupt or FIFO capabilities.
 - 3:EXT_FALLING_EDGE External clocking falling edge, supported by A/D's with DMA, Interrupt or FIFO capabilities.
 - 4:MMTIMER Multi-Media system timer, supported by the 4012AD and 4112AD series in the Windows 95/98 environment only.

The default Clock Source is 0:SOFTWARE_CONVERT

- DBL** **Sample Rate (Hertz:1000)** Sample rate is the requested number of samples per second the VI acquires from the specified channel. This parameter defaults to a rate of 1000.00 samples per second.
- I32** **Number of Buffers(1)** Specifies the number of buffers the VI allocates. This parameter defaults to 1, and should only be changed for multi-buffered acquisitions.
- U32** **Number of Samples (1000)** Number of samples is the number of single-channel samples the VI acquires before the acquisition completes. This parameter defaults to 1000.
- I32** **Iteration (init:0)** Iteration controls when the ADAC-LVi device subsystem allocation and device initialization is performed. If iteration is 0, the VI allocates the device and initializes it with the hardware specifications connected to the VI. The VI then starts the acquisition and reads the data. If iteration is greater than zero, then VI assumes that the current configuration is valid and starts the acquisition, then reads the data.
- TF** **Clear Acquisition (true)** Clear acquisition determines whether the VI releases the device after reading the specified number of samples. The VI should pass a TRUE to this parameter when reading the last set of samples for a given acquisition. The default is TRUE, which means that the VI reads data only once if this input remains unwired. You typically wire this input to the terminating condition of a loop, so that when the loop finishes the device is released.
- DBL** **TRIGGER INFO** TRIGGER INFO configures the analog input triggering mechanism on the specified Device Handle.

ADAC AI Wave Scan (con't.)

I32 **Trigger Mode (0:disabled)** Trigger Mode is a numeric value that sets the hardware trigger operation mode. The available options are:

0:DISABLED All triggering is disabled.

1:ABOUT_TRIG Collect (N) data points before and (N) data points after a trigger. Supported by A/D's ABOUT TRIGGER capabilities.

2:POST_TRIG Collect (N) data points after a trigger. supported by A/D's with DMA, Interrupt or FIFO capabilities.

3:PRE_TRIG Collect (N) data points before a trigger. Supported by A/D's ABOUT TRIGGER capabilities.

4:SCAN_TRIG Collect 1 data point for each channel after each trigger. Supported by A/D's SCAN TRIGGER capabilities.

The default Trigger Mode is 0:DISABLED

I32 **Trigger Source (0:disabled)** Trigger Source is a numeric value that sets the hardware triggering source. The available options are:

0:DISABLED All triggering sources are disabled.

1:EXTERNAL Trigger the acquisition from an external event. Supported by A/D's with DMA, Interrupt or FIFO capabilities.

2:CTR1 On-board counter 1 triggers the acquisition. Supported by A/D's with DMA, Interrupt or FIFO capabilities.

3:CTR2 On-board counter 2 triggers the acquisition. Supported by A/D's with DMA, Interrupt or FIFO capabilities.

The default Trigger Source is 0:DISABLED

I32 **Trigger Source Signal (0:falling)** Trigger Source Signal is a numeric value that sets the hardware triggering source signal. The available options are:

0:FALLING_EDGE Trigger on the falling edge of an event.

1:RISING_EDGE Trigger on the rising edge of an event.

The default Trigger Source Signal is 0:FALLING_EDGE

ADAC AI Wave Scan (con't.)

DBL **Trigger Rate (100)** Trigger Rate is a double value that specifies rate at which on-board triggers occur per second. This parameter is used when the trigger source is set to an on-board triggering source such as CTR1.

I32 **Post Sample Count** PostSampleCount specifies the number of samples to be obtained after an ABOUT TRIGGER mode trigger. This value typically ranges from 1 to 65535 samples. See your specific hardware capabilities for exact sample ranges.

I32 **Cycle Mode (Single:0)** Cycle Mode is a numeric value that sets the software cycle operation mode. Data can be collected in a single scan (SINGLE_CYCLE) filling buffer 0-n and then stopping acquisition or collected continuously (CONTINUOUS_CYCLE) filling buffers 0-n repeatedly.
The available options are:

0:SINGLE_CYCLE Complete acquisition scan of 0-n buffers and stop. Supported by all A/D's.

1:CONTINUOUS_CYCLE Restart acquisition, repeatedly loop on 0-n buffers. Supported by A/Ds with DMA, Interrupt or FIFO capabilities. For CONTINUOUS_CYCLE the Clock Source can not be set to SOFTWARE_CONVERT or an error will be generated immediately.

The default Cycle Mode is 0:SINGLE_CYCLE.

I32 **Data Transfer Method (DMA:1)** Data Transfer Method is a numeric value that sets the hardware transfer method. This parameter specifies how data from the A/D is to be transferred to buffers.
The available options are:

0:SOFTWARE Data is transferred via a software controlled loop, supported by all A/D's. For A/Ds that have a FIFO available, a Clock Source setting other than SOFTWARE_CONVERT will cause the START A/D VI to return immediately. Each call to the AI READ VI will then collect the data stored in the FIFO, providing asynchronous transfers.

1:DMA Data is transferred via DMA, supported by A/D's with DMA, capabilities.

2:IRQ Data is transferred via ISR, supported by A/D's with Interrupt, capabilities.

The default Data Transfer Method is 0:SOFTWARE.

ADAC AI Wave Scan (con't.)

- [DBL]** **timeout limit in sec** Time Limit in Seconds specifies the maximum length of time this VI waits for a done buffer. You express time limit in seconds. If this VI does not receive a completed buffer status prior to the timeout period, the VI returns an error.

- [ERR]** **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

- [SGL]** **Waveform Data** Waveform Data is a two-dimensional array containing analog input data in volts. The data appears in columns, where each column contains the data for a single channel. The second (bottom) dimension selects the channel. The first (top) dimension selects a single data point for that channel. This array is empty if convert to volts is FALSE.

- [DOB]** **BUFFER STATUS** ADAC-LVi buffers are not just arrays of data to a DAQ board's subsystem data, but are structures that contains the buffer type, size, data array, current sample count, hardware trigger points, completion flags, status flags, and error flags. Some of these fields remain constant for the duration of the acquisition. Others contain relevant information about the current condition of the buffer, and error conditions that can occurred during the current acquisition.

The three structure variables that provide state/status/error information are the DoneFlag, StatusFlag and ErrorFlags.

- [U32]** **BufferType** [U32] BufferType
Data Buffer type, DMA, Interrupt or Polling.

- [U32]** **BufferSize** [U32] BufferSize
Size of the buffer in bytes.

- [U32]** **BufferWrite** [U32] BufferWrite
Number of data samples in the buffer.

- [U32]** **BufferRead** [U32] BufferRead
Number of data samples obtained from a buffer.

- [U32]** **TriggerPoint** [U32] TriggerPoint
Buffer position number of the trigger point.

- [U32]** **TiggerPointStart** [U32] TriggerPointStart
Buffer position number of the pre-trigger data.

ADAC AI Wave Scan (con't.)**U32****DoneFlag** [U32] DoneFlag:

The Done Flag specifies the current state of a buffer. This flag can be set to any of the three following conditions.

BUFFER_IDLE = 0 The buffer is available for use, but acquisition has not started on this buffer.

BUFFER_DONE = 1 The buffer has been filled with samples and is available for use in the user application.

BUFFER_INUSE = 2 The buffer is currently being filled with samples

U32**StatusFlag** [U32] StatusFlag: The Status Flag specifies the completion status of a buffer. This flag can be set to any of the four following conditions.

BUFFER_EMPTY = 1 No samples are available in the buffer.

BUFFER_INCOMPLETE = 2 The output buffer has not read completely and input has not filled to capacity.

BUFFER_COMPLETE = 4 The output buffer has been read completely.

BUFFER_FULL = 8 The input buffer has filled to capacity.

I32**ErrorFlag** [i32] ErrorFlags:

The Error Flags specify the condition which has stopped a buffer from successfully completing. Although an error may be reported, this does not indicate the BUFFER_FULL flag is not set. This would be the case if an error was detected and enough samples were available to fill the buffer to capacity. One or more of these flags may also be set together. If any of these flags have set, checking the current hardware error conditions will usually reveal the actual condition that caused the acquisition runtime error. This flag can be set to any of the four following conditions.

BUFFER_STOPPED = 1 An error occurred and the buffer(s) are no longer being serviced.

BUFFER_OVERRUN = 2 An error occurred when attempting to place the next sample beyond the bounds of the buffer.

BUFFER_UNDERRUN = 4 An error occurred and the buffer is incomplete.

BUFFER_NEXTBUSY = 8 When attempting to access the next buffer, the current state of the DoneFlag indicated the buffer was not yet available for use.

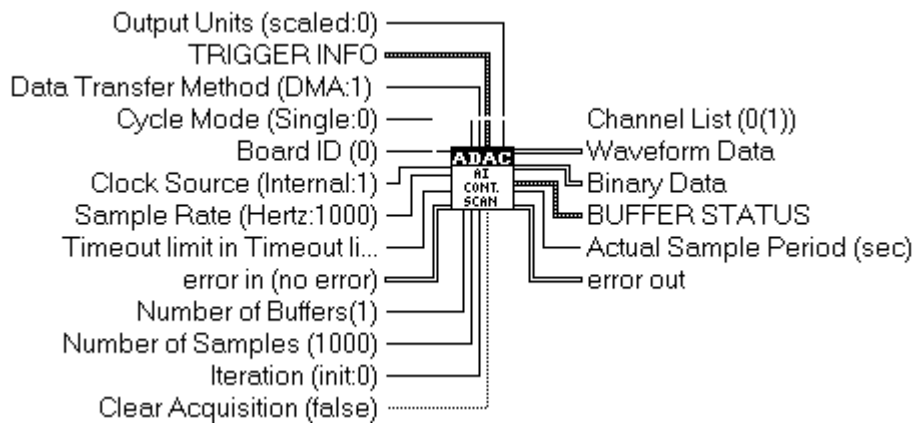
U32**BufferNum** [U32] BufferNum Buffer Number ID (0-n)

ADAC AI Wave Scan (con't.)

DBL **Actual Sample Period (sec)** Actual Sample Period is the actual interval between samples in seconds, which is the inverse of the actual sample rate in hertz. The actual sample period can differ from the requested sample rate, depending on the clocking capabilities of your selected device.

PF **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC AI Cont Scan



The ADAC AI Continuous Scan VI continuously collects the specified number of samples at the specified sample rate and returns the waveform data.

If the error in cluster contains an error from a previous VI call this VI does nothing and returns the error in cluster unmodified in the error out in a two dimensional array. This VI typically acquires voltage readings on a single channel only, but may be used to collect multiple channels into its 1 dimensional array.

I32 **Board ID (0)** Specifies the ADAC-LVi board ID number to be used in this acquisition. The ID numbers are defined in the configuration file ADAC-LVi.con located in the Windows main directory.

ADAC AI Cont Scan (con't)

abc **Channel List (0 (1))** Channel List is a string that specifies the analog input channel numbers to acquire data from. Channel List is typically set to acquire multiple channels, but may be used to collect single channels into its two dimensional array.

For boards that support input gain the channel list also specifies a separate gain value for each channel. Gains are specified in parentheses after the channel number. For example, "0(1),1(8),2(4)" is interpreted as channel 0 at a gain of 1, channel 1 at a gain of 8 and channel 2 at a gain of 4.

I32 **Clock Source (Internal:1)** Clock Source is a numeric value that sets the hardware clocking source. The available options are:

0:SOFTWARE_CONVERT Software clocked acquisition, supported by all A/D's.

1:INTERNAL On-board clocking acquisition, supported by A/D's with DMA, Interrupt or FIFO capabilities.

2:EXT_RISING_EDGE External clocking rising edge, supported by A/D's with DMA, Interrupt or FIFO capabilities.

3:EXT_FALLING_EDGE External clocking falling edge, supported by A/D's with DMA, Interrupt or FIFO capabilities.

4:MMTIMER Multi-Media system timer, supported by the 4012AD and 4112AD series in the Windows 95/98 environment only.

The default Clock Source is 0:SOFTWARE_CONVERT

DBL **Sample Rate (Hertz:1000)** Sample rate is the requested number of samples per second the VI acquires from the specified channel. This parameter defaults to a rate of 1000.00 samples per second.

I32 **Number of Buffers(1)** Specifies the number of buffers the VI allocates. This parameter defaults to 1, and should only be changed for multi-buffered acquisitions.

U32 **Number of Samples (1000)** Number of samples is the number of single-channel samples the VI acquires before the acquisition completes. This parameter defaults to 1000.

DBL **Timeout limit (sec)** Time Limit in Seconds specifies the maximum length of time this VI waits for a done buffer. You express time limit in seconds. If this VI does not receive a completed buffer status prior to the timeout period, the VI returns an error.

U16 **Output Units (scaled:0)** Output Units specifies whether the VI returns unscaled binary data or scaled voltage data.

ADAC AI Cont Scan (con't)

0:Scaled Return scaled voltage data only (default setting). The binary data array appears empty.

1:Binary Return binary data only. The scaled data array appears empty.

The VI executes faster if you select this value, because the VI does not perform scaling.



TRIGGER INFO TRIGGER INFO configures the analog input triggering mechanism on the specified Device Handle.



Trigger Mode (0:disabled) Trigger Mode is a numeric value that sets the hardware trigger operation mode. The available options are:

0:DISABLED All triggering is disabled.

1:ABOUT_TRIG Collect (N) data points before and (N) data points after a trigger. Supported by A/D's ABOUT TRIGGER capabilities.

2:POST_TRIG Collect (N) data points after a trigger. supported by A/D's with DMA, Interrupt or FIFO capabilities.

3:PRE_TRIG Collect (N) data points before a trigger. Supported by A/D's ABOUT TRIGGER capabilities.

4:SCAN_TRIG Collect 1 data point for each channel after each trigger. Supported by A/D's SCAN TRIGGER capabilities.

The default Trigger Mode is 0:DISABLED



Trigger Source (0:disabled) Trigger Source is a numeric value that sets the hardware triggering source. The available options are:

0:DISABLED All triggering sources are disabled.

1:EXTERNAL Trigger the acquisition from an external event. Supported by A/D's with DMA, Interrupt or FIFO capabilities.

2:CTR1 On-board counter 1 triggers the acquisition. Supported by A/D's with DMA, Interrupt or FIFO capabilities.

3:CTR2 On-board counter 2 triggers the acquisition. Supported by A/D's with DMA, Interrupt or FIFO capabilities.

The default Trigger Source is 0:DISABLED

ADAC AI Cont Scan (con't.)

I32 **Trigger Source Signal (0:falling)** Trigger Source Signal is a numeric value that sets the hardware triggering source signal. The available options are:

0:FALLING_EDGE Trigger on the falling edge of an event.

1:RISING_EDGE Trigger on the rising edge of an event.

The default Trigger Source Signal is 0:FALLING_EDGE

DBL **Trigger Rate (100)** Trigger Rate is a double value that specifies rate at which on-board triggers occur per second. This parameter is used when the trigger source is set to an on-board triggering source such as CTR1.

I32 **Post Sample Count** PostSampleCount specifies the number of samples to be obtained after an ABOUT TRIGGER mode trigger. This value typically ranges from 1 to 65535 samples. See your specific hardware capabilities for exact sample ranges.

I32 **Cycle Mode (Single:0)** Cycle Mode is a numeric value that sets the software cycle operation mode. Data can be collected in a single scan (SINGLE_CYCLE) filling buffer 0-n and then stopping acquisition or collected continuously (CONTINUOUS_CYCLE) filling buffers 0-n repeatedly. The available options are:

0:SINGLE_CYCLE Complete acquisition scan of 0-n buffers and stop. Supported by all A/Ds.

1:CONTINUOUS_CYCLE Restart acquisition, repeatedly loop on 0-n buffers.

Supported by
A/D's with DMA, Interrupt or FIFO capabilities.

For CONTINUOUS_CYCLE the Clock Source can not be set to SOFTWARE_CONVERT or an error will be generated immediately.

The default Cycle Mode is 0:SINGLE_CYCLE.

I32 **Iteration (init:0)** Iteration controls when the ADAC-LVi device subsystem allocation and device initialization is performed. If iteration is 0, the VI allocates the device and initializes it with the hardware specifications connected to the VI. The VI then starts the acquisition and reads the data. If iteration is greater than zero, then VI assumes that the current configuration is valid and starts the acquisition, then reads the data.

ADAC AI Cont Scan (cont.)

[TF] **Clear Acquisition (false)** Clear acquisition determines whether the VI releases the device after reading the specified number of samples. The VI should pass a TRUE to this parameter when reading the last set of samples for a given acquisition. The default is FALSE, which means that the VI reads data continuously if this input remains unwired. You typically wire this input to the terminating condition of a loop, so that when the loop finishes the device is released.

[53] **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

[I32] **Data Transfer Method (DMA:1)** Data Transfer Method is a numeric value that sets the hardware transfer method. This parameter specifies how data from the A/D is to be transferred to buffers. The available options are:

0:SOFTWARE Data is transferred via a software controlled loop, supported by all A/D's. For A/D's that have a FIFO available, a Clock Source setting other than SOFTWARE_CONVERT will cause the START A/D VI to return immediately. Each call to the AI READ VI will then collect the data stored in the FIFO, providing asynchronous transfers.

1:DMA Data is transferred via DMA, supported by A/D's with DMA, capabilities.

2:IRQ Data is transferred via ISR, supported by A/D's with Interrupt, capabilities.

The default Data Transfer Method is 0:SOFTWARE.

[56] **Waveform Data** Waveform Data is a two-dimensional array containing analog input data in volts. The data appears in columns, where each column contains the data for a single channel. The second (bottom) dimension selects the channel. The first (top) dimension selects a single data point for that channel. This array is empty if convert to volts is FALSE.

[116] **Binary Data** Binary Data is a two-dimensional array containing unscaled analog input data. The data appears in columns, where each column contains the data for a single channel. The second (or bottom) dimension selects the channel. The first (or top) dimension selects a single data point for that channel.

ADAC AI Cont Scan (con't.)

U32 **BUFFER STATUS** ADAC-LVi buffers are not just arrays of data to a DAQ board's subsystem data, but are structures that contains the buffer type, size, data array, current sample count, hardware trigger points, completion flags, status flags, and error flags. Some of these fields remain constant for the duration of the acquisition. Others contain relevant information about the current condition of the buffer, and error conditions that can occurred during the current acquisition.

The three structure variables that provide state/status/error information are the DoneFlag, StatusFlag and ErrorFlags.

U32 **BufferType** [U32] BufferType
Data Buffer type, DMA, Interrupt or Polling.

U32 **BufferSize** [U32] BufferSize
Size of the buffer in bytes.

U32 **BufferWrite** [U32] BufferWrite
Number of data samples in the buffer.

U32 **BufferRead** [U32] BufferRead
Number of data samples obtained from a buffer.

U32 **TriggerPoint** [U32] TriggerPoint
Buffer position number of the trigger point.

U32 **TiggerPointStart** [U32] TriggerPointStart
Buffer position number of the pre-trigger data.

U32 **DoneFlag** [U32] DoneFlag: The Done Flag specifies the current state of a buffer. This flag can be set to any of the three following conditions.
 BUFFER_IDLE = 0 The buffer is available for use, but acquisition has not started on this buffer.
 BUFFER_DONE = 1 The buffer has been filled with samples and is available for use in the user application.
 BUFFER_INUSE = 2 The buffer is currently being filled with samples

U32 **StatusFlag** [U32] StatusFlag: The Status Flag specifies the completion status of a buffer. This flag can be set to any of the four following conditions.

BUFFER_EMPTY = 1 No samples are available in the buffer.
 BUFFER_INCOMPLETE = 2 The output buffer has not read completely or and input has not filled to capacity.

ADAC AI Cont Scan (con't.)

BUFFER_COMPLETE = 4 The output buffer has been read completely.
 BUFFER_FULL = 8 The input buffer has filled to capacity.

I32 **ErrorFlag** [i32] ErrorFlags: The Error Flags specify the condition which has stopped a buffer from successfully completing. Although an error may be reported, this does not indicate the BUFFER_FULL flag is not set. This would be the case if an error was detected and enough samples were available to fill the buffer to capacity. One or more of these flags may also be set together. If any of these flags have set, checking the current hardware error conditions will usually reveal the actual condition that caused the acquisition runtime error. This flag can be set to any of the four following conditions.

BUFFER_STOPPED = 1 An error occurred and the buffer(s) are no longer being serviced.
 BUFFER_OVERRUN = 2 An error occurred when attempting to place the next sample beyond the bounds of the buffer.
 BUFFER_UNDERRUN = 4 An error occurred and the buffer is incomplete.
 BUFFER_NEXTBUSY = 8 When attempting to access the next buffer, the current state of the DoneFlag indicated the buffer was not yet available for use.

U32 **BufferNum** [U32] BuffNum Buffer Number ID (0-n)

DBL **Actual Sample Period (sec)** Actual Sample Period is the actual interval between samples in seconds, which is the inverse of the actual sample rate in hertz. The actual sample period can differ from the requested sample rate, depending on the clocking capabilities of your selected device.

ERR **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC AI Set Trigger Config



This VI configures triggering operations of an analog input acquisition.

ADAC AI Set Trigger Config (con't)

I32 **Device Handle (0)** Device Handle is a numeric value that identifies the board's A/D subsystem to be acted upon. The default Device Handle is 0.

I32 **TRIGGER INFO**

I32 **Trigger Mode** Trigger Mode is a numeric value that sets the hardware trigger operation mode. The available options are:

0:DISABLED All triggering is disabled.

1:ABOUT_TRIG Collect (N) data points before and (N) data points after a trigger.

2:POST_TRIG Collect (N) data points after a trigger.

3:PRE_TRIG Collect (N) data points before a trigger.

4:SCAN_TRIG Collect 1 data point for each channel after each trigger.

The default Trigger Mode is 0:DISABLED

I32 **Trigger Source** Trigger Source is a numeric value that sets the hardware triggering source. The available options are:

0:DISABLED All triggering sources are disabled.

1:EXTERNAL Trigger the acquisition from an external event.

2:CTR1 On-board counter 1 triggers the acquisition.

3:CTR2 On-board counter 2 triggers the acquisition.

The default Trigger Source is 0:DISABLED

I32 **Trigger Source Signal** Trigger Source Signal is a numeric value that sets the hardware triggering source signal. The available options are:

0:FALLING_EDGE Trigger on the falling edge of an event.

1:RISING_EDGE Trigger on the rising edge of an event.

The default Trigger Source Signal is 0:FALLING_EDGE

ADAC AI Set Trigger Config (con't)



Trigger Rate Trigger Rate is the time in seconds(Hertz) the a trigger clock re-triggers an acquisition. This parameter is used to set rate of an on-board triggering clock when selected form the Trigger Source setting above.



Post Sample Counts (-1: nochange) Post Sample Counts specifies the number of samples to be collected after the second trigger occurs. The Trigger Mode must be set to ABOUT_TRIGGER.



error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.



Device Handle out Device Handle out contains the value of Device Handle in.



error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

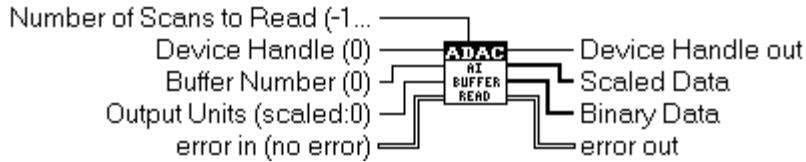
Advanced Analog Input VIs

This section describes the Advanced Analog Input VIs that are used to perform analog input functions. The advanced VIs provide the link to the ADAC DAQ boards. Easy, Utility, and Intermediate Analog Input VIs were all created using Advanced Analog Input VIs.

To access the ADAC Analog Input Easy VIs, choose **Functions >> User Libraries >> ADAC >> Analog Input >> Adv.** The Advanced Analog Input in a pop-up window:

- ADAC AI Buffer Read

ADAC AI Buffer Read



This VI reads data from a buffered data acquisition into a one-dimensional array in the specified data output units. Before reading any data, AI Read checks to see if the input cluster error in indicates that an error has already occurred. If so, this VI does not read any data, but passes the error information unmodified through error out. Otherwise, this VI reads the specified amount of data from a buffered analog input acquisition. See also `AL_GetBufferStatus.vi`.



Device Handle (0) Device Handle is a numeric value that identifies the board's A/D subsystem to be acted upon. The default Device Handle is 0.



Buffer Number (0) Buffer Number is the ID number of the buffer(0-n) to be obtained from the devices internal subsystem.



Number of Scans to Read (-1:all) Number of Scans to Read specifies the number of scans the VI retrieves from the acquisition buffer. The default input is -1, which tells LabVIEW to set number of scans to read equal to the value of the DAQ buffer size. Once a buffer is successfully read into LabVIEW, the buffer is placed back into the devices available buffer queue. If the Number of Scans to Read is less than the actual buffer size, all unread data will be lost.

ADAC AI Buffer Read (cont.)

[U16] **Output Units (scaled:0)** Output Units specifies whether the VI returns unscaled binary data or scaled voltage data.

0:Scaled Return scaled voltage data only (default setting). The binary data array appears empty.

1:Binary Return binary data only. The scaled data array appears empty.

2:Scaled & Binary Returns both scaled and binary data.

The VI executes faster if you select binary data only, because the VI does not perform scaling.

[E75] **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

[I32] **Device Handle out** Device Handle out contains the value of Device Handle in.

[SGL] **Scaled Data** Scaled Data is a one-dimensional array containing analog input data in Volts. The data appears in successive array elements, where each element contains the data for a single channel.

[I16] **Binary Data** Binary Data is a one-dimensional array containing analog input data in raw data. The data appears in successive array elements, where each element contains the data for a single channel.

[E75] **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

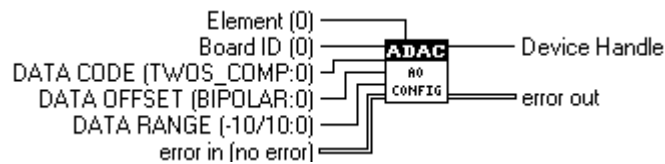
The Analog Output VIs

The Analog Output VIs are used to perform analog output functions.

To access the ADAC Analog Output VIs, choose **Functions >> User Libraries >> ADAC >> Analog Output**.

- ADAC AO Config
- ADAC AO Update Channel
- ADAC AO Clear

ADAC AO Config



The ADAC AO Config VI allocates and configures a device's D/A subsystem returning a Device Handle to be used in subsequent VI calls.

If the error in cluster contains an error from a previous VI call this VI does nothing and returns the error in cluster unmodified in the error out. In the event of a previous error the Device Handle is set to -1.



Board ID (0) Specifies the ADAC-LVi board ID number to be used in this acquisition. The ID numbers are defined in the configuration file ADAC-LVi.con located in the Windows main directory.



DATA CODE (TWOS_COMP:0) DATA CODE is a numeric value that sets the hardware data coding configuration of the device. The available options are:

0:TWOS_COMPLEMENT	Two's Complement data coding
1:STRAIGHT_BINARY	Straight Binary data coding
2:OFFSET_BINARY	Offset Binary data coding

The default DATA CODE is 0:TWOS_COMPLEMENT

ADAC AO Config (cont.)

I32 **DATA OFFSET (BIPOLAR:0)** DATA OFFSET is a numeric value that sets the hardware data offset configuration of the device.

The available options are:

0:BIPOLAR Bipolar data +/- V

1:UNIPOLAR Unipolar data 0-V

The default DATA OFFSET is 0:BIPOLAR

E51 **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

I32 **DATA RANGE (-10/10:0)** DATA RANGE is a numeric value that sets the hardware voltage data range of the device.

The available options are:

0:-5_5 +/- 5V

1:0_10 + 0-10V

2:-10_10 +/- 10V

The default DATA OFFSET is 2:-10_10

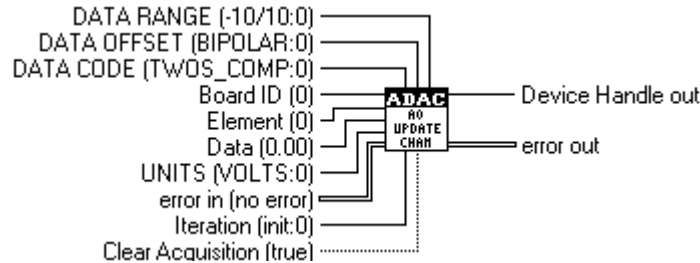
U16 **Element (0)** Element is a numeric value that specifies the DAQ hardware subsystems DAC number. For a two DAC board the first DAC would be Element 0 and the second DAC would be element 1.

The default Element is 0.

I32 **Device Handle** Device Handle is a numeric value that identifies the board's D/A subsystem to be acted upon in subsequent VI subsystem calls.

E51 **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC AO Update Channel



The ADAC AO Update Channel VI performs an immediate, **untimed** output on the specified DAC Element in the specified data units.

I32 **Board ID (0)** Specifies the ADAC-LVi board ID number to be used in this acquisition. The ID numbers are defined in the configuration file ADAC-LVi.con located in the Windows main directory.

DBL **Data (0.00)** Data is a double value that specifies the DAC channel output value.

I32 **UNITS (VOLTS:0)** UNITS is a numeric value that sets the units in which the DATA output parameter is specified. The available options are:

0:VOLTS The voltage to output

1:RAWDATA The hardware register bits output

2:CURRENT The current to output

The default UNITS 0:VOLTS

I32 **Iteration (init:0)** Iteration controls when the ADAC-LVi device subsystem allocation and device initialization is performed. If iteration is 0, the VI allocates the device and initializes it with the hardware specifications connected to the VI. The VI then starts the acquisition and reads the data. If iteration is greater than zero, then VI assumes that the current configuration is valid and starts the acquisition, then reads the data.

ADAC AO Update Channel (cont.)

TF **Clear Acquisition (true)** Clear acquisition determines whether the VI releases the device after reading the specified number of samples. The VI should pass a TRUE to this parameter when reading the last set of samples for a given acquisition. The default is TRUE, which means that the VI reads data only once if this input remains unwired. You typically wire this input to the terminating condition of a loop, so that when the loop finishes the device is released.

I32 **DATA CODE (TWOS_COMP:0)** DATA CODE is a numeric value that sets the hardware data coding configuration of the device.
The available options are:

- 0:TWOS_COMPLEMENT Two's Complement data coding
- 1:STRAIGHT_BINARY Straight Binary data coding
- 2:OFFSET_BINARY Offset Binary data coding

The default DATA CODE is 0: TWOS_COMPLEMENT

I32 **DATA OFFSET (BIPOLAR:0)** DATA OFFSET is a numeric value that sets the hardware data offset configuration of the device.
The available options are:

- 0:BIPOLAR Bipolar data +/- V
- 1:UNIPOLAR Unipolar data 0-V

The default DATA OFFSET is 0: BIPOLAR

I32 **DATA RANGE (-10/10:0)** DATA RANGE is a numeric value that sets the hardware voltage data range of the device.

The available options are:

- 0:-5_5 +/- 5V
- 1:0_10 + 0-10V
- 2:-10_10 +/- 10V

The default DATA OFFSET is 2:-10_10

ADAC AO Update Channel (cont.)

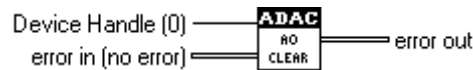
575 **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

U16 **Element (0)** Element is a numeric value that specifies the DAQ hardware subsystems DAC number. For a two DAC board the first DAC would be Element 0 and the second DAC would be element 1.

The default Element is 0.

I32 **Device Handle out** Device Handle is a numeric value that identifies the board's D/A subsystem to be acted upon in subsequent VI subsystem calls.

575 **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC AO Clear

This VI stops, frees all allocated resources and releases an D/A subsystem associated with the specified Device Handle. Once called, the Device Handle should be considered invalid, and not use in any subsequent subsystem calls.




This VI calls AL_ReleaseDevice to release the subsystem even if the error in cluster contains an error, or if any VIs called result in an error.

Before another acquisition can begin, you must call ADAC AO Config again to reallocate the device subsystem.

v Note

This VI will automatically release the ADAC-LVi environment if the specified Device Handle is the last valid ADAC-LVi Device Handle available. This behavior is controlled by the AutoLoad=YES setting in the ADAC-LVi.con file located in the main Windows directory.

ADAC AO Clear (con't)

-  **Device Handle (0)** Device Handle is a numeric value that identifies the board's A/D subsystem to be acted upon. The default Device Handle is 0.
-  **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
-  **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

Analog Output Examples

Examples for each Easy Analog Output VIs are provided, and located in the **ADAC Examples** in your LabVIEW directory. Example programs all provide a dialog box allowing the selection of any ADAC board installed in ADAC-LVi. If a board incapable of performing a particular example is selected, an error will occur indicating non-support.

The **Easy AO Update Channel** example demonstrates the use of ADAC AO Update Channel VI. The VI writes the specified voltage to a single analog output channel each time the UPDATE button is pressed.

The Digital I/O VIs

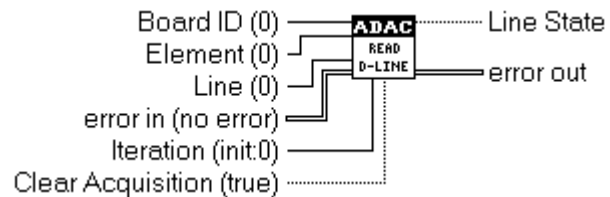
The Digital I/O VIs are used to perform digital input and output functions.

To access the ADAC Digital I/O VIs, choose **Functions >> User Libraries >> ADAC >> Digital I/O**.

- ADAC Read D(igital) Line
- ADAC Read D(igital) Port
- ADAC 5600 Config
- ADAC DIO Config
- ADAC DIO Read
- ADAC Write to D(igital) Line
- ADAC (DIO) Start Device
- ADAC DIO Clear
- ADAC Write to D(igital) Port

For advanced digital I/O functions, see the ADAC Digital I/O Library under the ADAC Advanced System VIs.

ADAC Read From D(igital) Line



The ADAC AI Read from Digital Line performs an immediate, **untimed** acquisition on the specified Digital port bit and returns the bit state acquired in a true(set) / false(clear) boolean value.

ADAC Read From D(igital) Line (con't.)

I32 **Board ID (0)** Specifies the ADAC-LVi board ID number to be used in this acquisition. The ID numbers are defined in the configuration file ADAC-LVi.con located in the Windows main directory.

U16 **Element (0)** Element is a numeric value that specifies element number of the Subsystem Type. Elements range from 0-n for all Subsystem Types.

The DIN, DOT typically have multiple elements, the first subsystem element is 0, the second is 1, and so forth.

The default Element is 0.

U16 **Line (0)** Line specifies the individual port bit or line to be used for I/O.

I32 **Iteration (init:0)** Iteration controls when the ADAC-LVi device subsystem allocation and device initialization is performed. If iteration is 0, the VI allocates the device and initializes it with the hardware specifications connected to the VI. The VI then starts the acquisition and reads the data. If iteration is greater than zero, then VI assumes that the current configuration is valid and starts the acquisition, then reads the data.

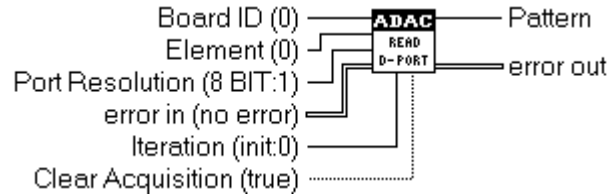
TF **Clear Acquisition (true)** Clear acquisition determines whether the VI releases the device after reading the specified number of samples. The VI should pass a TRUE to this parameter when reading the last set of samples for a given acquisition. The default is TRUE, which means that the VI reads data only once if this input remains unwired. You typically wire this input to the terminating condition of a loop, so that when the loop finishes the device is released.

EB **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

TF **Line State** Line State is a boolean value that contains the current digital line bit status.

EB **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Read D(igital) Port



The ADAC AI Read from Digital Port performs an immediate, **untimed** acquisition on the specified Digital port element and returns the data acquired in binary data units.

I32 **Board ID (0)** Specifies the ADAC-LVi board ID number to be used in this acquisition. The ID numbers are defined in the configuration file ADAC-LVi.con located in the Windows main directory.

U16 **Element (0)** Element is a numeric value that specifies element number of the Subsystem Type. Elements range from 0-n for all Subsystem Types.

The DIN, DOT typically have multiple elements, the first subsystem element is 0, the second is 1, and so forth.

The default Element is 0.

I32 **Port Resolution (8 BIT:1)** Port Resolution is a numeric value that whether the input or output port is read or written in 4, 8 or 16 bit transfers. The available options are:

0:4 BIT
1:8 BIT
2:16 BIT

The default value is 1:8 BIT

I32 **Iteration (init:0)** Iteration controls when the ADAC-LVi device subsystem allocation and device initialization is performed. If iteration is 0, the VI allocates the device and initializes it with the hardware specifications connected to the VI. The VI then starts the acquisition and reads the data. If iteration is greater than zero, then VI assumes that the current configuration is valid and starts the acquisition, then reads the data.

ADAC Read D(igital) Port (con't)

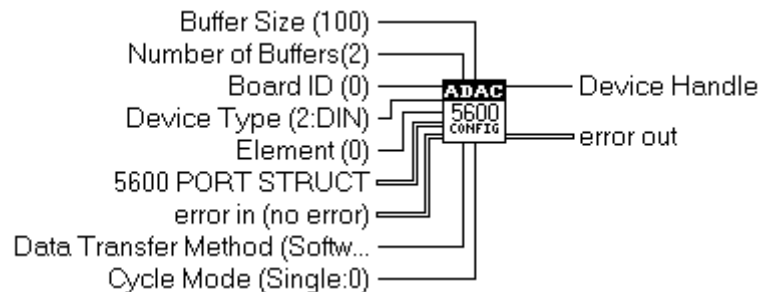
- TF** **Clear Acquisition (true)** Clear acquisition determines whether the VI releases the device after reading the specified number of samples. The VI should pass a TRUE to this parameter when reading the last set of samples for a given acquisition. The default is TRUE, which means that the VI reads data only once if this input remains unwired. You typically wire this input to the terminating condition of a loop, so that when the loop finishes the device is released.

- 5.0** **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

- U16** **Pattern** Pattern is a binary value that contains the current port data.

- 5.0** **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC 5600 Config



The ADAC 5600 DIO Config VI allocates and configures a device's digital input port (DIN 0-1) or DOT digital output port (DOT 0-3) subsystem returning a Device Handle to be used in subsequent VI calls. This VI is only used with ADAC **5600 Series** boards.

If the error in cluster contains an error from a previous VI call this VI does nothing and returns the error in cluster unmodified in the error out. In the event of a previous error the Device Handle is set to -1.

ADAC 5600 Config (con't.)

I32 **Board ID (0)** Specifies the ADAC-LVi board ID number to be used in this acquisition. The ID numbers are defined in the configuration file ADAC-LVi.con located in the Windows main directory.

U16 **Device Type (2:DIN)** Subsystem Type is a numeric value that specifies the board subsystem to be allocated. The available options are:

2:DIN Digital Input
3:DOT Digital Output

The default Subsystem Type is 2:DIN

U16 **Element (0)** Element is a numeric value that specifies the DIN or DOT port number to be allocated. The 5600 ports range from DIN0 to DIN1 and DOT0 to DOT3.

Note that DIN1 is not available when DIN0 is configured for 16 bit Port Resolution and DOT1 and DOT3 are not available when DOT0 or DOT2 is configured for 16 bit Port Resolution respectively.

The default Element is 0.

I32 **Data Transfer Method (Software:0)** Data Transfer Method is a numeric value that sets the hardware transfer method. Only 5600 series cards with DIN0 Interrupt capability can be configured for Interrupt operation. For all other DIN and DOT ports Data Transfer Method must be set to SOFTWARE, only single read or write operations are available and the 5600 PORT STRUCT cluster is not used. The available options are:

0:SOFTWARE Data is transferred via a single read or write operation. Supported by all 5600 ports.

1:DMA Not available on the 5600 series.
2:IRQ Data is transferred via ISR, supported by DIN0 only.

The default Data Transfer Method is 0:SOFTWARE.

5600 PORT STRUCT

U16 **Port Resolution** Port Resolution is a numeric value that represents the desired number of DIO port bits, either 8 or 16. The DIO ports DIN0 can be either 8 or 16 bit ports and DIN1 can only be an 8 bit port. When DIN0 is set for 16 bit resolution DIN1 is not available. The DIO ports DOT0 and DOT2 can be either 8 or 16 bit ports and DOT1 and DOT3 can only be an 8 bit port. When DOT0 or DOT2 is set for 16 bit resolution DOT1 or DOT 3 is not available respectively.

ADAC 5600 Config (con't.)

I32 **Port Mask** Port Mask is a binary value that indicates which bits are to be read during interrupt transfers. If the associated bit in the Port Mask control is set = "1" the bit is read, inactive bits will return "0".

I32 **A5600PORT**

I32 **Debounce** Debounce is a numeric value that indicates if the device's debounce circuitry is enabled or disabled. When Enabled the Time Constant value sets the time required for a stable input.

0:DISABLED Disable Debounce

1:ENABLED Enable Debounce

The default value is 1:ENABLED

I32 **Port Mode Specification** Port Mode Specification defines the pattern match condition.

0:DISABLED Pattern match is disabled.

1:AND All unmasked bit conditions must occur.

2:OR Any unmasked bit conditions must occur.

The default Port Mode Specification is 2:OR

I32 **Pattern Latch** Pattern Latch controls the Latch On Pattern ability. If the Pattern Latch is enabled, the state of the input port will be latched when a match condition occurs. The data will be held until the port is read within its interrupt handler. If a subsequent match occurs (port goes from match condition to non-match condition to a new match condition) before the port is read within its interrupt handler, an IP error condition will occur stopping the acquisition. The Ignore IP Error control can be set to TRUE to disable the IP error checking in the driver, but data may be lost when match conditions are occurring faster than they can be serviced.

I32 **Data Path Polarity** Data Path Polarity allows each bit for DIN0 to be individually inverted. If the associated bit is set "1", data read from that bit on DIN0 will be inverted. For detailed definitions see the 5600 manuals ADVANCED PROGRAMMING TECHNIQUES. When the Port Resolution is set for 16 bit mode, Data Path Polarity is set in WORDs b(1111111111111111).

ADAC 5600 Config (con't.)

I32 **Special IO Control** The SpecialIoControl controls the BIT CATCHER operation mode. Port DIN0 may be configured to return the state of the bit catcher, instead of the actual port data by setting the associated bit to "1" in the Special Io Control. If a bit is so programmed, the port will return "0" for that bit until a "1" occurs at the input. The port will then read back a "1" even if the "1" goes away. When the DIN0 port is serviced by the driver, each bit enabled and in the "1" state will read automatically reset and ready for the next input transition. For detailed definitions see the 5600 manuals ADVANCED PROGRAMMING TECHNIQUES. When the Port Resolution is set for 16 bit mode, Special Io Control is set in WORDs b(1111111111111111).

I32 **Pattern Polarity** The PatternPolarity, PatternTransition and PatternMask define the match condition for the Port Mode Specification AND / OR modes.

For example one can define a match condition as bit 1, bit 2 and bit 3 all set = "1" as a valid match (AND mode) by setting the following.

Port Mode Specification = AND
 PatternPolarity = (b) 1110
 PatternTransition = (b) 0
 PatternMask = (b) 1110

Alternately, one can define a match condition as any one or more of bit 1, bit 2 and bit 3 having set = "1" by changing the Port Mode Specification to OR mode.

PM(n)	PT(n)	PP(n)	Pattern Match Bit Definitions
0	0	0 or 1	Bit Masked Off
0	1	0 or 1	Any Transition
1	0	0	ZERO
1	0	1	ONE
1	1	0	ONE to ZERO Transition
1	1	1	ZERO to ONE Transition

For detailed definitions see the 5600 manuals ADVANCED PROGRAMMING TECHNIQUES. When the PortResolution=16, PatternPolarity, PatternTransition and PatternMask are set in WORDs b(1111111111111111).

ADAC 5600 Config (con't.)



Pattern Transition The PatternPolarity, PatternTransition and PatternMask define the match condition for the Port Mode Specification AND / OR modes.

For example one can define a match condition as bit 1, bit 2 and bit 3 all set = "1" as a valid match (AND mode) by setting the following.

Port Mode Specification = AND
 PatternPolarity = (b) 1110
 PatternTransition = (b) 0
 PatternMask = (b) 1110

Alternately, one can define a match condition as any one or more of bit 1, bit 2 and bit 3 having set = "1" by changing the Port Mode Specification to OR mode.

PM(n)	PT(n)	PP(n)	Pattern Match Bit Definitions
0	0	0 or 1	Bit Masked Off
0	1	0 or 1	Any Transition
1	0	0	ZERO
1	0	1	ONE
1	1	0	ONE to ZERO Transition
1	1	1	ZERO to ONE Transition

For detailed definitions see the 5600 manuals ADVANCED PROGRAMMING TECHNIQUES. When the PortResolution=16, PatternPolarity, PatternTransition and PatternMask are set in WORDs b(1111111111111111).



Pattern Mask The PatternPolarity, PatternTransition and PatternMask define the match condition for the Port Mode Specification AND / OR modes.

For example one can define a match condition as bit 1, bit 2 and bit 3 all set = "1" as a valid match (AND mode) by setting the following.

Port Mode Specification = AND
 PatternPolarity = (b) 1110
 PatternTransition = (b) 0
 PatternMask = (b) 1110

Alternately, one can define a match condition as any one or more of bit 1, bit 2 and bit 3 having set = "1" by changing the Port Mode Specification to OR mode.

ADAC 5600 Config (cont.)

PM(n)	PT(n)	PP(n)	Pattern Match Bit Definitions
0	0	0 or 1	Bit Masked Off
0	1	0 or 1	Any Transition
1	0	0	ZERO
1	0	1	ONE
1	1	0	ONE to ZERO Transition
1	1	1	ZERO to ONE Transition

For detailed definitions see the 5600 manuals ADVANCED PROGRAMMING TECHNIQUES. When the PortResolution=16, PatternPolarity, PatternTransition and PatternMask are set in WORDs b(1111111111111111).

I32 **Ignore IP Error** The Ignore IP Error setting controls the ADAC-LVi driver pattern match error condition. A setting of TRUE stops the generation of an error condition when a match condition occurs while another match is being serviced, indicating match conditions are occurring faster than they can be serviced.

I32 **Time Constant** The Time Constant value sets the time required for a stable input. Use the following equation to determine the value.

$$\text{TimeConstant} = \text{period}(\text{usec}) / 2(\text{usec})$$

Where period(usec) is the debounce time required.

For detailed definitions see the 5600 manuals ADVANCED PROGRAMMING TECHNIQUES.

I32 **Number of Buffers(2)** Specifies the number of buffers the VI allocates. Buffers are only when DIN0 is configured for Interrupt acquisitions.

I32 **Buffer Size (100)** Buffer Size is the number of data samples you each buffer to holds. Buffers Size is only set when DIN0 is configured for Interrupt acquisitions.

I32 **Cycle Mode (Single:0)** Cycle Mode is a numeric value that sets the software cycle operation mode. Data can be collected in a single scan (SINGLE_CYCLE) filling buffer 0-n and then stopping acquisition or collected continuously (CONTINUOUS_CYCLE) filling buffers 0-n repeatedly.

For DIO ports that do not support Interrupts, Cycle Mode must be set to SINGLE_CYCLE. Only single write operations are available and buffers are not used.

ADAC 5600 Config (con't)

See also the Data Transfer Method for DIO limitations. The available options are:

0:SINGLE_CYCLE Complete acquisition scan of 0-n buffers and stop. Supported by all DIO's.

1:CONTINUOUS_CYCLE Restart acquisition, repeatedly loop on 0-n buffers. Supported by DIN0 and DIN2 with Interrupt transfers.

The default Cycle Mode is 0:SINGLE_CYCLE.



error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

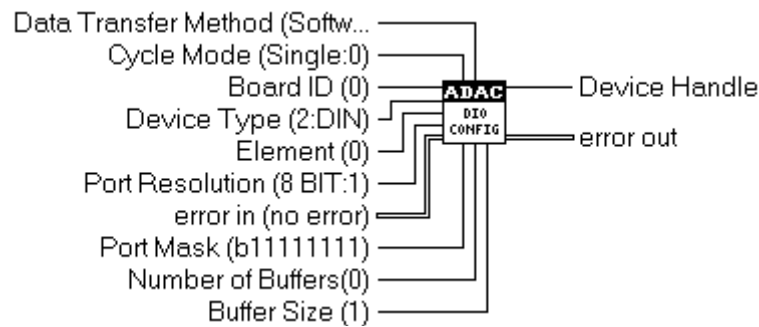


Device Handle Device Handle is a numeric value that identifies the board's DIO subsystem to be acted upon in subsequent VI subsystem calls.



error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC DIO Config



The ADAC DIO Config VI allocates and configures a device's digital input port (DIN 0-n) or DOT digital output port (DOT 0-n) subsystem returning a Device Handle to be used in subsequent VI calls.

ADAC DIO Config (con't.)

If the error in cluster contains an error from a previous VI call this VI does nothing and returns the error in cluster unmodified in the error out. In the event of a previous error the Device Handle is set to -1.



Board ID (0) Specifies the ADAC-LVi board ID number to be used in this acquisition. The ID numbers are defined in the configuration file ADAC-LVi.con located in the Windows main directory.



Device Type (2:DIN) Subsystem Type is a numeric value that specifies the board subsystem to be allocated. The available options are:

2:DIN Digital Input
3:DOT Digital Output

The default Subsystem Type is 2:DIN



Element (0) Element is a numeric value that specifies element number of the Subsystem Type. Elements range from 0-n for all Subsystem Types.

The DIN, DOT typically have multiple elements, the first subsystem element is 0, the second is 1, and so forth.

The default Element is 0.



Cycle Mode (Single:0) Cycle Mode is a numeric value that sets the software cycle operation mode. Data can be collected in a single scan (SINGLE_CYCLE) filling buffer 0-n and then stopping acquisition or collected continuously (CONTINUOUS_CYCLE) filling buffers 0-n repeatedly.

For DIO ports that do not support DMA or Interrupts Cycle Mode must be set to SINGLE_CYCLE. Only single read operations are available. The available options are:

0:SINGLE_CYCLE Complete acquisition scan of 0-n buffers and stop. Supported by all DIO's.

1:CONTINUOUS_CYCLE Restart acquisition, repeatedly loop on 0-n buffers. Supported by DIO's with DMA, Interrupt or FIFO capabilities.

The default Cycle Mode is 0:SINGLE_CYCLE.

ADAC DIO Config (con't.)

I32 **Data Transfer Method (Software:0)** Data Transfer Method is a numeric value that sets the hardware transfer method. This parameter specifies how data from the DIO is to be transferred to buffers.

For DIO ports that do not support DMA or Interrupts Data Transfer Method must be set to SOFTWARE. Only single read operations are available. The available options are:

- 0:SOFTWARE Data is transferred via a software controlled loop, supported by all DIO's.
- 1:DMA Data is transferred via DMA, supported by DIO's with DMA, capabilities.
- 2:IRQ Data is transferred via ISR, supported by DIO's with Interrupt, capabilities.

The default Data Transfer Method is 0:SOFTWARE.

I32 **Port Resolution (8 BIT:1)** Port Resolution is a numeric value that whether the input or output port is read or written in 4, 8 or 16 bit transfers. The available options are:
0:4 BIT
1:8 BIT
2:16 BIT

The default value is 1:8 BIT

I32 **Port Mask (b1111111)** Port Mask is a binary value that selects the operation bits. All bits set to one(1) in the Port Mask will acted upon during Interrupt or DMA transfers. Port Mask is only configured when Data Transfer Method is set for DMA or Interrupt acquisitions.

The default value is binary 11111111.

I32 **Number of Buffers(0)** Specifies the number of buffers the VI allocates. Buffers are only configured when Data Transfer Method is set for DMA or Interrupt acquisitions.

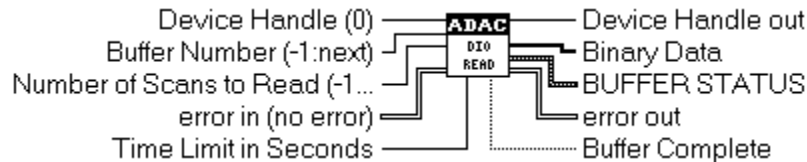
I32 **Buffer Size (1)** Buffer Size is the number of data samples you each buffer to holds. Buffer Size is only configured when Data Transfer Method is set for DMA or Interrupt acquisitions.

E7 **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

I32 **Device Handle** Device Handle is a numeric value that identifies the board's DIO subsystem to be acted upon in subsequent VI subsystem calls.

E7 **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC DIO Read



This VI reads data from an queued DIO buffer and returns the data acquired in binary data units.

The BUFFER STATUS cluster is useful in determining the current status of the acquisition by presenting the Buffer DoneFlag, StatusFlag and ErrorFlag conditions as described below. The BUFFER STATUS is especially useful in multi-buffered/continuous acquisition where no interaction takes place except from incoming buffers.

Before reading any data, DIO Read checks to see if the input cluster error in indicates that an error has already occurred. If so, this VI does not read any data, but passes the error information unmodified through error out.



Device Handle (0) Device Handle is a numeric value that identifies the board's DIO subsystem to be acted upon. The default Device Handle is 0.



Buffer Number (-1:next) Buffer Number is the ID number of the buffer to be obtained. The Buffers are numbered from 0-n as specified in the ADAC DIO Config VI. The default input is -1, which tells ADAC-LVi to return the next available buffer in the done queue. For multi-buffered or continuous acquisitions, setting Buffer Number to -1 provides the easiest method of obtaining the next buffer available in the done queue. Queued buffers are returned in a first done first available fashion.



Number of Scans to Read (-1:all) Number of Scans to Read specifies the number of scans the VI retrieves from the acquisition buffer. The default input is -1, which tells LabVIEW to set number of scans to read equal to the value of the DAQ buffer size. Once a buffer is successfully read into LabView the buffer is placed back into the device's available buffer queue. If the Number of Scans to Read is less than the actual buffer size, all unread data will be lost.



Time Limit in Seconds (1.00) Time Limit in Seconds specifies the maximum length of time this VI waits for a done buffer. You express time limit in seconds. If this VI does not receive a completed buffer status prior to the timeout period, the VI returns an error.



error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

ADAC DIO Read (con't.)

- I32** **Device Handle out** Device Handle out contains the value of Device Handle in.
- U16** **Binary Data** Binary Data is a two-dimensional array containing unscaled analog input data. The data appears in columns, where each column contains the data for a single channel. The second (or bottom) dimension selects the channel. The first (or top) dimension selects a single data point for that channel. This array is empty if Output Units is set to scaled.
- U05** **BUFFER STATUS** ADAC-LVi buffers are not just arrays of data to a DAQ board's subsystem data, but are structures that contains the buffer type, size, data array, current sample count, hardware trigger points, completion flags, status flags, and error flags. Some of these fields remain constant for the duration of the acquisition. Others contain relevant information about the current condition of the buffer, and error conditions that can occurred during the current acquisition.

The three structure variables that provide state/status/error information are the DoneFlag, StatusFlag and ErrorFlags.

- U32** **BufferType** [U32] BufferType
Data Buffer type, DMA, Interrupt or Polling.
- U32** **BufferSize** [U32] BufferSize
Size of the buffer in bytes.
- U32** **BufferWrite** [U32] BufferWrite
Number of data samples in the buffer.
- U32** **BufferRead** [U32] BufferRead
Number of data samples obtained from a buffer.
- U32** **TriggerPoint** [U32] TriggerPoint
Buffer position number of the trigger point.
- U32** **TiggerPointStart** [U32] TriggerPointStart
Buffer position number of the pre-trigger data.
- U32** **DoneFlag** [U32] DoneFlag:
The Done Flag specifies the current state of a buffer. This flag can be set to any of the three following conditions.

BUFFER_IDLE = 0 The buffer is available for use, but acquisition has not started on this buffer.

BUFFER_DONE = 1 The buffer has been filled with samples and is available

ADAC DIO Read (con't.)

for use in the user application.

BUFFER_INUSE = 2 The buffer is currently being filled with samples

U32**StatusFlag** [U32] StatusFlag:

The Status Flag specifies the completion status of a buffer. This flag can be set to any of the four following conditions.

BUFFER_EMPTY = 1 No samples are available in the buffer.

BUFFER_INCOMPLETE = 2 The output buffer has not read completely or and input has not filled to capacity.

BUFFER_COMPLETE = 4 The output buffer has been read completely.

BUFFER_FULL = 8 The input buffer has filled to capacity.

I32

ErrorFlag [i32] ErrorFlags: The Error Flags specify the condition which has stopped a buffer from successfully completing. Although an error may be reported, this does not indicate the BUFFER_FULL flag is not set. This would be the case if an error was detected and enough samples were available to fill the buffer to capacity. One or more of these flags may also be set together. If any of these flags have set, checking the current hardware error conditions will usually reveal the actual condition that caused the acquisition runtime error. This flag can be set to any of the four following conditions.

BUFFER_STOPPED = 1 An error occurred and the buffer(s) are no longer being serviced.

BUFFER_OVERRUN = 2 An error occurred when attempting to place the next sample beyond the bounds of the buffer.

BUFFER_UNDERRUN = 4 An error occurred and the buffer is incomplete.

BUFFER_NEXTBUSY = 8 When attempting to access the next buffer, the current state of the DoneFlag indicated the buffer was not yet available for use.

U32**BufferNum** [U32] BuffNum

Buffer Number ID (0-n)

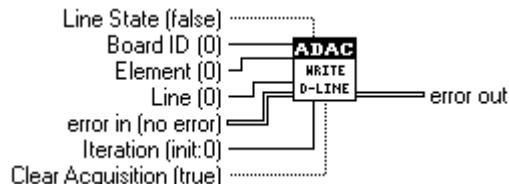
TF

Buffer Complete Buffer Complete is TRUE if this VI was able to successfully obtain a done buffer. A completed buffer may only indicate the buffer is no longer being serviced and the status is available, always check the StatusFlag and ErrorFlag.

Err

error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Write to D(igital) Line



The ADAC Write To Digital Line performs an immediate, **untimed** acquisition on the specified digital port bit and returns the bit state acquired in a true(set) / false(clear) boolean value.

I32 **Board ID (0)** Specifies the ADAC-LVi board ID number to be used in this acquisition. The ID numbers are defined in the configuration file ADAC-LVi.con located in the Windows main directory.

U16 **Element (0)** Element is a numeric value that specifies element number of the Subsystem Type. Elements range from 0-n for all Subsystem Types.

The DIN, DOT typically have multiple elements, the first subsystem element is 0, the second is 1, and so forth.




The default Element is 0.

U16 **Line (0)** Line specifies the individual port bit or line to be used for I/O.

I32 **Iteration (init:0)** Iteration controls when the ADAC-LVi device subsystem allocation and device initialization is performed. If iteration is 0, the VI allocates the device and initializes it with the hardware specifications connected to the VI. The VI then starts the acquisition and reads the data. If iteration is greater than zero, then VI assumes that the current configuration is valid and starts the acquisition, then reads the data.

TF **Clear Acquisition (true)** Clear acquisition determines whether the VI releases the device after reading the specified number of samples. The VI should pass a TRUE to this parameter when reading the last set of samples for a given acquisition. The default is TRUE, which means that the VI reads data only once if this input remains unwired. You typically wire this input to the terminating condition of a loop, so that when the loop finishes the device is released.

ADAC Write to D(igital) Line (con't)





-  **Line State (false)** (boolean) line state: The position of this boolean switch controls the setting or clearing of the line (or bit) at the port. If the line state switch is slid down to the off (or false) position the line is cleared. If the line state switch is slid up to the on (or true) position the line is set.
-  **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
-  **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC (DIO) Start Device



This VI starts digital I/O acquisition on the specified Device Handle. All configuration of the device must be completed before calling this VI.

Depending on the current ADAC-LVi transfer mode this call may not return until completed. See ADAC DIO Config VI for detailed information on asynchronous operations.

-  **Device Handle (0)** Device Handle is a numeric value that identifies the board's DIO subsystem to be acted upon. Start Device is only available to DIO subsystems configured for DMA or interrupt operations. The default Device Handle is 0.
-  **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
-  **Device Handle out** Device Handle out contains the value of Device Handle in.
-  **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC DIO Clear



This VI stops, frees all allocated resources and releases an DIO subsystem and association with the specified Device Handle. Once called the Device Handle should be considered invalid and not use in any subsequent subsystem calls.

This VI calls AL_StopDevice and AL_ReleaseDevice to stop and release the subsystem even if the error in cluster contains an error or if any VIs called result in an error.

Before another acquisition can begin, you must call ADAC DIO Config again to reallocate the device subsystem.

- v **Note** *This VI will automatically release the ADAC-LVi environment if the specified Device Handle is the last valid ADAC-LVi Device Handle available. This behavior is controlled by the AutoLoad=YES setting in the ADAC-LVi.con file located in the main Windows directory.*



Device Handle (0) Device Handle is a numeric value that identifies the board's DIO subsystem to be acted upon. The default Device Handle is 0.

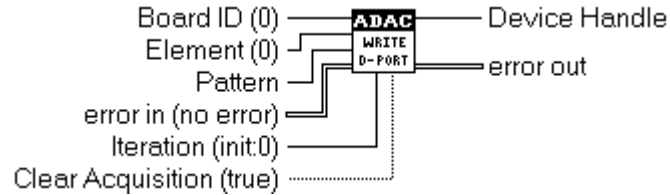


error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.



error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Write to D(igital) Port



The ADAC AI write to Digital Port performs an immediate, **untimed** acquisition on the specified Digital port element with the output pattern data.

I32 **Board ID (0)** Specifies the ADAC-LVi board ID number to be used in this acquisition. The ID numbers are defined in the configuration file ADAC-LVi.con located in the Windows main directory.

U16 **Element (0)** Element is a numeric value that specifies element number of the Subsystem Type. Elements range from 0-n for all Subsystem Types.

The DIN, DOT typically have multiple elements, the first subsystem element is 0, the second is 1, and so forth.

The default Element is 0.

I32 **Port Resolution (8 BIT:1)** Port Resolution is a numeric value that whether the input or output port is read or written in 4, 8 or 16 bit transfers. The available options are:





0:4 BIT
1:8 BIT
2:16 BIT

The default value is 1:8 BIT

U16 **Pattern** Pattern is a binary value that contains the output port data.

I32 **Iteration (init:0)** Iteration controls when the ADAC-LVi device subsystem allocation and device initialization is performed. If iteration is 0, the VI allocates the device and initializes it with the hardware specifications connected to the VI. The VI then starts the acquisition and reads the data. If iteration is greater than zero, then VI assumes that the current configuration is valid and starts the acquisition, then reads the data.

ADAC Write to D(igital) Port (cont.)

-  **Clear Acquisition (true)** Clear acquisition determines whether the VI releases the device after reading the specified number of samples. The VI should pass a TRUE to this parameter when reading the last set of samples for a given acquisition. The default is TRUE, which means that the VI reads data only once if this input remains unwired. You typically wire this input to the terminating condition of a loop, so that when the loop finishes the device is released.
-  **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
-  **Device Handle** Device Handle is a numeric value that identifies the board's DIO subsystem to be acted upon in subsequent VI subsystem calls.
-  **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

Digital I/O Examples

Examples for each Easy Digital Input VIs are provided, and located in the **ADAC Examples** in your LabVIEW directory. Example programs all provide a dialog box allowing the selection of any ADAC board installed in ADAC-LVi. If a board incapable of performing a particular example is selected, an error will occur indicating non-support.

The **Easy DIO Read Port** example demonstrates the use of ADAC Read from Digital Port VI. The example continuously reads a digital port in loop until the EXIT button is pressed.

The **Easy DIO Write Port** example demonstrates the use of ADAC Read from Digital Port VI. This VI writes the specified binary value to a single digital output port when the UPDATE button is pressed.

The **Advanced 5600DIO Acq Port** example demonstrates the use of the 5600 Series PORT STRUCT. This VI shows how to use the 5600 Series PORT STRUCT. It uses the 5600 board's interrupt capabilities to catch digital input lines changing state on each of the 16 digital input lines.

An occurrence is used for a continuous asynchronous acquisition. This is a non-timed acquisition, meaning that no clocking is used to control the acquisition rate. It is also a continuous circular buffered acquisition. This means that a software buffer is used between your DAQ board and LabVIEW. While data is being transferred from your board into one of the input buffers, LabVIEW is reading data from another buffer. The occurrence VI makes the acquisition asynchronous (allowing processor time for other things to run) by causing the loop to sleep until notified a buffer is available which is then read into LabVIEW.

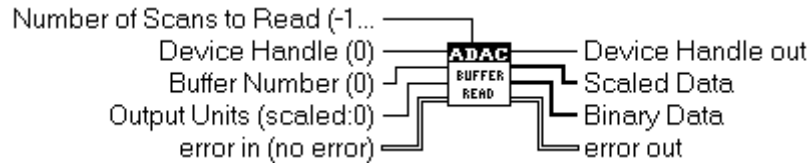
The Miscellaneous VIs

The Miscellaneous VIs are common VIs used by the Analog Input, Analog Output and Digital I/O VIs.

To access the ADAC Analog Output VIs, choose **Functions >> User Libraries >> ADAC >> Misc.**

- ADAC Buffer Read
- ADAC Occurrence Config
- ADAC Enum(erate) Boards
- ADAC Get Board Selections

ADAC Buffer Read



This VI reads data from a buffered data acquisition.

Before reading any data, Buffer Read checks to see if the input cluster error in indicates that an error has already occurred. If so, this VI does not read any data, but passes the error information unmodified through error out. Otherwise, this VI reads the specified amount of data from a buffered analog input acquisition. See also `AL_GetBufferStatus.vi`.

I32 **Device Handle (0)** Device Handle is a numeric value that identifies the board's subsystem to be acted upon. The default Device Handle is 0.

U32 **Buffer Number (0)** Buffer Number: is the ID number of the buffer(0-n) to be obtained from the devices internal subsystem.

ADAC Buffer Read (con't.)

[I32] **Number of Scans to Read (-1:all)** Number of Scans to Read specifies the number of scans the VI retrieves from the acquisition buffer. The default input is -1, which tells LabVIEW to set number of scans to read equal to the value of the DAQ buffer size. Once a buffer is successfully read into LabView the buffer is placed back into the devices' available buffer queue. If the Number of Scans to Read is less than the actual buffer size, all unread data will be lost.

[U16] **Output Units (scaled:0)** Output Units specifies whether the VI returns unscaled binary data or scaled voltage data.

0:Scaled Return scaled voltage data only (default setting). The binary data array appears empty.

1:Binary Return binary data only. The scaled data array appears empty.

2:Scaled & Binary Returns both scaled and binary data.

The VI executes faster if you select Binary data units, because the VI does not perform scaling.

[E5] **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

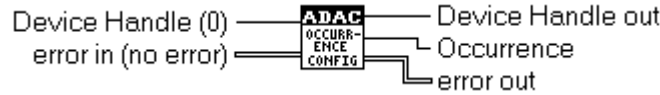
[I32] **Device Handle out** Device Handle out contains the value of Device Handle in.

[SGL] **Scaled Data** Scaled Data is a one-dimensional array containing analog input data in Volts. The data appears in successive array elements, where each element contains the data for a single channel.






[I16] **Binary Data** Binary Data is a one-dimensional array containing analog input data in raw data. The data appears in successive array elements, where each element contains the data for a single channel.

[E5] **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

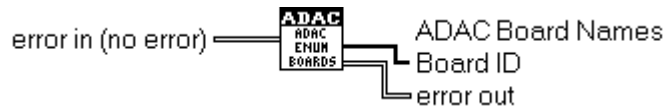
ADAC Occurrence Config



This VI creates a DAQ Event Occurrence. The output occurrence is wired to a wait on occurrence primitive which is used to provide asynchronous DAQ.





-  **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.
-  **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
-  **Device Handle out** Device Handle out contains the value of Device Handle in.
-  **Occurrence** occurrence is the occurrence value created. Wire this output to a wait on occurrence primitive. Wire the output of the primitive to the part of your diagram you wish to execute when the DAQ event happens and the occurrence is set. If LabVIEW is unable to create an occurrence, the occurrence value is set to the Not a Refnum file I/O constant.
-  **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Enum(erate) Boards

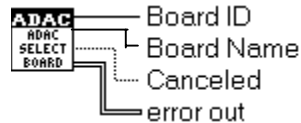


The ADAC Enum Boards VI reads in the ADAC-LVi.con file that contains the board names and ID numbers used to represent the DAQ boards. The information containing each board description is located in the ADAC-LVi.con configuration file located in your main Windows directory.

ADAC Enum(erate) Boards (cont.)




-  **error in (no error)** error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
-  **ADAC Board Names** ADAC Board Names contains ASCII strings representing all of the installed ADAC-LVi data acquisition boards.
-  **Board ID** Board ID contains the numeric board ID number that is used to identify the board in subsequent VI calls.
-  **error out** error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Get Board Selections



This VI presents a list of installed ADAC data acquisition boards and allows you to select the board you want to use.

The VI is set up to open its front panel when called and close when complete. The VI is closed when you click either the OK or Cancel button.

-  **Board Name** Board Name contains ASCII string representing the selected DAQ board.
-  **Board ID** Board ID contains the numeric board ID number that is used to identify the board in subsequent VI calls. If you click Cancel, Board ID will be set to -1.
-  **error out** error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

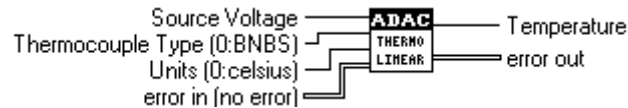
The Signal Conditioning VIs

The Signal Conditioning VIs are used convert analog input voltages read from resistance temperature detectors (RTDs), or thermocouples into units of temperature.

To access the ADAC Analog Output VIs, choose **Functions >> User Libraries >> ADAC >> Signal. Cond.**

- ADAC Thermo Linear (Convert Single Thermocouple Reading)
- ADAC TC Lin Buff (Convert Thermocouple Buffer)
- ADAC RTD Conv (Convert RTD Reading)

ADAC Thermo Linear (Convert Thermocouple Reading)



This VI Converts the specified thermocouple Source Voltage input array into the specified temperature units.



Source Voltage Source Voltage is a single value containing the input data voltage to be converted into temperature.



Thermocouple Type (0:BNBS) Thermocouple Type is a numeric value that specifies the NBS thermocouple polynomial equation for the conversion. The available options are:

0:BNBS	B-type thermocouple
1:KNBS	K-type thermocouple
2:SNBS	S-type thermocouple
3:TNBS	T-type thermocouple
4:JNBS	J-type thermocouple
5:RNBS	R-type thermocouple
6:ENBS	E-type thermocouple

The default Thermocouple Type is 0:BNBS

ADAC Thermo Linear (Convert Thermocouple Reading) (con't.)

Units (0:celsius) Units specifies the desired temperature units for the conversion. The available options are:

0:Celcius	Celsius degree
1:Fahrenheit	Fahrenheit degree
2:Kelvin	Kelvin degree
3:Rankine	Rankine

The default Units is 0:Celcius



error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

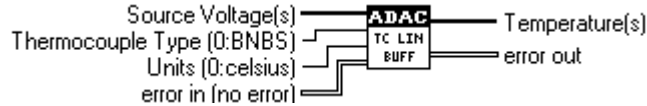


Temperature Temperature a single value containing the converted data in the specified temperature units.



error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC TC Lin Buff (Convert Thermocouple Buffer)



This VI Converts the specified thermocouple Source Voltages input array into the specified temperature units.

[SGL] **Source Voltage(s)** Source Voltage(s) is a one-dimensional array containing the input data in volts to be converted into temperature.

[U16] **Thermocouple Type (0:BNBS)** Thermocouple Type is a numeric value that specifies the NBS thermocouple polynomial equation for the conversion. The available options are:

0:BNBS	B-type thermocouple
1:KNBS	K-type thermocouple
2:SNBS	S-type thermocouple
3:TNBS	T-type thermocouple
4:JNBS	J-type thermocouple
5:RNBS	R-type thermocouple
6:ENBS	E-type thermocouple

The default Thermocouple Type is 0:BNBS

[U16] **Units (0:celsius)** Units specifies the desired temperature units for the conversion. The available options are:

0:Celcius	Celsius degree
1:Fahrenheit	Fahrenheit degree
2:Kelvin	Kelvin degree
3:Rankine	Rankine

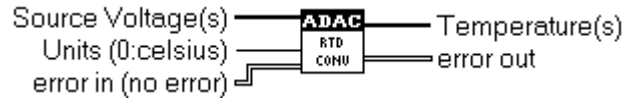
The default Units is 0:Celcius

[Err] **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

[SGL] **Temperature(s)** Temperature(s) is a one-dimensional array containing the converted data in the specified temperature units.

[Err] **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC RTD Conv (Convert RTD Reading)



This VI Converts the specified RTD Source Voltage(s) input array into temperature units.

[SGL] **Source Voltage(s)** Source Voltage(s) is a one-dimensional array containing the input data in volts to be converted into temperature.

[U16] **Units (0:celsius)** Units specifies the desired temperature units for the conversion. The available options are:

0:Celcius	Celsius degree
1:Fahrenheit	Fahrenheit degree
2:Kelvin	Kelvin degree
3:Rankine	Rankine

The default Units is 0:Celcius

[Err] **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

[SGL] **Temperature(s)** Destination buffer is a one-dimensional array containing the converted data in the specified temperature units.

[Err] **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

The Advanced System Function VIs - Program Flow

ADAC-LVi provides a complete set of VIs to allocate, configure, initialize and acquire data. All ADAC-LVi hardware devices have an associate **BoardID** that is set in the ADAC-LVi.CON file. In addition a given hardware device may have **subsystems** consisting of ADC0, DIN0, DIN1..., DOT0, DOT1 ..., DAC0 and DAC1... as defined in each device's capability file. The first step in setting up an ADAC device for use in LabVIEW is to allocate it. A device is allocated from the ADAC-LVi using the **BoardID** and **subsystem** type. Once allocated, the subsystem can then be configured using any of the available ADAC-LVi VIs within the subsystem's capabilities. Once the subsystem is configured, it would then be initialized and started.

The basic flow of the ADAC-LVi VIs are generic across all subsystems types as described above. The most important aspect of the flow is the **subsystem configuration**, this must be done before the subsystem is initialize and started. Two methods of subsystem configuration are available, the first known as the INI default setup which is configured when you first allocate your subsystem. The file ADAC-LVi.INI contains all available subsystem configurations that are read into your subsystem as the defaults. The second method of subsystem configuration is through the actual ADAC-LVi's VIs which are used to further configure your subsystem under program control.

The following flowcharts show the basic steps required to setup a device subsystem for use within LabView.

Basic ADAC-LVi Program Flow

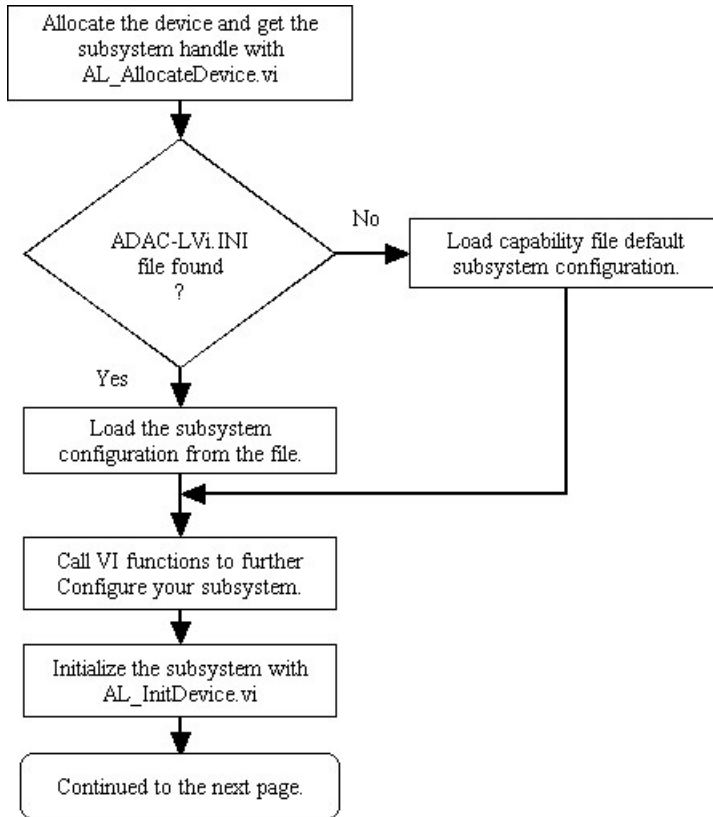


Figure 1: Basic Program Flow

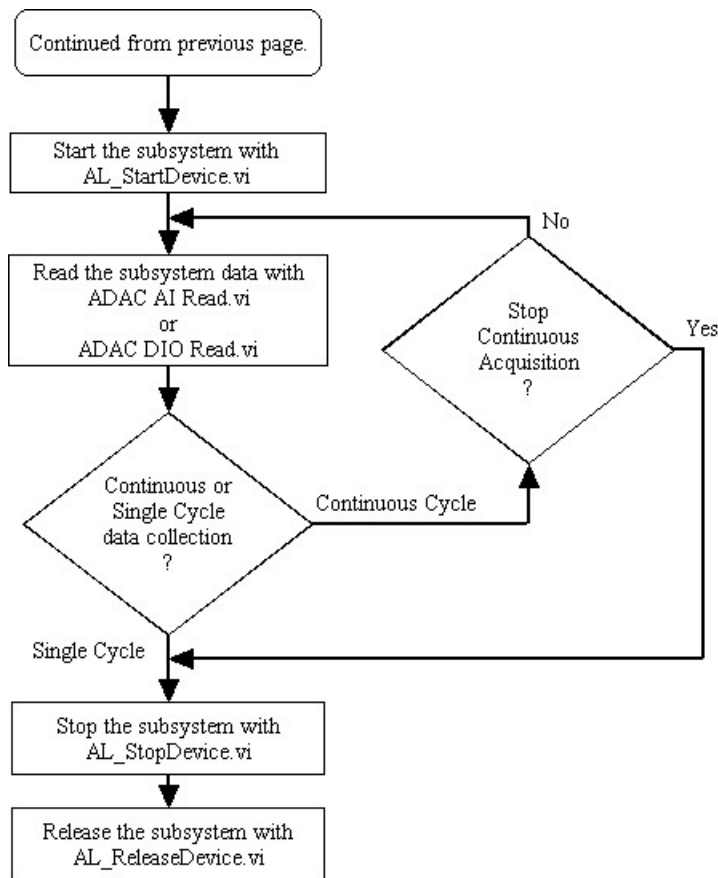


Figure 1: Basic Program Flow (cont.)

Continuous Asynchronous Program Flow

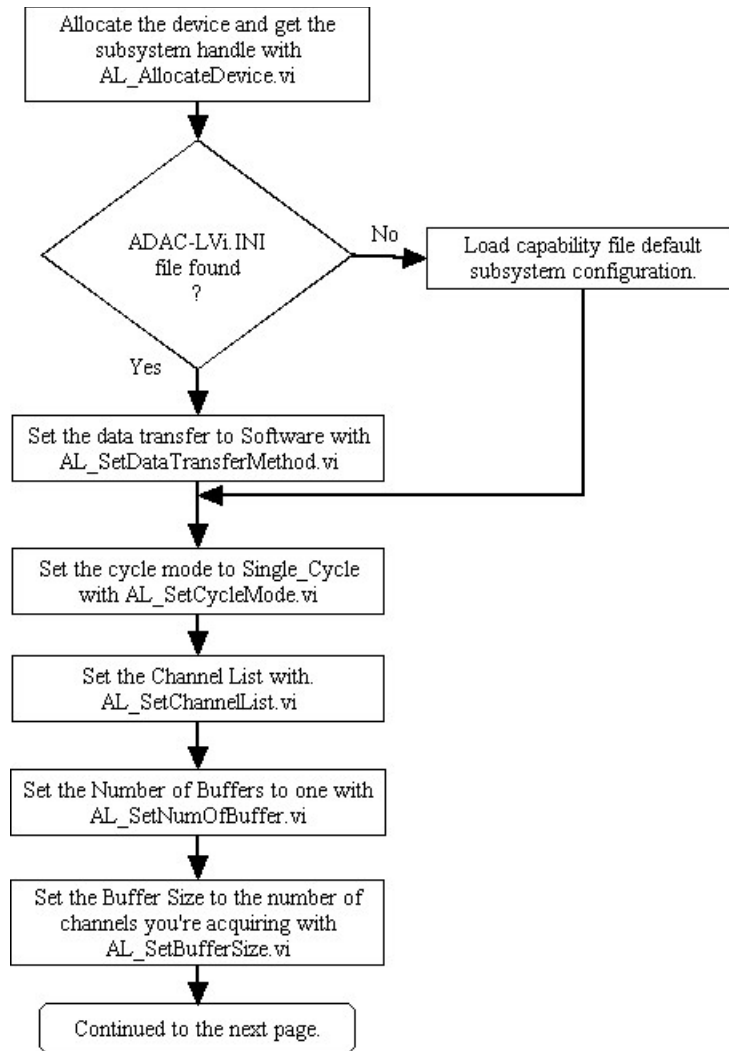


Figure 2: Continuous Asynchronous Program Flow

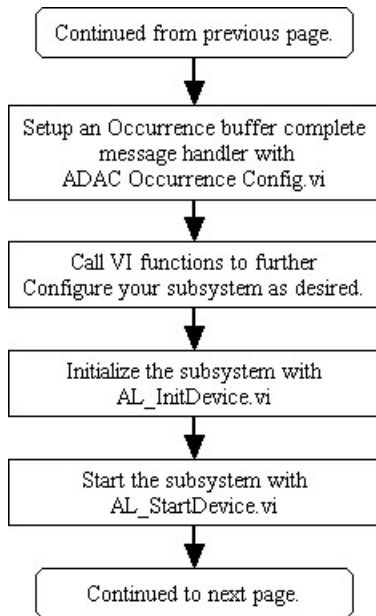


Figure 2: Continuous Asynchronous Program Flow (cont.)

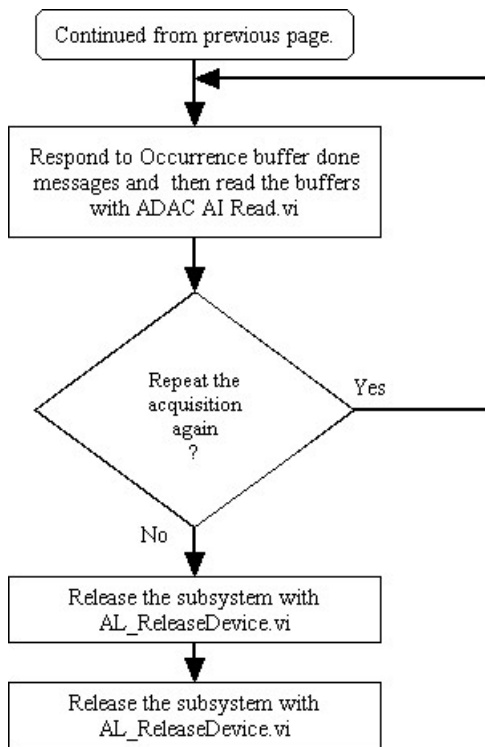


Figure 2: Continuous Asynchronous Program Flow (cont.)

Continuous Software Polling Program Flow

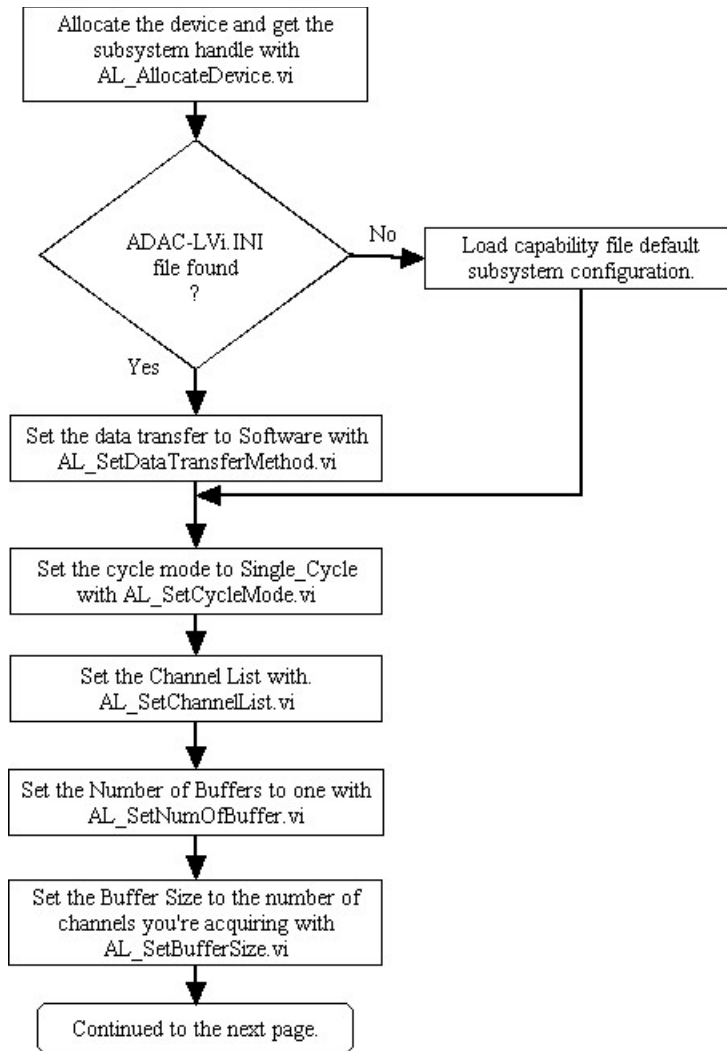


Figure 3: Continuous Software Polling Program Flow

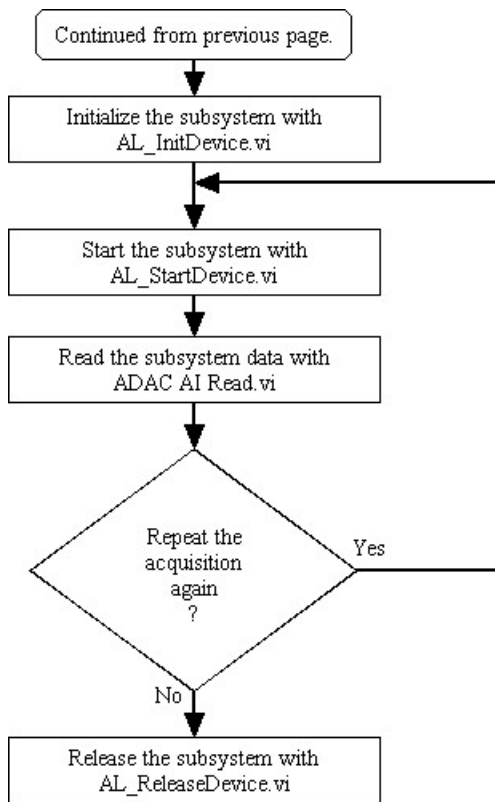


Figure 3: Continuous Software Polling Program Flow (cont.)

Single Point Analog Output Program Flow

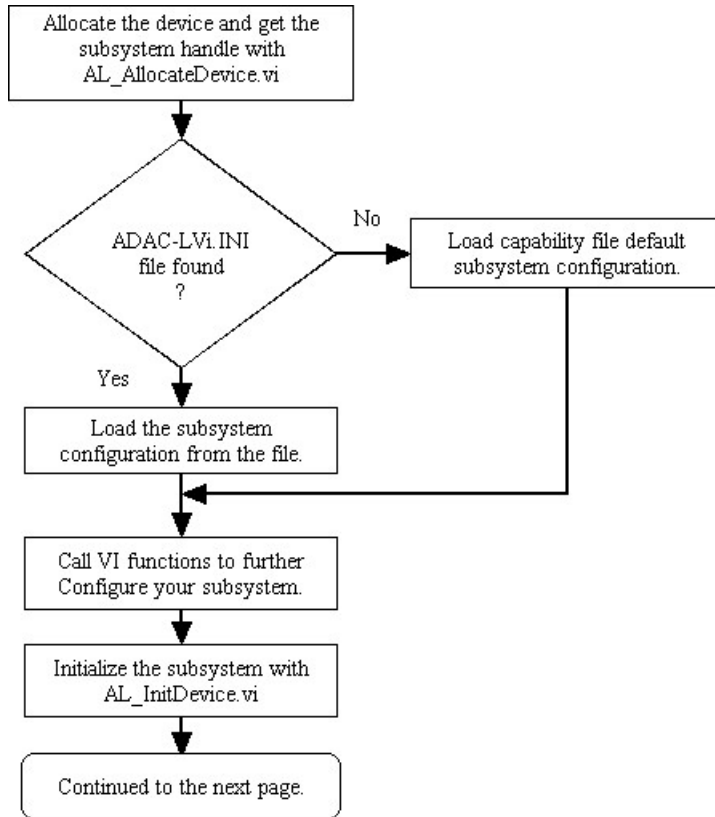


Figure 4: Single Point Analog Output Program Flow

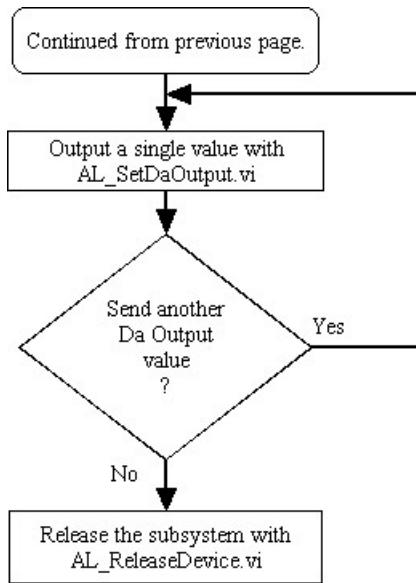


Figure 4: Single Point Analog Output Program Flow (cont.)

Single Point Digital I/O Program Flow

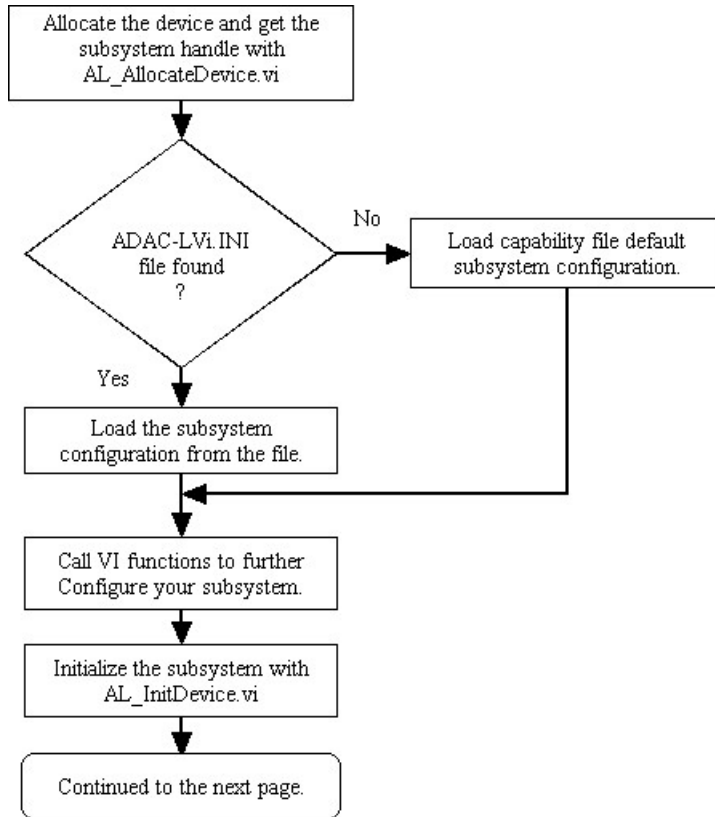


Figure 5: Single Point Digital I/O Program Flow

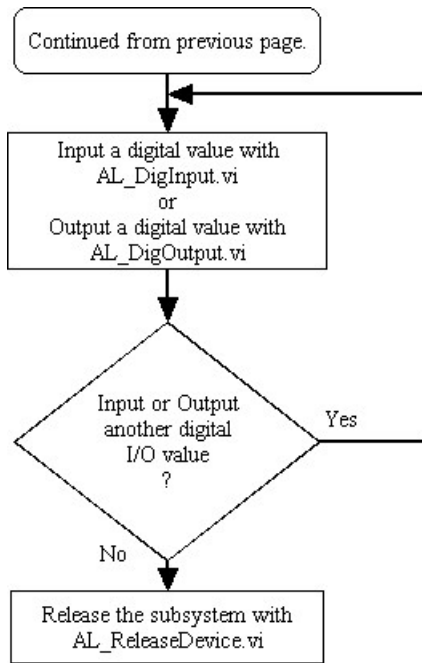


Figure 5: Single Point Digital I/O Program Flow (cont.)

The Advanced System Function VIs

The Advanced VIs are used to perform advance system functions.

To access the ADAC Analog Input VIs, choose **Functions >> User Libraries >> ADAC >> Adv Func.**

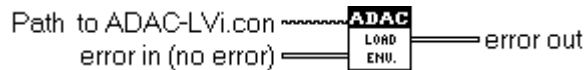
The Advanced System VIs are grouped by function in a series of libraries:

- **ADAC Environment Config**
- **ADAC Board Func Library**
- **ADAC Buffer Config Library**
- **ADAC Burst Config Library**
- **ADAC Channel Config Library**
- **ADAC Clock Config Library**
- **ADAC Data Convert Library**
- **ADAC Analog Ouput Library**
- **ADAC Digital I/O Library**
- **ADAC DMA Config Library**
- **ADAC Filter Config Library**
- **ADAC Gate Config Library**
- **ADAC Input Config Library**
- **ADAC IRQ (Interrupt) Config Library**
- **ADAC Device Function Library**
- **ADAC Panel Config Library**
- **ADAC Range Config Library**
- **ADAC CJ (Cold Junction) Config Library**
- **ADAC Trigger Config Library**

ADAC Environment Config

- ADAC Load Env
- ADAC Release Env
- ADAC Set Env String
- ADAC Get Env String

ADAC Load Env



This VI loads the ADAC-LVi environment. This VI must be called before any other ADAC-LVi calls can be made.

Note

The ADAC-LVi automatically loads the environment when a call to AllocateDevice is initiated and released when the last device handle in use is released. This behavior is controlled by the AutoLoad=YES setting in the ADAC-LVi.con file located in the main Windows directory.



Path to ADAC-LVi.con Specifies the complete drive, path and filename of the ADAC-LVi environment configuration file ADAC-LVi.con. If the full drive and path are not specified, the Windows directory is searched.

Notes:

The ADAC-LVi is set to automatically load the environment when a call to AllocateDevice is initiated and released when the last device handle in use is released. This behavior is controlled by the AutoLoad=YES setting in the ADAC-LVi.con file located in the main Windows directory.



error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.



error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Release Env



This VI releases the ADAC-LVi environment. This VI must be called before the main VI terminates to release the system resources allocated by the ADAC-LVi environment. This VI releases the ADAC-LVi environment even if (error in) contains an error. This VI is typically not required, see notes below.

Note:

The ADAC-LVi is set to automatically loads the environment when a call to AllocateDevice is initiated and released when the last device handle in use is released. This behavior is controlled by the AutoLoad=YES setting in the ADAC-LVi.con file located in the main Windows directory.

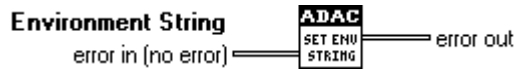


error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.



error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Set Env String



This VI sets ADAC-LVi user-defined environment description string.



Environment String Specifies the user define environment description string.



error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.



error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Get Env String



This VI returns the user defined environment description string.



error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.



Environment String Contains the user defined environment description string.



error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Board Func Library

- ADAC Get Board Hardware ID
- ADAC Get Board Hardware Version
- ADAC Get Board Driver Version
- ADAC Set Board String
- ADAC Get Board String
- ADAC Get Board Error

ADAC Get Board Hardware ID



This VI retrieves the board's hardware register ID number.



Board ID (0) Specifies the ADAC-LVi board ID number identified in the ADAC-LVi configuration file (.CON).



error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.



Board Hardware ID Contains the board's hardware register ID number.



error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Get Board Hardware Version



This VI retrieves the board's hardware version number.



Board ID (0) Specifies the ADAC-LVi board ID number identified in the ADAC-LVi configuration file (.CON).



error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.



Board Version Contains the board's hardware register version number.



error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Get Board Driver Version



This VI retrieves the board's driver version number.



Board ID (0) Specifies the ADAC-LVi board ID number identified in the ADAC-LVi configuration file (.CON).



error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

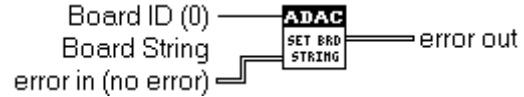


Board Driver Version Contains the board's driver version number.



error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Set Board String



This VI sets ADAC-LVi user-defined board description string.

132 **Board ID (0)** Specifies the ADAC-LVi board ID number identified in the ADAC-LVi configuration file (.CON).

abc **Board String** Specifies the user-defined board description string.

err **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

err **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Get Board String



This VI returns the user defined board description string.

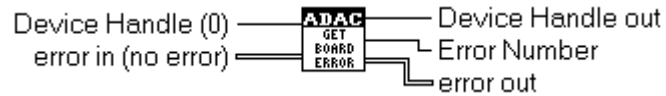
132 **Board ID (0)** Specifies the ADAC-LVi board ID number identified in the ADAC-LVi configuration file (.CON).

err **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

abc **Board String** Contains the user defined board description string.

err **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Get Board Error



This VI returns the last hardware error code that occurred on the specified device. A return error code of 0 indicates no hardware errors have occurred.



Device Handle (0) Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.



error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.



Device Handle out Device Handle out contains the value of Device Handle in.



Error Number Error Number is a numeric value that represents the last board hardware error code of the device. A return error code of 0 indicates no hardware errors have occurred.



error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Buffer Config Library

- ADAC Get Buffer Status
- ADAC Set Buffer Size
- ADAC Get Buffer Size
- ADAC Set Number of Buffers
- ADAC Get Number of Buffers
- ADAC Set Buffer
- ADAC Clear Buffer Done
- ADAC Get Auto Init Buffers
- ADAC Get Buffer Done Handler
- ADAC Set Auto Init Buffers
- ADAC Set Buffer Done Handler
- ADAC Get Num of Buffers

ADAC Get Buffer Status



This VI provides access to the device's buffers. Internally ADAC-LVi buffers are not just pointers to a DAQ board's subsystem data, but are pointers to a structure that contains the buffer type, size, current sample count, hardware trigger points, completion flags, status flags, and error flags. Some of these fields remain constant for the duration of the acquisition. Others contain relevant information about the current condition of the buffer and error conditions that can occur in the hardware device's subsystem during the current acquisition.

A completed buffer only indicates the BUFFER STATUS is available, always check the StatusFlag and ErrorFlag.

ADAC Get Buffer Status (con't.)

- I32** **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.
- I32** **BUFFER NUMBER (-1:next)** Specifies the ID number of the buffer from which the status information is obtained.
If Buffer Number is set to -1 the status is obtained from the next available LDSB buffer.
- E06** **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
- I32** **Device Handle out** Device Handle out contains the value of Device Handle in.
- E06** **BUFFER STATUS** ADAC-LVi buffers are not just arrays of data to a DAQ board's subsystem data, but are structures that contains the buffer type, size, data array, current sample count, hardware trigger points, completion flags, status flags, and error flags. Some of these fields remain constant for the duration of the acquisition. Others contain relevant information about the current condition of the buffer, and error conditions that can occurred during the current acquisition.

The three structure variables that provide state/status/error information are the DoneFlag, StatusFlag and ErrorFlags.

- U32** **BufferType** [U32] BufferType
Data Buffer type, DMA, Interrupt or Polling.
- U32** **BufferSize** [U32] BufferSize
Size of the buffer in bytes.
- U32** **BufferWrite** [U32] BufferWrite
Number of data samples in the buffer.
- U32** **BufferRead** [U32] BufferRead
Number of data samples obtained from a buffer.
- U32** **TriggerPoint** [U32] TriggerPoint
Buffer position number of the trigger point.
- U32** **TiggerPointPre** [U32] TriggerPointStart
Buffer position number of the pre-trigger data.
- U32** **DoneFlag** [U32] DoneFlag:
The Done Flag specifies the current state of a buffer. This flag can be set to any of the three following conditions.

ADAC Get Buffer Status (con't.)

BUFFER_IDLE = 0 The buffer is available for use, but acquisition has not started on this buffer.

BUFFER_DONE = 1 The buffer has been filled with samples and is available for use in the user application.

BUFFER_INUSE = 2 The buffer is currently being filled with samples



StatusFlag [U32] StatusFlag:

The Status Flag specifies the completion status of a buffer. This flag can be set to any of the four following conditions.

BUFFER_EMPTY = 1 No samples are available in the buffer.

BUFFER_INCOMPLETE = 2 The output buffer has not read completely or and input has not filled to capacity.

BUFFER_COMPLETE = 4 The output buffer has been read completely.

BUFFER_FULL = 8 The input buffer has filled to capacity.



ErrorFlag [i32] ErrorFlags:

The Error Flags specify the condition which has stopped a buffer from successfully completing. Although an error may be reported, this does not indicate the BUFFER_FULL flag is not set. This would be the case if an error was detected and enough samples were available to fill the buffer to capacity. One or more of these flags may also be set together. If any of these flags have set, checking the current hardware error conditions will usually reveal the actual condition that caused the acquisition runtime error. This flag can be set to any of the four following conditions.

BUFFER_STOPPED = 1 An error occurred and the buffer(s) are no longer being serviced.

BUFFER_OVERRUN = 2 An error occurred when attempting to place the next sample beyond the bounds of the buffer.

BUFFER_UNDERRUN = 4 An error occurred and the buffer is incomplete.

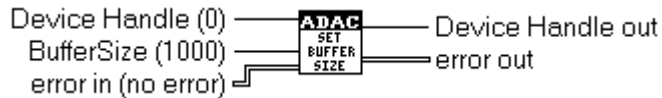
BUFFER_NEXTBUSY = 8 When attempting to access the next buffer, the current state of the DoneFlag indicated the buffer was not yet available for use.

ADAC Get Buffer Status (con't.)

U32 **BufferNum** [U32] BuffNum
Buffer Number ID (0-n)

TF **Buffer Complete** Buffer Complete is TRUE if this VI was able to successfully obtain a done buffer. A completed buffer may only indicate the buffer is no longer being serviced and the status is available, always check the StatusFlag and ErrorFlag.

Err **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Set Buffer Size

This VI sets the size of a device's internal data buffers in samples. The size can range from 0-n, the only limitation is the amount of available memory. See also SetNumBuffer.vi. to set how many buffers are rotated.

I32 **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.

I32 **BufferSize (1000)** Buffer Size is a numeric value that specifies the number of data samples each buffer contains.

Err **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.






I32 **Device Handle out** Device Handle out contains the value of Device Handle in.

Err **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

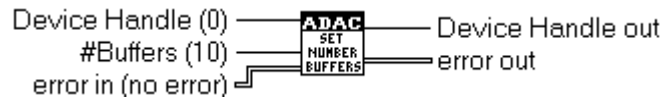
ADAC Get Buffer Size



This VI gets the size of a device's internal data buffers in samples.






-  **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.
-  **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
-  **Device Handle out** Device Handle out contains the value of Device Handle in.
-  **BufferSize** Buffer Size is a numeric value that specifies the number of data samples each buffer contains.
-  **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Set Number of Buffers








This VI sets the quantity of a device's internal data buffers. The quantity can range from 0-n, the only limitation is the amount of available memory. See also SetBufferSize.vi to set how many data samples are available in each buffer.

ADAC Set Number of Buffers (con't.)

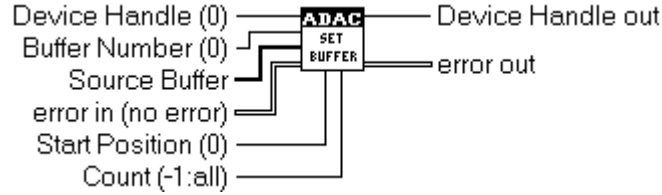
-  **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.
-  **#Buffers (10)** #Buffers is a numeric value that specifies the number of internal data buffers available to a device. Each buffer contains n samples specified by the buffer size parameter.
-  **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
-  **Device Handle out** Device Handle out contains the value of Device Handle in.
-  **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Get Number of Buffers

This VI gets the number of internal data buffers for a given device. The number of buffers can range from 0-n, the only limitation is the amount of available memory. See also SetNumOfBuffers.vi to set the number of data buffers.

-  **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.
-  **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
-  **Device Handle out** Device Handle out contains the value of Device Handle in.
-  **#Buffers** #Buffers is a numeric value that specifies the number of internal data buffers currently set for a device.
-  **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

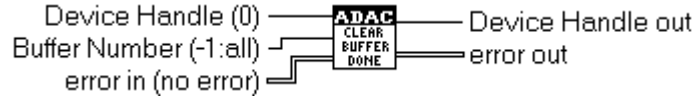
ADAC Set Buffer



This VI writes output data samples to the specified buffer Number created by the ADAC DIO config.vi. The write always begins at the Start Position and ends at the specified Count(s). Setting the Start Position to 0 and the count to -1 will copy all source data samples to the destination buffer.

- I32** **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.
- I32** **Buffer Number (0)** Specifies the ID number of the output buffer to be set. Buffers ID numbers range from (0 - n) according to the number of allocated buffers.
- I16** **Source Buffer** Source buffer is a one-dimensional array containing the output data to be copied to the specified ADAC-LVi Internal Buffer Number.
- I32** **Start Position (0)** Start Position specifies the starting sample number position in the source buffer to be copied.
- I32** **Count (-1:all)** Count specifies the number of samples to be copied from the Start Position of the source buffer. If the count is set to -1 all source buffer data points are copied.
- F4** **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
- I32** **Device Handle out** Device Handle out contains the value of Device Handle in.
- F4** **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Clear Buffer Done



The `AL_ClearBufferDoneFlag` function informs the ADAC-LVi driver that a buffer is available for use. All buffers are flagged as done when completed. It is the user's responsibility to inform the ADAC-LVi driver that a buffer is once again available. This function is therefore required when the ADAC-LVi cycle mode is set to `CONTINUOUS_CYCLE`, otherwise an error will be generated when the driver attempts to reuse a buffer that has not been flagged as available. Setting `IBuffNum` to -1 will reinitialize all buffers for the specified device, this is most useful in the `SINGLE_CYCLE` mode when all buffers have been completed.

This VI is available for use when `ADAC AI READ.VI` is not used. The `ADAC AI READ.VI` gets the current buffers status; if status is complete, it copies the data to LabVIEW and releases the buffer back to the ADAC-LVi driver.

- I32** **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.
- I32** **Buffer Number (-1:all)** Specifies the ID number of the buffer to be released back to the ADAC-LVi driver. Setting Buffer Number to -1 releases all buffers back, the -1 option is most useful in the `SINGLE_CYCLE` mode after all buffers have been completed.
- F7** **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
- I32** **Device Handle out** Device Handle out contains the value of Device Handle in.
- F7** **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Get Auto Init Buffers



This VI gets the software buffer done initialization authorization. All buffers are flagged as done when filled and it is the user application’s responsibility to inform the ADAC-LVi driver that the buffer is once again available.

The auto initialize is used most in the SINGLE_CYCLE mode when multiple calls to AL_StartAd is necessary to restart the acquisition. On each call to AL_StartAd.vi all buffers will be reinitialized.

- I32** **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.
- Bo** **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
- I32** **Device Handle out** Device Handle out contains the value of Device Handle in.
- TF** **Auto Init** Auto Init is a numeric value that represents the software buffer done initialization authorization.

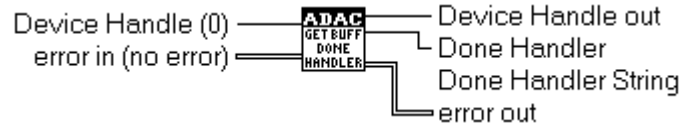
The available options are:

0:OFF Buffers are not auto initialized on AI_StartAD. The user is responsible for calling AL_ClearBufferDoneFlag on each buffer.

1:ON Buffers are auto initialized on AI_StartAD.

- Bo** **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Get Buffer Done Handler



This VI gets the buffer done message handler method. The internal ADAC-LVi buffers can be accessed in LabVIEW by either polling the buffer status information for a completed buffer (CHECK_BUFFER) or asynchronously by using a LabVIEW occurrence to receive buffer done messages (POST_MESSAGE_PARAMS). If POST_MESSAGE_PARAMS is selected an additional call to ADAC Occurrence Config.vi must be made to configure the occurrence.

The occurrence can then be set to "wait on occurrence", this effectively puts your VI to sleep saving precious processor time until a buffer done message is sent or the wait on occurrence times out.

The ADAC Occurrence Config.VI automatically sets the Done Handler to POST_MESSAGE_PARAMS and calls AL_SetBufferDoneHandlerMsg to configure the occurrence done buffer messages to LabVIEW.



Device Handle (0) Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.



error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.



Device Handle out Device Handle out contains the value of Device Handle in.



Done Handler Done Handler is a numeric value of the current hardware buffer done handler method set in the ADAC-LVi driver. The available options are:

0:CHECK_BUFFER Buffers are polled for completion.

1:POST_MESSAGE_PARAMS A windows message is posted to a LabVIEW occurrence. The information to configure the occurrence must be processed by a call to ADAC Occurrence Config.vi.

See the ADAC Occurrence Config.VI.

ADAC Get Buffer Done Handler (con't)

abc **Done Handler String** Done Handler String is the sting value of the current hardware buffer done handler method set in the ADAC-LVi driver.

The available options are:

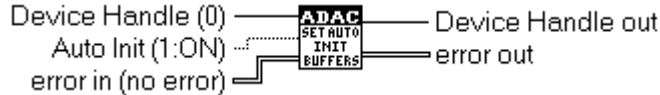
"CHECK_BUFFER" Buffers are polled for completion.

"POST_MESSAGE_PARAMS" A windows message is posted to a LabVIEW occurrence. The information to configure the occurrence must be processed by a call to ADAC Occurrence Config.vi.

See the ADAC Occurrence Config.VI.

FT **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Set Auto Init Buffers



This VI sets the software buffer done initialization authorization. All buffers are flagged as done when filled, and it is the user application's responsibility to inform the ADAC-LVi driver that the buffer is once again available.

The auto initialize is used most in the SINGLE_CYCLE mode when multiple calls to AL_StartAd are necessary to restart the acquisition. On each call to AL_StartAd.vi all buffers will be reinitialized.

I32 **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.

TF **Auto Init (1:ON)** Auto Init is a numeric value that sets the software buffer done initialization authorization.

ADAC Set Auto Init Buffers (con't.)

The available options are:

0:OFF Buffers are not auto initialized on AI_StartAD. The user is responsible for calling AL_ClearBufferDoneFlag on each buffer.

1:ON Buffers are auto initialized AI_StartAD.

The default Done Handler is 1:ON



error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

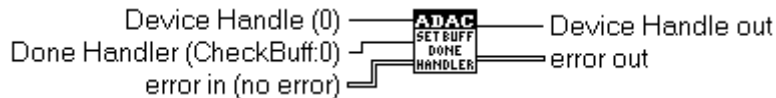


Device Handle out Device Handle out contains the value of Device Handle in.



error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Set Buffer Done Handler




This VI sets the buffer done message handler method. The internal ADAC-LVi buffers can be accessed in LabVIEW by either polling the buffer status information for a completed buffer (CHECK_BUFFER) or asynchronously by using a LabVIEW occurrence to receive buffer done messages (POST_MESSAGE_PARAMS). If POST_MESSAGE_PARAMS is selected, an additional call to ADAC Occurrence Config.vi must be made to configure the occurrence.

The occurrence can then be set to "wait on occurrence", this effectively puts your VI to sleep saving precious processor time until a buffer done message is sent or the wait on occurrence times out.

The ADAC Occurrence Config.VI automatically sets the Done Handler to POST_MESSAGE_PARAMS and calls AL_SetBufferDoneHandlerMsg to configure the occurrence done buffer messages to LabVIEW.

ADAC Set Buffer Done Handler

 **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.


 **Done Handler (CheckBuff:0)** Done Handler is a numeric value that sets the hardware buffer done handler method in the ADAC-LVi driver. The available options are:


0:CHECK_BUFFER Buffers are polled for completion.


1:POST_MESSAGE_PARAMS A windows message is posted to a LabVIEW occurrence. The information to configure the occurrence must be processed by a call to ADAC Occurrence Config.vi.

See the ADAC Occurrence Config.VI.

The default Done Handler is 0:CheckBuff

 **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

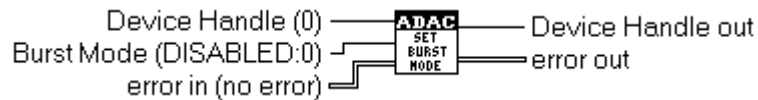
 **Device Handle out** Device Handle out contains the value of Device Handle in.

 **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Burst Config Library

- ADAC Set Burst Mode
- ADAC Set Burst Length
- ADAC Set Burst Rate
- ADAC Get Burst Struct

ADAC Set Burst Mode



This VI sets the hardware burst operation mode of the device. The mode can be either enabled or disabled.



Device Handle (0) Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.



Burst Mode (DISABLED:0) Burst Mode is a numeric value that sets the hardware Burst operation mode of the device. The available options are:

0:DISABLED	Disable burst Mode
1:ENABLED	Enable burst Mode

The default Burst Mode is 0:DISABLED.



error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

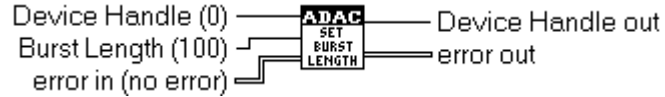


Device Handle out Device Handle out contains the value of Device Handle in.



error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Set Burst Length



Burst Length is a numeric value that sets the number of samples obtained from a device during a burst operation. This function is made available for hardware devices that provide burst length mechanism. When the hardware device starts a burst, it provides (n) more samples, then stops. The Burst samples specify the number of data samples to be obtained after a valid hardware burst trigger.

I32 **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.

I32 **Burst Length (100)** Burst Length is a numeric value that sets the number of samples obtained by the device during a burst operation. This function is made available for hardware devices that provide burst length mechanism. When the hardware device starts a burst, it provides (n) more samples, then stops. The Burst samples specify the number of data samples to be obtained after a valid hardware burst trigger.

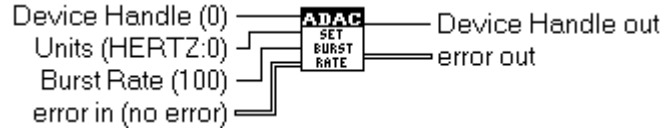
The available Burst length is device independent and is verified by ADAC-LVi against the min./max. burst length specified in the device's capabilities file.

Err **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

I32 **Device Handle out** Device Handle out contains the value of Device Handle in.

Err **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Set Burst Rate



This VI sets the hardware burst rate of the Device. The rate can be specified in hertz or tics.

I32 **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.

I32 **Units (HERTZ:0)** Units is a numeric value that specifies base units in which the Burst Rate parameter is specified. The available options are:

0:HERTZ	Rate is specified in units of hertz
1:TICS	Rate is the divisor to the Burst clock source

If TICS is chosen, the Burst Rate parameter is usually set as divisor of the Burst Source clock.

The default Unit is 0:HERTZ

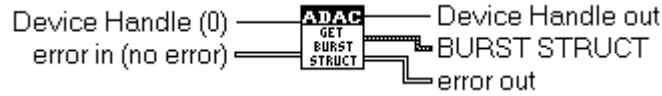
DBL **Burst Rate (100)** Burst Rate is a double value that specifies the rate at which burst clocks occur.

Err **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

I32 **Device Handle out** Device Handle out contains the value of Device Handle in.

Err **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Get Burst Struct



This VI provides access to the device's burst mode settings. See the associated burst mode setting VI for a full description of the BURST STRUCT cluster.

- I32** **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.
- F7** **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
- I32** **Device Handle out** Device Handle out contains the value of Device Handle in.
- 00a** **BURST STRUCT**
 - I32** **Burst Mode** Burst Mode is a numeric value that represent the current hardware Burst operation mode of the device. The available options are:
 0:DISABLED Disable burst Mode
 1:ENABLED Enable burst Mode
 - I32** **Burst Mode ID** Burst Mode ID is a numeric value that represents the current hardware Burst Mode ID number defined in the devices capabilities file. This ID is used internally by the ADAC-LVi driver and is typically not used in LabVIEW VI's.
 - I32** **Burst Rate Units** Burst Rate Units is a numeric value that specifies base units in which the Burst Rate parameter is specified. The available options are:
 0:HERTZ Rate is specified in units of hertz
 1:TICS Rate is the divisor to the Burst clock source

If TICS is chosen, the Burst Rate parameter is usually set as divisor of the Burst clock.

- DBL** **Burst Rate** Burst Rate is a double value that specifies the rate at which burst clocks occur.

ADAC Get Burst Struct (cont.)

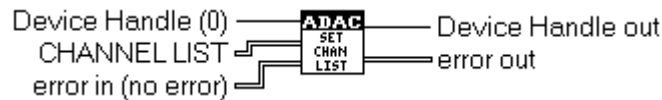
I32 **Burst Samples** Burst Samples is a numeric value that represents the number of samples obtained by the device during a burst operation. This parameter is used by hardware devices that provide burst length mechanism. When the hardware device starts a burst, it provides (n) more samples, then stops. The Burst samples specify the number of data samples obtained after a valid hardware burst trigger.

Error **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Channel Config Library

- ADAC Set Channel List
- ADAC Get Channel List
- ADAC Get Num of Chans

ADAC Set Channel List



This VI function sets the active channel and gain setting for the specified device. If a device supports gain settings, the list is considered a channel and gain list, and must be set appropriately. See examples below.

I32 **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.

5.1 **CHANNEL LIST** The CHANNEL LIST cluster is used to set the active channel and gain setting for the specified device.

I32 **Array Type** Array Type
 0:Use string list
 1:Use array of numbers

abc **String Channel List (empty)** String Channel List represents the character string list of channels.

For boards that support input gain the channel list also specifies a separate gain value for each channel. Gains are specified in parentheses after the channel number. For example, "0(1),1(8),2(4)" is interpreted as channel 0 at a gain of 1, channel 1 at a gain of 8 and channel 2 at a gain of 4.

Examples:

string without gains "0,1,2,3,4,5,6,7"

string with gains "0(1),1(2),2(4),3(8),4(1),5(2),6(4),7(8)"

ADAC Set Channel List (con't.)

[I32] **Numeric Channel List (0 (1))** Numeric Channel List represents the numeric array of channels.

For boards that support input gain the channel list also specifies a separate gain value for each channel. Gains are specified in parentheses after the channel number. For example, "0(1),1(8),2(4)" is interpreted as channel 0 at a gain of 1, channel 1 at a gain of 8 and channel 2 at a gain of 4.

Examples:

array without gains {0,1,2,3,4,5,6,7}

array with gains {0,1,1,2,2,4,3,8,4,1,5,2,6,4,7,8}

[Error] **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

[I32] **Device Handle out** Device Handle out contains the value of Device Handle in.

[Error] **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Get Channel List



This VI function retrieves the current active channel and gain settings for the specified device. If a device supports gain settings, the list is considered a channel and gain list. See examples below.

ADAC Get Channel List (con't.)

- I32** **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.
- E64** **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
- I32** **Device Handle out** Device Handle out contains the value of Device Handle in.
- E64** **CHANNEL LIST** The CHANNEL LIST cluster is used to Get the active channel and gain setting for the specified device.
- I32** **Array Type** Array Type
 0:Use string list
 1:Use array of numbers
- abc** **String Channel List)** String Channel List represents the character string list of channels.

 For boards that support input gain the channel list also specifies a separate gain value for each channel. Gains are specified in parentheses after the channel number. For example, "0(1),1(8),2(4)" is interpreted as channel 0 at a gain of 1, channel 1 at a gain of 8 and channel 2 at a gain of 4.

 Examples:
 string without gains "0,1,2,3,4,5,6,7"
 string with gains "0(1),1(2),2(4),3(8),4(1),5(2),6(4),7(8)"
- I32** **Numeric Channel List** Numeric Channel List represents the numeric array of channels.

 For boards that support input gain the channel list also specifies a separate gain value for each channel. Gains are specified in parentheses after the channel number. For example, "0(1),1(8),2(4)" is interpreted as channel 0 at a gain of 1, channel 1 at a gain of 8 and channel 2 at a gain of 4.

 Examples:
 array without gains {0,1,2,3,4,5,6,7}
 array with gains {0,1,1,2,2,4,3,8,4,1,5,2,6,4,7,8}
- E64** **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Get Num of Chans



This VI returns the number of channels currently configured in a device's channel list. Multiple occurrences of the same channel number count as multiple channels, not one.

132 **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.

5.1 **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

132 **Device Handle out** Device Handle out contains the value of Device Handle in.

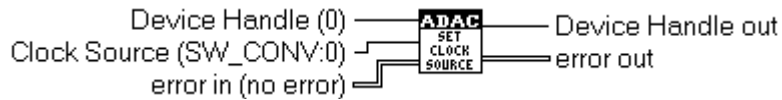
132 **Number Of Chans** Number Of Chans is a numeric value that specifies the number of channels currently configured in a device's channel list. Multiple occurrences of the same channel number count as multiple channels, not one.

5.1 **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Clock Config Library

- ADAC Set Clock Source
- ADAC Set Clock Rate
- ADAC Get Clock Structure
- ADAC Get Actual Clock Rate

ADAC Set Clock Source



This VI sets the hardware clocking source of the device. The source can be either software_convert, internal, ext_rising_edge or ext_falling_edge.



Device Handle (0) Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.



Clock Source (SW_CONV:0) Clock Source is a numeric value that sets the hardware clocking source of the device. The available options are:

0:SOFTWARE_CONVERT	Software polled acquisition
1:INTERNAL	On-board clocking
2:EXT_RISING_EDGE	External clocking rising edge
3:EXT_FALLING_EDGE	External clocking falling edge
4:MMTIMER	Multi-Media system time

The default Clock Source is 0:SOFTWARE_CONVERT



error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

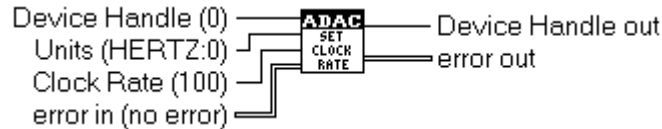


Device Handle out Device Handle out contains the value of Device Handle in.



error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Set Clock Rate



This VI sets the hardware clocking rate of the device. The rate can be specified in hertz or tics.

I32 **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.

I32 **Units (HERTZ:0)** Units is a numeric value that specifies base units in which the Clock Rate parameter is specified. The available options are:

0:HERTZ	Rate is specified in units of hertz
1:TICS	Rate is the divisor to the clock source
2:MSECONDS	Miliseconds, supported by the 4012AD and 4112AD series in the Windows 95/98 environment only. The 4012 or 4112 series can be set to either HERTZ or MSECONDS only.

If TICS is chosen, the Clock Rate parameter is usually set as divisor of the Clock Source selected.

The default Unit is 0:HERTZ

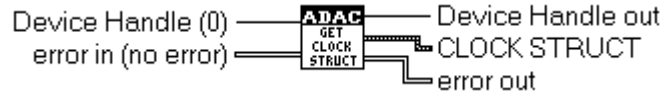
DBL **Clock Rate (100)** Clock Rate is a double value that specifies rate at which clocks occur.

Err **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

I32 **Device Handle out** Device Handle out contains the value of Device Handle in.

Err **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Get Clock Structure



This VI provides access to the device clock settings. See the associated clock setting VI for a full description of each setting returned to the CLOCK STRUCT cluster.

- I32** **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.
- Err** **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
- I32** **Device Handle out** Device Handle out contains the value of Device Handle in.
- Err** **CLOCK STRUCT**
 - I32** **Clock Source** Clock Source is a numeric value that represents the current hardware clocking source of the device. The available options are:

0:SOFTWARE_CONVERT	Software polled acquisition
1:INTERNAL	On-board clocking
2:EXT_RISING_EDGE	External clocking rising edge
3:EXT_FALLING_EDGE	External clocking falling edge
4:MMTIMER	Multi-Media system timer
 - I32** **Clock Source ID** Clock Source ID is a numeric value that represents the current hardware Clock Source ID number defined in the devices capabilities file. This ID is used internally by the ADAC-LVi driver and is typically not used in LabVIEW VI's.

ADAC Get Clock Structure (con't.)

I32 **Clock Rate Units** Clock Rate Units is a numeric value that specifies base units in which the Clock Rate parameter is specified. The available options are:

- 0:HERTZ Rate is specified in units of hertz (samples per second)
- 1:TICS Rate is the divisor programmed to the on-board clocking source.
- 2:MSECONDS Miliseconds, supported by the 4012AD and 4112AD series in the Windows 95/98 environment only. The 4012 or 4112 series can be set to either HERTZ or MSECONDS only.

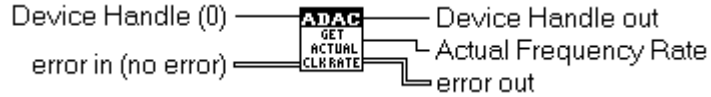
If TICS is chosen, the Clock Rate parameter is used to specify the divisor to the on-board CLK input source. The on-board clock source is typically an 8254 device that has a 1,000,000 CLK input source. The 1,000,000 source is divided from 2 to 65525 to set actual A/D clocking rate.

For boards that use 2 on-board clocks concatenated together, the divisor broken into two 16 bit WORDS. The high word divides the first CLK that feeds the second CLK that is further divided by the low word.

DBL **Clock Rate** Clock Rate is a double value that represent the current rate at which clocks occur.

ERR **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Get Actual Clock Rate



This function is used to determine the frequency actually available by a device's clocking hardware. The clock frequency specified in the `AL_SetClockRate.vi` function may not be achievable due to hardware limitations.

Note:

The `AL_SetClockRate` clocking rate parameters are not programmed to a device until it is initialized with `AL_InitializeDevice`. This function uses the clocking rate parameters from the last call to the `AL_SetClockRate` function to determine the actual available hardware rate and does not reflect the current device's programmed rate until the device has been initialized.

Once a device has been initialized, if the `AL_SetClockRate` is called again with different clocking rate parameters this function will not return the current device's programmed clocking frequency until the device is re-initialized.

Therefore this function only returns the actual rate that can be set by the `AL_SetClockRate` function. To get the current Clocking Rate programmed to the device, it must be initialized before calling this function.



error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.



Device Handle (0) Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.



Device Handle out Device Handle out contains the value of Device Handle in.



Actual Frequency Rate Actual Frequency Rate is the frequency that a device's clock was actually programmed. The clock frequency specified in the `AL_SetClockRate.vi` function may not be achievable due to hardware limitations.

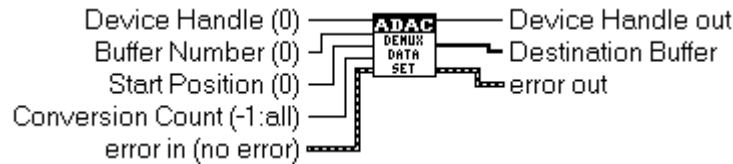


error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Data Convert Library

- ADAC Demux Data Set
- ADAC TC Temp
- ADAC RTD Temp

ADAC Demux Data Set



This VI converts the specified samples in the selected buffer to engineering voltage units.

[I32] **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.

[I32] **Buffer Number (0)** Specifies the ID number of the buffer to be converted to voltage.

[I32] **Start Position (0)** Start Position specifies the starting sample number in the ADAC-LVi buffer to be converted.

[I32] **Conversion Count (-1:all)** Conversion count specifies the number of samples to be converted from the Start Position. If Conversion count is set to -1, all samples from the specified Start Position to the end of the buffer are converted.

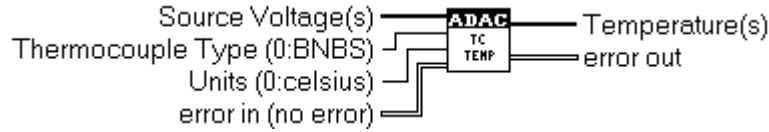
[E4] **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

[I32] **Device Handle out** Device Handle out contains the value of Device Handle in.

[SGL] **Destination Buffer** Destination buffer is a one-dimensional array containing the converted data in volts.

[E4] **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC TC Temp



This VI converts the specified thermocouple Source Voltage(s) input array into the specified temperature units.

[SGL] **Source Voltage(s)** Source Voltage(s) is a one-dimensional array containing the input data in volts to be converted into temperature.

[U16] **Thermocouple Type (0:BNBS)** Thermocouple Type is a numeric value that specifies the NBS thermocouple polynomial equation for the conversion. The available options are:

0:BNBS	B-type thermocouple
1:KNBS	K-type thermocouple
2:SNBS	S-type thermocouple
3:TNBS	T-type thermocouple
4:JNBS	J-type thermocouple
5:RNBS	R-type thermocouple
6:ENBS	E-type thermocouple

The default Thermocouple Type is 0:BNBS

[U16] **Units (0:celsius)** Units specifies the desired temperature units for the conversion. The available options are:

0:Celcius	Celsius degree
1:Fahrenheit	Fahrenheit degree
2:Kelvin	Kelvin degree
3:Rankine	Rankine

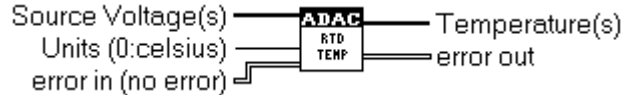
The default Units is 0:Celcius

[Err] **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

[SGL] **Temperature(s)** Temperature(s) is a one-dimensional array containing the converted data in the specified temperature units.

[Err] **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC RTD Temp



This VI converts the specified RTD Source Voltage(s) input array into temperature units.

[SGL] **Source Voltage(s)** Source Voltage(s) is a one-dimensional array containing the input data in volts to be converted into temperature.

[U16] **Units (0:celsius)** Units specifies the desired temperature units for the conversion. The available options are:

0: Celcius	Celsius degree
1: Fahrenheit	Fahrenheit degree
2: Kelvin	Kelvin degree
3: Rankine	Rankine

The default Units is 0: Celcius

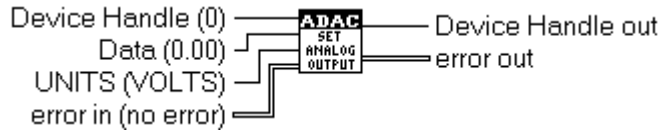
[Err] **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

[SGL] **Temperature(s)** Temperatures(s) Temperature(s) is a one-dimensional array containing the converted data in the specified temperature units.

[Err] **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Analog Output Library

ADAC Set Analog Output



This VI immediately sets a hardware Analog Output port to the specified DATA value. The DATA value can be specified in VOLTS, RAWDATA or CURRENT.

I32 **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.

DBL **Data (0.00)** Data is a double value that specifies the output.

↑ **UNITS (VOLTS)** UNITS is a numeric value that sets the units in which the DATA output parameter is specified. The available options are:

- 0:VOLTS The voltage to output
- 1:RAWDATA The hardware register bits output
- 2:CURRENT The current to output

The default UNITS 0:VOLTS

Err **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

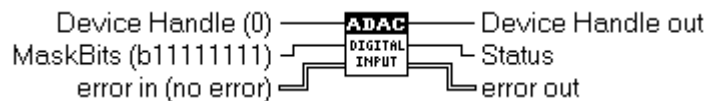
I32 **Device Handle out** Device Handle out contains the value of Device Handle in.

Err **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Digital I/O Library

- ADAC Digital Input
- ADAC Digital Output
- ADAC Digital Bits Test
- ADAC Set Port Res
- ADAC Set Port Mask
- ADAC Set 5600 Port Struct
- ADAC Get 5600 Port Struct

ADAC Digital Input



This VI immediately retrieves the 4, 8 or 16 bit input status of a digital input port.



Device Handle (0) Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.



MaskBits (b11111111) MaskBits is a binary value that selects the operation bits. All bits set to one (1) in the MaskBits will return their current state in the Status output. Unselected MaskBits will be set to zero (0). The default value is binary 11111111.



error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.



Device Handle out Device Handle out contains the value of Device Handle in.

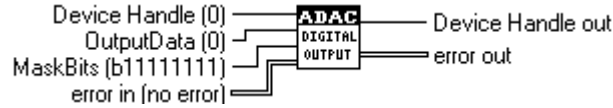


Status Status is a binary value that contains the current port data.



error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Digital Output



This VI immediately sends the specified OutputData to a digital output port.

I32 **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.

I32 **OutputData (0)** OutputData is a binary value that specifies the output data to be sent to the port.

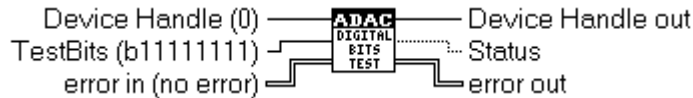
I32 **MaskBits (b11111111)** MaskBits is a binary value that selects the operation bits. All bits set to one(1) in the MaskBits will acted upon. Unselected MaskBits will not change state. The default value is binary 11111111.

F7 **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

I32 **Device Handle out** Device Handle out contains the value of Device Handle in.

F7 **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Digital Bits Test

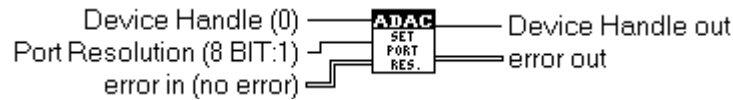


This VI immediately tests a digital input port for a TRUE or FALSE match condition specified in the TestBits parameter.

ADAC Digital Bits Test (con't.)

- I32** **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.
- I32** **TestBits (b11111111)** TestBits is a binary value that selects the operation bits. All bits set to one (1) in the TestBits will be checked for a TRUE (1) condition. The default value is binary 11111111.
- Err** **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
- I32** **Device Handle out** Device Handle out contains the value of Device Handle in.
- TF** **Status** Status is a Boolean value that is set TRUE if all bits specified in the TestBits input parameter are TRUE (1) and set to FALSE if any TestBits are FALSE (0).
- Err** **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Set Port Res(olution)






This VI specifies whether the input or output port is read or written in 4, 8 or 16 bit transfers.

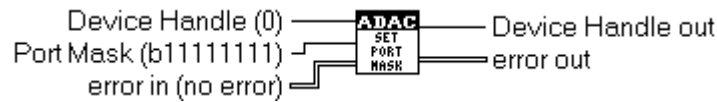
- I32** **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.
- I32** **Port Resolution (8 BIT:1)** Port Resolution is a numeric value that sets whether the input or output port is read or written in 4, 8 or 16 bit transfers. The available options are:
 0:4 BIT Not Available
 1:8 BIT
 2:16 BIT

The default value is 1:8 BIT






ADAC Set Port Res (con't.)

-  **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
-  **Device Handle out** Device Handle out contains the value of Device Handle in.
-  **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

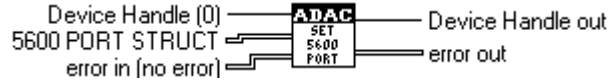
ADAC Set Port Mask



This VI specifies the input or output bit(s) that are to be acted upon during Interrupt or DMA transfers.

-  **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.
-  **Port Mask (b11111111)** Port Mask is a binary value that selects the operation bits. All bits set to one (1) in the Port Mask will acted upon during Interrupt or DMA transfers. The default value is binary 11111111.
-  **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
-  **Device Handle out** Device Handle out contains the value of Device Handle in.
-  **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Set 5600 Port Struct



This VI sets the digital I/O port settings for the 5600 Series cards. Refer to the 5600 manual for detailed information on each setting. This VI is only used with ADAC 5600 boards.

I32 **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.

F16 **5600 PORT STRUCT**

U16 **Port Resolution** Port Resolution is a numeric value that represents the desired number of DIO port bits, either 8 or 16. The DIO ports DIN0 can be either 8 or 16 bit ports and DIN1 can only be an 8 bit port. When DIN0 is set for 16 bit resolution DIN1 is not available. The DIO ports DOT0 and DOT2 can be either 8 or 16 bit ports and DOT1 and DOT3 can only be an 8 bit port. When DOT0 or DOT2 is set for 16 bit resolution DOT1 or DOT 3 is not available respectively.

I32 **Port Mask** Port Mask is a binary value that indicates which bits are to be read during interrupt transfers. If the associated bit in the Port Mask control is set ="1" the bit is read, unactive bits will return "0".

E08 **A5600PORT**

I32 **Debounce** Debounce is a numeric value that indicates if the device's debounce circuitry is enabled or disabled. When Enabled the Time Constant value sets the time required for a stable input.

0:DISABLED

1:ENABLED

The default value is 1:ENABLED

I32 **Port Mode Specification** Port Mode Specification defines the pattern match condition.

0:DISABLED Pattern match is disabled.

ADAC Set 5600 Port Struct (con't.)

- 1:AND All unmasked bit conditions must occur.
- 2:OR Any unmasked bit conditions must occur.

The default Port Mode Specification is 2:OR

I32 **Pattern Latch** Pattern Latch controls the Latch On Pattern ability. If the Pattern Latch is enabled, the state of the input port will be latched when a match condition occurs. The data will be held until the port is read within its interrupt handler. If a subsequent match occurs (port goes from match condition to non-match condition to a new match condition) before the port is read within its interrupt handler, an IP error condition will occur stopping the acquisition. The Ignore IP Error control can be set to TRUE to disable the IP error checking in the driver, but data may be lost when match conditions are occurring faster than they can be serviced.

I32 **Data Path Polarity** Data Path Polarity allows each bit for DIN0 to be individually inverted. If the associated bit is set "1", data read from that bit on DIN0 will be inverted. For detailed definitions see the 5600 manuals ADVANCED PROGRAMMING TECHNIQUES. When the Port Resolution is set for 16 bit mode, Data Path Polarity is set in WORDs b(1111111111111111).

I32 **Special IO Control** The SpecialIoControl controls the BIT CATCHER operation mode. Port DIN0 may be configured to return the state of the bit catcher, instead of the actual port data by setting the associated bit to "1" in the Special Io Control. If a bit is so programmed, the port will return "0" for that bit until a "1" occurs at the input. The port will then read back a "1" even if the "1" goes away. When the DIN0 port is serviced by the driver, each bit enabled and in the "1" state will read automatically reset and ready for the next input transition. For detailed definitions see the 5600 manuals ADVANCED PROGRAMMING TECHNIQUES. When the Port Resolution is set for 16 bit mode, Special Io Control is set in WORDs b(1111111111111111).

I32 **Pattern Polarity** The PatternPolarity, PatternTransition and PatternMask define the match condition for the Port Mode Specification AND / OR modes.

For example one can define a match condition as bit 1, bit 2 and bit 3 all set = "1" as a valid match (AND mode) by setting the following.

Port Mode Specification = AND
PatternPolarity = (b) 1110

ADAC Set 5600 Port Struct (con't.)

PatternTransition = (b) 0
 PatternMask = (b) 111

Alternately, one can define a match condition as any one or more of bit 1, bit 2 and bit 3 having set = "1" by changing the Port Mode Specification to OR mode.

PM(n)	PT(n)	PP(n)	Pattern Match Bit Definitions
0	0	0 or 1	Bit Masked Off
0	1	0 or 1	Any Transition
1	0	0	ZERO
1	0	1	ONE
1	1	0	ONE to ZERO Transition
1	1	1	ZERO to ONE Transition

For detailed definitions see the 5600 manuals ADVANCED PROGRAMMING TECHNIQUES. When the PortResolution=16, PatternPolarity, PatternTransition and PatternMask are set in WORDs b(1111111111111111).

I32

Pattern Transition The PatternPolarity, PatternTransition and PatternMask define the match condition for the Port Mode Specification AND / OR modes.

For example one can define a match condition as bit 1, bit 2 and bit 3 all set = "1" as a valid match (AND mode) by setting the following.

Port Mode Specification = AND
 PatternPolarity = (b) 1110
 PatternTransition = (b) 0
 PatternMask = (b) 1110

Alternately, one can define a match condition as any one or more of bit 1, bit 2 and bit 3 having set = "1" by changing the Port Mode Specification to OR mode.

PM(n)	PT(n)	PP(n)	Pattern Match Bit Definitions
0	0	0 or 1	Bit Masked Off
0	1	0 or 1	Any Transition
1	0	0	ZERO
1	0	1	ONE
1	1	0	ONE to ZERO Transition
1	1	1	ZERO to ONE Transition

ADAC Set 5600 Port Struct (con't.)

For detailed definitions see the 5600 manuals ADVANCED PROGRAMMING TECHNIQUES. When the PortResolution=16, PatternPolarity, PatternTransition and PatternMask are set in WORDs b(1111111111111111).

I32 **Pattern Mask** The PatternPolarity, PatternTransition and PatternMask define the match condition for the Port Mode Specification AND / OR modes.

For example one can define a match condition as bit 1, bit 2 and bit 3 all set = "1" as a valid match (AND mode) by setting the following.

Port Mode Specification = AND
 PatternPolarity = (b) 1110
 PatternTransition = (b) 0
 PatternMask = (b) 1110

Alternately, one can define a match condition as any one or more of bit 1, bit 2 and bit 3 having set = "1" by changing the Port Mode Specification to OR mode.

PM(n)	PT(n)	PP(n)	Pattern Match Bit Definitions
0	0	0 or 1	Bit Masked Off
0	1	0 or 1	Any Transition
1	0	0	ZERO
1	0	1	ONE
1	1	0	ONE to ZERO Transition
1	1	1	ZERO to ONE Transition

For detailed definitions see the 5600 manuals ADVANCED PROGRAMMING TECHNIQUES. When the PortResolution=16, PatternPolarity, PatternTransition and PatternMask are set in WORDs b(1111111111111111).

I32 **Ignore IP Error** The Ignore IP Error setting controls the ADAC-LVi driver pattern match error condition. A setting of TRUE stops the generation of an error condition when a match condition occurs while another match is being serviced, indicating match conditions are occurring faster than they can be serviced.

I32 **Time Constant** The Time Constant value sets the time required for a stable input. Use the following equation to determine the value.

$$\text{TimeConstant} = \text{period(usec)} / 2(\text{usec})$$

ADAC Set 5600 Port Struct (con't.)

Were period(usec) is the debounce time required.

For detailed definitions see the 5600 manuals ADVANCED PROGRAMMING TECHNIQUES.



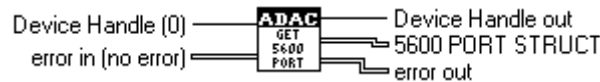
error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.



Device Handle out Device Handle out contains the value of Device Handle in.



error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Get 5600 Port Struct

This VI gets the digital I/O port settings for the 5600 Series cards. Refer to the 5600 manual for detailed information on each setting. This VI is only used with the 5600 Series.



Device Handle (0) Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.



error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.



Device Handle out Device Handle out contains the value of Device Handle in.



5600 PORT STRUCT



Port Resolution Port Resolution is a numeric value that represents the desired number of DIO port bits, either 8 or 16. The DIO ports DIN0 can be either 8 or 16 bit ports and DIN1 can only be an 8 bit port. When DIN0 is set for 16 bit

ADAC Get 5600 Port Struct (cont.)

resolution DIN1 is not available. The DIO ports DOT0 and DOT2 can be either 8 or 16 bit ports and DOT1 and DOT3 can only be an 8 bit port. When DOT0 or DOT2 is set for 16 bit resolution DOT1 or DOT 3 is not available respectively.

I32 **Port Mask** Port Mask is a binary value that indicates which bits are to be read during interrupt transfers. If the associated bit in the Port Mask control is set ="1" the bit is read, unactive bits will return "0".

E05 A5600PORT

I32 **Debounce** Debounce is a numeric value that indicates if the device's debounce circuitry is enabled or disabled. When Enabled the Time Constant value sets the time required for a stable input.

0:DISABLED

1:ENABLED

The default value is 1:ENABLED

I32 **Port Mode Specification** Port Mode Specification defines the pattern match condition.

0:DISABLED Pattern match is disabled.

1:AND All unmasked bit conditions must occur.

2:OR Any unmasked bit conditions must occur.

The default Port Mode Specification is 2:OR

I32 **Pattern Latch** Pattern Latch controls the Latch On Pattern ability. If the Pattern Latch is enabled, the state of the input port will be latched when a match condition occurs. The data will be held until the port is read within its interrupt handler. If a subsequent match occurs (port goes from match condition to non-match condition to a new match condition) before the port is read within its interrupt handler, an IP error condition will occur stopping the acquisition. The Ignore IP Error control can be set to TRUE to disable the IP error checking in the driver, but data may be lost when match conditions are occurring faster than they can be serviced.

ADAC Get 5600 Port Struct (con't.)

I32 **Data Path Polarity** Data Path Polarity allows each bit for DIN0 to be individually inverted. If the associated bit is set "1", data read from that bit on DIN0 will be inverted. For detailed definitions see the 5600 manuals ADVANCED PROGRAMMING TECHNIQUES. When the Port Resolution is set for 16 bit mode, Data Path Polarity is set in WORDs b(1111111111111111).

I32 **Special IO Control** The SpecialIoControl controls the BIT CATCHER operation mode. Port DIN0 may be configured to return the state of the bit catcher, instead of the actual port data by setting the associated bit to "1" in the Special Io Control. If a bit is so programmed, the port will return "0" for that bit until a "1" occurs at the input. The port will then read back a "1" even if the "1" goes away. When the DIN0 port is serviced by the driver, each bit enabled and in the "1" state will read automatically reset and ready for the next input transition. For detailed definitions see the 5600 manuals ADVANCED PROGRAMMING TECHNIQUES. When the Port Resolution is set for 16 bit mode, Special Io Control is set in WORDs b(1111111111111111).

I32 **Pattern Polarity** The PatternPolarity, PatternTransition and PatternMask define the match condition for the Port Mode Specification AND / OR modes.

For example one can define a match condition as bit 1, bit 2 and bit 3 all set = "1" as a valid match (AND mode) by setting the following.

```
Port Mode Specification = AND
PatternPolarity = (b) 1110
PatternTransition = (b) 0
PatternMask = (b) 1110
```

Alternately, one can define a match condition as any one or more of bit 1, bit 2 and bit 3 having set = "1" by changing the Port Mode Specification to OR mode.

PM(n)	PT(n)	PP(n)	Pattern Match Bit Definitions
0	0	0 or 1	Bit Masked Off
0	1	0 or 1	Any Transition
1	0	0	ZERO
1	0	1	ONE
1	1	0	ONE to ZERO Transition
1	1	1	ZERO to ONE Transition

ADAC Get 5600 Port Struct (con't.)

For detailed definitions see the 5600 manuals ADVANCED PROGRAMMING TECHNIQUES. When the PortResolution=16, PatternPolarity, PatternTransition and PatternMask are set in WORDs b(1111111111111111).



Pattern Transition The PatternPolarity, PatternTransition and PatternMask define the match condition for the Port Mode Specification AND / OR modes.

For example one can define a match condition as bit 1, bit 2 and bit 3 all set = "1" as a valid match (AND mode) by setting the following.

Port Mode Specification = AND
 PatternPolarity = (b) 1110
 PatternTransition = (b) 0
 PatternMask = (b) 1110

Alternately, one can define a match condition as any one or more of bit 1, bit 2 and bit 3 having set = "1" by changing the Port Mode Specification to OR mode.

PM(n)	PT(n)	PP(n)	Pattern Match Bit Definitions
0	0	0 or 1	Bit Masked Off
0	1	0 or 1	Any Transition
1	0	0	ZERO
1	0	1	ONE
1	1	0	ONE to ZERO Transition
1	1	1	ZERO to ONE Transition

For detailed definitions see the 5600 manuals ADVANCED PROGRAMMING TECHNIQUES. When the PortResolution=16, PatternPolarity, PatternTransition and PatternMask are set in WORDs b(1111111111111111).

ADAC Get 5600 Port Struct (con't.)

I32 **Pattern Mask** The PatternPolarity, PatternTransition and PatternMask define the match condition for the Port Mode Specification AND / OR modes.

For example one can define a match condition as bit 1, bit 2 and bit 3 all set = "1" as a valid match (AND mode) by setting the following.

```
Port Mode Specification = AND
PatternPolarity = (b) 1110
PatternTransition = (b) 0
PatternMask = (b) 1110
```

Alternately, one can define a match condition as any one or more of bit 1, bit 2 and bit 3 having set = "1" by changing the Port Mode Specification to OR mode.

PM(n)	PT(n)	PP(n)	Pattern Match Bit Definitions
0	0	0 or 1	Bit Masked Off
0	1	0 or 1	Any Transition
1	0	0	ZERO
1	0	1	ONE
1	1	0	ONE to ZERO Transition
1	1	1	ZERO to ONE Transition

For detailed definitions see the 5600 manuals ADVANCED PROGRAMMING TECHNIQUES. When the PortResolution=16, PatternPolarity, PatternTransition and PatternMask are set in WORDs b(1111111111111111).

I32 **Ignore IP Error** The Ignore IP Error setting controls the ADAC-LVi driver pattern match error condition. A setting of TRUE stops the generation of an error condition when a match condition occurs while another match is being serviced, indicating match conditions are occurring faster than they can be serviced.

ADAC Get 5600 Port Struct (con't.)

I32 **Time Constant** The Time Constant value sets the time required for a stable input. Use the following equation to determine the value.

$$\text{TimeConstant} = \text{period}(\text{usec}) / 2(\text{usec})$$

Where period(usec) is the debounce time required.

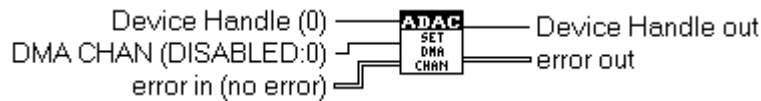
For detailed definitions see the 5600 manuals ADVANCED PROGRAMMING TECHNIQUES.

I51 **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC DMA Config Library

- ADAC Set DMA Chan
- ADAC Get DMA Chan

ADAC Set DMA Chan



This VI sets the hardware DMA channel of the device.



Device Handle (0) Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.



DMA CHAN (DISABLED:0) DMA Chan is a numeric value that sets the hardware DMA channel of the device. The available options are:

0:DISABLED	DMA disabled
1:DMA1	DMA channel 1
2:DMA2	DMA channel 2
3:DMA3	DMA channel 3
4:DMA5	DMA channel 5
5:DMA6	DMA channel 6
6:DMA7	DMA channel 7

The default DMA Chan is 0:DISABLED



error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

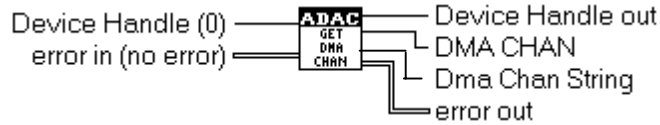


Device Handle out Device Handle out contains the value of Device Handle in.



error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Get DMA Chan



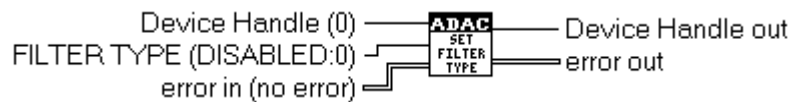
This VI gets the hardware DMA channel of the device. See the AL_SetDmaChan.VI for the available options.

- I32** **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.
- Err** **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
- I32** **Device Handle out** Device Handle out contains the value of Device Handle in.
- I32** **DMA CHAN** DMA Chan is a numeric value that sets the hardware DMA channel of the device. The available settings are:
 - 0:DISABLED DMA disabled
 - 1:DMA1 DMA channel 1
 - 2:DMA2 DMA channel 2
 - 3:DMA3 DMA channel 3
 - 4:DMA5 DMA channel 5
 - 5:DMA6 DMA channel 6
 - 6:DMA7 DMA channel 7
- abc** **DMA Chan String** DMA Channel String is a string value that represents the hardware DMA channel currently configured in the LDS.
- Err** **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Filter Config Library

- ADAC Set Filter Type
- ADAC Set Filter Freq
- ADAC Get Filter Struct

ADAC Set Filter Type



This VI sets the hardware filtering operation mode of the device. The mode can be disabled, Butterworth or Bessel. This VI is used only with ADAC 5508SCi.



Device Handle (0) Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.



FILTER TYPE (DISABLED:0) Filter Type is a numeric value that sets the hardware Filter operation mode of the device. The available options are:

0:DISABLED	Disable filtering
1:BUTTERWORTH	ButterWorth filter
2:BESSEL	Bessel filter

The default Filter Type is 0:DISABLED.



error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

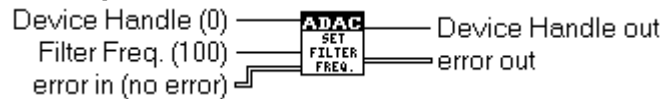


Device Handle out Device Handle out contains the value of Device Handle in.



error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

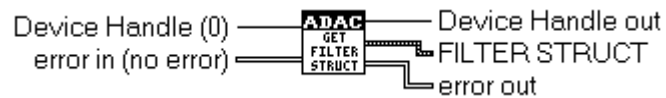
ADAC Set Filter Freq



This VI sets the hardware anti-alias filter clock frequency. The rate is specified in hertz. This VI is used only with ADAC 5508SCi.

- I32** **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.
- DBL** **Filter Freq. (100)** Filter Frequency is a double value that specifies rate at which anti-alias filter is configured at the factory.
- Err** **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
- I32** **Device Handle out** Device Handle out contains the value of Device Handle in.
- Err** **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Get Filter Struct



This VI provides access to the device's filter settings. See the associated Filter type setting VI for a full description of the FILTER STRUCT cluster. This VI is used only with ADAC 5508SCi.

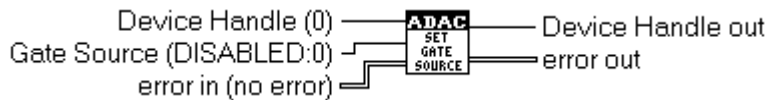
ADAC Get Filter Struct (cont.)

- I32** **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.
- E06** **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
- I32** **Device Handle out** Device Handle out contains the value of Device Handle in.
- E06** **FILTER STRUCT**
- I32** **Filter Type** Filter Type is a numeric value that represent the current hardware Filter operation mode of the device. The available options are:
- | | |
|---------------|--------------------|
| 0:DISABLED | Disable filtering |
| 1:BUTTERWORTH | ButterWorth filter |
| 2:BESSEL | Bessel filter |
- I32** **Filter Type ID** Filter Type ID is a numeric value that represents the current hardware Filter Type ID number defined in the devices capabilities file. This ID is used internally by the ADAC-LVi driver and is typically not used in LabVIEW VI's.
- D6L** **Filter Freq.** Filter Frequency is a double value that specifies rate at which anti-alias filter is configured at the factory.
- E06** **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Gate Config Library

- ADAC Set Gate Source
- ADAC Set Gate Level
- ADAC Set Software Gate
- ADAC Get Software Gate
- ADAC Get Gate Struct

ADAC Set Gate Source



This VI sets the hardware gating source of the device. The source can be either disabled swgate or extgate.



Device Handle (0) Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.



Gate Source (DISABLED:0) Gate Source is a numeric value that sets the hardware gate source of the device. The available options are:

0:DISABLED Hardware gating is disabled
 1:SWGATE Software control gating
 2:EXTGATE External gate control

The default Gate Source is 0:DISABLED



error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

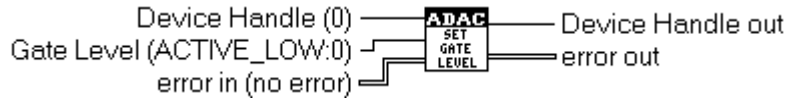


Device Handle out Device Handle out contains the value of Device Handle in.



error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Set Gate Level



This VI sets the hardware gating level of the device. The source can be either active_low or active_high.

I32 **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.

I32 **Gate Level (ACTIVE_LOW:0)** Gate Level is a numeric value that sets the hardware gate level of the device. The available options are:

0:ACTIVE_LOW Low level gate signal active
1:ACTIVE_HIGH High level gate signal active

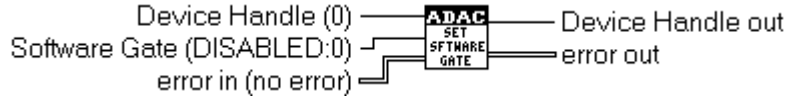
The default Gate Level is 0:ACTIVE_LOW

5.1 **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

I32 **Device Handle out** Device Handle out contains the value of Device Handle in.

5.1 **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Set Software Gate



This VI sets the hardware gate state of the device. The software gate can be either enabled or disabled.



Device Handle (0) Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.



Software Gate (DISABLED:0) Software Gate is a numeric value that sets the desired hardware gate state of the device. The available options are:

0:DISABLED The device is stopped
 1:ENABLED The device is enabled to run

The default SwGate state is 0:DISABLED



error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

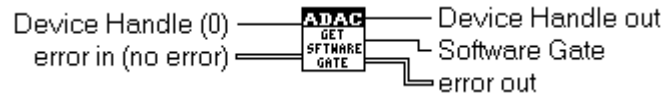


Device Handle out Device Handle out contains the value of Device Handle in.



error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Get Software Gate



This VI gets the hardware gate state of the device. The software gate can be either enabled or disabled.



Device Handle (0) Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.



error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.



Device Handle out Device Handle out contains the value of Device Handle in.



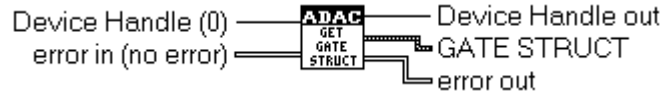
Software Gate Software Gate is a numeric value that represents the current hardware gate state of the device. The available states are:

- 0:DISABLED The device is stopped
- 1:ENABLED The device is enabled to run



error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Get Gate Struct



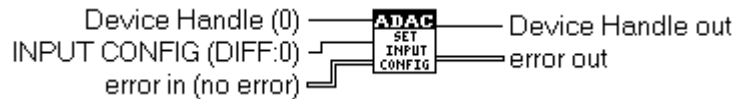
This VI provides access to the device's gate settings. See the associated gate setting VI for a full description of each setting returned to the GATE STRUCT cluster.

- I32** **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.
- F75** **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
- I32** **Device Handle out** Device Handle out contains the value of Device Handle in.
- 906** **GATE STRUCT**
 - I32** **Gate Source** Gate Source is a numeric value that represents the current hardware gate source of the device. The available options are:
 - 0:DISABLED Hardware gating is disabled
 - 1:SWGATE Software control gating
 - 2:EXTGATE External gate control
 - I32** **Gate Source ID** Gate Source ID is a numeric value that represents the current hardware Gate Source ID number defined in the devices capabilities file. This ID is used internally by the ADAC-LVi driver and is typically not used in LabVIEW VI's.
 - I32** **Gate Level** Gate Level is a numeric value that represents the current hardware gate level of the device. The available options are:
 - 0:ACTIVE_LOW Low level gate signal active
 - 1:ACTIVE_HIGH High level gate signal active
- F75** **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Input Config Library

- ADAC Set Input Config
- ADAC Get Input Config

ADAC Set Input Config



This VI sets the hardware input configuration of the device.

I32 **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.

I32 **INPUT CONFIG (DIFF:0)** INPUT Config is a numeric value that sets the hardware input configuration of the device. The available options are:

0:DIFFERENTIAL	Differential inputs
1:SINGLE-ENDED	Single-ended inputs
2:PSEUDO-DIFFERENTIAL	Pseudo-Differential inputs

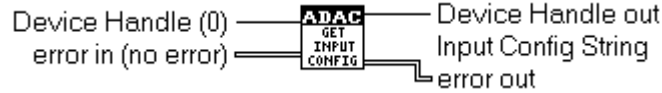
The default Input Config is 0:DIFFERENTIAL

5.1 **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

I32 **Device Handle out** Device Handle out contains the value of Device Handle in.

5.1 **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Get Input Config



This VI gets the hardware channel input configuration channel of the device. See the AL_SetInputConfig.VI for the available options.



Device Handle (0) Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.



error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.



Device Handle out Device Handle out contains the value of Device Handle in.



INPUT CONFIG INPUT Config is a numeric value that represents the hardware input configuration of the device. The available options are:

- | | |
|-----------------------|----------------------------|
| 0:DIFFERENTIAL | Differential inputs |
| 1:SINGLE-ENDED | Single-ended inputs |
| 2:PSEUDO-DIFFERENTIAL | Pseudo-Differential inputs |

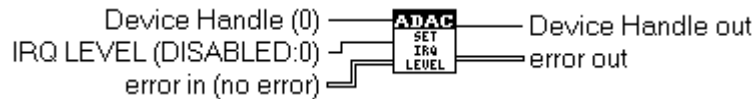


error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC IRQ (Interrupt) Config Library

- ADAC Set IRQ Level
- ADAC Get IRQ Level

ADAC Set IRQ Level



This VI sets the hardware IRQ level of the device.

Note:

The available settings are device dependant; see your ADAC hardware manual for valid settings.



Device Handle (0) Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.






IRQ LEVEL (DISABLED:0) IRQ LEVEL is a numeric value that sets the hardware IRQ level of the device. The available options are:

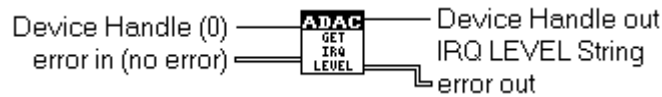
0:DISABLED	IRQ disabled
1:DISABLED	IRQ disabled
2:IRQ2	IRQ level 2
3:IRQ3	IRQ level 3
4:IRQ4	IRQ level 4
5:IRQ5	IRQ level 5
6:IRQ6	IRQ level 6
7:IRQ7	IRQ level 7
8:DISABLED	IRQ disabled
9:DISABLED	IRQ disabled
10:IRQ10	IRQ level 10
11:IRQ11	IRQ level 11
12:DISABLED	IRQ disabled
13:DISABLED	IRQ disabled
14:DISABLED	IRQ disabled
15:IRQ15	IRQ level 15

The default IRQ LEVEL is 0:DISABLED





ADAC Set IRQ Level (con't.)

-  **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
-  **Device Handle out** Device Handle out contains the value of Device Handle in.
-  **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Get IRQ Level



This VI gets the hardware IRQ level of the device. See the AL_SetIrqLevel.VI for the available options.

-  **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.
-  **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
-  **Device Handle out** Device Handle out contains the value of Device Handle in.
-  **IRQ LEVEL** IRQ LEVEL is a numeric value that represents the hardware IRQ level currently set in the device. The available options are:

0:DISABLED	IRQ disabled
1:DISABLED	IRQ disabled
2:IRQ2	IRQ level 2
3:IRQ3	IRQ level 3
4:IRQ4	IRQ level 4

ADAC Get IRQ Level

5:IRQ5	IRQ level 5
6:IRQ6	IRQ level 6
7:IRQ7	IRQ level 7
8:DISABLED	IRQ disabled
9:DISABLED	IRQ disabled
10:IRQ10	IRQ level 10
11:IRQ11	IRQ level 11
12:DISABLED	IRQ disabled
13:DISABLED	IRQ disabled
14:DISABLED	IRQ disabled
15:IRQ15	IRQ level 15

The default IRQ LEVEL is 0:DISABLED

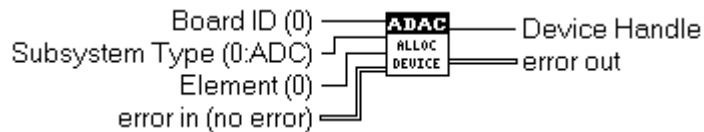


error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Device Function Library

- ADAC Allocate Device
- ADAC Release Device
- ADAC Set LDS String
- ADAC Get LDS String
- ADAC Init Device
- ADAC Start Device
- ADAC Stop Device
- ADAC Get Device Status
- ADAC Set Cycle Mode
- ADAC Set DTM (Data Transfer Method)
- ADAC Get Cycle Mode
- ADAC Get DTM (Data Transfer Method)

ADAC Allocate Device



The VI function allocates and loads a device subsystem into the ADAC-LVi logical device subsystem database (LDS). If the ADAC-LVi information file exists (ADAC-LVi.INI) as specified in the (ADAC-LVi.CON) file, then the LDS is configured per the settings specified in the (ADAC-LVi.INI) file. If the ADAC-LVi information file does NOT exist or a configuration parameter is not indicated within the (ADAC-LVi.INI) file, the parameter will be set to its associated capabilities file default(s) if available, otherwise the parameter is set to an ADAC-LVi internal default. See the individual function listing for its default behavior.

Multiple configurations for the same hardware device subsystem can also be allocated, but only one can be initialized and started at any given time.

ADAC Allocate Device (con't)

I32 **Board ID (0)** Specifies the ADAC-LVi board ID number from which the device subsystem is to be allocated. The board ID number is defined in the ADAC-LVi configuration file (.CON).

U16 **Subsystem Type (0:ADC)** Subsystem Type is a numeric value that specifies the board subsystem to be allocated. The available types are board dependent and are specified in the [Board] entry section of the specific board's capabilities (.CAP) file.

The available options are:

0:ADC Specifies a specific analog input subsystem on the board to be allocated.

1:DAC Specifies a specific analog output subsystem on the board to be allocated.

2:DIN Specifies a specific digital input subsystem on the board to be allocated.

3:DOT Specifies a specific digital output subsystem on the board to be allocated.

4:COUNTER Specifies a specific COUNTER subsystem on the board to be allocated.

The default Subsystem Type is 0:ADC

U16 **Element (0)** Element is a numeric value that specifies element number of the Subsystem Type. Elements range from 0-n for all Subsystem Types.

Most ADC Subsystems have only a single element (0), so the value is typically zero (0).

The other Subsystem Types DAC, DIN, DOT, COUNTER typically have multiple elements, the first subsystem element is 0, the second is 1, and so forth.

The default Element is 0.

E32 **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

I32 **Device Handle** Device Handle is a numeric value that is used to represent the board in subsequent VI calls to the allocated subsystem.

E32 **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Release Device



This VI releases a Logical Device Subsystem Database (LDS) and all associated memory previously allocated to the device. Further calls to the device subsystem will return an invalid LHL error code, all previously returned identifiers from the LDS to the VI that provide access to the device subsystem or its data buffers are considered invalid and must not be used by the VI program.

Note:

If the device is currently running, an error flag setting determines whether to stop and then release, or to return a non-critical error and keep the device running. See the advanced VI function `AL_SetErrorOnReleaseRunning` for more information.



Device Handle (0) Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.

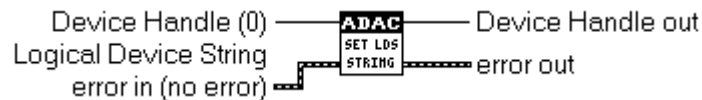


error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.








error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Set LDS String

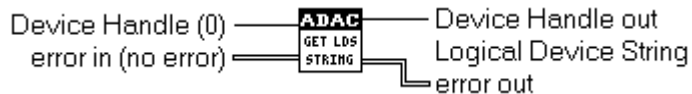


This VI sets user defined device description string.






ADAC Set LDS String

-  **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.
-  **Logical Device String** Specifies the user-defined Logical Device Subsystem (LDS) description string.
-  **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
-  **Device Handle out** Device Handle out contains the value of Device Handle in.
-  **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Get LDS String



This VI returns the device's user defined description string.

-  **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.
-  **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
-  **Device Handle out** Device Handle out contains the value of Device Handle in.
-  **Logical Device String** Contains the device's user defined description string.
-  **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Init Device



This VI initializes the actual hardware device to the current Logical Device Subsystem Database (LDSB) settings. This function must be called before attempting to call the AL_StartDevice VI call, failure to do so will result in an error. Once started, all ADAC-LVi calls that affect the functional state of the hardware device are still allowed, but the new setting is only reflected in LDSB, and **NOT** in the actual hardware device itself. ADAC-LVi VI's that control the running state are reflected in both the LDSB and actual hardware device itself. To update the actual state of the hardware device with any changes made to the LDSB, AL_InitDevice function must be called once again. This allows changes to be made to the functionality of the LDSB while the hardware device is still running. Multiple configurations for the same hardware device subsystem can also be allocated with AL_AllocateDevice, but only one can be initialized and started at any given time.

Note:





If the device is currently running, an error flag setting determines whether to stop and then release, or to return a non-critical error and keep the device running. See the advanced VI function AL_SetErrorOnReleaseRunning for more information.

- I32** **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.
- F7** **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
- I32** **Device Handle out** Device Handle out contains the value of Device Handle in.
- F7** **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Start Device







This VI puts a device in the run mode. Depending on the current ADAC-LVi transfer mode, this call may not return until completed. See *AL_SetDataTransferMode.vi* for more details on foreground and background operations.

-  **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.
-  **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
-  **Device Handle out** Device Handle out contains the value of Device Handle in.
-  **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

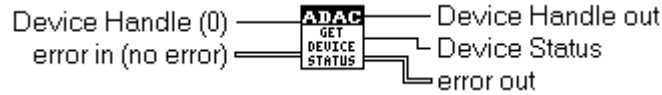
ADAC Stop Device



This VI disables the current acquisition, and all logical device operations are stopped.

-  **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem being stopped.. The default Device Handle is 0.
-  **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
-  **Device Handle out** Device Handle out contains the value of Device Handle in.
-  **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

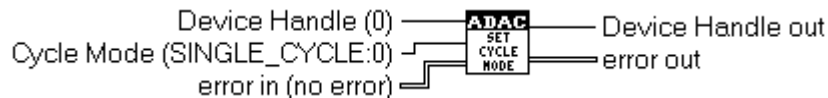
ADAC Get Device Status



This VI gets the current running Device Status of the device.

- I32** **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.
- F32** **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
- I32** **Device Handle out** Device Handle out contains the value of Device Handle in.
- I32** **Device Status** Device Status is a numeric value that represents the current hardware status of the device. The status is either:
 - 0:Stopped, or
 - 1:Running.
- F32** **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Set Cycle Mode



This VI sets the software cycle operation mode of the device. The mode can be either single cycle or continuous.

ADAC Set Cycle Mode (cont.)

I32 **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.

I32 **Cycle Mode (SINGLE_CYCLE:0)** Cycle Mode is a numeric value that sets the software cycle operation mode of the device. The available options are:

0:SINGLE_CYCLE complete acquisition and stop

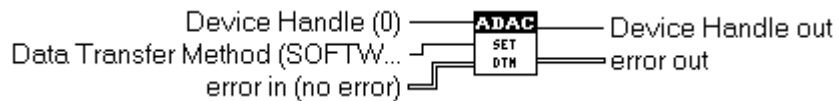
1:CONTINUOUS_CYCLE repeat acquisition

The default Cycle Mode is 0:SINGLE_CYCLE.

E7 **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

I32 **Device Handle out** Device Handle out contains the value of Device Handle in.

E7 **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Set DTM (Data Transfer Method)

This VI sets the Data Transfer Method of the device. The mode can be SOFTWARE, DMA, IRQ or C40PORT.

ADAC Set DTM (Data Transfer Method) (con't.)

I32 **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.

I32 **Data Transfer Method (SOFTWARE:0)** Data Transfer Method is a numeric value that sets the hardware transfer method of the device. The available options are:

- 0:SOFTWARE Software transfers
- 1:DMA DMA transfers
- 2:IRQ Interrupt transfers
- 3:C40PORT C40 port transfers

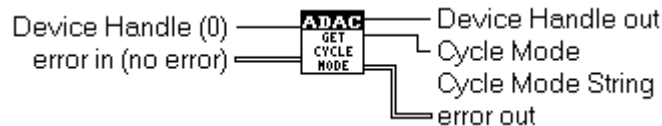
The default Data Transfer Method is 0:SOFTWARE.

E4 **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

I32 **Device Handle out** Device Handle out contains the value of Device Handle in.

E4 **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Get Cycle Mode






This VI gets the software cycle operation mode of the device. See the AL_SetCycleMode.VI for the available options.

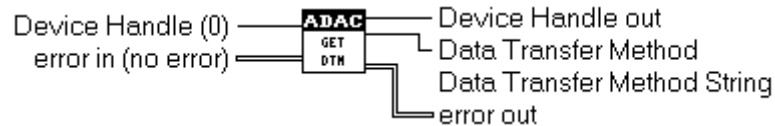
I32 **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.

E4 **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.






ADAC Get Cycle Mode (con't)

-  **Device Handle out** Device Handle out contains the value of Device Handle in.
-  **Cycle Mode** Cycle Mode is a numeric value that represents the software cycle operation mode currently configured in the device.
-  **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Get DTM (Data Transfer Method)



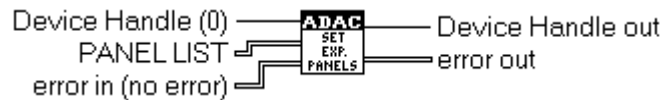
This VI gets the Data Transfer Method of the device. See the AL_SetDataTransfeMethod.VI for the available options.

-  **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.
-  **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
-  **Device Handle out** Device Handle out contains the value of Device Handle in.
-  **Data Transfer Method** Data Transfer Method is a numeric value that represents the hardware transfer method currently configured in the device.
-  **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Panel Config Library

- ADAC Set Expansion Panels
- ADAC Set Expansion Panel Gains
- ADAC Get Expansion Panel Structure
- ADAC Get Expansion Panel Gains Structure

ADAC Set Expansion Panels



This VI function sets the board's input panel type(s) in the LDS. The available panel type(s) are specified in the device capabilities file.



Device Handle (0) Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.



PANEL LIST



Array Type Array Type is a numeric value that specifies the type of panel list in use. The available options are:

- 0:STRING_LIST Character string list of panels
- 1:ARRAY_LIST Numeric array of panels

The default array type is 0:STRING_LIST.

Examples:

- String "PNLNAME-TC, PNLNAME-TC, PNLNAME-16, PNLNAME-32"
- Array {0,0,1,2};



String Panel List String List of Panel types. The Panel names are defined in the devices capabilities file.

Examples:

- String "PNLNAME-TC, PNLNAME-TC, PNLNAME-16, PNLNAME-32"

Set Expansion Panels (con't)

[I32] **Numeric Panel List** Numeric array of Panel types. The Panel ID numbers are defined in the devices capabilities file.

Examples:

Array {0,0,1,2};



error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

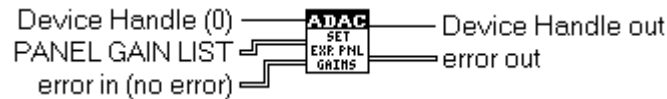


Device Handle out Device Handle out contains the value of Device Handle in.



error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Set Expansion Panel Gains



This VI function sets the board's input panel channel gain(s) for the specified device. The available panel gains are typically fixed in hardware and are specified in the device capabilities file.



Device Handle (0) Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.



PANEL GAIN LIST



Array Type Array Type is a numeric value that specifies the type of panel gain list in use. The available options are:

0:STRING_LIST	Character string list of panel gains
1:ARRAY_LIST	Numeric array of panel gains

Set Expansion Panel Gains (con't)

The default array type is 0:STRING_LIST.

Examples:

- String "0(100),1(100),2(100),3(100),4(200),5(250),6(250),7(250)"
- Array {0,100,1,100,2,100,3,100,4,200,5,250,6,250,7,250};



String Panel Gain List String Panel Gain List specifies the panels fixed gain in a character sting format.

Examples:

String "0(100),1(100),2(100),3(100),4(200),5(250),6(250),7(250)"



Numeric Panel Gain List Numeric Panel Gain List specifies the panels fixed gain in a numeric array format.

Examples:

Array {0,100,1,100,2,100,3,100,4,200,5,250,6,250,7,250};



error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

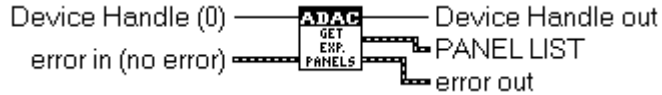


Device Handle out Device Handle out contains the value of Device Handle in.



error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Get Expansion Panel Structure



This VI function gets the board's input panel type(s) in the LDS. The available panel type(s) are specified in the device capabilities file.



Device Handle (0) Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.



error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.



Device Handle out Device Handle out contains the value of Device Handle in.



PANEL LIST



Array Type Array Type is a numeric value that represents the type of panel list in use. The available options are:

0:STRING_LIST	Character string list of panels
1:ARRAY_LIST	Numeric array of panels

Examples:

- String "PNLNAME-TC, PNLNAME-TC, PNLNAME-16, PNLNAME-32"
- Array {0,0,1,2};



String Panel List String Panel List represents the current panels. The actual panel names are defined in the devices capabilities file.

Examples:

String "PNLNAME-TC, PNLNAME-TC, PNLNAME-16, PNLNAME-32"



Numeric Panel List Numeric Panel List represents the current panels. The actual panel ID numbers are defined in the devices capabilities file.

Examples:

- Array {0,0,1,2};



error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Get Expansion Panel Gain Structure



This VI function gets the board's input panel fixed gain(s) for the specified device. The available panel gain(s) are typically fixed in hardware and specified in the device capabilities file.

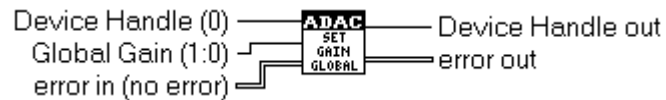
- [I32]** **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.
- [E]** **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
- [I32]** **Device Handle out** Device Handle out contains the value of Device Handle in.
- [E]** **PANEL GAIN LIST**
 - [I32]** **Array Type** Array Type is a numeric value that specifies the type of panel gain list in use. The available options are:

0:STRING_LIST	Character string list of panel gains
1:ARRAY_LIST	Numeric array of panel gains
 - The default array type is 0:STRING_LIST.
 - Examples:
 - String "0(100),1(100),2(100),3(100),4(200),5(250),6(250),7(250)"
 - Array {0,100,1,100,2,100,3,100,4,200,5,250,6,250,7,250};
 - [abc]** **String Panel Gain List** String Panel Gain List specifies the panels fixed gain in a character sting format. Examples:
 - String "0(100),1(100),2(100),3(100),4(200),5(250),6(250),7(250)"
 - [I32]** **Numeric Panel Gain List** Numeric Panel Gain List specifies the panels fixed gain in a numeric array format. Examples:
 - Array {0,100,1,100,2,100,3,100,4,200,5,250,6,250,7,250};
- [E]** **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Range Config Library

- ADAC Set Gain Global
- ADAC Set Data Code
- ADAC Get Data Code
- ADAC Set Data Offset
- ADAC Get Data Offset
- ADAC Set Data Range
- ADAC Get Data Range

ADAC Set Gain Global



This VI globally sets all hardware channel gain codes of the device. The available gain settings are device dependant, see your ADAC hardware manual for valid settings



Device Handle (0) Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.





Global Gain (1:0) Global Gain is a numeric value that globally sets all channel list gains to the specified gain setting. The available options are:


0:1	Gain of 1
1:2	Gain of 2
2:4	Gain of 4
3:5	Gain of 5
4:8	Gain of 8
5:10	Gain of 10
6:100	Gain of 100
7:500	Gain of 500

The default Global Gain is 0:1

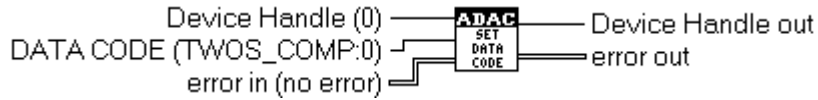
ADAC Set Gain Global

 **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.


 **Device Handle out** Device Handle out contains the value of Device Handle in.


 **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Set Data Code




This VI sets the hardware data code configuration of the device.


 **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.


 **DATA CODE (TWOS_COMP:0)** DATA CODE is a numeric value that sets the hardware data coding configuration of the device. The available options are:

- 0:TWOS_COMPLEMENT Two's Complement data coding
- 1:STRAIGHT_BINARY Straight Binary data coding
- 2:OFFSET_BINARY Offset Binary data coding

The default DATA CODE is 0:TWOS_COMPLEMENT

 **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

 **Device Handle out** Device Handle out contains the value of Device Handle in.

 **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

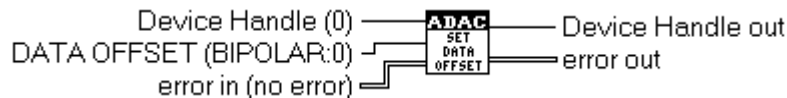
ADAC Get Data Code



This VI gets the hardware data code setting of the specified device.

- I32** **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.
- F75** **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
- I32** **Device Handle out** Device Handle out contains the value of Device Handle in.
- I32** **DATA CODE** DATA CODE is a numeric value that represents the hardware data coding configuration of the device.
The available options are:
 - 0:TWOS_COMPLEMENT Two's Complement data coding
 - 1:STRAIGHT_BINARY Straight Binary data coding
 - 2:OFFSET_BINARY Offset Binary data coding
- F75** **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Set Data Offset



This VI sets the hardware data offset configuration of the device.

ADAC Set Data Offset (con't.)

I32 **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.

I32 **DATA OFFSET (BIPOLAR:0)** DATA OFFSET is a numeric value that sets the hardware data offset configuration of the device.
The available options are:

0:BIPOLAR	Bipolardata +/- V
1:UNIPOLAR	Unipolar data 0-V

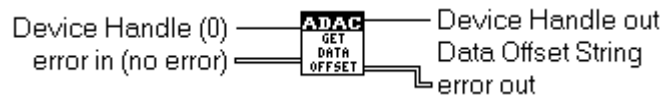
The default DATA OFFSET is 0:BIPOLAR

Err **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

I32 **Device Handle out** Device Handle out contains the value of Device Handle in.

Err **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Get Data Offset



This VI gets the hardware data offset of the device. See the AL_SetDataOffset.VI for the available options.

I32 **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.

Err **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

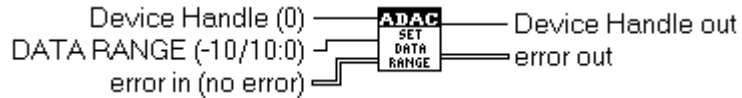
I32 **Device Handle out** Device Handle out contains the value of Device Handle in.

ADAC Get Data Offset (con't.)

I32 **DATA OFFSET** DATA OFFSET is a numeric value that represents the hardware data offset configuration of the device. The available options are:

0:BIPOLAR Bipolar data +/-V
1:UNIPOLAR Unipolar data 0-V

E4 **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Set Data Range

This VI sets the hardware data voltage range of the device.

I32 **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.

I32 **DATA RANGE (-10/10:0)** DATA RANGE is a numeric value that sets the hardware voltage data range of the device. The available options are:

0:-5_5 +/- 5V
1:0_10 + 0-10V
2:-10_10 +/- 10V

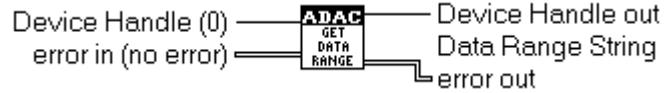
The default DATA OFFSET is 2:-10_10

E4 **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

I32 **Device Handle out** Device Handle out contains the value of Device Handle in.

E4 **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Get Data Range



This VI gets the hardware data voltage range of the device from the ADAC driver. See the `AL_SetDataRange.VI` for the available options.

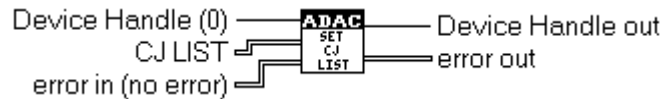
- I32**
Device Handle (0) Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.
- E7**
error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
- I32**
Device Handle out Device Handle out contains the value of Device Handle in.
- I32**
DATA RANGE DATA RANGE is a numeric value that represents the current hardware voltage data range of the device.
 The available options are:

0:-5_5	+/- 5V
1:0_10	+ 0-10V
2:-10_10	+/- 10V
- E7**
error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC CJ (Cold Junction) Config Library

- ADAC Set CJ List
- ADAC Set CJ Global
- ADAC Get CJ List

ADAC Set CJ List



This VI function sets the thermocouple list settings in the LDS. The CJ listing is comprised of a channel number and CJ type. The CJ list may contain channel/CJ settings that are not currently set in the channel listing itself, only those channels selected within the channel list will be set to the appropriate CJ type when running. The CJ ID type setting or the Numeric CJ List can be obtained from the capabilities file.



Device Handle (0) Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.



CJ LIST



Array Type Array Type is a numeric value that specifies the type of CJ list in use. The available options are:

0:STRING_LIST	Character string list of CJ types
1:ARRAY_LIST	Numeric array of CJ types

The default CJ Type is 0:STRING_LIST.

Examples:

string

"0(BNBS),1(JNBS),2(KNBS),3(SNBS),4(ENBS),5(TNBS),6(RNBS),7(BNBS)"

"

array {0,1000, 1,1001, 2,1002, 3,1003, 4,1004, 5,1005, 6,1006, 7,1007 }

ADAC Set CJ List

[abc] **String CJ List** CJ List specifies the desired CJ types in a character format. The CJ names are defined in the devices capabilities file.

Examples:

string

"0(BNBS),1(JNBS),2(KNBS),3(SNBS),4(ENBS),5(TNBS),6(RNBS),7(BNBS)"

[I32] **Numeric CJ List** Numeric array of CJ types. The CJ ID numbers are defined in the devices capabilities file.

Examples:

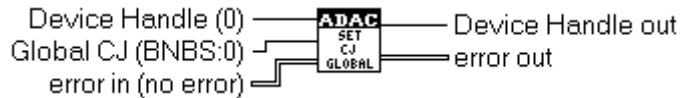
Array {0,1000, 1,1001, 2,1002, 3,1003, 4,1004, 5,1005, 6,1006, 7,1007}

[Err] **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

[I32] **Device Handle out** Device Handle out contains the value of Device Handle in.

[Err] **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Set CJ Global



This VI globally sets all CJ list thermocouple types to the specified Global CJ.

[I32] **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.

ADAC Set CJ Global (con't)

- I32** **Global CJ (BNBS:0)** Global CJ is a numeric value that globally sets all CJ channels to the specified CJ setting. The available options are:
- | | | | |
|--------|--------|--------------|----------|
| 0:BNBS | TYPE B | Thermocouple | (NON CJ) |
| 1:JNBS | TYPE J | Thermocouple | (NON CJ) |
| 2:KNBS | TYPE K | Thermocouple | (NON CJ) |
| 3:ENBS | TYPE E | Thermocouple | (NON CJ) |
| 4:TNBS | TYPE T | Thermocouple | (NON CJ) |
| 5:RNBS | TYPE R | Thermocouple | (NON CJ) |
| 6:SNBS | TYPE S | Thermocouple | (NON CJ) |

The default Global CJ is 0:BNBS

- E75** **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
- I32** **Device Handle out** Device Handle out contains the value of Device Handle in.
- E75** **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Get CJ List

This VI function retrieves the current CJ settings in the ADAC driver. The CJ listing is comprised of a channel number and CJ type. The CJ list may contain channel/CJ settings that are not currently set in the channel listing itself. The CJ ID type setting in the array can be obtained from the capabilities file.

- I32** **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.
- E75** **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.
- I32** **Device Handle out** Device Handle out contains the value of Device Handle in.

ADAC Get CJ List (con't)

CJ LIST

[I32] **Array Type** Array Type is a numeric value that specifies the type of CJ list in use. The available options are:
0:STRING_LIST Character string list of CJ types
1:ARRAY_LIST Numeric array of CJ types

The default CJ Type is 0:STRING_LIST.

Examples:

string "0(BNBS),1(JNBS),2(KNBS),3(SNBS),4(ENBS),5(TNBS),

6(RNBS),7(BNBS)"

array {0,1000, 1,1001, 2,1002, 3,1003, 4,1004, 5,1005, 6,1006, 7,1007 }

[abc] **String CJ List** CJ List specifies the current CJ types in a character format. The CJ names are defined in the devices capabilities file.

Examples:

string

"0(BNBS),1(JNBS),2(KNBS),3(SNBS),4(ENBS),5(TNBS),6(RNBS),7(BNBS)

"

[I32] **Numeric CJ List** Numeric array of CJ types. The CJ ID numbers are defined in the devices capabilities file.

Examples:

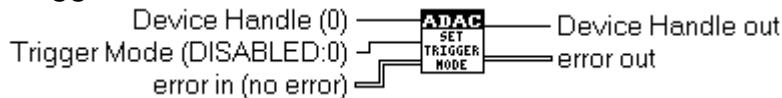
Array {0,1000, 1,1001, 2,1002, 3,1003, 4,1004, 5,1005, 6,1006, 7,1007 }

[Err] **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Trigger Config Library

- ADAC Set Trigger Mode
- ADAC Set Trigger Source
- ADAC Set Trigger Source Signal
- ADAC Set Trigger Rate
- ADAC Set Post Sample Count
- ADAC Get Trigger Structure

ADAC Set Trigger Mode



This VI sets the hardware trigger operation mode of the device. The mode can be disabled, about_trig, post_trig, pre_trig or scan_trig.



Device Handle (0) Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.



Trigger Mode (DISABLED:0) Trigger Mode is a numeric value that sets the hardware trigger operation mode of the device. The available options are:

0:DISABLED	All triggering is disabled
1:ABOUT_TRIG	Collect (N) data points before and (N) data points after a trigger
2:POST_TRIG	Collect (N) data points after a trigger
3:PRE_TRIG	Collect (N) data points before a trigger
4:SCAN_TRIG	Collect 1 data point for each channel

The default Trigger Mode is 0:DISABLED



error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

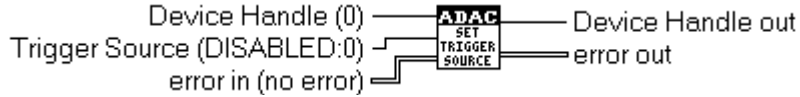


Device Handle out Device Handle out contains the value of Device Handle in.



error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Set Trigger Source



This VI sets the hardware trigger source mode of the device. The source can be disabled, external, ctr1 or ctr2.

I32 **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.

I32 **Trigger Source (DISABLED:0)** Trigger Source is a numeric value that sets the hardware triggering source of the device. The available options are:

- | | |
|------------|--|
| 0:DISABLED | All triggering sources are disabled |
| 1:EXTERNAL | Trigger the acquisition from an external event |
| 2:CTR1 | On-board counter 1 triggers the acquisition |
| 3:CTR2 | On-board counter 2 triggers the acquisition |

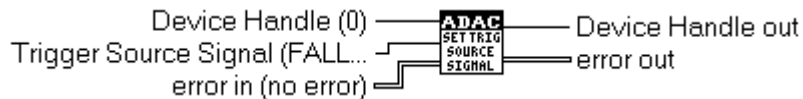
The default Trigger Source is 0:DISABLED

F7 **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

I32 **Device Handle out** Device Handle out contains the value of Device Handle in.

F7 **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Set Trigger Source Signal



This VI sets the hardware trigger source signal of the device. The source signal can be either falling_edge or rising_edge.

ADAC Set Trigger Source Signal (con't.)

I32 **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.

I32 **Trigger Source Signal (FALLING_EDGE:0)** Trigger Source Signal is a numeric value that sets the hardware triggering source signal of the device. The available options are:

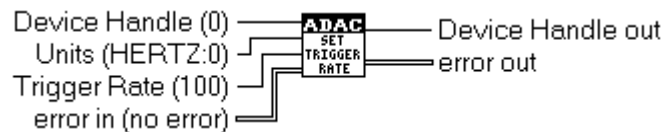
0:FALLING_EDGE Trigger on the falling edge of an event
1:RISING_EDGE Trigger on the rising edge of an event

The default Trigger Source Signal is 0:FALLING_EDGE

Err **error in (no error)** Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

I32 **Device Handle out** Device Handle out contains the value of Device Handle in.

Err **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Set Trigger Rate

This VI sets the hardware trigger rate of the device. The rate can be specified in hertz or tics.

I32 **Device Handle (0)** Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.

I32 **Units (HERTZ:0)** Units is a numeric value that specifies base units in which the Trigger Rate parameter is specified. The available options are:

0:HERTZ Rate is specified in units of hertz

ADAC Set Trigger Rate (con't)

1:TICS Rate is the divisor to the trigger clock source

If TICS is chosen, the Trigger Rate parameter is usually set as divisor of the Trigger Source clock selected.

The default Unit is 0:HERTZ



Trigger Rate (100) Trigger Rate is a double value that specifies rate at which triggers occur.



error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.

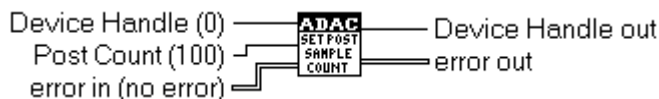


Device Handle out Device Handle out contains the value of Device Handle in.



error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Set Post Sample Count



This VI specifies the post trigger samples in the device. This function is made available for hardware devices that provide a post trigger counting mechanism. When the hardware device receives a trigger, it provides (n) more samples, then stops. The post trigger samples specifies the number of data samples to be obtained after a valid hardware trigger has been received.



Device Handle (0) Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.



Post Count (100) Post Count is a numeric value that sets the number of samples to be obtained after a trigger has been received. The available range is device dependent and is verified against the min./max. range specified in the device's capabilities file. Post Count is used with ABOUT_TRIG and PRE_TRIG trigger modes. The default value is 100 post samples.

ADAC Set Post Sample Count (con't)

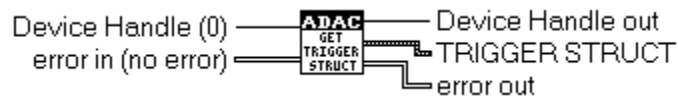
error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.



Device Handle out Device Handle out contains the value of Device Handle in.



error out Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

ADAC Get Trigger Structure

This VI provides access to the device trigger settings. See the associated trigger setting VI for a full description of the TRIGGER STRUCT cluster.



Device Handle (0) Device Handle is a numeric value that is used to identify the device subsystem. The default Device Handle is 0.



error in (no error) Error in describes error conditions occurring before this VI executes. This cluster defaults to no error.



Device Handle out Device Handle out contains the value of Device Handle in.

**TRIGGER STRUCT**

Trigger Mode Trigger Mode is a numeric value that sets the hardware trigger operation mode of the device. The available options are:

- | | |
|--------------|--|
| 0:DISABLED | All triggering is disabled |
| 1:ABOUT_TRIG | Collect (N) data points before and (N) data points after a trigger |
| 2:POST_TRIG | Collect (N) data points after a trigger |
| 3:PRE_TRIG | Collect (N) data points before a trigger |
| 4:SCAN_TRIG | Collect 1 data point for each channel |

The default Trigger Mode is 0:DISABLED

ADAC Get Trigger Structure (con't)

I32 **Trigger Mode ID** Trigger Mode ID is a numeric value that represents the current hardware trigger mode ID number defined in the devices capabilities file. This ID is used internally by the ADAC-LVi driver and is typically not used in LabVIEW VI's.

I32 **Trigger Source** Trigger Source is a numeric value that sets the hardware triggering source of the device. The available options are:

0:DISABLED	All triggering sources are disabled
1:EXTERNAL	Trigger the acquisition from an external event
2:CTR1	On-board counter 1 triggers the acquisition
3:CTR2	On-board counter 2 triggers the acquisition

The default Trigger Source is 0:DISABLED

I32 **Trigger Source ID** Trigger Source ID is a numeric value that represents the current hardware trigger source ID number defined in the devices capabilities file. This ID is used internally by the ADAC-LVi driver and is typically not used in LabVIEW VI's.

I32 **Trigger Source Signal** Trigger Source Signal is a numeric value that sets the hardware triggering source signal of the device. The available options are:

0:FALLING_EDGE	Trigger on the falling edge of an event
1:RISING_EDGE	Trigger on the rising edge of an event

The default Trigger Source Signal is 0:FALLING_EDGE

I32 **Trigger Source Signal ID** Trigger Source Signal ID is a numeric value that represents the current hardware trigger source signal ID number defined in the devices capabilities file. This ID is used internally by the ADAC-LVi driver and is typically not used in LabVIEW VI's.

I32 **Trigger Rate Units** Trigger Rate Units is a numeric value that specifies base units in which the Trigger Rate parameter is specified. The available options are:

0:HERTZ	Rate is specified in units of hertz
1:TICS	Rate is the divisor to the trigger clock source

If TICS is chosen, the Trigger Rate parameter is usually set as divisor of the Trigger Source clock selected.

ADAC Get Trigger Structure (con't.)

DBL **Trigger Rate** Trigger Rate is a double value that specifies rate at which triggers occur.

I32 **Post Sample Count** Post Count is a numeric value that sets the number of samples to be obtained after a trigger has been received. The available range is device dependent and is verified against the min./max. range specified in the device's capabilities file.

Post Count is used with ABOUT_TRIG and PRE_TRIG trigger modes.

ERR **error out** Error out contains error information. If the error in cluster indicated an error, the error out cluster contains the same information. Otherwise, error out describes the error status of this VI.

3

Supported Boards and Subsystems Listed by VI Functions

Easy Analog Input VIs

Adac AI Acquire Waveform.vi & Adac AI Acquire Waveforms.vi

Supported Boards:

5401DMF, 5402DMF, 5403DHR, 5404DHR, 5500HR, 5508SHR, 5508LC, 5508LF, 5508SCi, 5516DMA, 5525MF, 5550MF, 5801MF, 5802MF, 5803HR and 5804HR, 4012AD, 4112AD, PCI5500, PCI5501, PCI5502, PCI5503, PCI5504

Adac AI Sample Channel.v & Adac AI Sample Channels.vi

Supported Boards:

5401DMF, 5402DMF, 5403DHR, 5404DHR, 5500HR, 5508SHR, 5508LC, 5508LF, 5508SCi, 5516DMA, 5525MF, 5550MF, 5801MF, 5802MF, 5803HR 5804HR, 5508BG, 5508TC, 5508RTD, 5500MF, 4012AD, 4112AD, PCI5500, PCI5501, PCI5502, PCI5503, PCI5504

Intermediate Analog Input VIs

Adac AI Config.vi & Adac AI Clear.vi & Adac AI Read.vi & Adac AI Start.vi

Supported Boards:

5401DMF, 5402DMF, 5403DHR, 5404DHR, 5500HR, 5508SHR, 5508LC, 5508LF, 5508SCi, 5516DMA, 5525MF, 5550MF, 5801MF, 5802MF, 5803HR 5804HR, 5508BG, 5508TC, 5508RTD, 5500MF, 4012AD, 4112AD, PCI5500, PCI5501, PCI5502, PCI5503, PCI5504

Analog Input Utility VIs

Adac Read One Scan.vi

Supported Boards:

5401DMF, 5402DMF, 5403DHR, 5404DHR, 5500HR, 5508SHR, 5508LC, 5508LF, 5508SCi, 5516DMA, 5525MF, 5550MF, 5801MF, 5802MF, 5803HR 5804HR, 5508BG, 5508TC, 5508RTD, 5500MF, 4012AD, 4112AD, PCI5500, PCI5501, PCI5502, PCI5503, PCI5504

Adac AI Waveform Scan.vi & Adac AI Continuous Scan.vi

Supported Boards:

5401DMF, 5402DMF, 5403DHR, 5404DHR, 5500HR, 5508SHR, 5508LC, 5508LF, 5508SCi, 5516DMA, 5525MF, 5550MF, 5801MF, 5802MF, 5803HR and 5804HR, 4012AD, 4112AD, PCI5500, PCI5501, PCI5502, PCI5503, PCI5504

Adac AI SetTriggerConfig.vi

Supported Boards:

5401DMF, 5402DMF, 5403DHR, 5801MF, 5802MF, 5803HR and 5804HR, PCI5500, PCI5501, PCI5502, PCI5503, PCI5504

Advanced Analog Input VIs

Adac AI Buffer Read.vi

Supported Boards:

5401DMF, 5402DMF, 5403DHR, 5404DHR, 5500HR, 5508SHR, 5508LC, 5508LF, 5508SCi, 5516DMA, 5525MF, 5550MF, 5801MF, 5802MF, 5803HR 5804HR, 5508BG, 5508TC, 5508RTD, 5500MF, 4012AD, 4112AD, PCI5500, PCI5501, PCI5502, PCI5503, PCI5504

Easy Analog Output VIs

ADAC AO Update Channel.vi

Supported Boards:

5504DA, 5525MF, 5401DMF, 5402DMF, 5403DHR, 5404DHR, 5550MF, 5801MF, 5802MF, 5803HR 5804HR, 4412DA, 4412DAcL, PCI5501, PCI5502, PCI5503, PCI5504

Intermediate Analog Output VIs

Adac AO Config.vi & Adac AO Clear.vi

Supported Boards:

5504DA, 5525MF, 5401DMF, 5402DMF, 5403DHR, 5404DHR, 5550MF, 5801MF, 5802MF, 5803HR 5804HR, 4412DA, 4412DAcL, PCI5501, PCI5502, PCI5503, PCI5504

Easy Digital I/O VIs

ADAC Read from Digital Port.vi & ADAC Read from Digital Line.vi

Supported Boards:

5401DMF, 5402DMF, 5403DHR, 5404DHR, 5500HR, 5508SHR, 5516DMA, 5525MF, 5550MF, 5801MF, 5802MF, 5803HR 5804HR, 5532TTL, 5632TTL, 5608ACI, 5616CCI, 5616DCI, 4608ACO, 4608DCO, 4616ACI, 4616CCI, 4616DCI, 4616HCO, 4616OII, 4616RLY, 4632TTL, PCI5500, PCI5501, PCI5502, PCI5503, PCI5504

ADAC Write to Digital Port.vi & ADAC Write to Digital Line.vi

Supported Boards:

5401DMF, 5402DMF, 5403DHR, 5404DHR, 5500HR, 5508SHR, 5516DMA, 5525MF, 5550MF, 5801MF, 5802MF, 5803HR 5804HR, 5532TTL, 5632TTL, 5608ACO, 5616DCO, 4608ACO, 4608DCO, 4616ACI, 4616CCI, 4616DCI, 4616HCO, 4616OII, 4616RLY, 4632TTL, PCI5500, PCI5501, PCI5502, PCI5503, PCI5504

Intermediate Digital I/O VIs

Adac DIO Config

Supported Boards:

5401DMF, 5402DMF, 5403DHR, 5404DHR, 5500HR, 5508SHR, 5516DMA, 5525MF, 5550MF, 5801MF, 5802MF, 5803HR 5804HR, 5532TTL, 5632TTL, 5608ACI, 5616CCI, 5616DCI, 5608ACO, 5616DCO, 4608ACO, 4608DCO, 4616ACI, 4616CCI, 4616DCI, 4616HCO, 4616OII, 4616RLY, 4632TTL, PCI5500, PCI5501, PCI5502, PCI5503, PCI5504

Adac DIO Clear.vi

Supported Boards:

5401DMF, 5402DMF, 5403DHR, 5404DHR, 5500HR, 5508SHR, 5516DMA, 5525MF, 5550MF, 5801MF, 5802MF, 5803HR 5804HR, 5532TTL, 5632TTL, 5608ACI, 5616CCI, 5616DCI, 5608ACO, 5616DCO, 4608ACO, 4608DCO, 4616ACI, 4616CCI, 4616DCI, 4616HCO, 4616OII, 4616RLY, 4632TTL, PCI5500, PCI5501, PCI5502, PCI5503, PCI5504

Adac DIO Read.vi & Adac DIO Start.vi

Supported Boards:

5532TTL, 5632TTL, 5608ACI, 5616CCI, 5616DCI, PCI5500, PCI5501, PCI5502, PCI5503, PCI5504

Advanced Digital I/O VIs

Adac DIO 5600 Config.vi

Supported Boards:

5632TTL, 5608ACI, 5616CCI, 5616DCI

Miscellaneous VIs

Adac Buffer Read.vi

Supported Boards:

5401DMF, 5402DMF, 5403DHR, 5404DHR, 5500HR, 5508SHR, 5508LC, 5508LF, 5508SCi, 5516DMA, 5525MF, 5550MF, 5801MF, 5802MF, 5803HR 5804HR, 5508BG, 5508TC, 5508RTD, 5500MF, 5532TTL, 5632TTL, 5608ACI, 5616CCI, 5616DCI, 4012AD, 4112AD, PCI5500, PCI5501, PCI5502, PCI5503, PCI5504

Adac Occurrence Config.vi

Supported Boards:

5401DMF, 5402DMF, 5403DHR, 5404DHR, 5500HR, 5508SHR, 5508LC, 5508LF, 5508SCi, 5516DMA, 5525MF, 5550MF, 5801MF, 5802MF, 5803HR 5804HR, 5508BG, 5508TC, 5508RTD, 5532TTL, 5632TTL, 5608ACI, 5616CCI, 5616DCI, 4012AD, 4112AD, PCI5500, PCI5501, PCI5502, PCI5503, PCI5504

Adac Enum Boards.vi & Adac Get Board Sections.vi

Supported Boards:

All ADAC Boards

Signal Conditioning VIs

Adac Convert Thermocouple Reading.vi & Adac Convert Thermocouple Buffer.vi

Supported Boards:

5401DMF, 5402DMF, 5403DHR, 5404DHR, 5508LF, 5508SCi, 5525MF, 5550MF, 5801MF, 5802MF, 5803HR 5804HR, 5508TC, 4012AD, 4112AD, PCI5501, PCI5502, PCI5503, PCI5504

Adac Convert RTD Reading.vi

Supported Boards:

5508RTD, 4012AD

Advanced Function Environment VIs:

AL_LoadEnvironment.vi &...AL_ReleaseEnvironment.vi... & AL_GetEnvString.vi & AL_SetEnvString.vi

Supported Boards:

All ADAC Boards

Advanced Function Board VIs:

AL_GetBoardHardwareId.vi & AL_GetBoardHardwareVersion.vi

Supported Boards:

5401DMF, 5402DMF, 5403DHR, 5404DHR, 5801MF, 5802MF, 5803HR, 5804HR, PCI5500, PCI5501, PCI5502, PCI5503, PCI5504

AL_GetBoardDriverVersion.vi & AL_SetBoardString.vi & AL_GetBoardString.vi & AL_GetBoardError.vi

Advanced Function Board VIs (con't)

Supported Boards:
All ADAC Boards

Advanced Function Device Configuration VIs:

AL_AllocateDevice.vi & AL_ReleaseDevice.vi & AL_SetLdsString.vi & AL_GetLdsString.vi & AL_InitDevice.vi

Supported Subsystems:
All ADAC Board Subsystems

AL_StartDevice.vi & AL_StopDevice.vi

Supported Subsystems:
5400 Series: ADC0
PCI5500: ADC0
PCI55xx: ADC0, DAC0, DAC1
5500HR: ADC0
5508LC: ADC0
5508LF: ADC0
5508SCi: ADC0
5508SHR: ADC0
5508TC/BG/RTD: ADC0
5516DMA: ADC0
5525/50MF: ADC0
5532TTL: DIN0, DIN1, DIN2, DIN3, DOT0, DOT1, DOT2, DOT3
5600 Series: DIN0
5800 Series: ADC0
4012AD Series: ADC0
4112AD Series: ADC0

AL_SetCycleMode.vi & AL_GetCycleMode.vi

Supported Subsystems:	
5400 Series: ADC0, Options:	SINGLE_CYCLE or CONTINUOUS_CYCLE
PCI5500: ADC0, Options:	SINGLE_CYCLE or CONTINUOUS_CYCLE
PCI55xx: ADC0, Options:	SINGLE_CYCLE or CONTINUOUS_CYCLE
PCI55xx: DAC0,DAC1, Options:	SINGLE_CYCLE or CONTINUOUS_CYCLE
5500HR: ADC0, Options:	SINGLE_CYCLE or CONTINUOUS_CYCLE
5500MF: ADC0, Options:	SINGLE_CYCLE
5504DA: DAC0, DAC1, DAC2, DAC3, Options:	SINGLE_CYCLE
5508LC: ADC0, Options:	SINGLE_CYCLE or CONTINUOUS_CYCLE
5508LF: ADC0, Options:	SINGLE_CYCLE or CONTINUOUS_CYCLE
5508SCi: ADC0, Options:	SINGLE_CYCLE or CONTINUOUS_CYCLE
5508SHR: ADC0, Options:	SINGLE_CYCLE or CONTINUOUS_CYCLE
5508TC/BG/RTD: ADC0, Options:	SINGLE_CYCLE or CONTINUOUS_CYCLE
5516DMA: ADC0, Options:	SINGLE_CYCLE or CONTINUOUS_CYCLE
5525/50MF: ADC0, Options:	SINGLE_CYCLE or CONTINUOUS_CYCLE
5532TTL, DOT0, DOT1, DOT2, DOT3, Options:	SINGLE_CYCLE or CONTINUOUS_CYCLE
5532TTL: DIN0, DIN1, DIN2, DIN3, Options:	SINGLE_CYCLE or CONTINUOUS_CYCLE

Advanced Function Device Configuration VIs (con't)

5600 Series: DIN0	SINGLE_CYCLE or CONTINUOUS_CYCLE
5600 Series: DIN1, DOT0-3	SINGLE_CYCLE
5800 Series: ADC0, Options:	SINGLE_CYCLE or CONTINUOUS_CYCLE
4012AD Series: ADC0, Options:	SINGLE_CYCLE or CONTINUOUS_CYCLE
4112AD Series: ADC0, Options:	SINGLE_CYCLE or CONTINUOUS_CYCLE
4412DA: DAC0, DAC1, DAC2, DAC3 Options:	SINGLE_CYCLE
4412DACL: DAC0, DAC1, DAC2, DAC3 Options:	SINGLE_CYCLE
4608ACO Options:	SINGLE_CYCLE
4608DCO Options:	SINGLE_CYCLE
4616ACI Options:	SINGLE_CYCLE
4616CCI Options:	SINGLE_CYCLE
4616DCI Options:	SINGLE_CYCLE
4616HCO Options:	SINGLE_CYCLE
4616OII Options:	SINGLE_CYCLE
4616RLY Options:	SINGLE_CYCLE
4632TTL Options:	SINGLE_CYCLE

AL_SetDataTransferMethod.vi & AL_GetDataTransferMethod.vi

Supported Subsystems:

5400 Series: ADC0, Options:	SOFTWARE, DMA or INTERRUPT
PCI5500: ADC0, Options:	SOFTWARE or DMA
PCI55xx: ADC0, Options:	SOFTWARE or DMA
PCI55xx: DAC0, DAC10, Options:	SOFTWARE or DMA
5500HR: ADC0, Options:	SOFTWARE, DMA or INTERRUPT
5500MF: ADC0, Options:	SOFTWARE
5504DA: DAC0, DAC1, DAC2, DAC3, Options:	SOFTWARE
5508LC: ADC0, Options:	SOFTWARE, DMA or INTERRUPT
5508LF: ADC0, Options:	SOFTWARE, DMA or INTERRUPT
5508SCi: ADC0, Options:	SOFTWARE, DMA or INTERRUPT
5508SHR: ADC0, Options:	SOFTWARE, DMA or INTERRUPT
5508TC/BG/RTD: ADC0, Options:	SOFTWARE or INTERRUPT
5516DMA: ADC0, Options:	SOFTWARE, DMA or INTERRUPT
5525/50MF: ADC0, Options:	SOFTWARE, DMA or INTERRUPT
5532TTL: DIN0, DIN1, DIN2, DIN3, Options:	SOFTWARE or INTERRUPT
5532TTL: DOT0, DOT1, DOT2, DOT3, Options:	SOFTWARE or INTERRUPT
5600 Series: DIN0	SOFTWARE or INTERRUPT
5600 Series: DIN1, DOT0-3	SOFTWARE
5800 Series: ADC0, Options:	SOFTWARE, DMA or INTERRUPT
4012AD Series: ADC0, Options:	SOFTWARE
4112AD Series: ADC0, Options:	SOFTWARE
4608ACO Options:	SOFTWARE
4608DCO Options:	SOFTWARE
4616ACI Options:	SOFTWARE
4616CCI Options:	SOFTWARE
4616DCI Options:	SOFTWARE
4616HCO Options:	SOFTWARE
4616OII Options:	SOFTWARE
4616RLY Options:	SOFTWARE

Advanced Function Device Configuration VIs (con't)

4632TTL Options: SOFTWARE

AL_GetDeviceStatus.vi

Supported Subsystems:

5400 Series: ADC0, DAC0, DAC1, DIN0, DIN1, DIN2, DIN3, DOT1 and DOT3

PCI5500: ADC0, DIN0, DIN1, DOT0, and DOT1

PCI55xx Series: ADC0, DAC0, DAC1, DIN0, DIN1, DIN2, DIN3, DOT0, DOT1, DOT2, and DOT3

5500HR: ADC0, DIN0, DIN1, DOT0 and DOT1

5500MF: ADC0, DIN0, DIN1, DIN2, DIN3, DOT0, DOT1, DOT2 and DOT3

5504DA: DAC0, DAC1, DAC2 and DAC3

5508LC: ADC0

5508LF: ADC0

5508SCi: ADC0

5508SHR: ADC0, DIN0, DIN1, DOT0 and DOT1

5508TC/BG/RTD: ADC0

5516DMA: ADC0, DIN0, DIN1, DOT0, DOT1

5525/50MF: ADC0, DAC0, DAC1, DIN0, DIN1, DIN2, DIN3, DOT0, DOT1, DOT2 and DOT3

5532TTL: DIN0, DIN1, DIN2, DIN3, DOT0, DOT1, DOT2, DOT3

5600 Series: DIN0, DIN1, DOT0-3

5800 Series: ADC0, DAC0, DAC1, DIN0, DIN1, DIN2, DIN3, DOT1 and DOT3

4012AD Series: ADC0

4112AD Series: ADC0

4412DA: DAC0, DAC1, DAC2, DAC3

4412DAcL: DAC0, DAC1, DAC2, DAC3

4608ACO: DOT0

4608DCO: DOT0

4616ACI: DIN0-1

4616CCI: DIN0-1

4616DCI: DIN0-1

4616HCO: DOT0-1

4616OII: DIN0-1

4616RLY: DOT0

4632TTL: DIN0-3, DOT0-3

Advanced Function Analog Output VIs:

AL_SetDaOutput.vi

Supported Subsystems:

5400 Series: DAC0 and DAC1

PCI55xx Series: DAC0 and DAC1

5504DA: DAC0, DAC1, DAC2 and DAC3

5525/50MF: DAC0 and DAC1

5800 Series: DAC0 and DAC1

4412DA: DAC0, DAC1, DAC2, DAC3

4412DAcL: DAC0, DAC1, DAC2, DAC3

Advanced Function Digital I/O VIs:

AL_DigInput.vi & AL_DigBitsTest.vi

Supported Subsystems:

5400 Series: DIN0, DIN1, DIN2 and DIN3

PCI5500: DIN0, DIN1, DOT0, and DOT1

PCI55xx Series: DIN0, DIN1, DIN2, DIN3, DOT0, DOT1, DOT2, and DOT3

5500HR: DIN0 and DIN1

5500MF: DIN0, DIN1, DIN2 and DIN3

5508SHR: DIN0 and DIN1

5516DMA: DIN0 and DIN1

5525/50MF: DIN0, DIN1, DIN2 and DIN3

5532TTL: DIN0, DIN1, DIN2 and DIN3

5600 Series: DIN0, DIN1, DOT0-3

5800 Series: DIN0, DIN1, DIN2 and DIN3

4608ACO: DOT0

4608DCO: DOT0

4616ACI: DIN0-1

4616CCI: DIN0-1

4616DCI: DIN0-1

4616HCO: DOT0-1

4616OII: DIN0-1

4616RLY: DOT0

4632TTL: DIN0-3, DOT0-3

AL_DigOutput.vi

Supported Subsystems:

5400 Series: DOT1 and DOT3

PCI5500: DOT0, and DOT1

PCI55xx Series DOT0, DOT1, DOT2, and DOT3

5500HR: DOT0 and DOT1

5500MF: DOT0, DOT1, DOT2 and DOT3

5508SHR: DOT0 and DOT1

5516DMA: DOT0 and DOT1

5525/50MF: DOT0, DOT1, DOT2 and DOT3

5532TTL: DOT0, DOT1, DOT2 and DOT3

5600 Series: DOT0-3

5800 Series: DOT1 and DOT3

4608ACO: DOT0

4608DCO: DOT0

4616HCO: DOT0, DOT1

4616RLY: DOT0

4632TTL: DOT0-3

AL_SetPortMask.vi

Supported Subsystems:

5532TTL:

5600 Series ACI, DCI, CCI, TTL: DIN0

4616ACI: DIN0

4616CCI: DIN0

Advanced Function Triggering VIs: (con't)

5508SCi: ADC0, Options	DISABLED, EXTERNAL or CTR1
5508SHR: ADC0, Options	DISABLED or CTR2
5525/50MF: ADC0, Options:	DISABLED or CTR2
5800 Series: ADC0, Options:	DISABLED or EXTERNAL

AL_SetTriggerSourceSignal.vi

Supported Subsystems:	
5400 Series: ADC0, Options:	FALLING_EDGE or RISING_EDGE
5800 Series: ADC0, Options:	FALLING_EDGE or RISING_EDGE
PCI5500: ADC0, Options:	FALLING_EDGE or RISING_EDGE
PCI55xx Series: ADC0, Options:	FALLING_EDGE or RISING_EDGE
PCI55xx Series: DAC0, Options:	FALLING_EDGE or RISING_EDGE

AL_SetTriggerRate.vi

Supported Subsystems:	
5500HR: ADC0, Options:	15.259 to 15000
5508LC: ADC0, Options:	15.259 to 100000
5508LF: ADC0, Options:	15.259 to 100000
5508SCi: ADC0, Options:	15.259 to 100000
5508SHR: ADC0, Options:	61.036 to 47000

AL_SetPostSampleCount.vi

Supported Subsystems:	
5400 Series: ADC0, Options:	0 to 65535used for PRE_TRIG or ABOUT_TRIG Triggermode
5800 Series: ADC0, Options:	0 to 65535used for PRE_TRIG or ABOUT_TRIG Triggermode
PCI5500:	0 to 65535used for PRE_TRIG or ABOUT_TRIG Triggermode
PCI55xx Series:	0 to 65535used for PRE_TRIG or ABOUT_TRIG Triggermode

AL_GetTriggerStruct.vi

Supported Subsystems:	
5400 Series: ADC0	
5500HR: ADC0	
5508LC: ADC0	
5508LF: ADC0	
5508SCi: ADC0	
5508SHR: ADC0	
5508TC/BG/RTD: ADC0	
5516DMA: ADC0	
5525/50MF: ADC0	
5800 Series: ADC0	
PCI5500: ADC0	
PCI55xx Series: ADC0, DAC0	

Advanced Function Clcking VIs:

AL_SetClockSource.vi

Supported Subsystems:

5400 Series: ADC0, Options:	SOFTWARE_CONVERT, INTERNAL, EXT_RISING_EDGE or EXT_FALLING_EDGE
PCI5500: ADC0 Options:	SOFTWARE_CONVERT, INTERNAL, EXT_RISING_EDGE or EXT_FALLING_EDGE
PCI55xx Series: ADC0 Options:	SOFTWARE_CONVERT, INTERNAL, EXT_RISING_EDGE or EXT_FALLING_EDGE
PCI55xx Series: DAC0 Options:	SOFTWARE_CONVERT, INTERNAL, EXT_RISING_EDGE or EXT_FALLING_EDGE
5500HR: ADC0, Options:	SOFTWARE_CONVERT, INTERNAL, or EXT_FALLING_EDGE
5508LC: ADC0, Options:	SOFTWARE_CONVERT, INTERNAL, or EXT_FALLING_EDGE
5508LF: ADC0, Options:	SOFTWARE_CONVERT, INTERNAL, or EXT_FALLING_EDGE
5508SCi: ADC0, Options:	SOFTWARE_CONVERT, INTERNAL, or EXT_FALLING_EDGE
5508SHR: ADC0, Options:	SOFTWARE_CONVERT, INTERNAL, or EXT_FALLING_EDGE
5508TC/BG/RTD:ADC0, Options:	SOFTWARE_CONVERT, INTERNAL, or EXT_FALLING_EDGE
5516DMA: ADC0, Options:	SOFTWARE_CONVERT, INTERNAL, or EXT_FALLING_EDGE
5525/50MF: ADC0, Options:	SOFTWARE_CONVERT, INTERNAL, or EXT_FALLING_EDGE
5800 Series: ADC0, Options:	SOFTWARE_CONVERT, INTERNAL, EXT_RISING_EDGE or EXT_FALLING_EDGE
4012AD Series: ADC0, Options:	SOFTWARE or MMTIMER
4412AD Series: ADC0, Options:	SOFTWARE or MMTIMER

AL_SetClockRate.vi

Supported Subsystems:

5401, 5402, 5801 and 5802: ADC0,	1.164188e-3 to 333000
5403, 5404, 5803 and 5804: ADC0,	1.164188e-3 to 100000
PCI5500: ADC0,	2.3283e-3 to 100000
PCI5501 Series: ADC0,	2.3283e-3 to 100000
PCI5502 Series: ADC0,	2.3283e-3 to 100000
PCI5503 Series: ADC0,	2.3283e-3 to 200000
PCI5504 Series: ADC0,	2.3283e-3 to 200000
PCI55xx Series: DAC0, DAC1	2.3283e-3 to 200000
5500HR: ADC0,	15.259 to 15000
5508LC: ADC0,	15.259 to 100000
5508LF: ADC0,	15.259 to 100000
5508SCi: ADC0,	15.259 to 100000
5508SHR: ADC0,	61.036 to 47000
5516DMA: ADC0,	15.259 to 60000
5525/50MF: ADC0,	15.259 to 40000
4012AD Series: ADC0,	1 to 1000
4112AD Series: ADC0,	1 to 1000

Advanced Function Clocking VIs: (con't)

AL_GetClockStruct.vi

Supported Subsystems:

5400 Series: ADC0
PCI5500: ADC0
PCI5501 Series: ADC0, DAC0, DAC1
PCI5502 Series: ADC0, DAC0, DAC1
PCI5503 Series: ADC0, DAC0, DAC1
PCI5504 Series: ADC0, DAC0, DAC1
5500HR: ADC0
5508LC: ADC0
5508LF: ADC0
5508SCi: ADC0
5508SHR: ADC0
5508TC/BG/RTD: ADC0
5516DMA: ADC0
5525/50MF: ADC0
5800 Series: ADC0
4012AD Series: ADC0
4112AD Series: ADC0

AL_GetActualClockRate.vi

Supported Subsystems:

5400 Series: ADC0
PCI5500: ADC0
PCI55xx Series: ADC0, DAC0, DAC1
5500HR: ADC0
5508LC: ADC0
5508LF: ADC0
5508SCi: ADC0
5508SHR: ADC0
5516DMA: ADC0
5525/50MF: ADC0
5800 Series: ADC0
4012AD Series: ADC0
4112AD Series: ADC0

Advanced Function Gating VIs:

AL_SetGateSource.vi

Supported Subsystems:

5400 Series: ADC0, Options:	DISABLED, SWGATE or EXTGATE
5800 Series: ADC0, Options:	DISABLED, SWGATE or EXTGATE
PCI5500: ADC0, Options:	DISABLED, SWGATE or EXTGATE
PCI55xx Series: ADC0, Options:	DISABLED, SWGATE or EXTGATE
PCI55xx Series: DAC0, Options:	DISABLED, SWGATE or EXTGATE

AL_SetGateLevel.vi

Supported Subsystems:

5400 Series: ADC0, Options:	ACTIVE_HIGH or ACTIVE_LOW
5800 Series: ADC0, Options:	ACTIVE_HIGH or ACTIVE_LOW
PCI5500: ADC0, Options:	ACTIVE_HIGH or ACTIVE_LOW
PCI55xx Series: ADC0, Options:	ACTIVE_HIGH or ACTIVE_LOW
PCI55xx Series: DAC0, Options:	ACTIVE_HIGH or ACTIVE_LOW

AL_GetGateStruct.vi

Supported Subsystems:

5400 Series: ADC0
5800 Series: ADC0
PCI5500: ADC0
PCI55xx Series: ADC0
PCI55xx Series: DAC0

AL_SetSwGate.vi & AL_GetSwGate.vi

Supported Subsystems:

5400 Series: ADC0, Options:	DISABLED or ENABLED
5800 Series: ADC0, Options:	DISABLED or ENABLED
PCI5500: ADC0, Options:	DISABLED or ENABLED
PCI55xx Series: ADC0, Options:	DISABLED or ENABLED
PCI55xx Series: DAC0, Options:	DISABLED or ENABLED

Advanced Function Burst VIs:

AL_SetBurstMode.vi

Supported Subsystems:

5400 Series: ADC0, Options:	DISABLED or ENABLED
5525/50MF: ADC0, Options:	DISABLED or ENABLED
5800 Series: ADC0, Options:	DISABLED or ENABLED
PCI5500: ADC0, Options:	DISABLED or ENABLED
PCI55xx Series: ADC0, Options:	DISABLED or ENABLED

Advanced Function Burst VIs: (con't)

AL_SetBurstLength.vi

Supported Subsystems:
 5400 Series: ADC0, Options: 1 to 256
 5800 Series: ADC0, Options: 1 to 256
 PCI5500: ADC0, Options: 1 to 65535
 PCI55xx Series: ADC0, Options: 1 to 65535

AL_SetBurstRate.vi

Supported Subsystems:
 5401, 5402, 5801 and 5802: ADC0, Options: 3 to 64 specified in Tics
 5403, 5404, 5803 and 5804: ADC0, Options: 10 to 64 specified in Tics
 5525/50MF: ADC0, Options: 25 to 65535 specified in Tics
 PCI5500: ADC0, Options: 1 to 65526
 PCI5501: ADC0, Options: 1 to 65526
 PCI5502: ADC0, Options: 1 to 65526
 PCI5503: ADC0, Options: 1 to 65531
 PCI5504: ADC0, Options: 1 to 65531

AL_GetBurstStruct.vi

Supported Subsystems:
 5400 Series: ADC0
 5525/50MF: ADC0
 5800 Series: ADC0
 PCI5500: ADC0
 PCI55xx Series: ADC0

Advanced Function Channels VIs:

AL_SetChannelList.vi & AL_GetChannelList.vi & AL_GetNumberOfChans.vi

Supported Subsystems:
 5401, 5402, 5801 and 5802: ADC0, Options: Min/Max Channel, Gains = 1, 2, 4, 8
 5403, 5404, 5803 and 5804: ADC0, Options: Min/Max Channel, Gains = 1, 10, 100, 500
 PCI5500: ADC0, Options: Min/Max Channel
 PCI5501: ADC0, Options: Min/Max Channel, Gains = 1, 2, 4, 8
 PCI5502: ADC0, Options: Min/Max Channel, Gains = 1, 10, 100
 PCI5503: ADC0, Options: Min/Max Channel, Gains = 1, 2, 4, 8
 PCI5504: ADC0, Options: Min/Max Channel, Gains = 1, 10, 100
 5500HR: ADC0, Options: Min/Max Channel
 5500MF: ADC0, Options: Min/Max Channel
 5508LC: ADC0, Options: Min/Max Channel
 5508LF: ADC0, Options: Min/Max Channel, Gains = 1, 2, 4, 8 or 1, 2, 5, 10
 5508SCi: ADC0, Options: Min/Max Channel, Gains = 1, 2, 4, 8 or 1, 2, 5, 10
 5508SHR: ADC0, Options: Min/Max Channel
 5508TC/BG/RTD: ADC0, Options: Min/Max Channel, Gains = 1, 2, 4, 8
 5516DMA: ADC0, Options: Min/Max Channel, Gains = 1, 2, 4, 8
 5525/50MF: ADC0, Options: Min/Max Channel, Gains = 1, 2, 4, 8

Advanced Function Channels VIs: (con't)

4012AD Series: ADC0, Options: Min/Max Channel, Gains = 1, 2, 4, 8
4112AD Series: ADC0, Options: Min/Max Channel, Gains = 1, 2, 4, 8

Advanced Function Expansion Panels VIs:

AL_SetExpPanels.vi & AL_GetExpPanelStruct.vi

Supported Subsystems:

5400 Series: ADC0, Options:	TB5800-16, TB5800-TC or TB5800-64
PCI5500: ADC0, Options:	NONE, PCI_TB5500_8
PCI55xx: ADC0, Options:	NONE, PCI_TB5500_16, PCI_TB5500_64 or PCI_TB5500_TC
5508LF: ADC0, Options:	NONE, 5508EXi
5508SCi: ADC0, Options:	NONE, 5508EXi
5525/50MF: ADC0, Options:	NONE, A4012HLEX or A4012TCEX
5800 Series: ADC0, Options:	TB5800-16, TB5800-TC or TB5800-64
4012AD Series: ADC0, Options:	A4012HLEX, A4012TCEX, A4012CLEX, A4012AMEX, A4012CSEX, A4012BGEX
4112AD Series: ADC0, Options:	A4012WRSX, A4112HCVX

AL_SetExpPanelGains.vi & AL_GetExpPanelGainStruct.vi

Supported Subsystems:

5400 Series: ADC0, Options:	Min/Max Channel, NO Limit on panel gain
PCI55xx: ADC0, Options:	Min/Max Channel, NO Limit on panel gain
5508LF: ADC0, Options:	Min/Max Channel, NO Limit on panel gain
5508SCi: ADC0, Options:	Min/Max Channel, NO Limit on panel gain
5800 Series: ADC0, Options:	Min/Max Channel, NO Limit on panel gain
4012AD Series: ADC0, Options:	Min/Max Channel, NO Limit on panel gain
4112AD Series: ADC0, Options:	Min/Max Channel, NO Limit on panel gain

Advanced Function Thermocouple Configuration VIs:

AL_SetCjList.vi & AL_GetCjList.vi & AL_SetCjGlobal.vi

Supported Subsystems:

5400 Series: ADC0, Options:	Min/Max Channel, BNBS,JNBS,KNBS,SNBS,ENBS,TNBS,RNBS
PCI55xx: ADC0, Options:	Min/Max Channel, BNBS,JNBS,KNBS,SNBS,ENBS,TNBS,RNBS
5508SCi: ADC0, Options:	Min/Max Channel, BNBS,JNBS,KNBS,SNBS,ENBS,TNBS,RNBS
5508TC: ADC0, Options:	Min/Max Channel, BNBS,JNBS,KNBS,SNBS,ENBS,TNBS,RNBS
5525/50MF: ADC0, Options:	Min/Max Channel, BNBS,JNBS,KNBS,SNBS,ENBS,TNBS,RNBS
5800 Series: ADC0, Options:	Min/Max Channel, BNBS,JNBS,KNBS,SNBS,ENBS,TNBS,RNBS
4012AD Series: ADC0, Options:	Min/Max Channel, BNBS,JNBS,KNBS,SNBS,ENBS,TNBS,RNBS
4112AD Series: ADC0, Options:	Min/Max Channel, BNBS,JNBS,KNBS,SNBS,ENBS,TNBS,RNBS

Advanced Function DMA Configuration Vis:

AL_SetDmaChan.vi & AL_GetDmaChan.vi

Supported Subsystems:

5400 Series: ADC0, Options:	DISABLED, DMA5, DMA6, DMA7
5500HR: ADC0, Options:	DISABLED, DMA1, DMA2, DMA3
5508LC: ADC0, Options:	DISABLED, DMA1, DMA2, DMA3
5508LF: ADC0, Options:	DISABLED, DMA1, DMA2, DMA3
5508SCi: ADC0, Options:	DISABLED, DMA1, DMA2, DMA3
5508SHR: ADC0, Options:	DISABLED, DMA1, DMA2, DMA3
5516DMA: ADC0, Options:	DISABLED, DMA1, DMA2, DMA3
5525/50MF: ADC0, Options:	DISABLED, DMA1, DMA2, DMA3
5800 Series: ADC0, Options:	DISABLED, DMA5, DMA6, DMA7

Advanced Function IRQ Configuration VIs:

AL_GetIrqLevel.vi & AL_SetIrqLevel.vi

Supported Subsystems:

5400 Series: ADC0, Options:	DISABLED, IRQ3, IRQ5, IRQ7, IRQ10, IRQ11, IRQ15
5500HR: ADC0, Options:	DISABLED, IRQ2, IRQ3, IRQ4, IRQ5, IRQ6, IRQ7
5508LC: ADC0, Options:	DISABLED, IRQ2, IRQ3, IRQ4, IRQ5, IRQ6, IRQ7
5508LF: ADC0, Options:	DISABLED, IRQ2, IRQ3, IRQ4, IRQ5, IRQ6, IRQ7
5508SCi: ADC0, Options:	DISABLED, IRQ2, IRQ3, IRQ4, IRQ5, IRQ6, IRQ7
5508SHR: ADC0, Options:	DISABLED, IRQ2, IRQ3, IRQ4, IRQ5, IRQ6, IRQ7
5508TC/BG/RTD: ADC0, Options:	DISABLED, IRQ2, IRQ3, IRQ4, IRQ5, IRQ6, IRQ7
5516DMA: ADC0, Options:	DISABLED, IRQ2, IRQ3, IRQ4, IRQ5, IRQ6, IRQ7
5525/50MF: ADC0, Options:	DISABLED, IRQ2, IRQ3, IRQ4, IRQ5, IRQ6, IRQ7
5600 Series ACI, DCI, CCI, TTL: DIN0, Options:	DISABLED, IRQ2, IRQ3, IRQ4, IRQ5, IRQ6, IRQ7
5800 Series: ADC0, Options:	DISABLED, IRQ3, IRQ5, IRQ7, IRQ10, IRQ11, IRQ15

Advanced Function Input Configuration VIs:

AL_SetInputConfig.vi & AL_GetInputConfig.vi

Supported Subsystems:

5400 Series: ADC0, Options:	DIFFERENTIAL, SINGLE-ENDED or PSEUDO-DIFFERENTIAL
5800 Series: ADC0, Options:	DIFFERENTIAL, SINGLE-ENDED or PSEUDO-DIFFERENTIAL
PCI5500: ADC0, Options:	SINGLE-ENDED
PCI55xx: ADC0, Options:	DIFFERENTIAL, SINGLE-ENDED or PSEUDO-DIFFERENTIAL must be set in the ADAC-LVI.CON file

Advanced Function Data Codes VIs:

AL_SetDataCode.vi & AL_GetDataCode.vi

Supported Subsystems:

5400 Series: DAC0, Options:	TWOS COMPLEMENT, STRAIGHT_BINARY or OFFSET_BINARY
5500HR: ADC0, Options:	TWOS COMPLEMENT, STRAIGHT_BINARY or OFFSET_BINARY
5500MF:ADC0, Options:	TWOS COMPLEMENT, STRAIGHT_BINARY or OFFSET_BINARY
5504DA:DAC0-3, Options:	TWOS COMPLEMENT, STRAIGHT_BINARY or OFFSET_BINARY
5508LC: ADC0, Options:	TWOS COMPLEMENT, STRAIGHT_BINARY or OFFSET_BINARY
5508LF: ADC0, Options:	TWOS COMPLEMENT, STRAIGHT_BINARY or OFFSET_BINARY
5508SCi: ADC0, Options:	TWOS COMPLEMENT, STRAIGHT_BINARY or OFFSET_BINARY
5508TC/BG/RTD: ADC0, Options:	TWOS COMPLEMENT, STRAIGHT_BINARY or OFFSET_BINARY
5516DMA: ADC0, Options:	TWOS COMPLEMENT, STRAIGHT_BINARY or OFFSET_BINARY
5525/50MF: DAC0 and ADC0, Options:	TWOS COMPLEMENT, STRAIGHT_BINARY or OFFSET_BINARY
5800 Series: DAC0, Options:	TWOS COMPLEMENT, STRAIGHT_BINARY or OFFSET_BINARY
4012AD Series: ADC0, Options:	TWOS COMPLEMENT, STRAIGHT_BINARY or OFFSET_BINARY
4112AD Series: ADC0, Options:	TWOS COMPLEMENT, STRAIGHT_BINARY or OFFSET_BINARY
4412DA: DAC0-3, Options:	TWOS COMPLEMENT, STRAIGHT_BINARY or OFFSET_BINARY
4412DAcL: DAC0-3, Options:	TWOS COMPLEMENT, STRAIGHT_BINARY or OFFSET_BINARY

AL_SetDataOffset.vi & AL_GetDataOffset.vi

Supported Subsystems:

5400 Series: ADC0 and DAC0, Options:	BIPOLAR or UNIPOLAR
PCI5500: ADC0, Options:	BIPOLAR or UNIPOLAR must be set in the ADAC-LVI.CON file
PCI55xx: ADC0, Options:	BIPOLAR or UNIPOLAR must be set in the ADAC-LVI.CON file
5500HR: ADC0, Options:	BIPOLAR or UNIPOLAR
5500MF:ADC0, Options:	BIPOLAR or UNIPOLAR
5504DA:DAC0, Options:	BIPOLAR or UNIPOLAR
5508LC: ADC0, Options:	BIPOLAR or UNIPOLAR
5508LF: ADC0, Options:	BIPOLAR or UNIPOLAR
5508SCi: ADC0, Options:	BIPOLAR or UNIPOLAR
5508SHR: ADC0, Options:	BIPOLAR or UNIPOLAR
5508TC/BG/RTD: ADC0, Options:	BIPOLAR or UNIPOLAR
5516DMA: ADC0, Options:	BIPOLAR or UNIPOLAR
5525/50MF: ADC0 and DAC0, Options:	BIPOLAR or UNIPOLAR
5800 Series: ADC0 and DAC0, Options:	BIPOLAR or UNIPOLAR
4012AD: ADC0, Options:	BIPOLAR or UNIPOLAR

Advanced Function Data Codes VIs (con't)

4112AD, ADC0, Options: BIPOLAR or UNIPOLAR
 4412DA: DAC0-3, Options: BIPOLAR or UNIPOLAR
 4412DACL: DAC0-3 Options: BIPOLAR or UNIPOLAR

AL_SetDataRange.vi & AL_GetDataRange.vi

Supported Subsystems:

5400 Series: DAC0, Options: -10_10 or 0_10
 5500HR: ADC0, Options: -10_10, -5_5 or 0_10
 5500MF: ADC0, Options: -10_10, -5_5 or 0_10
 5504DA: DAC0, Options: -10_10, -5_5 or 0_10
 5508LC: ADC0, Options: -10_10, -5_5 or 0_10
 5508LF: ADC0, Options: -10_10, -5_5 or 0_10
 5508SCi: ADC0, Options: -10_10, -5_5 or 0_10
 5508TC/BG/RTD: ADC0, Options: -10_10, -5_5 or 0_10
 5516DMA: ADC0, Options: -10_10, -5_5 or 0_10
 5525/50MF: ADC0 and DAC0, Options: -10_10, -5_5 or 0_10
 5800 Series: DAC0, Options: -10_10 or 0_10
 4012AD Series: ADC0, Options: -10_10, -5_5 or 0_10
 4112AD Series: ADC0, Options: -10_10, -5_5 or 0_10
 4412AD: DAC0-3, Options: -10_10, -5_5 or 0_10
 4412DACL: DAC0-3, Options: -10_10, -5_5 or 0_10

AL_SetGainGlobal.vi

Supported Subsystems:

5401, 5403, 5801 and 5803: ADC0, Options: 1, 2, 4, 8
 5402, 5404, 5802 and 5804: ADC0, Options: 1, 10, 100, 500
 PCI5501: ADC0, Options: 1, 2, 4, 8
 PCI5502: ADC0, Options: 1, 10, 100
 PCI5503: ADC0, Options: 1, 2, 4, 8
 PCI5504: ADC0, Options: 1, 10, 100
 5508LF: ADC0, Options: 1, 2, 4, 8 or 1, 2, 5, 10
 5508SCi: ADC0, Options: 1, 2, 4, 8 or 1, 2, 5, 10
 5508TC/BG/RTD: ADC0, Options: 1, 2, 4, 8
 5516DMA: ADC0, Options: 1, 2, 4, 8
 5525/50MF: ADC0, Options: 1, 2, 4, 8
 4012AD: ADC0, Options: 1, 2, 4, 8
 4112AD: ADC0, Options: 1, 5, 10, 20, 100, 200, 500, 1000

Advanced Function Filters VIs:

AL_SetFilterType.vi

Supported Subsystems:

5508SCi: ADC0, Options: DISABLED, BUTTERWORTH, BESSEL

AL_SetFilterFreq.vi

Supported Subsystems:

Advanced Function Filters VIs (con't)

5508SCi: ADC0 0 to 100000

AL_GetFilterStruct.vi

Supported Subsystems:

5508SCi: ADC0

Advanced Function Buffers VIs:

AL_ClearBufferDoneFlag.vi

Supported Subsystems:

5400 Series: ADC0

PCI5500: ADC0

PCI55xx: ADC0

PCI55xx: DAC0, DAC1

5500HR: ADC0

5500MF: ADC0

5508LC: ADC0

5508LF: ADC0

5508SCi: ADC0

5508SHR: ADC0

5508TC/BG/RTD: ADC0

5516DMA: ADC0

5525/50MF: ADC0

5532TTL: DIN0, DIN1, DIN2, DIN3, DOT0, DOT1, DOT2, DOT3

5600 Series ACI, DCI, CCI, TTL: DIN0

5800 Series: ADC0

4012AD Series: ADC0

4112AD Series: ADC0

AL_SetBuffer.vi

Supported Subsystems:

5532TTL: DOT0, DOT1, DOT2, DOT3

AL_GetBufferStatus.vi

Supported Subsystems:

5400 Series: ADC0

PCI5500: ADC0

PCI55xx: ADC0

PCI55xx: DAC0, DAC1

5500HR: ADC0

5500MF: ADC0

5508LC: ADC0

5508LF: ADC0

5508SCi: ADC0

5508SHR: ADC0

5508TC/BG/RTD: ADC0

5516DMA: ADC0

5525/50MF: ADC0

Advanced Function Buffers VIs (con't)

5532TTL: DIN0, DIN1, DIN2, DIN3, DOT0, DOT1, DOT2, DOT3
5600 Series ACI, DCI, CCI, TTL: DIN0
5800 Series: ADC0
4012AD Series: ADC0
4112AD Series: ADC0

AL_SetBufferDoneHandler.vi & AL_GetBufferDoneHandler.vi

Supported Subsystems:
5400 Series: ADC0
PCI5500: ADC0
PCI55xx: ADC0
PCI55xx: DAC0, DAC1
5500HR: ADC0
5500MF: ADC0
5504DA: DAC0
5508LC: ADC0
5508LF: ADC0
5508SCi: ADC0
5508SHR: ADC0
5508TC/BG/RTD: ADC0
5516DMA: ADC0
5525/50MF: ADC0
5532TTL: DIN0, DIN1, DIN2 and DIN3
5532TTL: DOT0, DOT1, DOT2 and DOT3
5600 Series ACI, DCI, CCI, TTL: DIN0
5800 Series: ADC0
4012AD Series: ADC0
4112AD Series: ADC0

AL_SetNumOfBuffers.vi...& AL_GetNumOfBuffers.vi

Supported Subsystems:
5400 Series: ADC0
PCI5500: ADC0
PCI55xx: ADC0
PCI55xx: DAC0, DAC1
5500HR: ADC0
5500MF: ADC0
5508LC: ADC0
5508LF: ADC0
5508SCi: ADC0
5508SHR: ADC0
5508TC/BG/RTD: ADC0
5516DMA: ADC0
5525/50MF: ADC0
5532TTL: DIN0, DIN1, DIN2, DIN3, DOT0, DOT1, DOT2, DOT3
5600 Series ACI, DCI, CCI, TTL: DIN0
5800 Series: ADC0
4012AD Series: ADC0

Advanced Function Buffers VIs (con't)

4112AD Series: ADC0

AL_SetBufferSize.vi & AL_GetBufferSize.vi

Supported Subsystems:

5400 Series: ADC0

PCI5500: ADC0

PCI55xx: ADC0

PCI55xx: DAC0, DAC1

5500HR: ADC0

5500MF: ADC0

5508LC: ADC0

5508LF: ADC0

5508SCi: ADC0

5508SHR: ADC0

5508TC/BG/RTD: ADC0

5516DMA: ADC0

5525/50MF: ADC0

5532TTL: DIN0, DIN1, DIN2, DIN3, DOT0, DOT1, DOT2, DOT3

5600 Series ACI, DCI, CCI, TTL: DIN0

5800 Series: ADC0

4012AD Series: ADC0

4112AD Series: ADC0

AL_SetAutoInitBuffers.vi & AL_GetAutoInitBuffers.vi

Supported Subsystems:

All ADAC Board Subsystems that Require Buffers

Data Conversion:

AL_DemuxDataSet.vi

Supported Subsystems:

5400 Series: ADC0

PCI5500: ADC0

PCI55xx: ADC0

5500HR: ADC0

5500MF: ADC0

5508LC: ADC0

5508LF: ADC0

5508SCi: ADC0

5508SHR: ADC0

5508TC/BG/RTD: ADC0

5516DMA: ADC0

5525/50MF: ADC0

5800 Series: ADC0

4012AD Series: ADC0

4112AD Series: ADC0

Data Conversion (con't):

AL_TcTemp.vi

Supported Subsystems:

5400 Series: ADC0, Options:

PCI55xx: ADC0

5508SCi: ADC0, Options:

5508TC: ADC0, Options:

5525/50MF: ADC0, Options:

5800 Series: ADC0, Options:

4012AD Series: ADC0, Options:

4112AD Series: ADC0, Options:

AL_RtdTemp.vi

Supported Subsystems:

5508RTD: ADC0, Options:

4012AD Series: ADC0, Options:

4

ADAC-LVi Error Codes

ALERR_NOERRORS	1
ALERR_NOT_SUPPORTED	-1

MEMORY

ALERR_MEMORY_LOW	-100	Memory allocation failed
ALERR_DMA_MEMORY_LOW	-101	DMA memory allocation failed increase ADACDMABUFFERSIZE in system.ini file.

GENERIC

ALERR_ARRAY_PTR	-200	Invalid array pointer passed to function
ALERR_STRING_PTR	-201	Invalid string pointer passed to function
ALERR_MAXSTRING	-202	Input string size exceeds MAX length
ALERR_MAXARRAY	-203	Input array size exceeds MAX length
ALERR_INVALID_STRINGLIST	-204	Input string format invalid

Generic Codes (con't)

ALERR_DESTINATION_STRLEN	-205	Input parameter destination string length is less than the source string length
ALERR_MINARRAY	-206	Input array size exceeds MIN length

BOARD STRUCTURE

ALERR_BOARD_STRUCT_PTR	-300	Invalid BOARD struct pointer.
ALERR_BOARD_ID	-301	No BOARD struct for the specified Board ID exist.
ALERR_BOARD_CAPSFILE_STRPTR	-302	The BOARD struct CAP's file pointer is invalid.
ALERR_BOARD_NOBOARDS	-303	No board configurations found in .con file
ALERR_BOARD_MAX_ADCCAPS	-304	Max. ADC caps structures have been allocated
ALERR_BOARD_MAX_DACCAPS	-305	Max. DAC caps structures have been allocated
ALERR_BOARD_MAX_DINCAPS	-306	Max. DIN caps structures have been allocated
ALERR_BOARD_MAX_DOTCAPS	-307	Max. DOT caps structures have been allocated
ALERR_BOARD_MAX_CTRCAPS	-308	Max. CTR caps structures have been allocated
ALERR_BOARD_MAXSTRING	-309	The specified BOARD string exceeds max. length

LOGICAL DEVICE HANDLES

ALERR_LHLD	-400	The LHLD specified does not exist
ALERR_LHLD_MAX	-401	The maximum LHLD have already been allocated

LDS STRUCTURE

ALERR_LDS_STRUCT_PTR	-500	Invalid LDS struct pointer.
ALERR_INTERNAL_LDS_TYPE	-501	Unknown Logical Device Subsystem type.
ALERR_LDS_MAXSTRING	-502	The LDS type string specified in a call to adlib exceeds the maximum allowed length.
ALERR_LDS_MAXALLOCATED	-503	The maximum LDS have been allocated.
ALERR_LDS_NOCAPS	-504	No CAPS found.
ALERR_LDS_NOCAPSADC	-505	No CAPSADC found.
ALERR_LDS_NOCAPSDAC	-506	No CAPSDAC found.
ALERR_LDS_NOCAPSDIN	-507	No CAPSDIN found.
ALERR_LDS_NOCAPSDOT	-508	No CAPSDOT found.
ALERR_LDS_NOCAPSCTR	-509	No CAPSCTR found.

CAPS STRUCTURE

ALERR_CAPS_TYPE	-600	Unknown Capabilities Subsystem type.
ALERR_CAPS_STRUCT_PTR	-601	Invalid CAPS struct pointer.
ALERR_CAPSADC_STRUCT_PTR	-602	Invalid CAPSADC struct pointer.
ALERR_CAPSDAC_STRUCT_PTR	-603	Invalid CAPSCTRIO struct pointer.
ALERR_CAPSDIGIO_STRUCT_PTR	-604	Invalid CAPSDIGIO struct pointer.
ALERR_CAPSCTRIO_STRUCT_PTR	-605	Invalid CAPSCTRIO struct pointer.

ENVIRONMENT STRUCTURE

ALERR_ENV_STRUCT_PTR	-700	Invalid ENV struct pointer.
ALERR_ENV_LOADED	-701	The Environment is already loaded.
ALERR_ENV_NOTLOADED	-702	The Environment is NOT loaded.
ALERR_ENV_MAXSTRING	-703	The specified Environment is too long.

OPTIONS STRUCTURE

ALERR_OPTION_STRUCT_PTR	-800	Invalid OPTIONS struct pointer.
ALERR_OPTION_STRPTR	-801	Invalid option string pointer.
ALERR_OPTION_STRING	-802	Invalid option string format.

OPTIONS STRUCTURE (con't)

ALERR_OPTION_NAME_STRPTR	-803	Invalid option string name pointer.
ALERR_OPTION_NAME_STRING	-804	Invalid option string name.
ALERR_OPTION_ID_STRING	-805	Invalid option string ID.
ALERR_OPTION_CONFIG_TYPE	-806	Invalid option string configuration type.
ALERR_OPTION_MAXSTRING	-807	The option string exceeds the maximum allowed length.

FILE I/O

ALERR_FILE_FOUND	-900	The specified file can not be found.
------------------	------	--------------------------------------

INI FILE

ALERR_INI_FILEEXIST	-1000	The INFO file (.INI) can not be found.
ALERR_INI_PATH_STRPTR	-1001	Invalid .INI string pointer.
ALERR_INI_SECTION_STRPTR	-1002	Invalid .INI [SECTION] string pointer.
ALERR_INI_ENTRY_STRPTR	-1003	Invalid .INI ENTRY string pointer.
ALERR_INI_SECTION_UNKNOWN	-1004	Unknown .INI [SECTION].
ALERR_INI_BOARDID_RANGE	-1005	Invalid .INI BoardId setting.
ALERR_INI_DTM_STRING	-1006	Invalid .INI DataTransMethod setting.
ALERR_INI_CM_STRING	-1007	Invalid .INI CycleMode setting.
ALERR_INI_DMAMODE_STRING	-1008	Invalid .INI DmaMode setting.

INI FILE (con't)

ALERR_INI_DMACHAN_STRING	-1009	Invalid .INI DmaChan setting.
ALERR_INI_IRQMODE_STRING	-1010	Invalid .INI IrqMode setting.
ALERR_INI_IRQLEVEL_STRING	-1011	Invalid .INI IrqLevel setting.
ALERR_INI_ARBREINIT_STRING	-1012	Invalid .INI ArbReinit setting.
ALERR_INI_LOGERRORS_STRING	-1013	Invalid .INI LogErrors setting.
ALERR_INI_ERRONBUFFOVERRUN_STRING	-1014	Invalid .INI ErrorOnBufferOverrun setting.
ALERR_INI_ERRONRELRUNNING_STRING	-1015	Invalid .INI ErrorOnReleaseRunning setting.
ALERR_INI_BUFFERSIZE_LOW	-1016	Invalid .INI BufferSize setting.
ALERR_INI_NUMBUFFER_LOW	-1017	Invalid .INI NumBuffers setting.
ALERR_INI_BUFFERNOTIFYMETHOD_STRING	-1018	Invalid .INI BufferNotificationMethod setting
ALERR_INI_AUTOINITBUFFERS_STRING	-1019	Invalid .INI AutoInitBuffers setting
ALERR_INI_ERRONTRIGGEROVERRUN_STRING	-1020	Invalid .INI ErrOnTriggerOverrun setting
ALERR_INI_MINSTARTCHAN_STRING	-1100	Invalid .INI MinStartChan setting.
ALERR_INI_MAXENDCHAN_STRING	-1101	Invalid .INI MaxEndChan setting.
ALERR_INI_SIGNALPATH_STRING	-1102	Invalid .INI SignalPath setting.
ALERR_INI_INPUTCONFIG_STRING	-1103	Invalid .INI InputConfig setting.
ALERR_INI_CLKMODE_STRING	-1104	Invalid .INI ClkMode setting.

INI FILE (con't)

ALERR_INI_CLKSOURCE_STRING	-1105	Invalid .INI ClkSource setting.
ALERR_INI_CLKSOURCESIGNAL_STRING	-1106	Invalid .INI ClkSourceSignal setting.
ALERR_INI_CLKRATE_STRING	-1107	Invalid .INI ClkRate setting.
ALERR_INI_CLKRATE_RANGE	-1108	Invalid .INI ClkRate range.
ALERR_INI_CLKRATEUNITS_STRING	-1109	Invalid .INI ClkRateUnits setting.
ALERR_INI_BURSTLENGTH_STRING	-1110	Invalid .INI BurstLength setting.
ALERR_INI_BURSTLENGTH_RANGE	-1111	Invalid .INI BurstLength range.
ALERR_INI_BURSTRATE_STRING	-1112	Invalid .INI BurstRate setting.
ALERR_INI_BURSTRATE_RANGE	-1113	Invalid .INI BurstRate range.
ALERR_INI_BURSTRATEUNITS_STRING	-1114	Invalid .INI BurstRateUnits setting.
ALERR_INI_TRIGMODE_STRING	-1115	Invalid .INI TrigMode setting.
ALERR_INI_TRIGSRC_STRING	-1116	Invalid .INI TrigSource setting.
ALERR_INI_TRIGSRCSIGNAL_STRING	-1117	Invalid .INI TrigSourceSignal setting.
ALERR_INI_TRIGRATE_STRING	-1118	Invalid .INI TrigRate setting.
ALERR_INI_TRIGRATE_RANGE	-1119	Invalid .INI MinTrigRate or MaxTrigRate setting.
ALERR_INI_TRIGRATEUNITS_STRING	-1120	Invalid .INI TrigRateUnits setting.
ALERR_INI_DATACODE_STRING	-1121	Invalid .INI DataCode setting.
ALERR_INI_DATAOFFSET_STRING	-1122	Invalid .INI DataOffset setting.
ALERR_INI_DATASPAN_STRING	-1123	Invalid .INI DataSpan setting.

INI FILE (con't)

ALERR_INI_DATARANGE_STRING	-1124	Invalid .INI DataRange setting.
ALERR_INI_OUTPUTCONFIG_STRING	-1125	Invalid .INI OutputConfig setting.
ALERR_INI_EXPPANEL_STRING	-1126	Invalid .INI ExpansionPanel n setting.
ALERR_INI_POSTSAMPLECOUNT_STRING	-1127	Invalid .INI PostSampleCounts setting.
ALERR_INI_POSTSAMPLECOUNT_RANGE	-1128	Invalid .INI PostSampleCounts range.
ALERR_INI_GATESRC_STRING	-1129	Invalid .INI GatrSource setting.
ALERR_INI_GATESRCLEVEL_STRING	-1130	Invalid .INI GateSourceLevel setting.
ALERR_INI_BURSTMODE_STRING	-1131	Invalid .INI BurstMode setting.
ALERR_INI_FILTERTYPE_STRING	-1132	Invalid .INI Filtertype setting.
ALERR_INI_FILTERFREQ_STRING	-1133	Invalid .INI FilterFreq setting.
ALERR_INI_FILTERFREQ_RANGE	-1134	Invalid .INI FilterFreq range.
ALERR_INI_HANDSHAKE_STRING	-1135	Invalid .INI HandShake setting.
ALERR_INI_PORTRESOLUTION_STRING	-1136	Invalid .INI PortResolution setting.
ALERR_INI_PORTMASK_STRING	-1137	Invalid .INI PortMask setting.
ALERR_INI_PORTRMASK_RANGE	-1138	Invalid .INI PortMask range.
ALERR_INI_CTRMODE_STRING	-1139	Invalid .INI CtrMode setting.
ALERR_INI_CTRRATEUNITS_STRING	-1140	Invalid .INI CtrRateUnits setting.
ALERR_INI_CTRRATE_STRING	-1141	Invalid INI CtrRate setting.
ALERR_INI_PACKEDDATA_STRING	-1142	Invalid .INI PackedData setting.

INI FILE (con't)

ALERR_INI_CLKOUTPUT_STRING	-1143	Invalid .INI ClockOutput setting.
ALERR_INI_TRIGOUTPUT_STRING	-1144	Invalid .INI TrigOutput setting.

CAPABILITIES FILE

ALERR_CAPS_FILEEXIST	-2000	The Capabilities (.CAP) can not be found.
ALERR_CAPS_MFG_STRING	-2001	Invalid .CAP manufacture setting.
ALERR_CAPS_MODEL_STRING	-2002	Invalid .CAP model setting.

BOARD

ALERR_CAPS_IOBASEADDRSEL_STRING	-2100	Invalid .CAP IoBaseAddrSelect setting.
ALERR_CAPS_MINIOBASEARRR_STRING	-2101	Invalid .CAP MinIoBaseAddress setting.
ALERR_CAPS_MAXIOBASEARRR_STRING	-2102	Invalid .CAP MaxIoBaseAddress setting.

DRIVER

ALERR_CAPS_DRIVERVERSION_STRING	-2200	Invalid .CAP DriverVersion setting.
ALERR_CAPS_DRIVERNAME_STRING	-2201	Invalid .CAP DriverName setting.
ALERR_CAPS_BOARDVERSION_STRING	-2202	Invalid .CAP BoardVersionSupport setting.
ALERR_CAPS_BOARDIDSUPPORT_STRING	-2203	Invalid .CAP BoardVersionId setting.
ALERR_CAPS_VERSION_STRING	-2204	Invalid .CAP file Version setting.

COMMON DEVICE SETTINGS

ALERR_CAPS_IRQSHAREABLE_STRING	-2300	Invalid .CAP IrqShareable setting.
ALERR_CAPS_DTM_STRING	-2301	Invalid .CAP DataTransMethod setting.
ALERR_CAPS_DTM_ENTRY	-2302	CAP DataTransMethod entry does not exist.
ALERR_CAPS_CM_STRING	-2303	Invalid .CAP CycleMode setting.
ALERR_CAPS_CM_ENTRY	-2304	CAP CycleMode entry does not exist.
ALERR_CAPS_BUFFERNOTIFYMETHOD_STRING	-2305	Invalid .CAP buffer notification setting.
ALERR_CAPS_DMAMODE_STRING	-2307	Invalid .CAP DmaMode setting.
ALERR_CAPS_DMACHAN_STRING	-2308	Invalid .CAP DmaChan setting.
ALERR_CAPS_IRQMODE_STRING	-2309	Invalid .CAP IrqMode setting.
ALERR_CAPS_IRQLEVEL_STRING	-2310	Invalid .CAP IrqLevel setting.
ALERR_CAPS_BUFFERSUPPORT_STRING	-2211	Invalid .CAP BoardVersionId setting
ALERR_CAPS_MINBUFFERS_STRING	-2312	Invalid .CAP MinStartChan setting.
ALERR_CAPS_MAXBUFFERS_STRING	-2313	Invalid .CAP MaxEndChan setting.
ALERR_CAPS_MINBUFFERSIZE_STRING	-2314	Invalid .CAP Min buffer size setting.
ALERR_CAPS_MAXBUFFERSIZE_STRING	-2315	Invalid .CAP Max buffer size setting.
ALERR_CAPS_PACKEDFIFO_STRING	-2316	Invalid .CAP PackedFifoSupport setting.

ADC

ALERR_CAPS_GAIN_STRING	-2351	Invalid .CAP GAIN string format
ALERR_CAPS_MINSTARTCHAN_STRING	-2352	Invalid .CAP MinStartChan setting.
ALERR_CAPS_MAXENDCHAN_STRING	-2353	Invalid .CAP MaxEndChan setting.
ALERR_CAPS_ARBCHAN_STRING	-2354	Invalid .CAP ArbChan setting.
ALERR_CAPS_ARBGAIN_STRING	-2355	Invalid .CAP ArbGain setting.
ALERR_CAPS_CLKMODE_STRING	-2356	Invalid .CAP ClkMode setting.
ALERR_CAPS_CLKSOURCE_STRING	-2357	Invalid .CAP ClkSource setting.
ALERR_CAPS_CLKSOURCESIGNAL_STRING	-2358	Invalid .CAP ClkSourceSignal setting.
ALERR_CAPS_MINCLKRATE_STRING	-2359	Invalid .CAP MinClkRate setting.
ALERR_CAPS_MAXCLKRATE_STRING	-2360	Invalid .CAP MaxClkRate setting.
ALERR_CAPS_MINBURSTLENGTH_STRING	-2361	Invalid .CAP MinBurstLength setting.
ALERR_CAPS_MAXBURSTLENGTH_STRING	-2362	Invalid .CAP MaxBurstLength setting.
ALERR_CAPS_MINBURSTRATE_STRING	-2363	Invalid .CAP MinBurstRate setting.
ALERR_CAPS_MAXBURSTRATE_STRING	-2364	Invalid .CAP MaxBurstRate setting.
ALERR_CAPS_TRIGMODE_STRING	-2365	Invalid .CAP TrigMode setting.
ALERR_CAPS_TRIGSRC_STRING	-2366	Invalid .CAP TrigSource setting.
ALERR_CAPS_TRIGSRC SIGNAL_STRING	-2367	Invalid .CAP TrigSourceSignal setting.
ALERR_CAPS_MINTRIGRATE_STRING	-2368	Invalid .CAP MinTrigRate setting.
ALERR_CAPS_MAXTRIGRATE_STRING	-2369	Invalid .CAP MaxTrigRate setting.

ADC (con't)

ALERR_CAPS_CJ_STRING	-2370	Invalid .CAP CJ setting.
ALERR_CAPS_ARBCJ_STRING	-2371	Invalid .CAP ArbCj setting.
ALERR_CAPS_DATACODE_STRING	-2372	Invalid .CAP DataCode setting.
ALERR_CAPS_DATAOFFSET_STRING	-2373	Invalid .CAP DataOffset setting.
ALERR_CAPS_DATASPAN_STRING	-2374	Invalid .CAP DataSpan setting.
ALERR_CAPS_FIFOSIZE_STRING	-2375	Invalid .CAP FifoSize setting.
ALERR_CAPS_DATARANGE_STRING	-2376	Invalid .CAP DataRange setting.
ALERR_CAPS_OUTPUTCONFIG_STRING	-2377	Invalid .CAP OutputConfig setting.
ALERR_CAPS_MAXEXPPANELS_STRING	-2378	Invalid .CAP MaxNumExpPanels setting.
ALERR_CAPS_EXPPANEL_STRING	-2379	Invalid .CAP ExpPanel setting.
ALERR_CAPS_MAXEXPPANELS_MAX	-2380	Invalid .CAP MaxNumExpPanels to High.
ALERR_CAPS_BYTEPERSMPL_STRING	-2381	Invalid .CAP BytesPerSample setting.
ALERR_CAPS_MINPOSTSAMPLE_STRING	-2382	Invalid .CAP MinPostSamples setting.
ALERR_CAPS_MAXPOSTSAMPLE_STRING	-2383	Invalid .CAP MaxPostSamples setting.
ALERR_CAPS_GATESRC_STRING	-2384	Invalid .CAP GateSource setting.
ALERR_CAPS_GATESRCLEVEL_STRING	-2385	Invalid .CAP GateSourceLevel setting.
ALERR_CAPS_BURSTMODE_STRING	-2386	Invalid .CAP BurstMode setting.
ALERR_CAPS_SIGNALPATH_STRING	-2387	Invalid .CAP SignalPath setting.
ALERR_CAPS_INPUTCONFIG_STRING	-2388	Invalid .CAP InputConfig setting.

ADC (con't)

ALERR_CAPS_DATAMASK_STRING	-2389	Invalid .CAP DataMask setting.
ALERR_CAPS_FILTERTYPE_STRING	-2390	Invalid .CAP FilterType setting.
ALERR_CAPS_MINFILTERFREQ_STRING	-2391	Invalid .CAP MinFilterFreq setting.
ALERR_CAPS_MAXFILTERFREQ_STRING	-2392	Invalid .CAP MaxFilterFreq setting.
ALERR_CAPS_HANDSHAKE_STRING	-2393	Invalid .CAP HandShake setting.
ALERR_CAPS_PORTRESOLUTION_STRING	-2394	Invalid .CAP PortResolution setting.
ALERR_CAPS_MINPORTMASK_STRING	-2395	Invalid .CAP MinPortMask setting.
ALERR_CAPS_MAXPORTMASK_STRING	-2396	Invalid .CAP MaxPortMask setting.
ALERR_CAPS_CLKOUTPUT_STRING	-2397	Invalid .CAP ClockOutput setting.
ALERR_CAPS_TRIGOUTPUT_STRING	-2398	Invalid .CAP TrigOutput setting.
ALERR_CAPS_INPUTCONFIG_STRING	-2399	Invalid .CAP InputConfig setting.
ALERR_CAPS_DATAOFFSET_STRING	-2400	Invalid .CAP DataOffset setting.
ALERR_CAPS_CTRMODE_STRING	-2401	Invalid .CAP CtrMode setting.
ALERR_CAPS_MINRATE_STRING	-2402	Invalid .CAP MinRate setting.
ALERR_CAPS_MAXRATE_STRING	-2403	Invalid .CAP MaxRate setting.

CONFIGURATION FILE

ALERR_CON_FILEEXIST	-3000	The Configuration (.CON) can not be found.
ALERR_CON_FILE_MAXSTRING	-3001	The (.CON) file string exceeds the maximum allowed length.
ALERR_CON_FILE_STRPTR	-3002	Invalid .CON file string pointer.
ALERR_CON_BOARDSECTION_STRPTR	-3003	Invalid .CON board section pointer.
ALERR_CON_BOARDSECTION_STRING	-3004	Invalid .CON board section string
ALERR_CON_BOARDID_SAME	-3005	Invalid .CON BoardId setting.
ALERR_CON_BOARDID_RANGE	-3006	Invalid .CON BoardId setting.
ALERR_CON_IOBASEADDR_STRING	-3007	Invalid .CON IoBaseAddr setting.
ALERR_CON_BOARDCAPSFILE_STRING	-3008	Invalid .CON capabilities file setting.
ALERR_CON_DRVPATH_STRING	-3009	Invalid .CON driver Path setting.
ALERR_SYSTEMDRV_STRING	-3010	Invalid .CON System driver setting

DEVICE SUBSYSTEM CHANNELS

ALERR_CHANLIST_LISTTYPE	-4000	Invalid Channel list struct list lType, is it an Array or String list.
ALERR_CHANLIST_STRPTR	-4001	Invalid Channel list string pointer.
ALERR_CHANLIST_STRING	-4002	Invalid Channel list string.
ALERR_CHANLIST_MAXSTRLEN	-4003	Invalid Channel list string length.
ALERR_CHANLIST_ARRAYPTR	-4004	Invalid Channel list array pointer.
ALERR_CHANLIST_ARRAY_LENTHPTR	-4005	Invalid Channel list array length pointer.

DEVICE SUBSYSTEM CHANNELS (continued)

ALERR_CHANLIST_MAXARRAYLEN	-4006	Invalid Channel list array length.
ALERR_CHANLIST_MINCHAN	-4007	Invalid Channel in list
ALERR_CHANLIST_MAXCHAN	-4008	Invalid Channel in list
ALERR_CHANLIST_NONSEQ	-4009	Non-sequential Channels are not supported.
ALERR_GLOBALGAIN_STRPTR	-4010	Invalid Global Gain string pointer.
ALERR_GLOBALGAIN_GAIN	-4011	Invalid Gain type.
ALERR_CHANGAINLIST_LISTTYPE	-4050	Invalid Channel list struct list IType, is it an Array or String list.
ALERR_CHANGAINLIST_STRPTR	-4051	Invalid Channel list string pointer.
ALERR_CHANGAINLIST_STRING	-4052	Invalid Channel list string.
ALERR_CHANGAINLIST_MAXSTRLEN	-4053	Invalid Channel list string length.
ALERR_CHANGAINLIST_ARRAYPTR	-4054	Invalid Channel list array pointer.
ALERR_CHANGAINLIST_ARRAY_LENTHPTR	-4055	Invalid Channel list array length pointer.
ALERR_CHANGAINLIST_MAXARRAYLEN	-4056	Invalid Channel list array length.
ALERR_CHANGAINLIST_MINCHAN	-4057	Invalid Channel in list
ALERR_CHANGAINLIST_MAXCHAN	-4058	Invalid Channel in list
ALERR_CHANGAINLIST_NONSEQ	-4059	Non-sequential Channels are not supported.

DEVICE SUBSYSTEM CHANNELS (continued)

ALERR_CHANGAINLIST_SYNTAX	-4060	Invalid ChanGainList syntax
ALERR_CHANGAINLIST_ARBGAINS	-4061	Arbitrary Gains are not supported.
ALERR_CHANGAINLIST_GAINTYPE	-4062	Invalid Gain type

DEVICE SUBSYSTEM EXPANSION PANEL GAINS

ALERR_EXPPANELGAINLIST_LISTTYPE	-4100	Invalid ExpPanelGainList structure list lType
ALERR_EXPPANELGAINLIST_STRPTR	-4101	Invalid ExpPanelGainList string pointer.
ALERR_EXPPANELGAINLIST_STRING	-4102	Invalid ExpPanelGainList string.
ALERR_EXPPANELGAINLIST_MAXSTRLEN	-4103	Invalid ExpPanelGainList string length.
ALERR_EXPPANELGAINLIST_GAINSETTING	-4104	Invalid ExpPanelGainList gain setting.
ALERR_EXPPANELGAINLIST_ARRAYPTR	-4105	Invalid ExpPanelGainList array pointer.
ALERR_EXPPANELGAINLIST_ARRAY_LENGTHPTR	-4106	Invalid ExpPanelGainList array length pointer.
ALERR_EXPPANELGAINLIST_MAXARRAYLEN	-4107	Invalid ExpPanelGainList array length.
ALERR_EXPPANELGAINLIST_MINCHAN	-4108	Invalid ExpPanelGainList channel.
ALERR_EXPPANELGAINLIST_MAXCHAN	-4109	Invalid ExpPanelGainList channel.
ALERR_EXPPANELGAINLIST_SYNTAX	-4110	Invalid ExpPanelGainList syntax.

DEVICE SUBSYSTEM INPUT CONFIGURATION LIST

ALERR_INPUTCONFIG_LISTTYPE	-4120	Invalid InputConfig structure list lType.
ALERR_INPUTCONFIG_STRPTR	-4121	Invalid InputConfig string pointer.
ALERR_INPUTCONFIG_STRING	-4122	Invalid InputConfig string.
ALERR_INPUTCONFIG_MAXSTRLEN	-4123	Invalid InputConfig string length.
ALERR_INPUTCONFIG_GAINSETTING	-4124	Invalid InputConfig gain setting.
ALERR_INPUTCONFIG_ARRAYPTR	-4125	Invalid InputConfig array pointer.
ALERR_INPUTCONFIG_ARRAY_LENGTHPTR	-4126	Invalid InputConfig array length pointer.
ALERR_INPUTCONFIG_MAXARRAYLEN	-4127	Invalid InputConfig array length.
ALERR_INPUTCONFIG_MINCHAN	-4128	Invalid InputConfig min channel specified.
ALERR_INPUTCONFIG_MAXCHAN	-4129	Invalid InputConfig max channel specified.
ALERR_INPUTCONFIG_SYNTAX	-4130	Invalid InputConfig list syntax
ALERR_GLOBAL INPUTCONFIG_STRPTR	-4131	Invalid global InputConfig string pointer.
ALERR_GLOBAL INPUTCONFIG_TYPE	-4132	Invalid global InputConfig type

DEVICE SUBSYSTEM DATA OFFSET LIST

ALERR_DATAOFFSET_LISTTYPE	-4140	Invalid DataOffset structure list lType.
ALERR_DATAOFFSET_STRPTR	-4141	Invalid DataOffset string pointer.
ALERR_DATAOFFSET_STRING	-4142	Invalid DataOffset string.
ALERR_DATAOFFSET_MAXSTRLEN	-4143	Invalid DataOffset string length.

DEVICE SUBSYSTEM DATA OFFSET LIST (continued)

ALERR_DATAOFFSET_GAINSETTING	-4144	Invalid DataOffset gain setting.
ALERR_DATAOFFSET_ARRAYPTR	-4145	Invalid DataOffset array pointer.
ALERR_DATAOFFSET_ARRAY_LENGTHPTR	-4146	Invalid DataOffset array length pointer.
ALERR_DATAOFFSET_MAXARRAYLEN	-4147	Invalid DataOffset array length.
ALERR_DATAOFFSET_MINCHAN	-4148	Invalid DataOffset min channel specified.
ALERR_DATAOFFSET_MAXCHAN	-4149	Invalid DataOffset max channel specified.
ALERR_DATAOFFSET_SYNTAX	-4150	Invalid DataOffset list syntax.
ALERR_GLOBAL_DATAOFFSET_STRPTR	-4151	Invalid global DataOffset string pointer.
ALERR_GLOBAL_DATAOFFSET_TYPE	-4152	Invalid global DataOffset type.

DEVICE SUBSYSTEM EXPANSION PANEL

ALERR_EXPPANELLIST_LISTTYPE	-4200	Invalid ExpPanelList structure list lType.
ALERR_EXPPANELLIST_STRPTR	-4201	Invalid ExpPanelList string pointer.
ALERR_EXPPANELLIST_STRING	-4202	Invalid ExpPanelList string.
ALERR_EXPPANELLIST_MAXSTRLEN	-4203	Invalid ExpPanelList string length.
ALERR_EXPPANELLIST_SETTING	-4204	Invalid ExpPanelList gain setting.
ALERR_EXPPANELLIST_ARRAYPTR	-4205	Invalid ExpPanelList array pointer.
ALERR_EXPPANELLIST_ARRAY_LENGTHPTR	-4206	Invalid ExpPanelList array length pointer.

DEVICE SUBSYSTEM EXPANSION PANEL (continued)

ALERR_EXPPANELLIST_MAXARRAYLEN	-4207	Invalid ExpPanelList array length.
ALERR_EXPPANELLIST_MAXPANELS	-4208	Invalid Number of ExpPanels specified.
ALERR_EXPPANELLIST_PANELTYPE	-4209	Invalid ExpPanelList panel type.

DEVICE SUBSYSTEM THERMOUCOUPLES

ALERR_CJLIST_LISTTYPE	-4401	Invalid CJ structure list IType.
ALERR_CJLIST_STRPTR	-4402	Invalid CJ string pointer.
ALERR_CJLIST_STRING	-4403	Invalid CJ string.
ALERR_CJLIST_MAXSTRLEN	-4404	Invalid CJ string length.
ALERR_CJLIST_ARRAYPTR	-4405	Invalid CJ array pointer.
ALERR_CJLIST_ARRAY_LENGTHPTR	-4406	Invalid CJ array length pointer.
ALERR_CJLIST_MAXARRAYLEN	-4407	Invalid CJ array length.
ALERR_CJLIST_ARBCJ	-4408	Arbitrary CJs are not supported.
ALERR_CJLIST_MINCHAN	-4409	Invalid CJ channel specified.
ALERR_CJLIST_MAXCHAN	-4410	Invalid CJ channel specified.
ALERR_CJLIST_CJTYPE	-4411	Invalid CJ type
ALERR_CJLIST_SYNTAX	-4412	Invalid CJ list syntax
ALERR_GLOBALCJ_STRPTR	-4413	Invalid global CJ string pointer.
ALERR_GLOBALCJ_CJTYPE	-4414	Invalid global CJ type

TRIGGERS

ALERR_TRIGGERMODE_UNSUPPORTED	-4500	Trigger Mode is not supported.
ALERR_TRIGGERMODE_OPTION	-4501	Specified trigger mode source is not supported.
ALERR_TRIGGER_RATELOW	-4502	Invalid trigger rate specified
ALERR_TRIGGER_RATEHIGH	-4503	Invalid trigger rate specified
ALERR_TRIGGER_MINPOSTSAMPLECOUNT	-4504	Invalid post sample count specified
ALERR_TRIGGER_MAXPOSTSAMPLECOUNT	-4505	Invalid post sample count specified.
ALERR_TRIGGER_POSTSMPLCNT_BUFFSIZE	-4506	Specified Post sample counts > bufferSize.

TRIGGER SOURCES

ALERR_TRIGGERSOURCE_OPTION	-4507	Specified Trigger source is not supported.
ALERR_TRIGGERSOURCE_UNSUPPORTED	-4510	Trigger sources is not supported.

TRIGGER SIGNALS

ALERR_TRIGGERSOURCE SIGNAL_UNSUPPORTED	-4508	Triggering is not supported.
ALERR_TRIGGERSOURCE SIGNAL_OPTION	-4509	Specified trigger signal source is not supported.

TRIGGER OUTPUTS

ALERR_TRIGGERSOURCE_UNSUPPORTED	-4511	Triggering Output is not supported.
ALERR_TRIGGERSOURCE SIGNAL_OPTION	-4512	Specified trigger signal source is not supported.

CLOCKING

ALERR_CLOCKING_UNSUPPORTED	-4600	Clocking is not supported.
ALERR_CLOCKING_OPTION	-4601	Specified Clock option is not supported.
ALERR_CLOCKING_RATELOW	-4602	Invalid clock rate specified.
ALERR_CLOCKING_RATEHIGH	-4603	Invalid clock rate specified.

CLOCK SIGNALS

ALERR_CLOCKSIGNAL_UNSUPPORTED	-4604	Clocking is not supported.
ALERR_CLOCKSIGNAL_OPTION	-4605	Specified Clock option is not supported.

CLOCK OUTPUTS

ALERR_CLOCKSOUTPUT_UNSUPPORTED	-4606	Clock Output is not supported.
ALERR_CLOCKOUTPUT_OPTION	-4607	Specified Clock Output is not supported.

DATA CODE

ALERR_DATACODE_UNSUPPORTED	-4700	Data Coding is not supported.
ALERR_DATACODE	-4701	Specified Data Code option is not valid.

DATA OFFSET

ALERR_DATAOFFSET_UNSUPPORTED	-4800	Data Offset is not supported.
ALERR_DATAOFFSET	-4801	Specified Data Offset option is not valid.

DMA

ALERR_DMA_MODES_UNSUPPORTED	-4900	DMA Modes are not supported.
ALERR_DMA_MODE	-4901	Specified DMA Modes option is not valid.
ALERR_DMA_CHANS_UNSUPPORTED	-4902	DMA Channels are not supported.
ALERR_DMA_CHAN	-4903	Specified DMA Channel option is not valid.
ALERR_DMA_INUSE	-4904	Specified DMA Channel is already in use.
ALERR_SETDMASTATUS_FLAG	-4905	The status flag specified is unknown.

INTERRUPTS

ALERR_IRQ_LEVELS_UNSUPPORTED	-5000	IRQ Levels are not supported.
ALERR_IRQ_LEVEL	-5001	Specified IRQ Level option is not valid.
ALERR_IRQ_INUSE	-5002	Specified IRQ Level is already in use.
ALERR_SETIRQSTATUS_FLAG	-5003	The status flag specified is unknown.
ALERR_SETIRQSTATUS_OWNER	-5004	Invalid IRQ owner setting status flags.

SIGNAL PATHS

ALERR_SIGNALPATH_UNSUPPORTED	-5100	Signal Paths are not supported.
ALERR_SIGNALPATH	-5101	Specified Signal Path option is not valid.

DATA TRANSFER METHOD

ALERR_DTM_UNSUPPORTED	-5200	DTM Methods are not supported.
ALERR_DTM	-5201	Specified DTM Method option is not valid.

CYCLE MODE

ALERR_CM_UNSUPPORTED	-5300	CM Methods are not supported.
ALERR_CM	-5301	Specified CM option is not valid.
ALERR_CM_NUMBUFFERS	-5302	Selected CM option is not valid for the specified number of buffers.

DATA SPAN

ALERR_DATASPAN_UNSUPPORTED	-5400	Data Span is not supported.
ALERR_DATASPAN	-5401	Specified Data Span option is not valid.

DATA RANGE

ALERR_DATARANGE_UNSUPPORTED	-5500	Data Range is not supported.
ALERR_DATARANGE	-5501	Specified Data Range option is not valid.

INPUT CONFIG

ALERR_INPUTCONFIG_UNSUPPORTED	-5600	Input Config is not supported.
ALERR_INPUTCONFIG	-5601	Specified Input Config option is not valid.

OUTPUT CONFIG

ALERR_OUTPUTCONFIG_UNSUPPORTED	-5700	Signal Paths are not supported.
ALERR_OUTPUTCONFIG	-5701	Specified Signal Path option is not valid.

BUFFER NOTIFICATION METHOD

ALERR_SETBUFFDONEHANDLER_UNSUPPORTED	-5800	Buffer Notification Methods are not supported.
ALERR_SETBUFFDONEHANDLER_METHOD	-5801	Specified buffer Notification Method option is not valid.
ALERR_SETBUFFDONEHANDLERMSG_UNSUPPORTED	-5802	Buffer PostMessage Notification is not supported.
ALERR_SETBUFFDONEHANDLERFUNC_UNSUPPORTED	-5803	Buffer CallBack Notification's are not supported.
ALERR_SETBUFFDONEHANDLERMSGPARAMS_UNSUPPORTED	-5804	Buffer PostMessage Parameters are not supported.

START DEVICE

ALERR_DEVICE_BUSY	-5900	The device is already running.
ALERR_DEVICE_UNINITIALIZED	-5901	The device is not initialized.

DATA BUFFER CONTROL ERRORS

ALERR_BUFFER_HANDLER	-6000	Invalid handler specified
ALERR_BUFFER_SIZE	-6001	Invalid buffer size specified
ALERR_BUFFER_NUMBEROF	-6002	Invalid number of buffers specified
ALERR_BUFFER_TYPE	-6003	Invalid buffer type specified

DATA BUFFER CONTROL ERRORS (con't)

ALERR_BUFFER_NOTIFY_METHOD	-6004	Invalid buffer notification method specified
ALERR_BUFFER_NOTSUPPORTED	-6005	Buffer(s) are not supported on this LHL D
ALERR_BUFFER_POINTER	-6006	Buffer pointer equal to NULL
ALERR_BUFFER_ERROR	-6007	An error condition occurred in the previous buffer making the next DONE_BUFFER unavailable.
ALERR_DONEBUFFER_NOTREADY	-6008	The next buffer has not completed
ALERR_BUFFER_STARTPOINT	-6009	The specified buffer StartPoint position number is out of range for the available buffer size.
ALERR_BUFFER_MAXCOUNT	-6010	The specified number of buffer counts exceeds the available buffer size.
ALERR_BUFFER_STATUS_PTR	-6011	Invalid LPDATABUFFSTATUS specified
ALERR_BUFFER_NUMBER	-6012	Invalid buffer number specified
ALERR_BUFFER_INTERNAL	-6013	Unable to locate specified buffer

BURST MODE CONTROL ERRORS

ALERR_BURSTMODE_UNSUPPORTED	-7000	Burst mode is not supported on this LHL D
ALERR_BURSTMODE_OPTION	-7001	Invalid burst mode specified
ALERR_BURST_RATELOW	-7002	Burst mode rate too low for this LHL D
ALERR_BURST_RATEHIGH	-7003	Burst mode rate too high for this LHL D
ALERR_BURST_MINLENGTH	-7004	Burst mode length too low for this LHL D
ALERR_BURST_MAXLENGTH	-7005	Burst mode length too high for this LHL D

GATING

ALERR_GATING_UNSUPPORTED	-8000	Gating is not supported.
ALERR_GATING_OPTION	-8001	Specified Gate option is not supported.
ALERR_SETSWGATE_UNSUPPORTED	-8002	SetSwGate function is not supported.
ALERR_GETSWGATE_UNSUPPORTED	-8003	GetSwGate function is not supported.

GATE LEVELS

ALERR_GATELEVELS_UNSUPPORTED	-9000	Gate levels are not supported.
ALERR_GATELEVEL_OPTION	-9001	Specified Gate level option is not supported.

DRIVERS

ALERR_DRV_NOT_LOADED	-10002	The board's device driver is not loaded
ALERR_DRV_ADDR_STRUCT_PTR	-10004	Invalid Driver address struct pointer
ALERR_DRV_STOPDEV_ADDR_PTR	-10005	Driver StopDevice function not found
ALERR_DRV_DTM_ADDR_PTR	-10006	Driver DataTransMethod function not found
ALERR_DRV_CM_ADDR_PTR	-10007	Driver CycleMode function not found
ALERR_DRV_SETDMA_ADDR_PTR	-10008	Driver SetDma function not found
ALERR_DRV_SETIRQ_ADDR_PTR	-10009	Driver SetIrq function not found
ALERR_DRV_INIT_ADDR_PTR	-10010	Driver InitAddress function not found
ALERR_DRV_SETCHANGAIN_ADDR_PTR	-10011	Driver SetChanGain function not found
ALERR_DRV_SETCJ_ADDR_PTR	-10012	Driver SetCj function not found
ALERR_DRV_SETBURSTMODE_ADDR_PTR	-10013	Driver SetBurstMode function not found
ALERR_DRV_SETCLKMODE_ADDR_PTR	-10014	Driver SetClkMode function not found
ALERR_DRV_SETCLKSRC_ADDR_PTR	-10015	Driver SetClkSrc function not found
ALERR_DRV_SETCLKSRCSIG_ADDR_PTR	-10016	Driver SetClkSrcSig function not found
ALERR_DRV_SETTRIGMODE_ADDR_PTR	-10017	Driver SetTrigMode function not found
ALERR_DRV_SETTRIGSRC_ADDR_PTR	-10018	Driver SetTrigSrc function not found
ALERR_DRV_SETTRIGSRCSIG_ADDR_PTR	-10019	Driver SetTrigSrcSig function not found
ALERR_DRV_SETINPUT_ADDR_PTR	-10020	Driver SetInputConfig function not found
ALERR_DRV_SETDATACODE_ADDR_PTR	-10021	Driver SetDataCode function not found

DRIVERS (con't)

ALERR_DRV_SETDATAOFFSET_ADDR_PTR	-10022	Driver SetDataOffset function not found.
ALERR_DRV_SETDATASPAN_ADDR_PTR	-10023	Driver SetDataSpan function not found.
ALERR_DRV_SETSIGNALPATH_ADDR_PTR	-10024	Driver SetSignalPath function not found.
ALERR_DRV_SETDATARANGE_ADDR_PTR	-10025	Driver SetDataRange function not found.
ALERR_DRV_SETOUTPUTCONFIG_ADDR_PTR	-10026	Driver SetOutputConfig function not found.
ALERR_DRV_SETBUFFERS_ADDR_PTR	-10027	Driver SetBuffers function not found.
ALERR_DRV_SETDAOUTPUT_ADDR_PTR	-10028	Driver SetDaOutput function not found.
ALERR_DRV_BUFFERNOTIFYMETHOD_ADDR_PTR	-10029	Driver SetBuffNotifyMethod function not found.
ALERR_DRV_SETPOSTSMPLCOUNTS_ADDR_PTR	-10030	Driver SetOutputConfig function not found.
ALERR_DRV_STARTDEV_ADDR_PTR	-10031	Driver StartDevice function not found.
ALERR_DRV_GETDEVSTATUS_ADDR_PTR	-10032	Driver GetDeviceStatus function not found.
ALERR_DRV_DIGINPUT_ADDR_PTR	-10033	Driver Digital Input function not found.
ALERR_DRV_DIGBITSTEST_ADDR_PTR	-10034	Driver Digital Bit Test function not found.
ALERR_DRV_DIGOUTPUT_ADDR_PTR	-10035	Driver Digital Output function not found.
ALERR_DRV_SETGATESRC_ADDR_PTR	-10036	Driver SetGateSrc function not found.
ALERR_DRV_SETGATESRCLEVEL_ADDR_PTR	-10037	Driver SetGateSrcLevel function not found.
ALERR_DRV_SETSWGATE_ADDR_PTR	-10038	Driver SetSWGate function not found.

DRIVERS (con't)

ALERR_DRV_GETSWGATE_ADDR_PTR	-10039	Driver GetSWGate function not found.
ALERR_DRV_BOARDHWID_ADDR_PTR	-10040	Driver GetBoardID function not found.
ALERR_DRV_BOARDHWVER_ADDR_PTR	-10041	Driver GetBoardVer function not found.
ALERR_DRV_DRIVERVER_ADDR_PTR	-10042	Driver GetDrvVer function not found.
ALERR_DRV_BOARDERROR_ADDR_PTR	-10043	Driver GetBrdError function not found.
ALERR_DRV_BUFFERMESSAGEHANDLER_ADDR_PTR	-10044	Driver SetBuffmessagehandler function not found.
ALERR_DRV_BUFFERCALLBACKFUNC_ADDR_PTR	-10045	Driver SetBuffcallbackfunc function not found
ALERR_DRV_DEMUXDATA_ADDR_PTR	-10046	Driver DemuxData function not found
ALERR_DRV_DEMUXDATASET_ADDR_PTR	-10047	Driver DemuxDataSet function not found
ALERR_DRV_SETFILTERTYPE_ADDR_PTR	-10048	Driver SetfilterType function not found
ALERR_DRV_SETHANDSHAKE_ADDR_PTR	-10049	Driver SetHandShake function not found
ALERR_DRV_SETPORTRESOLUTION_ADDR_PTR	-10050	Driver SetPortResolution function not found
ALERR_DRV_SETERRONTRIGOVERRUN_ADDR_PTR	-10051	Driver SetErrOnTrigOverrun function not found.
ALERR_DRV_SETPORTCONTROL_ADDR_PTR	-10052	Driver SetPortControl function not found.
ALERR_DRV_BUFFERMESSAGEHANDLERPARAMS_ADDR_PTR	-10053	Driver SetBuffMessageHandlerParams function not found.

DRIVERS (con't)

ALERR_DRV_GETACTUALCLKRATE_ADDR_PTR	-10054	Driver GetactualClkRate function not found.
ALERR_DRV_SETPACKEDDATA_ADDR_PTR	-10055	Driver SetPackedData function not found
ALERR_DRV_SETCLKOUTPUT_ADDR_PTR	-10056	Driver SetClkOutput function not found
ALERR_DRV_SETTRIGOUTPUT_ADDR_PTR	-10057	Driver SetTriggerOutput function not found.
ALERR_DRV_SETINPUTCONFIGLIST_ADDR_PTR	-10058	Driver InputConfigList function not found.
ALERR_DRV_SETDATAOFFSETLIST_ADDR_PTR	-10059	Driver DataOffsetList function not found.
ALERR_DRV_SETCTRMODE_ADDR_PTR	-10060	Driver CtrMode function not found.
ALERR_DRV_COUNTEROUT_ADDR_PTR	-10061	Driver CounterOut function not found.
ALERR_DRV_COUNTERIN_ADDR_PTR	-10062	Driver CounterIn function not found.

DEVICE DRIVER HANDLES

ALERR_LHDRVSUBSYS	-10100	The LHDRVSUBSYS specified does not exist.
ALERR_LHDRVSUBSYS_MAX	-10101	The maximum LHDRVSUBSYS have already been allocated.

RUNTIME ERRORS

ALERR_RELEASE_RUNNING	-10200	An attempt to release the LHL D was made while the LHL D was running.
-----------------------	--------	---

FILTERS

ALERR_FILTERS_UNSUPPORTED	-10300	Filtering is not supported on this LHL D.
ALERR_FILTERS_OPTION	-10301	Invalid Filter Type specified.
ALERR_FILTERS_FREQLOW	-10302	Filter frequency to low for this LHL D.
ALERR_FILTERS_FREQHIGH	-10303	Filter frequency to high for this LHL D.

HANDSHAKE

ALERR_HANDSHAKING_UNSUPPORTED	-10400	Handshaking is not supported on this LHL D
ALERR_HANDSHAKE_OPTION	-10401	Invalid Handshake specified

PORT RESOLUTION

ALERR_PORTRESOLUTION_UNSUPPORTED	-10500	PortResolution is not supported on this LHL.
ALERR_PORTRESOLUTION_OPTION	-10501	Invalid PortResolution specified.
ALERR_PORTMASK_MINMASK	-10502	Invalid PortMask minimum.
ALERR_PORTMASK_MAXMASK	-10503	Invalid PortMask maximum.
ALERR_PORTCONTROL_STRUCT_PTR	-10504	Invalid PortControl structure pointer.
ALERR_PORTCONTROL_STRUCT_UNSUPPORTED	-10505	PortControl structure is not supported.

CTR MODE

ALERR_CTRMODE_UNSUPPORTED	-10600	Ctr Mode is not supported on this LHL.
ALERR_CTRMODE_OPTION	-10601	Invalid Ctr Mode specified.
ALERR_COUNTER_RATELOW	-10700	Invalid Counter rate specified.
ALERR_COUNTER_RATEHIGH	-10701	Invalid Counter rate specified.

ADAC-LVi TIMEOUTS

ALERR_TIMEOUT	-11000	The specified operation has passed its specified timeout period.
---------------	--------	--

ENVIRONMENT SPECIFIC ERROR CODES

ALERR_LOADLIBRARY_ERROFFSET	-100000	ADLIB ERROR CODE OFFSET.
ALERR_LOADLIBRARY	-100001	System was out of memory, executable file was corrupt, or relocations were invalid.
ALERR_LOADLIBRARY_FILE	-100002	File was not found.
ALERR_LOADLIBRARY_PATH	-100003	Path was not found.
ALERR_LOADLIBRARY_LINK	-100005	Attempt was made to dynamically link to a task, or there was a sharing or network-protection error.
ALERR_LOADLIBRARY_DATASEG	-100006	Library required separate data segments for each task.
ALERR_LOADLIBRARY_MEM	-100008	There was insufficient memory to start the application.
ALERR_LOADLIBRARY_VER	-100010	Windows version was incorrect.
ALERR_LOADLIBRARY_INV	-100011	Executable file was invalid. Either it was not a Windows application or there was an error in the .EXE image.
ALERR_LOADLIBRARY_OPSYS	-100012	Application was designed for a different operating system.
ALERR_LOADLIBRARY_DOS40	-100013	Application was designed for MS-DOS 4.0.
ALERR_LOADLIBRARY_UNKNOWN	-100014	Type of executable file was unknown.
ALERR_LOADLIBRARY_REALMODE	-100015	Attempt was made to load a real-mode application (developed for an earlier version of Windows).
ALERR_LOADLIBRARY_SECONDINST	-100016	Attempt was made to load a second instance of an executable file containing multiple data segments that were not marked read only.

ENVIRONMENT SPECIFIC ERROR CODES (con't)

ALERR_LOADLIBRARY_COMPRESSED	-100019	Attempt was made to load a compressed executable file. The file must be decompressed before it can be loaded.
ALERR_LOADLIBRARY_DLLMISSING	-100020	Dynamic-link library (DLL) file was invalid. One of the DLLs required to run this application was corrupt.
ALERR_LOADLIBRARY_32BITREQ	-100021	Application requires Microsoft Windows 32-bit extensions.

-10XXXX

Note that 32 bit load library errors are a combination of GetLastError() and AL_LOADLIBRARY_ERROROFFSET. To determine the exact error that occurred subtract AL_LOADLIBRARY_ERROROFFSET from the Error Code returned and see the SDK Error Codes.

ADLIB INTERNAL ERROR CODES

ALERR_INTERNAL_ISARBITRARY_PARAMETER	-200000	Parameter validation failed.
ALERR_INTERNAL_ISNUMSEQLIST_PARAMETER	-200001	Parameter validation failed.

BOARD ERROR CODES

DRVERR_DRVMAIN_PROCSTATE =	-401000	Invalid DriverMain procedure specified.
DRVERR_BOARD_NOT_PRESENT =	-401001	Unable to locate specified board.
DRVERR_INV_HBRDINST =	-401002	Invalid Board instance specified.
DRVERR_DEVICE_UNINITIALIZED =	-401003	The device has not been initialized.
DRVERR_MEMORY_LOW =	-401004	Unable to allocate memory.
DRVERR_MAXBOARDS_ALLOCATED =	-401005	The maximum boards have already been allocated.
DRVERR_BOARD_RESET_TIMEOUT =	-401006	Unable to reset the specified board.
DRVERR_PRODUCTID_UNMATCHED =	-401007	The product ID is incorrect for the specified board.
DRVERR_INV_BUFFER =	-401008	Invalid buffer in call to driver.
DRVERR_DEVICE_INUSE =	-401009	The specified device is busy.
DRVERR_SYSTEM_DRIVER_NOTFOUND =	-401010	System Driver Not Found.
DRVERR_SYSTEM_DRIVER_IOCTL =	-401011	An error occurred during an IOCTL Driver call.
DRVERR_BOARD_RESET_FAILED =	-401012	The driver was unable to properly reset the specified device.
DRVERR_THREAD_BUSY_TIMEOUT =	-401013	The internal driver is not responding to STOPAD command.
DRVERR_SYSTEM_DMATRANS_FAILED =	-401014	The System driver DMA Transfer initiation failed.
DRVERR_THREAD_READY_TIMEOUT =	-401015	An Internal Driver Thread has stopped responding to the system.

BOARD ERROR CODES (con't)

DRVERR_INVALID_DRVSUBSYS =	-401016	The specified driver subsystem does not exist.
DRVERR_MMTIMER_ALLOCATION	-401017	The specified driver subsystem does not exist.
DRVERR_SYSTEM_DRIVER_CREATE_EVENT	401018	The system driver can not insert an event object.
DRVERR_SYSTEM_DRIVER_IRP_INSERT	401019	The system driver can not insert an IRP Buffer.
DRVERR_CYCLEMODE_TRANSMETHOD_CONFLICT =	-401100	A conflict between the CycleMode and transfermethod has been specified, creating an invalid setup.
DRVERR_BUFFNOTIFY_TRANS_METHOD_CONFLICT =	-401101	A conflict between the buffer notification method and Transfermethod has been specified, creating an invalid setup.
DRVERR_DMA_ARBCJ_CHANNELS_NOTSUPPORTED =	-401102	Aribtrary thermocouple channels are not supported with DMA.
DRVERR_CYCLEMODE_CLKSOURCE_CONFLICT =	-401103	A conflict between the CycleMode and ClockSource has been specified, creating an invalid setup.
DRVERR_CLKSOURCE_TRANSMETHOD_CONFLICT =	-401104	A conflict between the ClockSource and TransferMethod has been specified, creating an invalid setup.

BOARD ERROR CODES (con't)

DRVERR_ARBCJ_CHANNELS_NOTSUPPORTED =	-401105	Arbitrary thermocouple channels are not supported.
DRVERR_DMA_ARBCHANS_NOTSUPPORTED =	-401106	Arbitrary channels are not supported.
DRVERR_DMA_ARBGAINS_NOTSUPPORTED =	-401107	Arbitrary gains are not supported.
DRVERR_TRIGMODE_CLKSOURCE_CONFLICT = -	401108	A conflict between the TriggerMode and ClockSource has been specified, creating an invalid setup.
DRVERR_TRIGMODE_ARBCJ_CONFLICT =	-401109	A conflict between the TriggerMode and arbitrary CJ channels has been specified, creating an invalid setup.
DRVERR_TRIGMODE_ARBGAIN_CONFLICT =	-401110	A conflict between the TriggerMode and arbitrary Gains has been specified, creating an invalid setup.
DRVERR_TRIGMODE_NONSEQCHAN_CONFLICT =	-401111	A conflict between the TriggerMode and nonsequential channels has been specified, creating an invalid setup.
DRVERR_BURSTMODE_CLKSOURCE_CONFLICT	-401112	A conflict between the BurstMode and ClockSource has been specified, creating an invalid setup.

BOARD ERROR CODES (con't)

DRVERR_TRIGSOURCE_BURSTMODE_CONFLICT	-401113	A conflict between the TriggerSource and BurstMode has been specified, creating an invalid setup.
DRVERR_TRIGSOURCE_CLKSOURCE_CONFLICT	-401114	A conflict between the TriggerSource and ClkSource has been specified, creating an invalid setup.
DRVERR_TRIGSOURCE_TRIGMODE_CONFLICT	-401115	A conflict between the TriggerSource and TriggerMode has been specified, creating an invalid setup.
DRVERR_GATESOURCE_TRANSMETHOD_CONFLICT	-401116	A conflict between the GateSource and Transfer Method has been specified creating an invalid setup.
DRVERR_TRIGSOURCE_TRANSMETHOD_CONFLICT	-401117	A conflict between the TriggerSource and Transfer Method has been specified, creating an invalid setup.
DRVERR_C40PORT_ARBCJ_CHANNES_NOT SUPPORTED =	-401118	Arbitrary thermocouple channels are not supported over the C40 Port.
DRVERR_DAOUTPUT_TRANSFERMETHOD_CONFLICT=	-401119	A conflict between the DAC Output and Transfer Method has been specified, creating an invalid setup.
DRVERR_CLKSOURCE_COUNTER0_CONFLICT=	-401120	A conflict between the Clock Source and Counter 0 has been specified, creating an invalid setup.

BOARD ERROR CODES (con't)

DRVERR_CLKSOURCE_TIMER1_CONFLICT=	-401121	A conflict between the Clock Source and Timer 1 has been specified, creating an invalid setup.
DRVERR_CLKSOURCE_TIMER0_CONFLICT=	-401122	A conflict between the Clock Source and Timer 0 has been specified, creating an invalid setup.
DRVERR_CLKSOURCE_COUNTER1_CONFLICT=	-401123	A conflict between the Clock Source and Counter 1 has been specified, creating an invalid setup.
DRVERR_DRVERR_CLKRATEUNITS_INVALID=	-401124	An invalid ClockRateUnits setting was specified.
DRVERR_GAINCHANLIST_MINLEN =	-401200	The specified channel gain list is invalid.
DRVERR_GAINCHANLIST_MAXLEN =	-401201	The specified channel gain list is invalid.
DRVERR_GAINCHANLIST_ARRAY_PTR	-401202	Invalid channel gain list.
DRVERR_PANELGAINLIST_MINLEN =	-401300	The specified panel gain list is invalid.
DRVERR_PANELGAINLIST_MAXLEN =	-401301	The specified panel gain list is invalid.
DRVERR_PANELGAINLIST_ARRAY_PTR =	-401302	Invalid panel gain list.
DRVERR_EXPPANEL_MINLEN =	-401400	The specified expansion panel gain list is invalid.
DRVERR_EXPPANEL_MAXLEN =	-401401	The specified expansion panel gain list is invalid.
DRVERR_EXPPANEL_ARRAY_PTR =	-401402	Invalid expansion channel gain list.

BOARD ERROR CODES (con't)

DRVERR_CJLIST_MINLEN =	-401500	The specified CJ list is invalid.
DRVERR_CJLIST_MAXLEN =	-401501	The specified CJ list is invalid.
DRVERR_CJLIST_ARRAY_PTR =	-401502	Invalid CJ list.
DRVERR_ADLOWRATE =	-401600	The specified A/D Clocking rate is low.
DRVERR_ADWITHTC_RATEHIGH =	-401601	The desired A/D clocking rate with high.
DRVERR_DAOUTRANGE =	-401700	Invalid D/A range specified.
DRVERR_DA_NOTAVALIABLE =	-401701	D/A support is not available.
DRVERR_DAUNITS_INVALID =	-401702	The specified D/A unit is not available.
DRVERR_IRQ_INUSE =	-401800	The specified Interrupt level is already in use.
DRVERR_INTERRUPTS_NOTSUPPORTED =	-401801	Interrupts are not supported.
DRVERR_DMA_INUSE =	-401900	The specified DMA level is already in use.
DRVERR_DMA_NOTSUPPORTED =	-401901	DMA is not supported.
DRVERR_DMA_TRANSFER	-401902	DMA Transfer Error.

BOARD ERROR CODES (con't)

DRVERR_ADFIFO_OVERRUN =	-402000	The board's A/D FIFO overrun bit has set.
DRVERR_BURST =	-402001	The board's A/D Burst error bit has set.
DRVERR_CLOCK =	-402002	The board's A/D Clock error bit has set.
DRVERR_ADDATA_OVERRUN =	-402003	The board's A/D DATA overrun bit has set.
DRVERR_TRIGGER_OVERRUN =	-402004	The board's trigger overrun bit has set.
DRVERR_INTERRUPT_OVERRUN =	-402005	The board's interrupt rate is too fast for the driver to respond.
DRVERR_DADATA_UNDERRUN=	-402006	The board's D/A data underrun bit has set.
DRVERR_DAFIFO_UNDERRUN=	-402007	The board's D/A data underrun bit has set.
DRVERR_BUFFER_OVERRUN =	-403000	The Current buffer has overrun.
DRVERR_BUFFER_NEXTBUSY =	-403001	The next buffer in sequence is not available for use by the driver.
DRVERR_GETDEVSTATUS_PARAM =	-404000	Invalid status function argument parameter specified.
DRVERR_STOPDEV_NOTSUPPORTED =	-405000	The requested operation is not supported.
DRVERR_INITDEV_NOTSUPPORTED =	-405001	The requested operation is not supported.
DRVERR_STARTDEV_NOTSUPPORTED =	-405002	The requested operation is not supported.
DRVERR_SETCYCLEMODE_NOTSUPPORTED =	-405003	The requested operation is not supported.

BOARD ERROR CODES (con't)

DRVERR_SETDATATRANSMETHOD_NOTSUPPORTED =	-405004	The requested operation is not supported.
DRVERR_SETGCLIST_NOTSUPPORTED =	-405005	The requested operation is not supported.
DRVERR_SETPANELLIST_NOTSUPPORTED =	-405006	The requested operation is not supported.
DRVERR_SETCJLIST_NOTSUPPORTED =	-405007	The requested operation is not supported.
DRVERR_SETCLOCKSOURCE_NOTSUPPORTED =	-405008	The requested operation is not supported.
DRVERR_SETTRIGGERMODE_NOTSUPPORTED =	-405009	The requested operation is not supported.
DRVERR_SETTRIGGERSOURCE_NOTSUPPORTED =	-405010	The requested operation is not supported.
DRVERR_SETTRIGGERSOURCESIG_NOTSUPPORTED =	-405011	The requested operation is not supported.
DRVERR_SETGATESOURCE_NOTSUPPORTED =	-405012	The requested operation is not supported.
DRVERR_SETSWGATE_NOTSUPPORTED =	-405013	The requested operation is not supported.
DRVERR_GETSWGATE_NOTSUPPORTED =	-405014	The requested operation is not supported.
DRVERR_GATEANDTRIG_NOTSUPPORTED =	-405015	The requested operation is not supported.
DRVERR_SETGATESOURCELEVEL_NOTSUPPORTED =	-405016	The requested operation is not supported.
DRVERR_SETDATACODE_NOTSUPPORTED =	-405017	The requested operation is not supported.

BOARD ERROR CODES (con't)

DRVERR_SETDATAOFFSET_NOTSUPPORTED =	-405018	The requested operation is not supported.
DRVERR_SETBURSTMODE_NOTSUPPORTED =	-405019	The requested operation is not supported.
DRVERR_SETINPUTCONFIG_NOTSUPPORTED =	-405020	The requested operation is not supported.
DRVERR_SETDATARANGE_NOTSUPPORTED =	-405021	The requested operation is not supported.
DRVERR_SETDAOUTPUT_NOTSUPPORTED =	-405022	The requested operation is not supported.
DRVERR_SETBUFFERS_NOTSUPPORTED =	-405023	The requested operation is not supported.
DRVERR_SETBUFFNOTIFYMETHOD_NOTSUPPORTED =	-405024	The requested operation is not supported.
DRVERR_CALLBACKHANDLER_INVALID =	-405025	The Specified call-back handler is invalid.
DRVERR_SETPOSTSAMPLECOUNTS_NOTSUPPORTED =	-405026	The requested operation is not supported.
DRVERR_SETBUFFMSGHANDLER_NOTSUPPORTED =	-405027	The requested operation is not supported.
DRVERR_GETDEVSTATUS_NOTSUPPORTED =	-405028	The requested operation is not supported.
DRVERR_DIGITALINPUT_NOTSUPPORTED =	-405029	The requested operation is not supported.
DRVERR_DIGITALOUTPUT_NOTSUPPORTED =	-405030	The requested operation is not supported.
DRVERR_BOARD_READYBIT_TIMEOUT =	-405031	The boards ready bit flag indicator is not setting.

BOARD ERROR CODES (con't)

DRVERR_SETFILTERTYPE_NOTSUPPORTED =	-405032	Filter types are not supported.
DRVERR_SETERRONTRIGOVERRUN_NOTSUPPORTED =	-405033	ErrOnTrigOverrun is not supported.
DRVERR_SETBUFFMSGHANDLERPARAMS_NOTSUPPORTED =	-405034	SetBuffMessageHandlerParams is not supported.
DRVERR_GETACTUALCLOCKRATE_NOTSUPPORTED =	-405035	GetActualClkRate is not supported or the clocking mode is set to external..
DRVERR_SETDATAOFFSETLIST_NOTSUPPORTED =	-405036	The requested operation is not supported.
DRVERR_SETTRIGGEROUTPUT_NOTSUPPORTED =	-405037	The requested operation is not supported.
DRVERR_SETCLOCKOUTPUT_NOTSUPPORTED =	-405038	The requested operation is not supported.
DRVERR_SETCTRMODE_NOTSUPPORTED =	-405039	The requested operation is not supported.
DRVERR_COUNTERIN_NOTSUPPORTED =	-405040	The requested operation is not supported.
DRVERR_COUNTEROUT_NOTSUPPORTED =	-405041	The requested operation is not supported.
DRVERR_SETINPUTCONFIGLIST_NOTSUPPORTED =	-405042	The requested operation is not supported.
DRVERR_GETBOARDERROR_NOTSUPPORTED =	-405043	The requested operation is not supported.
DRVERR_SETPACKEDDATA_NOTSUPPORTED =	-405044	The requested operation is not supported.

BOARD ERROR CODES (con't)

DRVERR_BOARD_FNEBIT_TIMEOUT =	-405045	The boards FNE bit is not setting after a conversion has been initiated.
DRVERR_BURSTRATE_LOW	-406000	The specified Burst rate is low.
DRVERR_BURSTRATE_HIGH	-406001	The specified Burst rate is high.
DRVERR_SCANRATE_LOW	-406100	The specified Scan rate is low.
DRVERR_SCANRATE_HIGH	-406101	The specified Scan rate is high.

Search/Quick Find

QUICK INFORMATION

Global Partners Web Site

What's New

Automatically receive e-mail updates on new products, software upgrades, and special promotions!

 **IOtech e-News**

IOtech announces that it has acquired the low-cost, high-quality, multifunction, [PCI data acquisition product line of ADAC Corporation](#).

What's new in Data Acquisition? Find out on IOtech's "[What's New](#)" page. Read IOtech's EE magazine cover story on using [Linux for Data Acquisition](#).

Featured Product

ZonicBook™
Medallion™ Series with
eZ-Analyst™ Software



Portable
Real-Time Vibration
& Acoustic Analysis System



FREE IOtech Catalog



Request Your Copy Today!

Quick Answers
to all your data acquisition needs



Ask An Expert

Easy Lease Program



IOtech offers an easy way to get needed equipment **NOW!**

[[HOME](#) | [PRODUCTS](#) | [TECH SUPPORT](#) | [CONTACT](#) | [SEARCH/MAP](#) | [ABOUT IOTECH](#) | [SHOP ONLINE](#)]

® [Copyright 2000, IOtech Inc.](#)

[Privacy Policy](#)