

# **Acquitek**

## **Data Acquisition Products**

### Software Development Kit

User Manual  
Sept 2006

Information in this document is subject to change without notice.  
© Copyright 2004, Acquitek, SAS. All rights reserved.



Acquitek is a trademark of Acquitek, SAS. Other trademarks and trade names may be used in this document to refer to either the entities claiming the marks and names or their products. Acquitek, disclaims any proprietary interest in trademarks and trade names other than its own.

Acquitek makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Acquitek shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this manual.

Acquitek, SAS.  
1 bis rue Marcel Paul  
91300 Massy,  
France

## TABLE OF CONTENTS

<b>Function Reference</b> .....	<b>6</b>
<i>XDA_Ain_Check</i> .....	7
<i>XDA_Ain_ConvertToVolts</i> .....	8
<i>XDA_Ain_Deinterleave</i> .....	9
<i>XDA_Ain_GetParm</i> .....	10
<i>XDA_Ain_MB_Check</i> .....	12
<i>XDA_Ain_MB_Copy</i> .....	13
<i>XDA_Ain_MB_Proc</i> .....	14
<i>XDA_Ain_ReadRaw</i> .....	15
<i>XDA_Ain_ReadVolts</i> .....	16
<i>XDA_Ain_SetBuffer</i> .....	17
<i>XDA_Ain_SetClock</i> .....	18
<i>XDA_Ain_SetParm</i> .....	19
<i>XDA_Ain_SetPll</i> .....	22
<i>XDA_Ain_SetScanRate</i> .....	23
<i>XDA_Ain_SetSequence</i> .....	24
<i>XDA_Ain_SetTrigger</i> .....	25
<i>XDA_Ain_Start</i> .....	27
<i>XDA_Ain_Stop</i> .....	28
<i>XDA_Ain_Wait</i> .....	29
<i>XDA_Aout_Check</i> .....	30
<i>XDA_Aout_ConvertToRaw</i> .....	31
<i>XDA_Aout_GetParm</i> .....	32
<i>XDA_Aout_Preload</i> .....	33
<i>XDA_Aout_Reload_Proc</i> .....	34
<i>XDA_Aout_SetBuffer</i> .....	35
<i>XDA_Aout_SetClock</i> .....	36
<i>XDA_Aout_SetParm</i> .....	37
<i>XDA_Aout_SetSequence</i> .....	39

---

<i>XDA_Aout_SetTrigger</i> .....	40
<i>XDA_Aout_Start</i> .....	41
<i>XDA_Aout_Stop</i> .....	42
<i>XDA_Aout_Update</i> .....	43
<i>XDA_Aout_Wait</i> .....	44
<i>XDA_Aout_WriteRaw</i> .....	45
<i>XDA_Aout_WriteVolts</i> .....	46
<i>XDA_Board_Cleanup</i> .....	47
<i>XDA_Board_GetInfo</i> .....	48
<i>XDA_Board_GetParm</i> .....	52
<i>XDA_Board_Init</i> .....	54
<i>XDA_Board_SetParm</i> .....	<b>Erreur ! Signet non défini.</b>
<i>XDA_CT_Config</i> .....	57
<i>XDA_CT_Gate</i> .....	59
<i>XDA_CT_GetParm</i> .....	60
<i>XDA_CT_Read</i> .....	61
<i>XDA_CT_SetParm</i> .....	62
<i>XDA_Din_Check</i> .....	63
<i>XDA_Din_Config</i> .....	64
<i>XDA_Din_Read</i> .....	65
<i>XDA_Din_SetBuffer</i> .....	66
<i>XDA_Din_SetClock</i> .....	67
<i>XDA_Din_SetSequence</i> .....	68
<i>XDA_Din_SetupTrigger</i> .....	69
<i>XDA_Din_Start</i> .....	70
<i>XDA_Din_Stop</i> .....	71
<i>XDA_Dout_Check</i> .....	72
<i>XDA_Dout_Config</i> .....	73
<i>XDA_Dout_SetBuffer</i> .....	74
<i>XDA_Dout_SetClock</i> .....	75
<i>XDA_Dout_SetSequence</i> .....	76
<i>XDA_Dout_Start</i> .....	77
<i>XDA_Dout_Stop</i> .....	78
<i>XDA_Dout_Write</i> .....	79

---

<b>C Programming Information .....</b>	<b>80</b>
<i>Programming Notes</i> .....	80
<i>Sample Programs</i> .....	80
ain.c: .....	81
ain-buf.c: .....	82
ain-mb.c: .....	83
ain-raw.c: .....	84
ain-trig.c: .....	85
aio.c: .....	86
aio-ibuf.c: .....	87
aio-obuf.c: .....	88
aout.c: .....	89
aout-buf.c: .....	90
aout-mb.c: .....	91
aout-raw: .....	92
aout-trig.c: .....	93
brd-sync.c: .....	94
ct.c: .....	96
din.c: .....	97
din-buf.c: .....	98
dio.c: .....	99
dout.c: .....	100
dout-buf.c: .....	101
echo_mode.c: .....	102
fft.c: .....	103
info.c: .....	104
mul-brd.c: .....	105
mux.c: .....	106
pll.c: .....	107
pxi.c: .....	108
temp.c: .....	109
<b>Microsoft® Visual Basic® Programming Information.....</b>	<b>110</b>
<i>Programming Notes</i> .....	110
<i>Sample Programs</i> .....	110
ain .....	111
ain-buf .....	112
ain-buf2 .....	113
ain-buf3 .....	114

aincheck.....	115
ain-mb.....	116
ain-raw.....	117
aintrig.....	118
aout.....	119
aout-buf.....	120
aout-buf2.....	121
aout-raw.....	122
din.....	123
din-buf.....	124
dio.....	125
dio-line.....	126
dout-buf.....	127
getinfo.....	128
temp.....	129
<b>Microsoft® Visual C++™ Programming Information .....</b>	<b>130</b>
<i>Programming Notes.....</i>	<i>130</i>
<i>Sample Programs.....</i>	<i>130</i>
ain-buf.....	131
aincheck.....	132
dinbuf.....	133
dinline.....	134
dioport.....	135
<b>Acquitek Data Acquisition ActiveX Control .....</b>	<b>136</b>
Overview.....	136
Property Pages.....	137
Notes.....	140
<b>Using Acquitek Data Acquisition SDK Signal Processing Features.....</b>	<b>141</b>
<i>Block FFT.....</i>	<i>141</i>
Introduction.....	141
Usage.....	145
Benchmarks.....	149

## Function Reference

Every XDADAQ function has the following format:

**status = functionName(parameter 1, parameter 2, ... parameter  $n$ )**

where  $n \geq 0$ . Each function returns the status in the form of an error code.

Status	Description
Zero	Executed successfully
Greater than zero	Returned a warning
Less than zero	Critical Error occurred

Please refer to the **Function Return Code Section** for a more detailed description of every possible return value.

## ***XDA\_Ain\_Check***

### **Prototype**

C: i32 XDA\_Ain\_Check(i32 device, i32 \*running, i32 \*count)

VB: XDA\_Ain\_Check (ByVal device As Long, running As Long, count As Long) As Long

### **Description**

Checks the status of a buffered analog input operation.

### **Parameters**

device	Device number of the input device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
running	This variable will be assigned a value of zero if the operation is finished, nonzero if the operation is still running.
count	This variable will be assigned a value equal to the number of samples recovered so far.

---

## ***XDA\_Ain\_ConvertToVolts***

### **Prototype**

C: i32 XDA\_Ain\_ConvertToVolts(i32 device, i32 chan, i32 count, void \*raw, f64 \*volts)

VB: XDA\_Ain\_ConvertToVolts (ByVal device As Long, ByVal chan As Long, ByVal count As Long, raw As Any, volts As Double) As Long

### **Description**

Converts an array of raw input values to an array of voltages. This function uses the specified channel's current parameters in performing the conversion. To convert a single value, pass count = 1 and a pointer to the raw value to convert. In VB, pass a variable containing the raw value to be converted.

### **Parameters**

device	Device number of the input device as specified in Acquitek Control Center. This is used in determining the voltage range to use for the conversion. Valid device numbers range from 1 to 63.
chan	Zero-based analog input channel from which the data was read. This is also used in determining the voltage range to use for the conversion. Valid values vary based on device type.
count	Number of values to convert.
raw	An array containing the raw values to be converted. This should be of the type appropriate to the data returned by the board (for CH/CM, this is i16 or u16). In Visual Basic, pass the first element of the array.
volts	The memory location where the new array of voltages will be placed. In Visual Basic, pass the first element of the array.

## ***XDA\_Ain\_Deinterleave***

### **Prototype**

C: i32 XDA\_Ain\_Deinterleave(i32 device, i32 numChans, i32 count, void \*buffer)

VB: XDA\_Ain\_Deinterleave (ByVal device As Long, ByVal numChans As Long, ByVal count As Long, buffer As Any) As Long

### **Description**

Groups an array of raw input values recovered from a buffered analog input by channel. When data is stored in a buffer during a multiple-channel analog input operation, it is stored in the order that it is captured. For example, if two channels were read, this function puts data from the first channel into the first half of the buffer and data from the second channel into the second half of the buffer.

### **Parameters**

device	Device number of the input device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
numChans	Number of channels from which the data was gathered.
count	Total number of samples in the buffer.
buffer	Pointer to a buffer containing input values. In Visual Basic, pass the first element of the array.

## ***XDA\_Ain\_GetParm***

### **Prototype**

C: i32 XDA\_Ain\_GetParm(i32 device, i32 chan, i32 index, f64 \*value)

VB: XDA\_Ain\_GetParm (ByVal device As Long, ByVal chan As Long, ByVal index As Long, value As Double) As Long

### **Description**

Gets the current value of the specified analog input parameter.

### **Parameters**

device	Device number of the input device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
chan	Analog input channel to which the parameter pertains.
index	Input parameter to check. See below.
value	This variable will be assigned the current value of the specified parameter.

### **Analog Input Parameters**

#### ***XDA\_AIN\_PARM\_POLARITY***

Polarity. This will be XDA\_BIPOLAR or XDA\_UNIPOLAR.

#### ***XDA\_AIN\_PARM\_RANGE***

Maximum value of voltage range. If the channel's polarity is XDA\_BIPOLAR, the minimum voltage will be  $-1 * \text{this value}$ . If the channel's polarity is XDA\_UNIPOLAR, the minimum voltage will be 0.

#### ***XDA\_AIN\_PARM\_COUPLING***

See XDA\_Ain\_SetParm() for a description.

#### ***XDA\_AIN\_PARM\_MODE***

See XDA\_Ain\_SetParm() for a description.

#### ***XDA\_AIN\_PARM\_MULTIBUFFER***

Number of buffers for a multi-buffered input on the specified input channel. The default is 0 (disabled). This is a global setting so chan should be passed as 0.

#### ***XDA\_AIN\_PARM\_TRIGGER\_OFFSET***

Gets offset (in total samples) from the start of an analog input buffer to the beginning of the captured data. Since the offset is in total samples, if this is a multichannel capture, the returned value should be divided by the number of channels captured. If trigger is disabled, this will return 0.

#### ***XDA\_AIN\_PARM\_TRIGGER\_NOISEREJECT***

Enables (value=1) or disables (value=0, default) noise rejection for the analog input trigger.

#### ***XDA\_AIN\_PARM\_FFT\_BLOCK\_MODE***

Enables (value=1) or disables (value=0, default) block FFT mode.

*XDA\_AIN\_PARM\_FFT\_SIZE*

Sets the number of points for FFT. Valid values are powers of 2 from 16 to 4096. Default is 1024.

*XDA\_AIN\_PARM\_FFT\_WINDOW\_TYPE*

Sets the window type for the FFT. Valid values are:

FFT\_WINDOW\_TYPE\_RECTANGLE  
FFT\_WINDOW\_TYPE\_BLACKMAN  
FFT\_WINDOW\_TYPE\_HAMMING  
FFT\_WINDOW\_TYPE\_HANNING  
FFT\_WINDOW\_TYPE\_BARTLETT  
FFT\_WINDOW\_TYPE\_WELCH

Default is FFT\_WINDOW\_TYPE\_RECTANGLE.

*XDA\_AIN\_PARM\_STAR\_CTRLR\_OUTPUT*

Enables PXI Star Trigger lines to be triggered when the input trigger on the PXI Star Trigger Controller board is triggered. Only valid on PXI devices when they are in the PXI Star Trigger Controller slot (Slot 2). Bits are numbered 0 – 12 so valid values are from 0x0000 to 0x1FFF. Default is 0. This is a global setting so chan should be passed as 0.

*XDA\_AIN\_PARM\_BURST\_SIZE*

See XDA\_Ain\_SetParm() for a description.

*XDA\_AIN\_PARM\_TRIGGER\_DELAY*

See XDA\_Ain\_SetParm() for a description.

*XDA\_AIN\_PARM\_PLL\_LOCKED*

Returns 1 if PLL error has been less than PLL lockedThreshold for xx consecutive PLL input pulses, returns 0 otherwise.

*XDA\_AIN\_PARM\_PLL\_UNLOCKED\_COUNT*

Unlocked count increments on every PLL input pulse on which PLL\_LOCKED == 0.

Unlocked count is only cleared upon reading. Therefore, unlocked count can be periodically checked during sampling to determine if the PLL has lost lock, independent of the current locked state.

*XDA\_AIN\_PARM\_PLL\_ERR\_MEAN*

The PLL error is accumulated on every PLL input pulse. Upon reading this parameter, the mean of the PLL error is computed and the accumulator is cleared. The reported error is fractional where one pulse period is equal to one.

*XDA\_AIN\_PARM\_PLL\_ERR\_RMS*

The PLL squared error is accumulated on every PLL input pulse. Upon reading this parameter, the square root of the mean of the PLL squared error is computed and the accumulator is cleared. The reported error is fractional where one pulse period is equal to one.

## ***XDA\_Ain\_MB\_Check***

### **Prototype**

C: i32 XDA\_Ain\_MB\_Check(i32 device, i32 \*status, i32 \*running)

VB: XDA\_Ain\_MB\_Check (ByVal device As Long, status As Long, running As Long) As Long

### **Description**

Checks the status of a multi-buffered analog input operation. Multi-buffering must be enabled by using XDA\_Ain\_SetParm with XDA\_AIN\_PARM\_MULTIBUFFER before the input is started with XDA\_Ain\_Start.

### **Parameters**

device	Device number of the input device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
status	This variable will be assigned a value of zero if the current partial buffer is not ready, nonzero if it is ready.
running	This variable will be assigned a value of zero if the operation is finished, nonzero if the operation is still running.

## ***XDA\_Ain\_MB\_Copy***

### **Prototype**

C: i32 XDA\_Ain\_MB\_Copy(i32 device, i32 \*count, void \*buffer)

VB: XDA\_Ain\_MB\_Copy (ByVal device As Long, count As Long, buffer As Any) As Long

### **Description**

In a multi-buffered analog input operation, copies the current partial buffer into the memory location specified by buffer. Multi-buffering must be enabled by using XDA\_Ain\_SetParm with XDA\_AIN\_PARM\_MULTIBUFFER before the input is started with XDA\_Ain\_Start.

### **Parameters**

device	Device number of the input device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
count	This variable will be assigned a value corresponding to the number of samples that are transferred.
buffer	Memory location at which the partial buffer will be placed. In Visual Basic, pass the first element of the array. This should be of the type appropriate to the data returned by the board (for CH/CM, this is a 16-bit integer).

## ***XDA\_Ain\_MB\_Proc***

### **Prototype**

C: `i32 XDA_Ain_MB_Proc(i32 device, i32 *count, void (*mbProc)(i32 count, void *currentSampPtr))`

VB: not applicable

### **Description**

In a multi-buffered analog input operation, calls a callback function which operates on the current partial buffer. Multi-buffering must be enabled by using `XDA_Ain_SetParm` with `XDA_AIN_PARM_MULTIBUFFER` before the input is started with `XDA_Ain_Start`.

### **Parameters**

device	Device number of the input device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
count	This variable will be assigned a value corresponding to the number of samples that are available.
mbProc	Void function which is called when a buffer is available for processing. The function referenced by the mbProc function pointer must take input parameters matching those specified in the <code>XDA_Ain_MB_Proc</code> function declaration.

## ***XDA\_Ain\_ReadRaw***

### **Prototype**

C: i32 XDA\_Ain\_ReadRaw(i32 device, i32 chan, i32 \*raw)

VB: XDA\_Ain\_ReadRaw (ByVal device As Long, ByVal chan As Long, raw As Long) As Long

### **Description**

Reads a raw integral value from the specified channel. Input range is set with XDA\_Ain\_SetParm and XDA\_AIN\_PARM\_RANGE.

### **Parameters**

device	Device number of the input device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
chan	Zero-based analog input channel from which the data will be read. Valid values vary based on device type.
raw	This variable will be assigned the value read from the device.

## ***XDA\_Ain\_ReadVolts***

### **Prototype**

C: i32 XDA\_Ain\_ReadVolts(i32 device, i32 chan, f64 \*volts)

VB: XDA\_Ain\_ReadVolts (ByVal device As Long, ByVal chan As Long, volts As Double)  
As Long

### **Description**

Reads the voltage from the specified channel. Input range is set with XDA\_Ain\_SetParm and XDA\_AIN\_PARM\_RANGE.

### **Parameters**

device	Device number of the input device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
chan	Zero-based analog input channel from which the data will be read. Valid values vary based on device type.
volts	This variable will be assigned the voltage read from the device.

## ***XDA\_Ain\_SetBuffer***

### **Prototype**

C: i32 XDA\_Ain\_SetBuffer(i32 device, i32 count, void \*buffer)

VB: XDA\_Ain\_SetBuffer (ByVal device As Long, ByVal count As Long, buffer() As Any)  
As Long

### **Description**

Locks a buffer and sets it up for DMA transfer in the next buffered analog input operation. To unlock the buffer, either call this function with count = 0 or lock a different buffer.

### **Parameters**

device	Device number of the input device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
count	Total number of samples that will be collected (also, the number of elements in the buffer). Set this to zero to unlock the buffer.
buffer	A pointer to the buffer where values will be stored. This should be of the type appropriate to the data returned by the board (for CH/CM, this is a 16-bit integer). In Visual Basic, pass the name of the array.

## ***XDA\_Ain\_SetClock***

### **Prototype**

C: i32 XDA\_Ain\_SetClock(i32 device, f64 rate, f64 \*actual)

VB: XDA\_Ain\_SetClock (ByVal device As Long, ByVal rate As Double, actual As Double)  
As Long

### **Description**

Sets the clock rate for a buffered analog input operation. This is used to determine the time between two consecutive samples. If the device can simultaneously sample all of the channels specified by XDA\_Ain\_SetSequence, this determines the time between samples on the same channel. If the device cannot simultaneously sample all of the specified channels, this determines the time between two consecutive samples, which will be on two different channels. In this case, XDA\_Ain\_SetScanRate is used to determine the time between samples on the same channel.

Full speed operation may not be possible with small buffer sizes. Buffer sizes with larger multiples of 4 KB, up to 128 KB, may be required to achieve uninterrupted transfers of multiple channels at high speeds.

### **Parameters**

device	Device number of the input device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
rate	Desired clock rate.
actual	This variable will be assigned the actual clock rate.

## ***XDA\_Ain\_SetParm***

### **Prototype**

C: i32 XDA\_Ain\_SetParm(i32 device, i32 chan, i32 index, f64 value)

VB: XDA\_Ain\_SetParm (ByVal device As Long, ByVal chan As Long, ByVal index As Long, ByVal value As Double) As Long

### **Description**

Sets an analog input parameter.

### **Parameters**

device	Device number of the input device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
chan	Analog input channel to which the parameter pertains.
index	Input parameter to set. See below.
value	Value to assign to the parameter.

### **Analog Input Parameters**

#### ***XDA\_AIN\_PARM\_POLARITY***

Sets polarity. Set this to XDA\_BIPOLAR or XDA\_UNIPOLAR.

#### ***XDA\_AIN\_PARM\_RANGE***

Sets maximum value of voltage range. If the channel's polarity is XDA\_BIPOLAR, the minimum voltage will be  $-1 * \text{this value}$ . If the channel's polarity is XDA\_UNIPOLAR, the minimum voltage will be 0.

#### ***XDA\_AIN\_PARM\_COUPLING***

Sets coupling. Set this to XDA\_AC, XDA\_DC, or XDA\_DC\_TERMINATED.

Some boards do not support XDA\_AC or XDA\_DC\_TERMINATED modes, and an error will be returned if the user attempts to set these.

The value of the termination impedance in XDA\_DC\_TERMINATED mode can be determined by logical AND of the *Ain\_features* member of the BoardInfo structure returned by a call to XDA\_Board\_GetInfo with either XDA\_AIN\_FEATURE\_50OHM or XDA\_AIN\_FEATURE\_75OHM.

See the user manual for your board for an explanation of the input coupling available on your board, and for the specified input impedance in XDA\_DC mode.

#### ***XDA\_AIN\_PARM\_MODE***

Sets mode for the channel. Set this to XDA\_DIFFERENTIAL, XDA\_SINGLE\_ENDED, or XDA\_PSEUDO\_DIFF. See the user manual for your board for a description of these analog input modes.

#### *XDA\_AIN\_PARM\_MULTIBUFFER*

Sets number of buffers for a multi-buffered input on the specified input channel. The default is 0 (disabled). This is currently limited to two buffers. This is a global setting so chan should be passed as 0.

#### *XDA\_AIN\_PARM\_TRIGGER\_NOISEREJECT*

Enables (value=1) or disables (value=0, default) noise rejection for the analog input trigger.

#### *XDA\_AIN\_PARM\_FFT\_BLOCK\_MODE*

Enables (value=1) or disables (value=0, default) block FFT mode.

#### *XDA\_AIN\_PARM\_FFT\_SIZE*

Sets the number of points for FFT. Valid values are powers of 2 from 16 to 4096. Default is 1024.

#### *XDA\_AIN\_PARM\_FFT\_WINDOW\_TYPE*

Sets the window type for the FFT. Valid values are:

FFT\_WINDOW\_TYPE\_RECTANGLE  
FFT\_WINDOW\_TYPE\_BLACKMAN  
FFT\_WINDOW\_TYPE\_HAMMING  
FFT\_WINDOW\_TYPE\_HANNING  
FFT\_WINDOW\_TYPE\_BARTLETT  
FFT\_WINDOW\_TYPE\_WELCH

Default is FFT\_WINDOW\_TYPE\_RECTANGLE.

#### *XDA\_AIN\_PARM\_STAR\_CTRLR\_OUTPUT*

Enables PXI Star Trigger lines to be triggered when the input trigger on the PXI Star Trigger Controller board is triggered. Only valid on PXI devices when they are in the PXI Star Trigger Controller slot (Slot 2). Bits are numbered 0 – 12. Valid values are 0 to 0x1FFF. Default is 0. The function will fail with XDA\_ERR\_PXI\_NOT\_ENABLED if the specified lines aren't enabled for output in Acquitek Control Center. This is a global setting so chan should be passed as 0.

#### *XDA\_AIN\_PARM\_SW\_TRIGGER*

Initiates a software trigger for XDA\_AIN\_TRIGGER\_SOFTWARE. This is only supported on PXI boards. This is a global setting so chan should be passed as 0.

#### *XDA\_AIN\_PARM\_BURST\_SIZE*

Applicable to a triggered input capture. When set to 0 (default), XDA\_Ain\_Start() captures the number of samples specified in its count parameter. When set to a non-zero number, XDA\_Ain\_Start() captures for this number of sample clocks following a trigger, then waits for the next trigger. This repeats until the number of samples specified in the XDA\_Ain\_Start() count parameter are captured. The burst size should be chosen as a function of sample rate so that the minimum time between triggers is at least 10 microseconds. Note that XDA\_AIN\_BURST\_SIZE is specified in sample clocks, so for multichannel capture, the total number of samples capture per burst will be numberOfChannels\*XDA\_AIN\_PARM\_BURST\_SIZE. Must be a multiple of 64 and less than 1,000,000. Cannot be using simultaneously with pre-triggering. This is a global setting so chan should be passed as 0. Not available on CM/XM

#### *XDA\_AIN\_PARM\_TRIGGER\_DELAY*

Applicable to a triggered input capture. Samples are not captured until the number of sample clocks equal to the value specified by XDA\_AIN\_PARM\_TRIGGER\_DELAY have elapsed. Default value is 0. Negative values indicate pretrigger. Must be between -1,000,000 and 1,000,000. Positive values of trigger delay are not available on the CM/XM. Negative values of trigger delay are available on the XM. This is a global setting so chan should be passed as 0. This parameter must be set prior to calling XDA\_Ain\_SetTrigger() for proper functionality. If set to a large value, XDA\_Board\_SetParm may need to be called with XDA\_BOARD\_PARM\_AIN\_MEMORY to allocate more memory to input.

#### *XDA\_AIN\_PARM\_EXT\_CLK\_DIVISOR*

Sets up the register for an external input clock. The external clock needs to be at least 1MHz. The clock needs to be  $2 * \text{desired sample rate} * \text{chanCount}$ . This parameter needs to be set at  $2 * \text{chanCount}$ . This is currently only implemented on the CH.

## ***XDA\_Ain\_SetPll***

### **Prototype**

C: i32 XDA\_Ain\_SetPll (i32 device, i32 mode, i32 sampPerPulse, f64 bw, f64 zeta, f64 lockThreshold)

VB: XDA\_Ain\_SetPll (ByVal device As Long, ByVal mode As Long, ByVal sampPerPulse As Long, ByVal bw As Double, ByVal zeta As Double, ByVal lockThreshold As Double) As Long

### **Description**

Starts or stops analog sample clock synchronization to pulses input on Gate0.

### **Parameters**

device	Device number of the input device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
mode	Synchronous, asynchronous or disable. This value can be either XDA_SYNC or XDA_ASYNC or XDA_PLL_DISABLE. If this is set to XDA_ASYNC, the PLL locking will be performed in the background and other operations can be performed. If it is set to XDA_SYNC, the function will return when the PLL has locked or when a timeout, specified by XDA_Board_SetParm() and XDA_BOARD_PARM_TIMEOUT, is reached. If it is set to XDA_PLL_DISABLE the PLL is disabled and the remaining parameters ignored.
sampPerPulse	PLL locks to pulses input via Gate0. The PLL will generate a sampling clock which captures sampPerPulse samples on each channel on each input pulse.
bw	PLL bandwidth, in cycles/sec (i.e. Hertz)
zeta	PLL damping factor
lockThreshold	When PLL error is less than this fractional part of the PLL input pulse period, the PLL is considered locked.

#### Notes:

Counter/Timer 0 cannot be used simultaneously with PLL mode.

Call XDA\_Ain\_SetClock() with the maximum value of the sampling clock (i.e. samplesPerPulsePerChannel \* maxPulsesPerSecond \* numberOfChannels.) prior to calling XDA\_Ain\_SetPLL().

If disabling PLL, call XDA\_Ain\_SetClock() again to reset the sampling clock to a known value.

---

## ***XDA\_Ain\_SetScanRate***

### **Prototype**

C: i32 XDA\_Ain\_SetScanRate(i32 device, f64 rate, f64 \*actual)

VB: XDA\_Ain\_SetScanRate (ByVal device As Long, ByVal rate As Double, actual As Double) As Long

### **Description**

Sets the scan rate for a buffered analog input operation. If the device cannot sample all of the specified channels simultaneously, this will be the rate used to determine the time between samples of the same channel. If this is set to zero, the first channel in the sequence will be sampled one clock (determined by XDA\_Ain\_SetClock) after the last channel in the sequence. If the device does not support scanning, this must be set to zero, otherwise, this function will return XDA\_ERR\_RATE.

Note: this function is not yet implemented and must be set to zero.

### **Parameters**

device	Device number of the input device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
rate	Desired rate at which to scan the input. This must be a factor of the clock rate set with XDA_Ain_SetClock and at most the clock rate divided by the number of channels in the sequence. If this is zero, the board will capture at the rate set by XDA_Ain_SetClock.
actual	This variable will be assigned the actual rate at which the device will scan. If it is zero, the variable will not be assigned a value.

## ***XDA\_Ain\_SetSequence***

### **Prototype**

C: i32 XDA\_Ain\_SetSequence(i32 device, i32 count, i32 \*channels, f64 \*ranges)

VB: XDA\_Ain\_SetSequence (ByVal device As Long, ByVal count As Long, channels As Long, ranges As Double) As Long

### **Description**

Sets the channel sequence for a buffered analog input operation.

### **Parameters**

device	Device number of the input device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
count	The number of channels to scan.
channels	An array of zero-based analog input channels to use in the buffered analog input. In Visual Basic, pass the first element of the array.
ranges	An array of ranges that will correspond to the channels. In Visual Basic, pass the first element of the array.

## ***XDA\_Ain\_SetTrigger***

### **Prototype**

C: i32 XDA\_Ain\_SetTrigger(i32 device, i32 chan, f64 trig\_volts, i32 flags)

VB: XDA\_Ain\_SetTrigger (ByVal device As Long, ByVal chan As Long, ByVal trig\_volts As Double, ByVal flags As Long) As Long

### **Description**

Sets a trigger for analog input. When XDA\_Ain\_Start is called after this, capture will begin when the trigger is generated. Offset from the beginning of the buffer to the captured data can be found using XDA\_Ain\_GetParm with XDA\_AIN\_PARM\_TRIGGER\_OFFSET.

### **Parameters**

device	Device number of the input device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
chan	Zero-based analog input channel to use for trigger. Valid values vary based on device type. This is ignored for external, digital, and PXI star triggers and should be passed as 0.
trig_volts	Voltage to trigger analog input. When the input voltage on the specified channel reaches this voltage, the trigger will be generated. This is ignored for digital and PXI star triggers and should be passed as 0.
flags	Specifies options for the trigger (see below).

#### ***XDA\_TRIGGER\_FALLING***

trigger is generated when analog signal goes from above specified voltage to below specified voltage.

#### ***XDA\_TRIGGER\_RISING***

trigger is generated when analog signal goes from below specified voltage to above specified voltage.

#### ***XDA\_TRIGGER\_EXTERNAL***

use external trigger. Must be TTL level rising edge signal on CM/XM.  
use A/D SeqSt pin on CM/XM.  
Pretriggering not available with this trigger on CM/XM

#### ***XDA\_TRIGGER\_SOFTWARE***

use software initiated trigger. Only supported on PXI boards.

#### ***XDA\_TRIGGER\_PXI\_STAR***

use PXI Star Trigger. Only supported on PXI boards.

*XDA\_TRIGGER\_DIGITAL*

use digital trigger. See XDA\_Din\_SetupTrigger for details.

Not available on CM/XM

*XDA\_TRIGGER\_NONE*

disable trigger.

Note:

XDA\_TRIGGER\_RISING and XDA\_TRIGGER\_FALLING can be or'ed with the other flags. By default it is or'ed with XDA\_TRIGGER\_FALLING.

## ***XDA\_Ain\_Start***

### **Prototype**

C: i32 XDA\_Ain\_Start(i32 device, i32 modeFlags, i32 count)

VB: XDA\_Ain\_Start (ByVal device As Long, ByVal modeFlags As Long, ByVal count As Long) As Long

### **Description**

Starts a buffered analog input operation using parameters specified by XDA\_Ain\_SetBuffer, XDA\_Ain\_SetParm, XDA\_Ain\_SetClock, XDA\_Ain\_SetScanRate, and XDA\_Ain\_SetSequence. This may not be used simultaneously with XDA\_Din\_Start.

### **Parameters**

device	Device number of the input device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
modeFlags	Synchronous or asynchronous. This value can be either XDA_SYNC or XDA_ASYNC. If this is set to XDA_ASYNC, the buffered input will be performed in the background and other operations can be performed. If it is set to XDA_SYNC, the function will return when the scan is complete or when a timeout, specified by XDA_Board_SetParm and XDA_BOARD_PARM_TIMEOUT, is reached.
count	Total number of samples to collect. (2 bytes per sample for 12- or 16-bit boards.) This must be a multiple of 4096 bytes (for CH/CM, this would be a multiple of 2048 samples).

## ***XDA\_Ain\_Stop***

### **Prototype**

C: i32 XDA\_Ain\_Stop(i32 device)

VB: XDA\_Ain\_Stop (ByVal device As Long) As Long

### **Description**

Stops the current buffered analog input operation.

### **Parameters**

device            Device number of the input device as specified in Acquitek Control Center.  
Valid device numbers range from 1 to 63.

## ***XDA\_Ain\_Wait***

### **Prototype**

C: i32 XDA\_Ain\_Wait(i32 device, i32 (\*cbfunc)(i32 count, void \* buffer))

VB: NA

### **Description**

Block until either the "buffer full" event or a timeout event occurs. The timeout is set by XDA\_BOARD\_PARM\_TIMEOUT. If a callback is specified, it will be invoked upon receipt of the half-buffer full event, and the return value will be interpreted as a 'iterate' or 'exit' flag. A positive non-zero value will cause this routine to continue collecting data (and invoking the callback on each half-buffer event); a negative or zero value will cause this routine to terminate data collection and return control to the caller. If the callback is NULL, control will be returned to the caller when either event occurs.

### **Parameters**

device	Device number of the input device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
cbfunc	A function which is called when a buffer is available for processing. The function referenced by the cbfunc function pointer must take input parameters matching those specified in the XDA_Ain_Wait function declaration. It should return a positive value to continue capturing; or negative or zero to get returned to the main program.

## ***XDA\_Aout\_Check***

### **Prototype**

C: i32 XDA\_Aout\_Check(i32 device, i32 \*running, i32 \*rsvd)

VB: XDA\_Aout\_Check (ByVal device As Long, running As Long, rsvd As Long) As Long

### **Description**

Checks the status of an analog output operation.

### **Parameters**

device	Device number as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
running	This variable will be assigned a value of zero if the operation is finished, nonzero if the operation is still running.
rsvd	Reserved variable. Set to zero.

## ***XDA\_Aout\_ConvertToRaw***

### **Prototype**

C: i32 XDA\_Aout\_ConvertToRaw(i32 device, i32 chan, i32 count, f64 \*volts, void \*raw)

VB: XDA\_Aout\_ConvertToRaw (ByVal device As Long, ByVal chan As Long, ByVal count As Long, volts As Double, raw As Any) As Long

### **Description**

Converts an array of voltages into an array of raw output values. This function uses the specified channel's current parameters in performing the conversion. To convert a single value, pass count = 1 and a pointer to the voltage to convert. In VB, pass a variable containing the voltage.

### **Parameters**

device	Device number of the output device as specified in Acquitek Control Center. This is used in determining the voltage range to use for the conversion. Valid device numbers range from 1 to 63.
chan	Zero-based analog output channel on which the data will be written. This is also used in determining the voltage range to use for the conversion. Valid values vary based on device type.
count	Number of values to convert.
volts	An array containing the voltages to be converted. In Visual Basic, pass the first element of the array.
raw	The memory location where the new array of raw output values will be placed. This should be of the type appropriate to the data to be output by the board (for CH/CM, this is a 16-bit integer). In Visual Basic, pass the first element of the array.

## ***XDA\_Aout\_GetParm***

### **Prototype**

C: i32 XDA\_Aout\_GetParm(i32 device, i32 chan, i32 index, f64 \*value)

VB: XDA\_Aout\_GetParm (ByVal device As Long, ByVal chan As Long, ByVal index As Long, value As Double) As Long

### **Description**

Gets the current value of the specified analog output parameter.

### **Parameters**

device	Device number of the output device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
chan	Analog output channel to which the parameter pertains.
index	Output parameter to check. See “Parameters.”
value	This variable will be assigned the current value of the specified parameter.

### **Analog Output Parameters**

#### ***XDA\_AOUT\_PARM\_POLARITY***

Polarity. This will be XDA\_BIPOLAR or XDA\_UNIPOLAR.

#### ***XDA\_AOUT\_PARM\_RANGE***

Maximum value of voltage range. If the channel’s polarity is XDA\_BIPOLAR, the minimum voltage will be -1 \* this value. If the channel’s polarity is XDA\_UNIPOLAR, the minimum voltage will be 0.

#### ***XDA\_AOUT\_PARM\_COUPLING***

Coupling. This will be XDA\_AC, XDA\_DC, or XDA\_DC\_TERMINATED.

#### ***XDA\_AOUT\_PARM\_UPDATEMODE***

Update mode for the output channel. If this is set to 0 (immediate, which is the default), the voltage being output by the channel will be updated immediately after a new value is written to the channel. If it is set to 1 (delayed), The voltage being output will not be updated until XDA\_Aout\_Update is called.

## ***XDA\_Aout\_Preload***

### **Prototype**

C: i32 XDA\_Aout\_Preload( i32 device, i32 reserved )

VB: NA

### **Description**

Preloads the waveform into the onboard memory. This is normally done with XDA\_Aout\_Start is executed. XDA\_Aout\_SetBuffer and XDA\_Aout\_ReloadProc must be called before this function

### **Parameters**

device	Device number of the output device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
reserved	This is currently reserved parameter and will be ignored.

## ***XDA\_Aout\_Reload\_Proc***

### **Prototype**

C: i32 XDA\_Aout\_Reload\_Proc(i32 device, u32 (\*func)( i32 size, void \* buf));

VB: NA

### **Description**

Sets the function that will be called when the (half) buffer is empty and ready for refilling. The function passed as func needs to have the same parameters as the declaration of u32 (\*func)( i32 size, void \* buf). If the function does not return the size as passed to it, this indicates the end of the buffer and output will be stopped after the completion of the current buffer.

### **Parameters**

device	Device number of the output device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
func	Function that will reload the buffer when it is empty. Should return size passed to it. Otherwise indicates end of the multibuffer.

## ***XDA\_Aout\_SetBuffer***

### **Prototype**

C: i32 XDA\_Aout\_SetBuffer(i32 device, i32 count, void \*buffer)

VB: XDA\_Aout\_SetBuffer (ByVal device As Long, ByVal count As Long, buffer() As Any)  
As Long

### **Description**

Locks a buffer and sets it up for DMA transfer in the next buffered analog output operation. To unlock the buffer, either call this function with count = 0 or lock a different buffer.

### **Parameters**

device	Device number of the output device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
count	Total number of values that will be output (also, the number of elements that are in the buffer). Set this to zero to unlock the buffer.
buffer	Buffer to use for output. The range for the values in this buffer will vary based on the resolution of the device and its current polarity setting. This should be of the type appropriate to the data returned by the board (for CH/CM, this is a 16-bit integer). In Visual Basic, pass the name of the array.

## **XDA\_Aout\_SetClock**

### **Prototype**

C: i32 XDA\_Aout\_SetClock(i32 device, f64 rate, f64 \*actual)

VB: XDA\_Aout\_SetClock (ByVal device As Long, ByVal rate As Double, actual As Double) As Long

### **Description**

Sets the clock rate for buffered analog output. This is used to determine the time between two consecutive writes. If the device can simultaneously write to all of the channels specified by XDA\_Aout\_SetSequence, this determines the time between writes on the same channel. If the device cannot simultaneously write to all of the specified channels, this determines the time between two consecutive writes, which will be on two different channels.

Full speed operation may not be possible with small buffer sizes. Buffer sizes with larger multiples of 4 KB, up to 128 KB, may be required to achieve uninterrupted transfers of multiple channels at high speeds.

### **Parameters**

device	Device number of the output device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
rate	Desired clock rate.
actual	This variable will be assigned the actual clock rate.

## ***XDA\_Aout\_SetParm***

### **Prototype**

C: i32 XDA\_Aout\_SetParm(i32 device, i32 chan, i32 index, f64 value)

VB: XDA\_Aout\_SetParm (ByVal device As Long, ByVal chan As Long, ByVal index As Long, ByVal value As Double) As Long

### **Description**

Sets an analog output parameter.

### **Parameters**

device	Device number of the output device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
chan	Analog output channel to which the parameter pertains.
index	Output parameter to set. See below.
value	Value to assign to the parameter.

### **Analog Output Parameters**

#### ***XDA\_AOUT\_PARM\_POLARITY***

Sets polarity. Set this to XDA\_BIPOLAR or XDA\_UNIPOLAR.

#### ***XDA\_AOUT\_PARM\_RANGE***

Sets maximum value of voltage range. If the channel's polarity is XDA\_BIPOLAR, the minimum voltage will be -1 \* this value. If the channel's polarity is XDA\_UNIPOLAR, the minimum voltage will be 0.

#### ***XDA\_AOUT\_PARM\_COUPLING***

Sets coupling. Set this to XDA\_AC, XDA\_DC, or XDA\_DC\_TERMINATED.

#### ***XDA\_AOUT\_PARM\_UPDATEMODE***

Sets update mode for the output channel. If this is set to 0 (immediate, which is the default), the voltage being output by the channel will be updated immediately after a new value is written to the channel. If it is set to 1 (delayed), The voltage being output will not be updated until XDA\_Aout\_Update is called.

#### ***XDA\_AOUT\_PARM\_BURST\_SIZE***

Sets the number of samples output on each trigger. Set to 0 (default) to output entire buffer on each trigger. This is a global setting so set chan to 0. This must be multiple of 32 and greater than 256. If trigger occur before the full BURST\_SIZE it outputted, it is ignored.

#### ***XDA\_AOUT\_PARM\_TRIGGER\_COUNT***

Sets the number of times to wait for output trigger. Set to 0 (default) for infinite. This is a global setting so set chan to 0.

*XDA\_AOUT\_PARM\_MULTIBUFFER*

Sets number of buffers for a multi-buffered output on a specific output channel. This by default is 0 (disable). It is currently limited to two buffers. This is a global setting so chan should be passed as 0.

*XDA\_AOUT\_PARM\_ECHO\_DELAY*

Sets the delay for the echo mode in samples. Set to 0 (disabled) by default. Minimum value is 320 samples. Maximum is limited to size of the onboard memory. This is a global setting so chan should be passed as 0.

## ***XDA\_Aout\_SetSequence***

### **Prototype**

C: i32 XDA\_Aout\_SetSequence(i32 device, i32 count, i32 \*channels)

VB: XDA\_Aout\_SetSequence (ByVal device As Long, ByVal count As Long, channels As Long) As Long

### **Description**

Sets the sequence of channels to use for buffered analog output.

### **Parameters**

device	Device number of the output device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
count	Number of channels that will be used in the buffered output operation.
channels	An array of analog output channels to use in the buffered output operation. In Visual Basic, pass the first element of the array.

## ***XDA\_Aout\_SetTrigger***

### **Prototype**

C: i32 XDA\_Aout\_SetTrigger(i32 device, i32 chan, f64 trig\_volts, i32 flags)

VB: XDA\_Aout\_SetTrigger (ByVal device As Long, ByVal chan As Long, ByVal trig\_volts As Double, ByVal flags As Long) As Long

### **Description**

Sets a trigger for analog output. When XDA\_Aout\_Start is called after this, output will begin when the trigger is generated. This is only supported on CH and XH series boards.

### **Parameters**

device	Device number as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
chan	Zero-based analog input channel to use for trigger. Valid values vary based on device type. This is ignored for digital and PXI star triggers and should be passed as 0.
trig_volts	Voltage to trigger analog input. When the input voltage on the specified channel reaches this voltage, the trigger will be generated. This is ignored for digital and PXI star triggers and should be passed as 0.
flags	Specifies options for the trigger (see below).

#### *XDA\_TRIGGER\_FALLING*

trigger is generated when analog signal goes from above specified voltage to below specified voltage.

#### *XDA\_TRIGGER\_RISING*

trigger is generated when analog signal goes from below specified voltage to above specified voltage.

#### *XDA\_TRIGGER\_EXTERNAL*

use external trigger.

#### *XDA\_TRIGGER\_PXI\_STAR*

use PXI Star Trigger. Only supported on PXI boards.

#### *XDA\_TRIGGER\_DIGITAL*

use digital trigger. See XDA\_Din\_SetupTrigger for details.

#### *XDA\_TRIGGER\_NONE*

disable trigger.

## ***XDA\_Aout\_Start***

### **Prototype**

C: i32 XDA\_Aout\_Start(i32 device, i32 count, i32 iterations)

VB: XDA\_Aout\_Start (ByVal device As Long, ByVal count As Long, ByVal iterations As Long) As Long

### **Description**

Starts a buffered analog output operation using parameters specified by XDA\_Aout\_SetBuffer, XDA\_Aout\_SetParm, XDA\_Aout\_SetClock, and XDA\_Aout\_SetSequence. This may not be used simultaneously with XDA\_Dout\_Start. The waveform will be output the number of times specified by iterations. If iterations is 0, the waveform will loop until stopped.

### **Parameters**

device	Device number of the output device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
count	Total number of samples in the buffer. (2 bytes per sample for 12- or 16-bit boards.) This must be a multiple of 64 bytes (for CH/CM, this would be a multiple of 32 samples).
iterations	Number of times to output the buffer or 0 for continuous output.

## ***XDA\_Aout\_Stop***

### **Prototype**

C: i32 XDA\_Aout\_Stop(i32 device)

VB: XDA\_Aout\_Stop (ByVal device As Long) As Long

### **Description**

Stops the current buffered analog output operation.

### **Parameters**

device      Device number of the output device as specified in Acquitek Control Center.  
Valid device numbers range from 1 to 63.

## ***XDA\_Aout\_Update***

### **Prototype**

C: i32 XDA\_Aout\_Update(i32 device)

VB: XDA\_Aout\_Update (ByVal device As Long) As Long

### **Description**

Updates the voltages being output by the device. If the device is set for immediate update, this is not necessary.

### **Parameters**

device            Device number of the output device as specified in Acquitek Control Center.  
Valid device numbers range from 1 to 63.

## ***XDA\_Aout\_Wait***

### **Prototype**

C: i32 XDA\_Aout\_Wait(i32 device, i32 timeout)

VB: NA

### **Description**

Block the program until the output stream is finished or there is a timeout.

### **Parameters**

device	Device number of the output device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
timeout	Sets timeout for analog output stream. Use -1 to wait indefinitely.

## ***XDA\_Aout\_WriteRaw***

### **Prototype**

C: i32 XDA\_Aout\_WriteRaw(i32 device, i32 chan, i32 raw)

VB: XDA\_Aout\_WriteRaw (ByVal device As Long, ByVal chan As Long, ByVal raw As Long) As Long

### **Description**

Writes a raw value to an analog output. Output range is set with XDA\_Aout\_SetParm and XDA\_AOUT\_PARM\_RANGE. If UPDATEMODE is set to zero (immediate; the default), this will immediately change the voltage being output. Otherwise, the device will wait until XDA\_Aout\_Update is called.

### **Parameters**

device	Device number of the output device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
chan	Zero-based analog output channel on which the data will be written. Valid values vary based on device type.
raw	Value to write to the specified channel.

## ***XDA\_Aout\_WriteVolts***

### **Prototype**

C: i32 XDA\_Aout\_WriteVolts(i32 device, i32 chan, f64 volts)

VB: XDA\_Aout\_WriteVolts (ByVal device As Long, ByVal chan As Long, ByVal volts As Double) As Long

### **Description**

Writes a voltage to an analog output. Output range is set with XDA\_Aout\_SetParm and XDA\_AOUT\_PARM\_RANGE. If UPDATEMODE is set to zero (immediate; the default), this will immediately change the voltage being output. Otherwise, the device will wait until XDA\_Aout\_Update is called.

### **Parameters**

device	Device number of the output device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
chan	Zero-based analog output channel on which the data will be written. Valid values vary based on device type.
volts	Voltage to be write to the specified channel.

## ***XDA\_Board\_Cleanup***

### **Prototype**

C: i32 XDA\_Board\_Cleanup(i32 device, i32 rsvd)

VB: XDA\_Board\_Cleanup (ByVal device As Long, ByVal rsvd As Long) As Long

### **Description**

Closes access to device resources. This function should always be called before a program exits.

### **Parameters**

device	Device number of the device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
rsvd	Reserved variable. Set this to zero.

## **XDA\_Board\_GetInfo**

### **Prototype**

C: i32 XDA\_Board\_GetInfo(i32 device, BoardInfo \*bi, i32 size)

VB: XDA\_Board\_GetInfo (ByVal device As Long, bi As BoardInfo, ByVal size As Long)  
As Long

### **Description**

Gets information about the specified device such as description, serial number, and specifications. (See below.)

### **Parameters**

device Device number of the device as specified in Acquitex Control Center. Valid device numbers range from 1 to 63.

bi This will hold the board information.

size Size of the BoardInfo structure. In C or C++, use sizeof(BoardInfo). In Visual Basic, use Len(BI) where BI is a variable of type BoardInfo.

### **BoardInfo Structure**

Board_name	Name of the specified device.
Board_serialNumber	Serial number of the specified device.
Board_ID	ID of the specified device.
Board_features	Contains bit flags (see “Board Feature Flags” below) describing features of the device.
Board_dio_count	Number of reversible digital I/O ports on the specified device.
Board_family	Board family. FAMILY_CH includes CH and XH series boards. FAMILY_CM includes CM and XM series boards.
Board_PXIgeographicAddress	PXI geographic address (slot number).
Board_rsvd	Reserved.
Ain_count	Number of analog inputs on the specified device.
Ain_bits	Resolution of the device’s A/D converter.
Ain_memory	Onboard memory (in bytes).Ain_features Contains bit flags (see “Analog Input Feature Flags” below) describing analog input features of the device.
Ain_min	Minimum analog input sample rate of the A/D converter(s).
Ain_max	Maximum analog input sample rate of the A/D converter(s).
Ain_ranges	Analog input voltage ranges of the specified device.
Ain_rsvd	Reserved.
Aout_count	Number of analog outputs on the specified device.
Aout_bits	Resolution of the device’s D/A converter.
Aout_memory	Onboard memory (in bytes).

Aout_features	Contains bit flags (see “Analog Output Feature Flags” below) describing analog output features of the device.
Aout_min	Minimum analog output sample rate of the D/A converter(s).
Aout_max	Maximum analog output sample rate of the D/A converter(s).
Aout_ranges	Analog output voltage ranges of the specified device.
Aout_rsvd	Reserved.
Din_count	Number of digital input ports (including reversible I/O ports).
Din_bits	Number of bits per digital input port.
Din_memory	Onboard memory (in bytes).
Din_features	Contains bit flags (see “Digital Input Feature Flags” below) describing digital input features of the device.
Din_min	Minimum digital input sample rate of the A/D converter(s).
Din_max	Maximum digital input sample rate of the A/D converter(s).
Din_ranges	Logic levels for digital input.
Din_rsvd	Reserved.
Dout_count	Number of digital output ports (including reversible I/O ports).
Dout_bits	Number of bits per digital output port.
Dout_memory	Onboard memory (in bytes).
Dout_features	Contains bit flags (see “Digital Output Feature Flags” below) describing digital output features of the device.
Dout_min	Minimum digital output sample rate of the D/A converter(s).
Dout_max	Maximum digital output sample rate of the D/A converter(s).
Dout_ranges	Logic levels for digital output.
Dout_rsvd	Reserved.
CT_count	Number of counter/timers.
CT_bits	Numbers of bits for each counter/timer.
rsvd	Reserved.

### Board Feature Flags

These are stored in Board\_features, in the BoardInfo structure.

XDA_BOARD_FEATURE_SERIALNUM	the device’s serial number can be read using the API
XDA_BOARD_FEATURE_TEMPSENSE	the device has an onboard temperature sensor
XDA_BOARD_FEATURE_PXI	the device is a PXI board

### Analog Input Feature Flags

These are stored in Ain\_features, in the BoardInfo structure.

---

XDA_AIN_FEATURE_BIPOLAR	the device supports bipolar input
XDA_AIN_FEATURE_UNIPOLAR	the device supports unipolar input
XDA_AIN_FEATURE_AC	the device supports AC (Hi-Z) coupling for input
XDA_AIN_FEATURE_DC	the device supports DC (Hi-Z) coupling for input
XDA_AIN_FEATURE_DC_TERMINATED	the device supports DC terminated (50- or 75-ohm) coupling for input
XDA_AIN_FEATURE_50OHM	the device supports 50-ohm terminated coupling for input
XDA_AIN_FEATURE_75OHM	the device supports 75-ohm terminated coupling for input
XDA_AIN_FEATURE_1HZCLOCK	the device's analog input clock has a resolution of 1 Hz
XDA_AIN_FEATURE_SIMULTANEOUS	the device supports multiple-channel simultaneous analog input
XDA_AIN_FEATURE_SCANNED	the device supports multiple-channel scanned analog input
XDA_AIN_FEATURE_SINGLE_ENDED	the device supports single-ended input
XDA_AIN_FEATURE_DIFFERENTIAL	the device supports differential input
XDA_AIN_FEATURE_PSEUDO_DIFF	the device supports pseudo-differential input

### Analog Output Feature Flags

These are stored in Aout\_features, in the BoardInfo structure.

XDA_AOUT_FEATURE_BIPOLAR	the device supports bipolar output
XDA_AOUT_FEATURE_UNIPOLAR	the device supports unipolar output
XDA_AOUT_FEATURE_AC	the device supports AC (Lo-Z) coupling for output
XDA_AOUT_FEATURE_DC	the device supports DC (Lo-Z) coupling for output
XDA_AOUT_FEATURE_DC_TERMINATED	the device supports DC terminated (50- or 75-ohm) coupling for output
XDA_AOUT_FEATURE_50OHM	the device supports 50-ohm terminated coupling for output
XDA_AOUT_FEATURE_75OHM	the device supports 75-ohm terminated coupling for output
XDA_AOUT_FEATURE_1HZCLOCK	the device's analog output clock has a resolution of 1 Hz
XDA_AOUT_FEATURE_RANGE	the device supports multiple voltage ranges for output
XDA_AOUT_FEATURE_FILTER	the device has an onboard reconstruction filter for analog output

### **Digital Input Feature Flags**

These are stored in `Din_features`, in the `BoardInfo` structure.

<code>XDA_DIN_FEATURE_STREAM</code>	the device supports streaming digital input
<code>XDA_DIN_FEATURE_PORTCFG</code>	the device's inputs are configurable by port
<code>XDA_DIN_FEATURE_LINECFG</code>	the device's inputs are configurable by line

### **Digital Output Feature Flags**

These are stored in `Dout_features`, in the `BoardInfo` structure.

<code>XDA_DOUT_FEATURE_STREAM</code>	the device supports streaming digital output
<code>XDA_DOUT_FEATURE_PORTCFG</code>	the device's outputs are configurable by port
<code>XDA_DOUT_FEATURE_LINECFG</code>	the device's outputs are configurable by line

## ***XDA\_Board\_GetParm***

### **Prototype**

C: i32 XDA\_Board\_GetParm(i32 device, i32 index, f64 \*value)

VB: XDA\_Board\_GetParm (ByVal device As Long, ByVal index As Long, value As Double) As Long

### **Description**

Gets the current value of the specified board parameter.

### **Parameters**

device Device number of the board as specified in Acquittek Control Center. Valid device numbers range from 1 to 63.  
index Board parameter to check. See below.  
value This variable will be assigned the current value of the specified parameter.

### **Board Parameters**

#### ***XDA\_BOARD\_PARM\_AIN\_MEMORY***

Gets current value of on board memory allocated to analog input. See XDA\_Board\_SetParm for details.

#### ***XDA\_BOARD\_PARM\_AOUT\_MEMORY***

Gets current value of on board memory allocated to analog output. See XDA\_Board\_SetParm for details.

#### ***XDA\_BOARD\_PARM\_AUX\_BNC***

Current setting for CH/XH auxiliary BNC connector. See XDA\_Board\_SetParm for details.

#### ***XDA\_BOARD\_PARM\_INT\_SYNC\_MASTER***

Current setting for CH/XH master mode. See XDA\_Board\_SetParm for details.

#### ***XDA\_BOARD\_PARM\_INT\_SYNC\_SLAVE***

Current setting for CH/XH slave mode. See XDA\_Board\_SetParm for details.

#### ***XDA\_BOARD\_PARM\_DLL\_VERSION***

Gets version number of currently installed DLL.

#### ***XDA\_BOARD\_PARM\_DLL\_BUILD***

Gets build number of currently installed DLL.

#### ***XDA\_BOARD\_PARM\_DSP\_MEMORY***

Gets current value of on board memory allocated to DSP functions. See XDA\_Board\_SetParm for details.

*XDA\_BOARD\_PARM\_EXTERNAL\_CLOCK*

On CH/XH, this parm functions identically to XDA\_BOARD\_PARM\_AUX\_BNC. On CM/XM, it enables external clock for input or output. See XDA\_Board\_SetParm for details.

*XDA\_BOARD\_PARM\_TEMPERATURE*

Current temperature of the board in degrees Celsius.

*XDA\_BOARD\_PARM\_TIMEOUT*

Timeout for the device for synchronous functions so that control will eventually be returned to the application if a problem occurs. The default is -1 (no timeout).

## ***XDA\_Board\_Init***

### **Prototype**

C: i32 XDA\_Board\_Init(i32 device, i32 rsvd)

VB: XDA\_Board\_Init (ByVal device As Long, ByVal rsvd As Long) As Long

### **Description**

Locates the device and initializes the onboard hardware to default conditions.

### **Parameters**

device	Device number of the device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
rsvd	Reserved variable. Set this to zero.

## ***XDA\_Board\_SetParm***

### **Prototype**

C: i32 XDA\_Board\_SetParm(i32 device, i32 index, f64 value)

VB: XDA\_Board\_SetParm (ByVal device As Long, ByVal index As Long, ByVal value As Double) As Long

### **Description**

Sets a board parameter.

### **Parameters**

device	Device number of the board as specified in Exacq Control Center. Valid device numbers range from 1 to 63.
index	Board parameter to set. See below.
value	Value to assign to the parameter.

### **Board Parameters**

#### ***XDA\_BOARD\_PARM\_AIN\_MEMORY***

The board has a large amount of on board RAM. The total amount of usable memory is available in the *Ain\_memory* member of the *BoardInfo* structure returned via a call to *XDA\_Board\_GetInfo()*. By default, most of the memory is allocated to the output, with additional memory allocated to the input and for DSP functionality, such as FFTs. In many applications, particularly when capturing data at very high speed, when capturing with multiple boards such that the PCI throughput is exceeded by the aggregate capture rate, or when running multichannel FFTs, more on board memory should be allocated to input or DSP. To avoid a *XDA\_ERR\_MEMORY\_SIZE* error, first reduce the *Aout* memory before increasing the allocation to *Ain* or DSP.

#### ***XDA\_BOARD\_PARM\_AOUT\_MEMORY***

See description under *XDA\_BOARD\_PARM\_AIN\_MEMORY*

#### ***XDA\_BOARD\_PARM\_AUX\_BNC***

Sets function of the CH/XH auxiliary BNC connector. Valid values are:

0 = Input - External trigger. This is the default.

1 = Output - *Ain* internal clock

2 = Output - *Aout* internal clock

0x100 = Input - receive *Ain* external clock

0x200 = Input - receive *Aout* external clock

0x400 = Input clock is sinewave (0 V threshold); otherwise TTL is assumed

#### ***XDA\_BOARD\_PARM\_INT\_SYNC\_MASTER***

Valid only on CH boards. In internal sync mode, a master board can drive *Ain* sample clock and trigger to slave boards via the digital I/O header and a ribbon connector

internal to the PC. This leaves the Aux BNC free for external trigger on the master. Do not set XDA\_BOARD\_PARM\_AUX\_BNC to anything other than external trigger mode when using internal sync master mode.

On CH, valid values are:

XDA\_BOARD\_SYNC\_DISABLE = disable internal sync master mode

XDA\_BOARD\_SYNC\_AIN = enable internal ain sync master mode

XDA\_BOARD\_SYNC\_AOUT = enable internal aout sync master mode

#### *XDA\_BOARD\_PARM\_INT\_SYNC\_SLAVE*

Valid only on CH boards. In internal sync mode, a master board can drive Ain sample clock and trigger to slave boards via the digital I/O header and a ribbon connector internal to the PC. Aux BNC can not be used for any purpose on a board configured in internal sync slave mode.

On CH, valid values are:

XDA\_BOARD\_SYNC\_DISABLE = disable internal sync master mode

XDA\_BOARD\_SYNC\_AIN = enable internal ain sync master mode

XDA\_BOARD\_SYNC\_AOUT = enable internal aout sync master mode

#### *XDA\_BOARD\_PARM\_DSP\_MEMORY*

See description under XDA\_BOARD\_PARM\_AIN\_MEMORY

#### *XDA\_BOARD\_PARM\_EXTERNAL\_CLOCK*

On CH/XH, this parm functions identically to XDA\_BOARD\_PARM\_AUX\_BNC.

On CM/XM, valid values are:

0 = Unused (input). This is the default.

0x100 = Input - receive Ain external clock

0x200 = Input - receive Aout external clock

#### *XDA\_BOARD\_PARM\_EXTERNAL\_MUX*

Switches B-125x external multiplexer attached to CH series boards or XH series boards with digital I/O. On XH boards, you must enable the Right Local Bus outputs on lines 0 and 1 or the function will fail with XDA\_ERR\_PXI\_NOT\_ENABLED.

You can control up to 8 boxes by passing in channel flags + box\_id \* 256.

Pass in the following bit flags to turn channels on/off:

0 = all channels off

1 = channel 1 on

2 = channel 2 on

4 = channel 3 on

8 = channel 4 on

15 = all 4 channels on

#### *XDA\_BOARD\_PARM\_TIMEOUT*

Sets a timeout for the device for synchronous functions so that control will eventually be returned to the application if a problem occurs.

## ***XDA\_CT\_Config***

### **Prototype**

C: i32 XDA\_CT\_Config(i32 device, i32 ct\_num, i32 count, i32 mode, i32 flags)

VB: XDA\_CT\_Config (ByVal device As Long, ByVal ct\_num As Long, ByVal count As Long, ByVal mode As Long, ByVal flags as Long) As Long

### **Description**

Configures settings for a specified counter.

### **Parameters**

device	Device number of the device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
ct_num	Zero-based counter/timer to configure. Valid values vary based on device type.
count	period from the beginning of one count to the beginning of the next count.
mode	counter mode (see below).

#### ***XDA\_CT\_MODE\_ONE\_SHOT***

On 2nd clock after rising edge of gate, output switches low, counter is loaded with count, and counter begins decrementing on each rising edge of clock. Output stays low until count reaches zero, then goes high until next rising edge of gate. Counting continues if gate goes low, and counter is set to count upon a rising edge of gate, even if it hasn't reached zero.

#### ***XDA\_CT\_MODE\_PULSE***

While gate is high, counter decrements. Upon reaching zero, output goes low for one clock period. Then the counter is reloaded with count and output goes high and the cycle repeats.

#### ***XDA\_CT\_MODE\_SQUARE***

While gate is high, counter decrements. Upon reaching 1/2 count, output goes low. When the counter reaches zero, the counter is reloaded with count and output goes high and the cycle repeats. Note: odd count values will have a duty cycle which is non-symmetric by one clock cycle.

#### ***XDA\_CT\_MODE\_STROBE***

On rising edge of gate, counter is loaded with count+1 and counter begins decrementing on each rising edge of clock. When counter reaches zero, output switches low for one clock cycle, then high. Counting continues if gate goes low, and counter is set to count upon a rising edge of gate, even if it hasn't reached zero.

#### ***XDA\_CT\_MODE\_COUNT***

This mode is only available on CH and XH boards. On every rising edge of gate, the count value decrements. When counter reaches zero, it is reset to count and the output goes low for one clock cycle.

*XDA\_CT\_MODE\_HI*

The output set high until the mode is changed. Count has no effect.

flags            sets counter/timer options.

*XDA\_CT\_FLAG\_GATE\_SW*

use software gate mode. Gate is controlled by XDA\_CT\_Gate.

*XDA\_CT\_FLAG\_GATE\_HW*

use hardware gate mode. Gate is controlled by external pin.

*XDA\_CT\_FLAG\_CLOCK\_INTERNAL*

use internal clock.

*XDA\_CT\_FLAG\_CLOCK\_EXTERNAL*

use external clock.

*XDA\_CT\_FLAG\_OUT\_INVERT*

invert counter/timer output pin state.

## ***XDA\_CT\_Gate***

### **Prototype**

C: i32 XDA\_CT\_Gate(i32 device, i32 ct\_num, i32 value)

VB: XDA\_CT\_Gate (ByVal device As Long, ByVal ct\_num As Long, ByVal value As Long) As Long

### **Description**

Switches a software counter/timer gate between digital logic high and digital logic low.

### **Parameters**

device	Device number of the device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
ct_num	Zero-based counter/timer gate to switch. Valid values vary based on device type.
value	set this to 1 for digital logic high, 0 for digital logic low.

## ***XDA\_CT\_GetParm***

### **Prototype**

C: i32 XDA\_CT\_GetParm(i32 device, i32 ct\_num, i32 index, f64 \*value)

VB: XDA\_CT\_GetParm (ByVal device As Long, ByVal ct\_num As Long, ByVal index As Long, value As Double) As Long

### **Description**

Gets the current value of the specified counter/timer parameter.

### **Parameters**

device	Device number of the device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
ct_num	Zero-based counter/timer number. Valid values vary based on device type.
index	Counter/timer parameter to check. See below.
value	This variable will be assigned the current value of the specified parameter.

### **Counter/Timer Parameters**

#### ***XDA\_CT\_PARM\_OUT\_ENABLE***

Enables/disables counter/timer output on digital I/O pins 8 and 9 for CH Series boards.

#### ***XDA\_CT\_PARM\_CLOCK\_MULTIPLIER***

The value of how much faster the counter clock runs than the clock associated with the counter. To get an expected result, multiply this value by a desired count when setting XDA\_CT\_Config, or divide the clock by this value when setting it.

## ***XDA\_CT\_Read***

### **Prototype**

C: i32 XDA\_CT\_Read(i32 device, i32 ct\_num, i32\* count)

VB: XDA\_CT\_Read (ByVal device As Long, ByVal ct\_num As Long, count As Long) As Long

### **Description**

Reads the count of a counter/timer without changing its value.

### **Parameters**

device	Device number of the device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
ct_num	Zero-based counter/timer to read. Valid values vary based on device type.
count	This variable will be assigned the current count of the counter/timer.

## ***XDA\_CT\_SetParm***

### **Prototype**

C: i32 XDA\_CT\_SetParm(i32 device, i32 ct\_num, i32 index, f64 value)

VB: XDA\_CT\_SetParm (ByVal device As Long, ByVal ct\_num As Long, ByVal index As Long, ByVal value As Double) As Long

### **Description**

Sets a counter/timer parameter.

### **Parameters**

device	Device number of the device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
ct_num	Zero-based counter/timer to configure. Valid values vary based on device type.
index	Counter/timer parameter to set. See below.
value	Value to assign to the parameter.

### **Counter/Timer Parameters**

*XDA\_CT\_PARM\_OUT\_ENABLE*

Enables/disables counter/timer output on digital I/O pins 8 and 9 for CH Series boards.

## ***XDA\_Din\_Check***

### **Prototype**

C: i32 XDA\_Din\_Check(i32 device, i32 \*running, i32 \*count)

VB: XDA\_Din\_Check (ByVal device As Long, running As Long, count As Long) As Long

### **Description**

Checks the status of a buffered digital input operation.

### **Parameters**

device	Device number of the input device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
running	This variable will be assigned a value of zero if the operation is finished, nonzero if the operation is still running.
count	This variable will be assigned a value equal to the number of samples recovered so far.

## ***XDA\_Din\_Config***

### **Prototype**

C: i32 XDA\_Din\_Config(i32 device, i32 port, i32 line, i32 mode)

VB: XDA\_Din\_Config (ByVal device As Long, ByVal port As Long, ByVal line As Long, ByVal mode As Long) As Long

### **Description**

Set the direction of a reversible digital input/output port to input. The line must be set to XDA\_ALL\_LINES, otherwise this function will return XDA\_ERR\_LINE\_NUMBER.

### **Parameters**

device	Device number of the device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
port	Digital port for which to set the direction.
line	Digital line for which to set the direction. Set this to XDA_ALL_LINES to configure the port.
mode	Reserved variable. Set this to zero.

## ***XDA\_Din\_Read***

### **Prototype**

C: i32 XDA\_Din\_Read(i32 device, i32 port, i32 line, i32 \*state)

VB: XDA\_Din\_Read (ByVal device As Long, ByVal port As Long, ByVal line As Long, state As Long) As Long

### **Description**

Reads from a digital input line or port.

### **Parameters**

device	Device number of the input device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
port	Zero-based digital port from which to read the value. The range for this value depends on the number of digital inputs on the device.
line	Zero-based line from which to read the value. To read the whole port, set this value to XDA_ALL_LINES.
state	This variable will be assigned a binary value corresponding to the current state of the digital input. If line is set to XDA_ALL_LINES, this will be an X-bit integer where X is the number of lines in the port. Otherwise, it will be 1 for on or 0 for off.

## ***XDA\_Din\_SetBuffer***

### **Prototype**

C: i32 XDA\_Din\_SetBuffer(i32 device, i32 count, void \*buffer)

VB: XDA\_Din\_SetBuffer (ByVal device As Long, ByVal count As Long, buffer() As Any)  
As Long

### **Description**

Locks a buffer and sets it up for DMA transfer in the next buffered digital input operation. To unlock the buffer, either call this function with count = 0 or lock a different buffer.

### **Parameters**

device	Device number of the input device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
count	Total number of samples that will be collected (also, the number of elements in the buffer). Set this to zero to unlock the buffer.
buffer	A pointer to the buffer where values will be stored. This should be of the type appropriate to the data returned by the board. In Visual Basic, pass the name of the array.

## ***XDA\_Din\_SetClock***

### **Prototype**

C: i32 XDA\_Din\_SetClock(i32 device, f64 rate, f64 \*actual)

VB: XDA\_Din\_SetClock (ByVal device As Long, ByVal rate As Double, actual As Double)  
As Long

### **Description**

Sets the clock rate for buffered digital input.

Full speed operation may not be possible with small buffer sizes. Buffer sizes with larger multiples of 4 KB, up to 128 KB, may be required to achieve uninterrupted transfers of multiple channels at high speeds.

### **Parameters**

device	Device number of the input device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
rate	Desired rate for the buffered input operation, in Hz. The range for this value depends on the capabilities of the device.
actual	This variable will be assigned the actual rate that will be used for the buffered input operation.

## ***XDA\_Din\_SetSequence***

### **Prototype**

C: i32 XDA\_Din\_SetSequence(i32 device, i32 count, i32 \*ports)

VB: XDA\_Din\_SetSequence (ByVal device As Long, ByVal count As Long, ports As Long)  
As Long

### **Description**

Sets the sequence of ports to use for buffered digital input. If any of the ports is set for output, it will reflect the current output state of the port.

Currently, 2 ports must be specified. On boards with 2 digital I/O ports, it must be 0 and 1. If the board has 4 digital I/O ports, it can be 0 and 1 or 2 and 3. If you're not interested in the data from one of the ports, it can be configured as an output.

### **Parameters**

device	Device number of the input device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
count	Number of ports to scan in the buffered input operation.
ports	An array of port numbers to scan. In Visual Basic, pass the first element of the array.

## ***XDA\_Din\_SetupTrigger***

### **Prototype**

C: i32 XDA\_Din\_SetupTrigger(i32 device, i32 port, i32 mask, i32 pattern, i32 flags)

VB: XDA\_Din\_SetupTrigger (ByVal device As Long, ByVal port As Long, ByVal mask As Long, ByVal pattern as Long, ByVal flags as Long) As Long

### **Description**

Sets the parameters for a digital trigger. This trigger is then enabled using XDA\_Aout\_SetTrigger. In most cases, the specified port should be set up as an input using XDA\_Din\_Config.

### **Parameters**

device	Device number as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
port	Port number to configure
mask	Enabled bits for trigger
pattern	For pattern trigger, this is the value to trigger on. For edge trigger, each bit controls rising (bit=1) or falling (bit=0).
flags	Defines type of trigger. Valid values are: XDA_DIN_FLAG_TRIGGER_PATTERN XDA_DIN_FLAG_TRIGGER_EDGE

## ***XDA\_Din\_Start***

### **Prototype**

C: i32 XDA\_Din\_Start(i32 device, i32 modeFlags, i32 count)

VB: XDA\_Din\_Start (ByVal device As Long, ByVal modeFlags As Long, ByVal count As Long) As Long

### **Description**

Starts a buffered digital input operation. This may not be used simultaneously with XDA\_Ain\_Start.

### **Parameters**

device	Device number of the input device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
modeFlags	Synchronous or asynchronous. This value can be either XDA_SYNC or XDA_ASYNC. If this is set to XDA_ASYNC, the buffered input will be performed in the background and other operations can be performed. If it is set to XDA_SYNC, the function will return when the scan is complete.
count	Total number of samples to collect. This must be a multiple of 4096 bytes (i.e. 4096 samples on an 8-bit port).

## ***XDA\_Din\_Stop***

### **Prototype**

C: i32 XDA\_Din\_Stop(i32 device)

VB: XDA\_Din\_Stop (ByVal device As Long) As Long

### **Description**

Stops the current buffered digital input operation.

### **Parameters**

device            Device number of the input device as specified in Acquitek Control Center.  
Valid device numbers range from 1 to 63.

## ***XDA\_Dout\_Check***

### **Prototype**

C: i32 XDA\_Dout\_Check(i32 device, i32 \*running, i32 \*rsvd)

VB: XDA\_Dout\_Check (ByVal device As Long, running As Long, rsvd As Long) As Long

### **Description**

Checks the status of a digital output operation.

### **Parameters**

device	Device number as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
running	This variable will be assigned a value of zero if the operation is finished, nonzero if the operation is still running.
rsvd	Reserved variable. Set to zero.

## ***XDA\_Dout\_Config***

### **Prototype**

C: i32 XDA\_Dout\_Config(i32 device, i32 port, i32 line, i32 mode)

VB: XDA\_Dout\_Config (ByVal device As Long, ByVal port As Long, ByVal line As Long, ByVal mode As Long) As Long

### **Description**

Set the direction of a reversible digital input/output port to output. The line must be set to XDA\_ALL\_LINES, otherwise this function will return XDA\_ERR\_LINE\_NUMBER.

### **Parameters**

device	Device number of the device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
port	Digital port for which to set the direction.
line	Digital line for which to set the direction. Set this to XDA_ALL_LINES to configure the port.
mode	Reserved variable. Set this to zero.

## ***XDA\_Dout\_SetBuffer***

### **Prototype**

C: i32 XDA\_Dout\_SetBuffer(i32 device, i32 count, void \*buffer)

VB: XDA\_Dout\_SetBuffer Alias (ByVal device As Long, ByVal count As Long, buffer() As Any) As Long

### **Description**

Locks a buffer and sets it up for DMA transfer in the next buffered digital output operation. To unlock the buffer, either call this function with count = 0 or lock a different buffer.

### **Parameters**

device	Device number of the output device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
count	Total number of values that will be output (also, the number of elements that are in the buffer). Set this to zero to unlock the buffer.
buffer	Buffer to use for output. This should be of the type appropriate to the data returned by the board. In Visual Basic, pass the name of the array.

## ***XDA\_Dout\_SetClock***

### **Prototype**

C: i32 XDA\_Dout\_SetClock(i32 device, f64 rate, f64 \*actual)

VB: XDA\_Dout\_SetClock (ByVal device As Long, ByVal rate As Double, actual As Double) As Long

### **Description**

Sets the clock rate for buffered digital output.

Full speed operation may not be possible with small buffer sizes. Buffer sizes with larger multiples of 4 KB, up to 128 KB, may be required to achieve uninterrupted transfers of multiple channels at high speeds.

### **Parameters**

device	Device number of the output device as specified in Acquiretek Control Center. Valid device numbers range from 1 to 63.
rate	Desired rate for the buffered output operation, in Hz. The range for this value depends on the capabilities of the device.
actual	This variable will be assigned the actual rate that will be used for the buffered output operation.

## ***XDA\_Dout\_SetSequence***

### **Prototype**

C: i32 XDA\_Dout\_SetSequence(i32 device, i32 count, i32 \*ports)

VB: XDA\_Dout\_SetSequence (ByVal device As Long, ByVal count As Long, ports As Long) As Long

### **Description**

Sets the sequence of ports to use for buffered digital output. If one of these ports is set for input, it will not be changed by the buffered output operation.

Currently, 2 ports must be specified. On boards with 2 digital I/O ports, it must be 0 and 1. If the board has 4 digital I/O ports, it can be 0 and 1 or 2 and 3. If you're not interested in the data from one of the ports, it can be configured as an input.

### **Parameters**

device	Device number of the output device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
count	Number of ports that will be used in the buffered output operation.
ports	An array of digital output ports to use in the buffered output operation. In Visual Basic, pass the first element of the array.

## ***XDA\_Dout\_Start***

### **Prototype**

C: i32 XDA\_Dout\_Start(i32 device, i32 count, i32 iterations)

VB: XDA\_Dout\_Start (ByVal device As Long, ByVal count As Long, ByVal iterations As Long) As Long

### **Description**

Starts a buffered digital output operation. This may not be used simultaneously with XDA\_Aout\_Start. The waveform will be output the number of times specified by iterations. If iterations is 0, the waveform will loop until stopped.

### **Parameters**

device	Device number of the output device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
count	Total number of samples in the buffer. This must be a multiple of 64 bytes (i.e. 64 samples on an 8-bit port).
iterations	Number of times to output the buffer or 0 for continuous output.

## ***XDA\_Dout\_Stop***

### **Prototype**

C: i32 XDA\_Dout\_Stop(i32 device)

VB: XDA\_Dout\_Stop (ByVal device As Long) As Long

### **Description**

Stops the current buffered digital output operation.

### **Parameters**

device            Device number of the output device as specified in Acquitek Control Center.  
Valid device numbers range from 1 to 63.

## ***XDA\_Dout\_Write***

### **Prototype**

C: i32 XDA\_Dout\_Write(i32 device, i32 port, i32 line, i32 state)

VB: XDA\_Dout\_Write (ByVal device As Long, ByVal port As Long, ByVal line As Long, ByVal state As Long) As Long

### **Description**

Writes a value to a digital output line or port.

### **Parameters**

device	Device number of the output device as specified in Acquitek Control Center. Valid device numbers range from 1 to 63.
port	Zero-based digital port on which to write the value. The range for this value depends on the number of digital outputs on the device.
line	Zero-based line on which to write the value. To write to the whole port, set this value to XDA_ALL_LINES.
state	Binary value corresponding to the state at which to set the digital output. If line is set to XDA_ALL_LINES, this will be an X-bit integer where X is the number of lines in the port. Otherwise, it will be 1 for on or 0 for off.

## **C Programming Information**

### ***Programming Notes***

Some information to note about using the Acquitek Data Acquisition SDK with the C programming language:

- XDADAQ.H should be included by any file that uses the Acquitek API. This file defines the necessary structs and constants and provides prototypes for Acquitek DA functions. This file is in the C:\Program Files\Acquitek\include directory.
- XDADAQ.lib contains the API functions. It should be linked with any program compiled by a Microsoft compiler that uses the Acquitek API. This file is in the C:\Program Files\Acquitek\lib directory.
- XDADAQ-Borland.lib should be linked with any program compiled by a Borland compiler that uses the Acquitek API.
- All Acquitek API functions use the WINAPI calling convention.
- The errCheck function included in the C samples is used to check for errors. Some helper functions are included in makewave.c and plot.c. In order to use these functions, helpers.h must be included and the proper file must be linked.

### ***Sample Programs***

Beginning on the next page, descriptions of sample C programs are listed, along with the functions they demonstrate and notes for how to connect wires to best demonstrate each sample.

## **ain.c:**

Reads the voltage from an analog input one time.

### **Demonstrates:**

XDA\_Ain\_ReadVolts  
XDA\_Board\_Cleanup

### **Wiring Notes:**

Connect voltage source to input channel

### **Usage:**

ain [-?] [-d#] [-i#]  
-? : prints usage  
-d# : device number  
-i# : input channel

**Default:** ain -d1 -i0

## **ain-buf.c:**

Test the streaming analog input. Deinterleave and average the data after it is collected.

### **Demonstrates:**

XDA\_Ain\_SetBuffer  
XDA\_Ain\_SetClock  
XDA\_Ain\_SetScanRate  
XDA\_Ain\_SetSequence  
XDA\_Ain\_Start  
XDA\_ASYNC  
XDA\_Ain\_Check  
XDA\_Ain\_Deinterleave  
XDA\_Ain\_ConvertToVolts  
XDA\_Board\_Cleanup

### **Wiring Notes:**

Connect a voltage source to the input channel

### **Usage:**

ain-buf [-?] [-d#] [-s#] [-c#] [-r#] [-f\*]

-? : prints usage

-d# : device number

-s# : number of samples

-c# : number of channels(1 to 4)

-r# : clock rate for sampling

-f\* : file name to output data

**Default:** ain-buf -d1 -s2048 -c1 -r100000

## ain-mb.c:

Sets up a multi-buffered input to read voltages. Prints the average input on each half-buffer or stores them to a file.

### Demonstrates:

```
XDA_Ain_SetParm
    XDA_AIN_PARM_MULTIBUFFER
XDA_Ain_SetBuffer
XDA_Ain_SetClock
XDA_Ain_SetScanRate
XDA_Ain_SetSequence
XDA_Ain_Start
    XDA_ASYNC
XDA_Ain_MB_Check
XDA_Ain_MB_Copy
XDA_Ain_Stop
XDA_Board_Cleanup
```

### Wiring Notes:

Connect voltage source to input channel

### Usage:

```
ain-mb [-?] [-d#] [-s#] [-i#] [-r#] [-f*]
-? : prints usage
-d# : device number
-s# : number of samples
-i# : input channel
-r# : clock rate for sampling
-f* : file name to output data
```

**Default:** ain-mb -d1 -s4096 -i1 -r100000

## **ain-raw.c:**

Reads raw data from an input channel using different voltage ranges. The ranges tested will vary based on board specifications.

### **Demonstrates:**

```
XDA_Board_GetInfo
    XDA_BIPOLAR
XDA_Ain_SetParm
    XDA_AIN_PARM_RANGE
XDA_Ain_ReadRaw
XDA_Board_Cleanup
```

### **Wiring Notes:**

Connect a voltage source to the input channel

### **Usage:**

```
ain-raw [-?] [-d#] [-i#]
-? : prints usage
-d# : device number
-i# : input channel
```

**Default:** ain-raw -d1 -i0

## ain-trig.c:

Test the streaming analog input with trigger. Starts capturing data after input voltage raising above threshold voltage. Deinterleave and average the data after it is collected.

### Demonstrates:

```
XDA_Ain_SetBuffer
XDA_Ain_SetTrigger
    XDA_TRIGGER_RISING
XDA_Ain_SetClock
XDA_Ain_SetScanRate
XDA_Ain_SetSequence
XDA_Ain_Start
    XDA_ASYNC
XDA_Ain_Check
XDA_Ain_Deinterleave
XDA_Ain_ConvertToVolts
XDA_Board_Cleanup
```

### Wiring Notes:

Connect a voltage source to the input channel.

### Usage:

```
ain-trig [-?] [-d#] [-s#] [-c#] [-t#] [-p#] [-r#] [-f*]
-? : prints usage
-d# : device number
-s# : number of samples
-c# : number of channels(1 to 4)
-p# : pre or post trigger delay
-r# : clock rate for sampling
-f* : file name to output data
```

**Default:** ain-trig -d1 -s524288 -t2.5 -c1 -p0 -r100000

## **aio.c:**

Tests analog I/O by writing different voltages to the output and then reading the voltages (many times) from the input.

### **Demonstrates:**

XDA\_Aout\_WriteVolts

XDA\_Ain\_ReadVolts

### **Wiring Notes:**

Connect the analog output channel specified to the analog input channel specified

### **Usage:**

aio [-?] [-d#] [-r#] [-i#] [-o#] [# # #...]

-? : prints usage

-d# : device number

-r# : number of read times

-i# : input channel

-o# : output channel

# # #... : voltages to output

**Default:** aio -d1 -r100 -i0 -o0 0 1 2 3 4 5

## **aio-ibuf.c:**

Outputs various voltages and read them in with an Input Buffer using the same board.

### **Demonstrates:**

XDA\_Ain\_SetBuffer  
XDA\_Ain\_SetClock  
XDA\_Ain\_SetScanRate  
XDA\_Ain\_SetSequence  
XDA\_Aout\_WriteVolts  
XDA\_Ain\_Start  
    XDA\_ASYNC  
XDA\_Ain\_Check  
XDA\_Ain\_ConvertToVolts  
XDA\_Board\_Cleanup

### **Wiring Notes:**

Connect the corresponding input/output channels together. i.e. Ain 0 to Aout 0.

### **Usage:**

aio-ibuf [-?] [-d#] [-s#] [-c#] [-r#] [# # #...]

-? : prints usage

-d# : device number

-s# : number of samples

-c# : channel, both input and output

-r# : clock rate for sampling

# # #... : values to output

**Default:** aio-ibuf -d1 -s2048 -c1 -r100000 0 1 2 3 4

## **aio-obuf.c:**

Outputs a sin wave and reads in a number of voltages

### **Demonstrates:**

XDA\_Aout\_ConvertToRaw  
XDA\_Aout\_SetClock  
XDA\_Aout\_SetBuffer  
XDA\_Aout\_SetSequence  
XDA\_Aout\_Start  
XDA\_Ain\_ReadVolts  
XDA\_Aout\_Check  
XDA\_Aout\_Stop  
XDA\_Aout\_SetBuffer  
XDA\_Board\_Cleanup

### **Wiring Notes:**

Connect the corresponding input/output channels together. i.e. Ain 0 to Aout 0.

### **Usage:**

aio-obuf [-?] [-d#] [-o#] [-r#] [-c#] [-i#]  
-? : prints usage  
-d# : device number  
-o# : output channel  
-r# : clock rate  
-c# : count of points  
-i# : number of iterations

**Default:** aio-obuf -d1 -o0 -r250000 -c1024 -i1

## **aout.c:**

Writes voltages to output channel one after the other

### **Demonstrates:**

XDA\_Aout\_WriteVolts

XDA\_Board\_Cleanup

### **Wiring Notes:**

Connect output channel to volt meter

### **Usage:**

aout [-?] [-d#] [-o#] [# # #...]

-? : prints usage

-d# : device number

-o# : output channel

# # #... : voltages to output

**Default:** aout -d1 -o0 0.0 1.0 2.0 3.0 4.0

## **aout-buf.c:**

Outputs a sin wave on one of the analog output channels

### **Demonstrates:**

XDA\_Aout\_ConvertToRaw  
XDA\_Aout\_SetClock  
XDA\_Aout\_SetBuffer  
XDA\_Aout\_SetSequence  
XDA\_Aout\_Start  
XDA\_Aout\_Stop  
XDA\_Board\_Cleanup

### **Wiring Notes:**

Connect a scope to the output channel to see the sin wave

### **Usage:**

aout-buf [-?] [-d#] [-o#] [-r#] [-c#] [-i#]  
-? : prints usage  
-d# : device number  
-o# : output channel  
-r# : clock rate  
-c# : count of points  
-i# : number of iterations

**Default:** aout-buf -d1 -o0 -r250000 -c1024 -i1

## **aout-mb.c:**

Sets up and output in multibuffer mode.

### **Demonstrates:**

XDA\_Aout\_SetClock  
XDA\_Aout\_SetParm  
    XDA\_AOUT\_PARM\_MULTIBUFFER  
XDA\_Aout\_SetBuffer  
XDA\_Aout\_SetSequence  
XDA\_Aout\_Reload\_Proc  
XDA\_Aout\_Preload  
XDA\_Aout\_Start  
XDA\_Aout\_Wait  
XDA\_Aout\_Stop  
XDA\_Aout\_SetBuffer  
XDA\_Board\_Cleanup

### **Wiring Notes:**

Connect scope to output channel

### **Usage:**

aout-mb [-d <device>] [-o <channel>] [-r <rate>] [-i <input file>] [-b <buffer size>]

### **Default:**

aout-mb -d 1 -o 0 -r 13.5e6

## **aout-raw:**

Test output using raw values rather than voltages.

### **Demonstrates:**

XDA\_Aout\_WriteRaw  
XDA\_Board\_Cleanup

### **Wiring Notes:**

Connect output channel to a volt meter

### **Usage:**

aout-raw [-?] [-d#] [-o#] [# # #...]

-? : prints usage

-d# : device number

-o# : output channel

# # #... : raw data to output

**Default:** aout-raw -d1 -o0 0 8000 16000 24000 32000

## **aout-trig.c:**

Outputs a sin wave every time the zeroth bit is triggered (raised on port 0)

### **Demonstrates:**

```
XDA_Aout_ConvertToRaw
XDA_Aout_SetClock
XDA_Aout_SetBuffer
XDA_Aout_SetSequence
XDA_Din_Config
    XDA_ALL_LINES
XDA_Din_SetupTrigger
    XDA_DIN_FLAG_TRIGGER_EDGE
XDA_Aout_SetTrigger
    XDA_TRIGGER_DIGITAL
XDA_Aout_Start
XDA_Aout_Stop
XDA_Board_Cleanup
```

### **Wiring Notes:**

Connect scope to output channel and switch/button to bit zero of digital port 0

### **Usage:**

```
aout-trig [-?] [-d#] [-o#] [-r#] [-c#] [-i#]
-? : prints usage
-d# : device number
-o# : output channel
-r# : clock rate
-c# : count of points
-i# : number of iterations
```

**Default:** aout-trig -d1 -o0 -r250000 -c1024 -i1

## **brd-sync.c:**

The Master uses the digital I/O ports to clock and trigger the Slave boards. Up to 8 boards have been tested in this configuration. Board 1 is always the master.

### **Demonstrates:**

```
XDA_Board_SetParm
    XDA_BOARD_PARM_AOUT_MEMORY
    XDA_BOARD_PARM_DSP_MEMORY
    XDA_BOARD_PARM_AIN_MEMORY
    XDA_BOARD_PARM_INT_SYNC_MASTER
    XDA_BOARD_PARM_INT_SYNC_SLAVE
XDA_Ain_SetBuffer
XDA_Ain_SetClock
XDA_Ain_SetSequence
XDA_Ain_GetParm
    XDA_AIN_PARM_EXT_CLK_DIVISOR
    XDA_AIN_PARM_TRIGGER_OFFSET
XDA_Dout_Config
    XDA_ALL_LINES
XDA_Ain_SetParm
    XDA_AIN_PARM_TRIGGER_DELAY
    XDA_AIN_PARM_EXT_CLK_DIVISOR
XDA_Ain_SetTrigger
    XDA_TRIGGER_RISING
    XDA_TRIGGER_EXTERNAL
    XDA_TRIGGER_DIGITAL
XDA_Din_Config
    XDA_ALL_LINES
    XDA_DIN_MODE_TERMINATED
XDA_Din_SetupTrigger
    XDA_DIN_FLAG_TRIGGER_EDGE
XDA_Ain_Start
    XDA_ASYNC
XDA_Ain_Check
XDA_Ain_Stop
XDA_Board_Cleanup
```

### **Wiring Notes:**

Connect the master board to the slave boards via the digital I/O header with a ribbon connector. Trigger the master with an external trigger. Connect voltage sources to all the analog inputs.

**Usage:**

brd-sync [-?] [-n#] [-c#] [-r#] [-s#] [-t#] [-f\*]

-? : prints usage

-n# : number of devices

-c# : number of channels

-r# : clock rate

-s# : sample size per channel

-t# : pretrigger sample size

-f\* : output file name

**Default:** brd-sync -n2 -c2 -r10000 -t100 -s4096

## **ct.c:**

Demonstrates a software gated counter in the mode of the users choice. Note: If using an external clock, must execute XDA\_CT\_Config before

### **Demonstrates:**

```
XDA_Board_GetInfo
    FAMILY_CH
XDA_Dout_Config
XDA_CT_SetParm
    XDA_CT_PARM_OUT_ENABLE
    XDA_CT_PARM_CLOCK_MULTIPLIER
XDA_Ain_SetClock
XDA_Aout_SetClock
XDA_CT_Config
    XDA_CT_MODE_HI
    XDA_CT_MODE_ONE_SHOT
    XDA_CT_MODE_PULSE
    XDA_CT_MODE_SQUARE
    XDA_CT_MODE_STROBE
    XDA_CT_MODE_COUNT
    XDA_CT_FLAG_GATE_SW
    XDA_CT_FLAG_OUT_INVERT
XDA_CT_Gate
```

### **Wiring Notes:**

Connect an LED or scope to the counter output pin (see manual for pin numbers)

### **Usage:**

```
ct [-?] [-d#] [-c#] [-r#] [-n#]
-? : prints usage
-d# : device number
-c# : counter number
-r# : clock rate
-n# : number to count from
-m# : mode of counter - see SDK for more details
    0 : HI
    1 : ONE_SHOT
    2 : PULSE
    3 : SQUARE
    4 : STROBE
    5 : COUNT
```

**Default:** ct -d1 -c0 -r10000 -n10000 -m3

## **din.c:**

Figures out how many digital inputs on the device and then checks the state of the whole port, then the individual bits. Ports are pulled high by default

### **Demonstrates:**

```
XDA_Board_GetInfo
XDA_Din_Config
    XDA_ALL_LINES
XDA_Din_Read
    XDA_ALL_LINES
XDA_Board_Cleanup
```

### **Wiring Notes:**

Connect digital signal to any digital port of the board

### **Usage:**

```
din [-?] [-d#]
-? : prints usage
-d# : device number
```

**Default:** din -d1

## **din-buf.c:**

Test the streaming digital input. Can be output to a file if specified.

### **Demonstrates:**

XDA\_Din\_Config  
    XDA\_ALL\_LINES  
XDA\_Din\_SetSequence  
XDA\_Din\_SetBuffer  
XDA\_Din\_SetClock  
XDA\_Din\_Start  
    XDA\_ASYNC  
XDA\_Din\_Check  
XDA\_Din\_Stop  
XDA\_Board\_Cleanup

### **Wiring Notes:**

Connect digital signal to input port 0 (8 bits).

### **Usage:**

din-buf [-?] [-d#] [-s#] [-r#] [-f\*]  
-? : prints usage  
-d# : device number  
-s# : number of samples  
-r# : clock rate for sampling  
-f\* : file name to output data

**Default:** din-buf -d1 -s2048 -r100000

## **dio.c:**

Tests digital I/O by writing different values to the output and then reading them from the input.

### **Demonstrates:**

XDA\_Din\_Config  
XDA\_Dout\_Config  
XDA\_Din\_Read  
XDA\_Dout\_Write  
XDA\_Board\_Cleanup

### **Wiring Notes:**

Connect the pins from the input port to the corresponding pins of the output port.

### **Usage:**

dio [-?] [-d#] [-o#] [-i#] [-pXX]  
-? : prints usage  
-d# : device number  
-o# : output port  
-i# : input port  
-pXX : hex pattern to output

**Default:** dio -d1 -o0 -i1 -paa

## **dout.c:**

Turns all (or inputted pattern) digital out lines of the output port, then off. Then cycles through turning each line on and off.

### **Demonstrates:**

```
XDA_Dout_Config
    XDA_ALL_LINES
XDA_Dout_Write
XDA_Board_Cleanup
```

### **Wiring Notes:**

Connect digital output port to LEDs or other indicator

### **Usage:**

```
dout [-?] [-d#] [-o#] [-pXX]
-? : prints usage
-d# : device number
-o# : output channel
-pXX : hex pattern to output
```

**Default:** `dout -d1 -o0 -p0`

## **dout-buf.c:**

Test the streaming digital output. Pattern can either be from a binary file or a command line argument.

### **Demonstrates:**

XDA\_Dout\_Config  
XDA\_Din\_Config  
XDA\_Dout\_SetClock  
XDA\_Dout\_SetSequence  
XDA\_Dout\_SetBuffer  
XDA\_Dout\_Start  
XDA\_Dout\_Stop  
XDA\_Board\_Cleanup

### **Wiring Notes:**

Connect LED or Logic analyser to Port 0 (8 bits) or Port 0/1 (16 bits)

### **Usage:**

dout-buf [-?] [-d#] [-c#] [-b#] [-l] [-f\*] (xxxx)

-? : prints usage

-d# : device number

-c# : clock speed (in Hz)

-b# : number of bits (8, 16)

-l : looping

-f\* : file for pattern (given pattern ignored)

(xxxx) : pattern (must have if no filename)

**Default:** dout-buf -d1 -c1000 -b8 (xxxx)

## **echo\_mode.c:**

Sets up multibuffer capture and echo mode.

### **Demonstrates:**

XDA\_Ain\_SetParm  
    XDA\_AIN\_PARM\_MULTIBUFFER  
XDA\_Ain\_SetBuffer  
XDA\_Ain\_SetSequence  
XDA\_Ain\_SetClock  
XDA\_Aout\_SetParm  
    XDA\_AOUT\_PARM\_ECHO\_DELAY  
XDA\_Aout\_SetClock  
XDA\_Aout\_SetSequence  
XDA\_Aout\_SetBuffer  
XDA\_Dout\_Config  
XDA\_Dout\_Write  
XDA\_Din\_SetupTrigger  
XDA\_Aout\_SetTrigger  
XDA\_Ain\_SetTrigger  
XDA\_Ain\_Start  
XDA\_Aout\_Start  
XDA\_Dout\_Write  
XDA\_Aout\_Stop  
XDA\_Ain\_Stop  
XDA\_Board\_Cleanup

### **Wiring Notes:**

Connect signal you want to echo on chan0, and scope to output chan0

### **Usage:**

echo\_mode [-d <device>] [-s <delay>] [-o <channel>] [-r <rate>]

### **Default:**

echo\_mode -d 1 -s 1000 -o 1 -r 10e6

## **fft.c:**

Start a one or two channel buffered analog input with FFT.

### **Demonstrates:**

```
XDA_Ain_SetParm
    XDA_AIN_PARM_FFT_BLOCK_MODE
    XDA_AIN_PARM_FFT_SIZE
    XDA_AIN_PARM_FFT_WINDOW_TYPE
    FFT_WINDOW_TYPE_HANNING
XDA_Ain_SetBuffer
XDA_Ain_SetSequence
XDA_Ain_SetClock
XDA_Ain_Start
    XDA_SYNC
XDA_Board_Cleanup
```

### **Wiring Notes:**

Connect a voltage source to the input channel

### **Usage:**

```
fft [-?] [-d#] [-c#] [-n#] [-r#] [-f*]
-? : prints usage
-d# : device number
-c# : number of channels (1 or 2)
-n# : number of fft points(16 to 4096 (2^#))
-r# : clock rate for sampling
-f* : file name to output data
```

**Default:** `fft -d1 -c1 -n1024 -r250000 -fft.dat`

## **info.c:**

Gets the board info from the device

### **Demonstrates:**

```
XDA_Board_GetInfo
    XDA_AIN_FEATURE_SIMULTANEOUS
    XDA_AIN_FEATURE_SCANNED
    XDA_AIN_FEATURE_BIPOLAR
    XDA_BIPOLAR
    XDA_AIN_FEATURE_UNIPOLAR
    XDA_UNIPOLAR
XDA_Board_Cleanup
```

### **Wiring Notes:**

None

### **Usage:**

```
info [-?] [-d#]
-? : prints usage
-d# : device number
```

**Default:** info -d1

## **mul-brd.c:**

Captures streaming analog input from 2 boards at the same time. If sample size, clock rate and number of channels are high enough, it will exceed the limits of the PCI throughput.

### **Demonstrates:**

```
XDA_Ain_SetBuffer
XDA_Ain_SetClock
XDA_Ain_SetSequence
XDA_Board_SetParm
    XDA_BOARD_PARM_AOUT_MEMORY
    XDA_BOARD_PARM_DSP_MEMORY
    XDA_BOARD_PARM_AIN_MEMORY
XDA_Ain_Start
    XDA_ASYNC
XDA_Ain_Check
XDA_Board_Cleanup
```

### **Wiring Notes:**

Connect voltage sources to the analog inputs of the boards.

### **Usage:**

```
mul-brd [-?] [-c#] [-r#] [-s#]
-? : prints usage
-c# : number of channels
-r# : clock rate
-s# : sample size per channel
```

**Default:** mul-brd -c1 -r10000 -s4096

**mux.c:**

Switch inputs on a B-125x external multiplexer box connected to a CH series board.

**Demonstrates:**

XDA\_Board\_SetParm  
XDA\_BOARD\_PARM\_EXTERNAL\_MUX

**Wiring Notes:**

None

**Usage:**

mux [-?] [-d#] [-i#] [-b#]  
-? : prints usage  
-d# : device number  
-i# : input number  
-b# : box id number

**Default:** mux -d1 -i0 -b0

## pll.c:

Enables PLL on Acquitek CM board, and starts a multibuffered capture with synchronized clock.

### Demonstrates:

```
XDA_Ain_SetParm
    XDA_AIN_PARM_MULTIBUFFER
    XDA_AIN_PARM_POLARITY
    XDA_BIPOLAR
XDA_Ain_SetBuffer
XDA_Ain_SetClock
XDA_Ain_SetSequence
XDA_Ain_SetPll
XDA_Ain_SetTrigger
    XDA_TRIGGER_EXTERNAL
    XDA_TRIGGER_RISING
XDA_Ain_Start
XDA_Ain_GetParm
XDA_Ain_MB_Check
XDA_Ain_MB_Copy
XDA_Ain_Stop
XDA_Board_SetParm
    XDA_BOARD_PARM_TIMEOUT
XDA_Board_Cleanup
```

### Wiring Notes:

Connect shaft encoder output (or similar source to which the sampling should lock) to Ctrl 0 Gate. Connect once per revolution signal (or similar reference) to A/D SeqSt.

### Usage:

```
pll [-?] [-d#] [-c#] [-s#] [-r#] [-t#] [-b#] [-z#] [-l#]
-? : prints usage
-d# : device number
-c# : number of channel
-s# : samples per pulse
-r# : clock rate
-t# : run time (in sec)
-b# : bandwidth (in rad/s)
-z# : damping coefficient
-l# : lock threshold
```

**Default:** pll -d1 -c2 -s256 -r1000 -t30 -b10 -z1 -l0.1

## **pxi.c:**

Board in PXI Star Trigger Controller slot triggers on a software trigger, and forward the trigger out on the star trigger lines. Other (up to 8 boards, 4 channels each) PXI board are configured to trigger via the star trigger input.

### **Demonstrates:**

```
XDA_Board_SetParm
    XDA_BOARD_PARM_AOUT_MEMORY
    XDA_BOARD_PARM_DSP_MEMORY
    XDA_BOARD_PARM_AIN_MEMORY
    XDA_BOARD_PARM_TIMEOUT
XDA_Ain_SetSequence
XDA_Ain_SetClock
XDA_Ain_SetBuffer
XDA_Ain_SetParm
    XDA_AIN_PARM_STAR_CTRLR_OUTPUT
    XDA_AIN_PARM_SW_TRIGGER
XDA_Ain_SetTrigger
    XDA_AIN_TRIGGER_SOFTWARE
    XDA_AIN_TRIGGER_PXI_STAR
    XDA_AIN_TRIGGER_RISING
XDA_Ain_Start
    XDA_ASYNC
XDA_Ain_Check
XDA_Ain_Stop
XDA_Ain_ConvertToVolts
XDA_Board_Cleanup
```

### **Wiring Notes:**

Connect voltages to the analog inputs of all the boards. Device 1 must be in the Star Trigger Controller slot of PXI chassis and PXI Slot must be set to 2 in Acquitek Control Center

### **Usage:**

```
pxi [-?] [-n#] [-c#] [-r#] [-s#] [-f*]
-? : prints usage
-n# : number of devices
-c# : number of channels
-r# : clock rate
-s# : sample size
-f* : output file name
```

**Default:** pxi -n2 -c1 -r10000 -s4096

## **temp.c:**

Gets the temperature of the device

### **Demonstrates:**

XDA\_Board\_GetParm  
    XDA\_BOARD\_PARM\_TEMPERATURE  
XDA\_Board\_Cleanup

### **Wiring Notes:**

None

### **Usage:**

temp [-?] [-d#]  
-? : prints usage  
-d# : device number

**Default:** temp -d1

## Microsoft® Visual Basic® Programming Information

### *Programming Notes*

Some information to note about using the Acquitek Data Acquisition SDK with Visual Basic:

- XDADAQ.bas contains prototypes of Acquitek DA API functions and should be added to Visual Basic projects that use the API.
- XDADAQ.dll contains the API functions and must be in the current directory or path of any system running a Visual Basic program that uses the Acquitek DA API.
- Some functions take arrays as arguments. Some of these will require the first element of the array to be passed; others will require the array name. See the corresponding pages of this document for more information.
- When a program calls a SetBuffer function in Visual Basic, XDADAQ.bas calls a corresponding SetBufferSA function. These functions use SAFEARRAY to ensure the proper handling of arrays. These functions will return XDA\_ERR\_SA\_LOCK if SafeArrayLock or SafeArrayUnlock returns an error.
- Visual Basic does not utilize unsigned data types. In unipolar mode on 16-bit boards (such as the CM Series), raw D/A or A/D values should be stored in a variable of the Integer data type. Values greater than 32767 will be stored as negative numbers. They can be converted to voltages as usual, with XDA\_Ain\_ConvertToVolts. To find the correct raw value, copy the value from the Integer to a variable of the Long data type, then add 65536 to negative values.
- The ErrCheck sub procedure included in the Visual Basic samples is used to check for errors.

### *Sample Programs*

Beginning on the next page, descriptions of sample Visual Basic programs are listed, along with the functions they demonstrate and notes for how to connect wires to best demonstrate each sample.

## ain

### **Description**

Writes a series of voltages to an analog output specified by the user and then reads them from an analog input (also specified by the user).

### **Demonstrates:**

XDA\_Aout\_WriteVolts

XDA\_Ain\_ReadVolts

### **Wiring Notes:**

Connect the specified analog output to the specified analog input. By default, this sample uses device 1, output channel 0, and input channel 0.

## **ain-buf**

### **Description**

Writes a series of voltages to an analog output specified by the user and then reads them from an analog input (also specified by the user) using buffered input.

### **Demonstrates:**

- XDA\_Ain\_SetSequence
- XDA\_Ain\_SetClock
- XDA\_Ain\_SetScanRate
- XDA\_Ain\_SetBuffer
- XDA\_Aout\_WriteVolts
- XDA\_Ain\_Start
- XDA\_Ain\_Check
- XDA\_Ain\_ConvertToVolts

### **Wiring Notes:**

Connect the specified analog output to the specified analog input. By default, this sample uses device 1, output channel 0, and input channel 0.

## **ain-buf2**

### **Description**

Writes a positive voltage to one channel (specified in the file) and a voltage of zero to another channel. Samples two input channels (specified in the file) using a buffered input.

### **Demonstrates:**

XDA\_Aout\_WriteVolts  
XDA\_Ain\_SetBuffer  
XDA\_Ain\_SetSequence  
XDA\_Ain\_SetClock  
XDA\_Ain\_SetScanRate  
XDA\_Ain\_Start  
XDA\_Ain\_Check  
XDA\_Ain\_ConvertToVolts

### **Wiring Notes:**

To see best results, connect analog output channels 0 and 1 to analog input channels 0 and 1, respectively. Channels not connected will show a voltage of (near) zero. This sample will most likely use device 1.

## **ain-buf3**

### **Description**

Conducts a single-channel buffered analog input of a channel specified in the file and displays some of the results.

### **Demonstrates:**

XDA\_Ain\_SetBuffer  
XDA\_Ain\_SetSequence  
XDA\_Ain\_SetClock  
XDA\_Ain\_SetScanRate  
XDA\_Ain\_Start  
XDA\_Ain\_Check  
XDA\_Ain\_ConvertToVolts

### **Wiring Notes:**

To see best results, connect an analog source to the analog input channel specified in the file. This sample will most likely use device 1, input channel 0.

## **aincheck**

### **Description**

Reads the input voltage from a channel (specified by the user).

### **Demonstrates:**

XDA\_Ain\_ReadVolts

### **Wiring Notes:**

To see best results, connect an analog source to the analog input channel specified by the user. By default, this sample uses device 1, input channel 0.

## **ain-mb**

### **Description**

Conducts a multi-buffered analog input on one channel (specified in the file) and displays average voltage for each partial buffer.

### **Demonstrates:**

- XDA\_Ain\_SetParm
- XDA\_Ain\_SetBuffer
- XDA\_Ain\_SetSequence
- XDA\_Ain\_SetClock
- XDA\_Ain\_SetScanRate
- XDA\_Ain\_Start
- XDA\_Ain\_MB\_Check
- XDA\_Ain\_MB\_Copy
- XDA\_Ain\_ConvertToVolts
- XDA\_Ain\_Stop

### **Wiring Notes:**

To see best results, connect an analog source to the analog input channel specified in the file. This sample will most likely use device 1, input channel 0.

## **ain-raw**

### **Description**

Writes a voltage to an analog output channel (specified by the user) and then reads the raw A/D values from an analog input (specified by the user) at different ranges. The ranges tested will vary based on board specifications.

### **Demonstrates:**

XDA\_Aout\_WriteVolts

XDA\_Board\_GetInfo

XDA\_Ain\_SetParm

XDA\_Ain\_ReadRaw

### **Wiring Notes:**

Connect the specified analog output to the specified analog input. This sample will most likely use device 1, output channel 0, and input channel 0.

## aintrig

### Description

Conducts a two-channel buffered output on analog output channels 0 and 1 of a device specified by the user of values between 0.4V and 3.2V. Sets up a trigger on input channel 0 and then conducts a triggered, buffered analog input operation.

### Demonstrates:

XDA\_Aout\_ConvertToRaw  
XDA\_Aout\_SetClock  
XDA\_Aout\_SetBuffer  
XDA\_Aout\_SetSequence  
XDA\_Ain\_SetBuffer  
XDA\_Ain\_SetClock  
XDA\_Ain\_SetSequence  
XDA\_Ain\_SetTrigger  
XDA\_Board\_SetParm  
XDA\_Aout\_Start  
XDA\_Ain\_Start  
XDA\_Aout\_Stop  
XDA\_Ain\_GetParm

### Wiring Notes:

Connect analog output channels 0 and 1 to analog input channels 0 and 1, respectively. This sample will most likely use device 1.

## **aout**

### **Description**

Writes a series of voltages to an analog output channel (specified by the user), sleeping one second between each write.

### **Demonstrates:**

XDA\_Aout\_WriteVolts

### **Wiring Notes:**

To see best results, measure the voltage from the analog output channel specified by the user. By default, this sample uses device 1, output channel 0.

## **aout-buf**

### **Description**

Starts a buffered analog output of voltages between 0.4V and 3.2V on an analog output channel specified by the user. Stops when the user clicks "Stop".

### **Demonstrates:**

XDA\_Aout\_ConvertToRaw

XDA\_Aout\_SetClock

XDA\_Aout\_SetBuffer

XDA\_Aout\_Start

XDA\_Aout\_Stop

### **Wiring Notes:**

To see best results, measure the voltage from the analog output channel specified by the user. By default, this sample uses device 1, output channel 0.

## **aout-buf2**

### **Description**

Conducts a two-channel buffered output on analog output channels 0 and 1 of a device specified by the user of values between 0.4V and 3.2V. At a specified interval, reads the input values from analog input channels 0 and 1 of another device specified by the user. Stops after it has read the channels a specified number of times.

### **Demonstrates:**

XDA\_Aout\_ConvertToRaw  
XDA\_Aout\_SetClock  
XDA\_Aout\_SetBuffer  
XDA\_Aout\_SetSequence  
XDA\_Aout\_Start  
XDA\_Aout\_Stop  
XDA\_Ain\_ReadVolts

### **Wiring Notes:**

Connect analog output channels 0 and 1 to analog input channels 0 and 1, respectively. This sample will most likely use device 1.

## **aout-raw**

### **Description**

Writes raw D/A values to an analog output channel specified by the user. Waits a specified amount of time between each write.

### **Demonstrates:**

XDA\_Aout\_WriteRaw

### **Wiring Notes:**

To see best results, measure the voltage from the analog output channel specified by the user. By default, this sample uses device 1, output channel 0.

## **din**

### **Description**

Figures out how many digital inputs the device (specified in the file) has, then reads the state of all the lines of a port (specified by the user).

### **Demonstrates:**

XDA\_Board\_GetInfo

XDA\_Din\_Config

XDA\_Din\_Read

### **Wiring Notes:**

To see best results, connect a digital signal to the digital input port specified by the user. This sample will most likely use device 1.

## **din-buf**

### **Description**

Conducts a buffered digital input on a port (specified by the user) and writes values to a different port (specified by the user). Values from 0 to 255 are written at a certain interval.

### **Demonstrates:**

- XDA\_Dout\_Config
- XDA\_Din\_Config
- XDA\_Dout\_Write
- XDA\_Din\_SetClock
- XDA\_Din\_SetBuffer
- XDA\_Din\_SetSequence
- XDA\_Din\_Start
- XDA\_Din\_Stop
- XDA\_Dout\_Write

### **Wiring Notes:**

Connect lines 0-7 on the digital port specified for output to lines 0-7 on the digital port specified for input. By default, this sample uses device 1, port 0 for input and device 1, port 1 for output.

## **dio**

### **Description**

Writes a binary value (specified by the user) to a digital output port (specified in the file) port and then reads from another specified port (specified in the file).

### **Demonstrates:**

XDA\_Dout\_Config

XDA\_Din\_Config

XDA\_Dout\_Write

XDA\_Din\_Read

### **Wiring Notes:**

Connect lines 0-7 on the digital port specified for output to lines 0-7 on the digital port specified for input. This sample will most likely use device 1, port 0 for output and device 1, port 1 for input.

## **dio-line**

### **Description**

Provides checkboxes for the user to turn digital output lines for a port (specified in the file) on or off. Shows the status of digital input lines on a different port (specified in the file).

### **Demonstrates:**

XDA\_Dout\_Config

XDA\_Din\_Config

XDA\_Dout\_Write

XDA\_Din\_Read

### **Wiring Notes:**

Connect lines 0-7 on the digital port specified for output to lines 0-7 on the digital port specified for input. By default, this sample uses device 1 and may not work properly if no device is connected and associated with that number. If using device 1 is not possible, change the device number in dio-line.frm.

## **dout-buf**

### **Description**

Conducts a buffered digital output on a port (specified in the file) with binary values between 0 and 255 (0x00 and 0xff). Reads an input port (specified in the file) at an interval specified in the file.

### **Demonstrates:**

XDA\_Dout\_Config  
XDA\_Din\_Config  
XDA\_Dout\_SetClock  
XDA\_Dout\_SetBuffer  
XDA\_Dout\_SetSequence  
XDA\_Dout\_Start  
XDA\_Dout\_Stop  
XDA\_Din\_Read

### **Wiring Notes:**

Connect lines 0-7 on the digital port specified for output to lines 0-7 on the digital port specified for input. By default, this sample uses device 1, port 0 for input and device 1, port 1 for output.

## **getinfo**

### **Description**

Gets information about a device (specified in the file) using the XDA\_Board\_GetInfo function. Displays it in a form.

### **Demonstrates:**

XDA\_Board\_GetInfo

### **Wiring Notes:**

By default, this sample uses device 1 and may not work properly if no device is connected and associated with that number. If using device 1 is not possible, change the device number in getinfo.frm.

## **temp**

### **Description**

Gets the current board temperature of a device (specified in the file) and updates it at an interval specified in the file.

### **Demonstrates:**

XDA\_Board\_GetParm

### **Wiring Notes:**

By default, this sample uses device 1 and may not work properly if no device is connected and associated with that number. If using device 1 is not possible, change the device number in temp.frm.

## **Microsoft® Visual C++™ Programming Information**

### ***Programming Notes***

Some information to note about using the Acquitek Data Acquisition SDK with Visual C++:

- XDADAQ.H should be included by any file that uses the Acquitek DA API. This file defines the necessary structs and constants and provides prototypes for Acquitek DA functions. This file is in the C:\Program Files\Acquitek\include directory.
- XDADAQ.lib contains the API functions should be linked with any program that uses the Acquitek DA API. This file is in the C:\Program Files \Acquitek\lib directory.
- All Acquitek DA API functions use the WINAPI calling convention.
- The errCheck function included in the Visual C++ samples is used to check for errors.

### ***Sample Programs***

Beginning on the next page, descriptions of sample Visual C++ programs are listed, along with the functions they demonstrate and notes for how to connect wires to best demonstrate each sample.

## **ain-buf**

### **Description**

Tests streaming input of one or two channels. Writes a constant voltage to a channel and then writes different voltages to another channel. Afterward, reads one or both of the analog input channels and shows average voltages. Note that voltages written to channel 0 will be read from input channel 0, etc.

### **Demonstrates:**

XDA\_Aout\_WriteVolts  
XDA\_Ain\_SetBuffer  
XDA\_Ain\_SetClock  
XDA\_Ain\_SetScanRate  
XDA\_Ain\_SetSequence  
XDA\_Ain\_Start  
XDA\_Ain\_Check  
XDA\_Ain\_Deinterleave  
XDA\_Ain\_ConvertToVolts

### **Wiring Notes:**

To see best results, connect analog output channels 0 and 1 to analog input channels 0 and 1, respectively. Channels not connected will show a voltage of (near) zero. This sample will most likely use device 1.

## **aincheck**

### **Description**

Reads an analog input channel and displays the voltage and raw A/D value.

### **Demonstrates:**

XDA\_Ain\_ReadRaw

XDA\_Ain\_ConvertToVolts

### **Wiring Notes:**

To see best results, connect an analog source to the analog input channel specified in the file. This sample will most likely use device 1, input channel 0.

## **dinbuf**

### **Description**

Sets up and runs a buffered digital input on the device/port defined in `dinbufDlg.cpp` and outputs the results to the edit box in the dialog.

### **Demonstrates:**

- XDA\_Din\_Config
- XDA\_Din\_SetClock
- XDA\_Din\_SetSequence
- XDA\_Din\_SetBuffer
- XDA\_Din\_Start
- XDA\_Din\_Check

### **Wiring Notes:**

To see best results, connect a digital signal to the digital input port specified in the file. This sample will most likely use device 1, port 0 for input.

## **dinline**

### **Description**

Reads the state of digital input lines on two ports. Includes a button to re-check the state.

### **Demonstrates:**

XDA\_Din\_Config

XDA\_Din\_Read

### **Wiring Notes:**

To see best results, connect a digital signal to the digital input port specified in the file. This sample will most likely use device 1, ports 0 and 1 for input.

## dioport

### Description

Demonstrates digital input and output by writing a value (most likely an 8-bit integer) to a digital output port and then reading the value from a digital input port. After this, the state of each digital input line on the input port is shown.

### Demonstrates:

XDA\_Din\_Config  
XDA\_Dout\_Config  
XDA\_Dout\_Write  
XDA\_Din\_Read

### Wiring Notes:

Connect lines 0-7 on the digital port specified for output (in dioportDlg.cpp) to lines 0-7 on the digital port specified for input (in dioportDlg.cpp). This sample will most likely use device 1, port 0 for output and device 1, port 1 for input.

## Acquitek Data Acquisition ActiveX Control

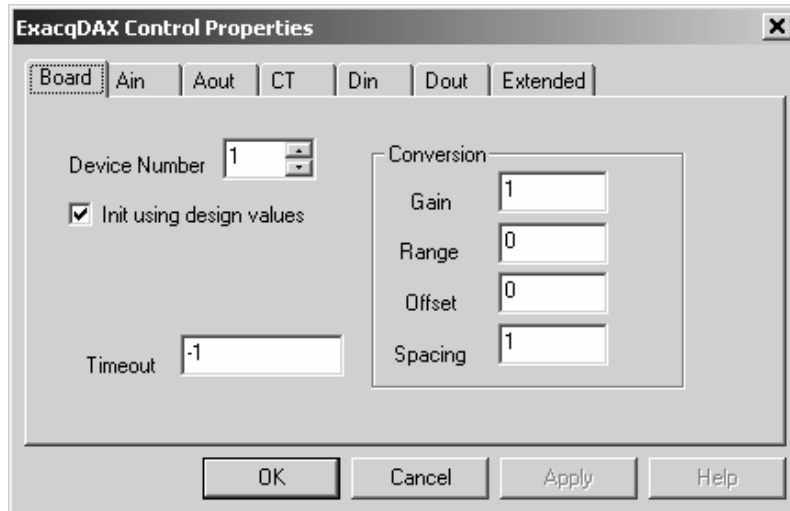
### Overview

The Acquitek Data Acquisition SDK enables full access to Acquitek Technologies hardware from within any environment that provides a mechanism for calling dynamic link libraries (DLLs). Some developers, however, may prefer working with Microsoft ActiveX controls rather than calling the driver DLL API directly. The Acquitek Data Acquisition ActiveX Control (XDADAX) wraps the SDK API and provides the same power through familiar methods and properties.

XDADAX supplies a method for each interface exported by the driver DLL. The parameters and return values are identical. The method names have been changed slightly to conform to ActiveX method naming requirements. To determine the corresponding XDADAX method name given a driver interface name, simply delete the XDA\_ prefix and remove all underscores within the name. For instance, XDA\_Ain\_SetClock() becomes AinSetClock(). Due to these similarities, no attempt will be made here to document usage and parameters for each XDADAX method. Please refer to the Acquitek Data Acquisition SDK manual for detailed information.

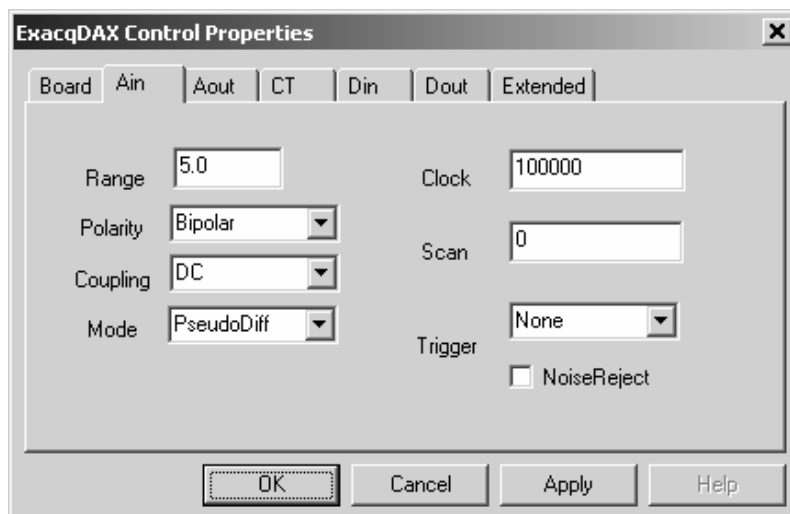
ActiveX makes it possible to simplify application programming in many cases. Rather than calling all of the methods necessary to set input type, sampling rates, and other hardware setup parameters, those details can be specified at design time using XDADAX property pages. Of course, all of those parameters can still be supplied as parameters to the various methods if the values are not known at design time.

## Property Pages



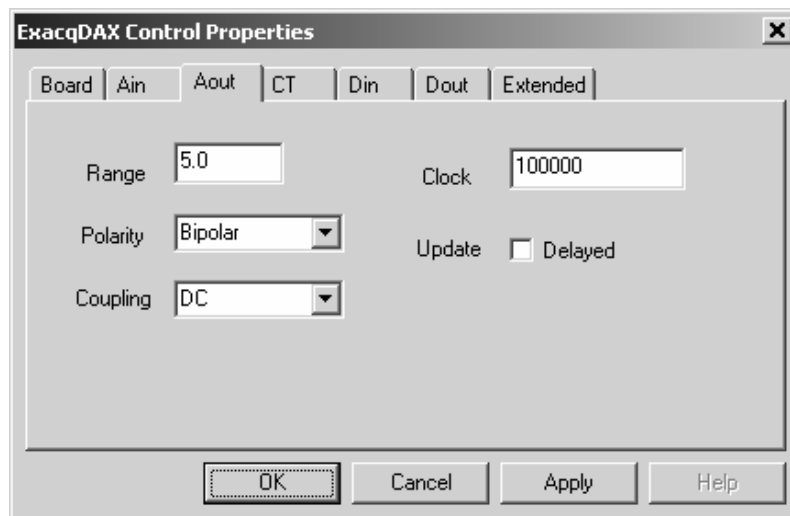
The screenshot shows the 'ExacqDAX Control Properties' dialog box with the 'Board' tab selected. The 'Device Number' is set to 1. The 'Init using design values' checkbox is checked. The 'Timeout' is set to -1. The 'Conversion' section contains: Gain (1), Range (0), Offset (0), and Spacing (1). Buttons for OK, Cancel, Apply, and Help are at the bottom.

The Board properties tab is used to set board level properties at design time. Set the Device Number property to the logical device number as specified in Acquitek Control Center. If the 'Init using design values' checkbox is cleared, none of the properties specified on the XDADAX property pages will be programmed into the hardware when an application using the control is executed. This will allow the application to initialize more quickly, but will also necessitate calling all of the appropriate control methods to set up the hardware. Each of the other properties corresponds to a XDA\_Board\_SetParm parameter documented in the Acquitek Data Acquisition SDK Manual.

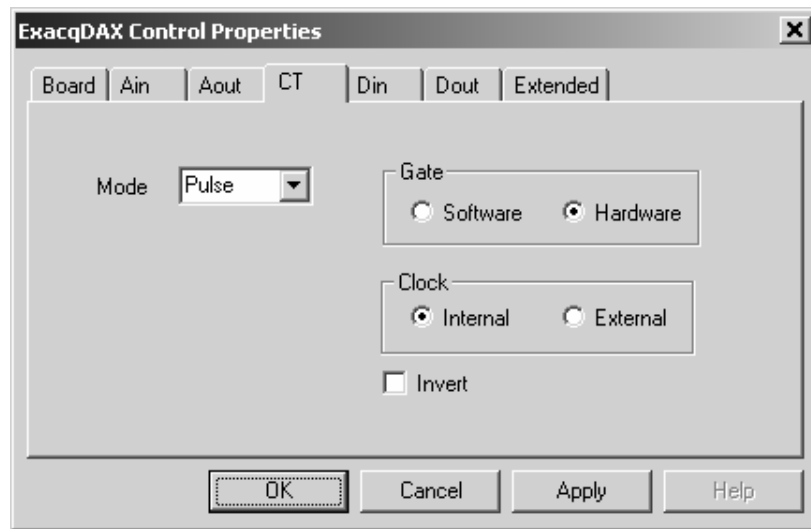


The screenshot shows the 'ExacqDAX Control Properties' dialog box with the 'Ain' tab selected. The 'Range' is 5.0, 'Polarity' is Bipolar, 'Coupling' is DC, and 'Mode' is PseudoDiff. The 'Clock' is 100000, 'Scan' is 0, and 'Trigger' is None. The 'NoiseReject' checkbox is unchecked. Buttons for OK, Cancel, Apply, and Help are at the bottom.

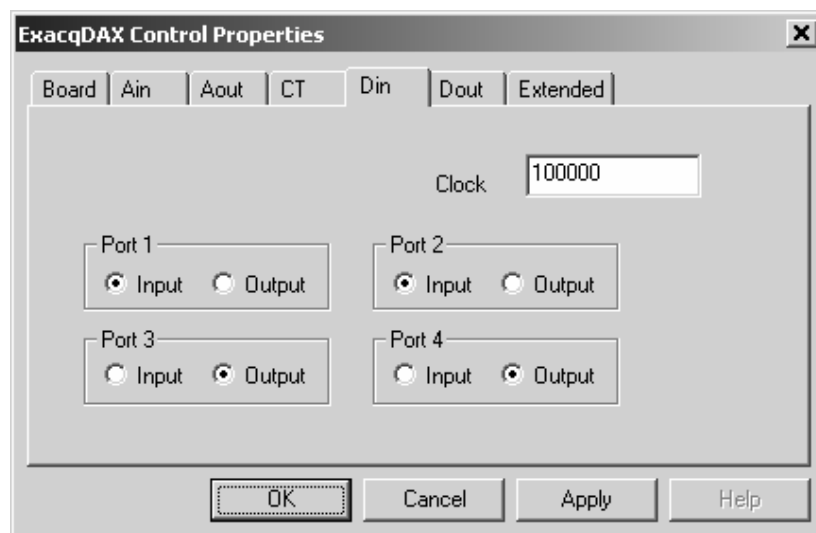
The Ain properties tab is used to set analog input properties at design time. The settings are applied to all analog input channels on the device. If the application requires differing configuration among the channels, the appropriate Ain methods should be called at run time. Each of the properties corresponds to an XDA\_Ain parameter documented in the Acquiretek Data Acquisition SDK Manual.



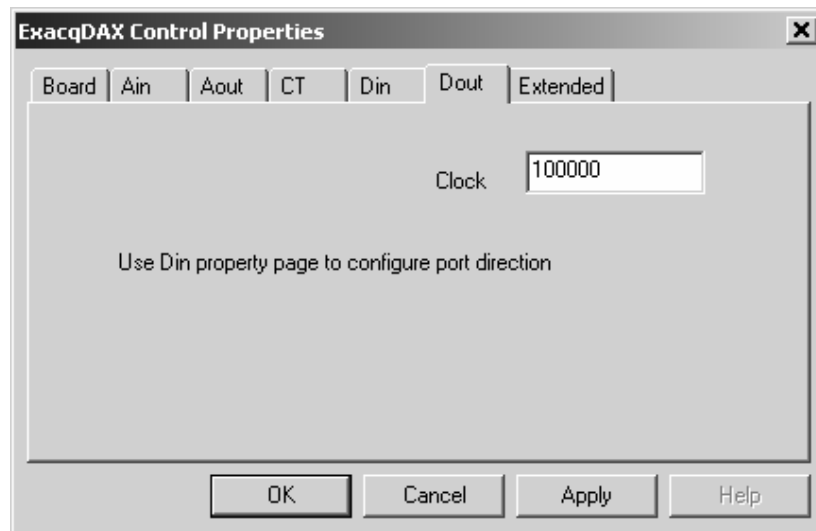
The Aout properties tab is used to set analog output properties at design time. The settings are applied to all analog input channels on the device. . If the application requires differing configuration among the channels, the appropriate Aout methods should be called at run time. Each of the properties corresponds to an XDA\_Aout parameter documented in the Acquiretek Data Acquisition SDK Manual.



The CT properties tab is used to set counter/timer properties at design time. The settings are applied to all counter/timers on the device. . If the application requires differing configuration among the CTs, the appropriate CT methods should be called at run time. Each of the properties corresponds to a XDA\_CT parameter documented in the Acquiretek Data Acquisition SDK Manual.



The Din properties tab is used to set digital input properties at design time. The sampling frequency and IO direction for each port can be specified.



The DOUT properties tab is used to set digital output properties at design time. The output clock frequency can be specified, but the port direction is taken from the Din property page.

## Notes

XDADAX is installed and registered when third party drivers are selected for installation using the Acquiretek Data Acquisition Setup installer. The control requires installation of the Acquiretek Data Acquisition WDM driver and DLL. These files are also installed by default when the Setup installer is run. Additionally, Acquiretek Control Center must be used to assign a logical device number before the hardware can be accessed by XDADAX.

To register XDADAX manually, open a command prompt and type:  
`regsvr32 [AcquiretekInstallDirectory]\thirdparty\XDADAX\XDADAX.ocx`

## Using Acquitek Data Acquisition SDK Signal Processing Features

Most Acquitek Technologies data acquisition boards are based upon a powerful digital signal processor (DSP). This DSP is programmed at the driver level to control the real time data acquisition functionality of the board, and is also available for general-purpose signal processing on the acquired data. Any signal processing performed on the DSP frees the host computer for higher level tasks such as graphical display, networked communications, etc. The Acquitek Data Acquisition SDK (XDADAQ SDK) provides a mechanism to utilize the general-purpose signal processing functions implemented in the DSP. The list of signal processing functions will continue to grow with future XDADAQ SDK releases

### **Block FFT**

#### **Introduction**

The Discrete Fourier Transform (DFT) is defined by the following equation.

$$X(k) = \sum_{n=0}^{N-1} x(n) * \exp(-j2\pi kn/N), \quad k = [0, N-1] \quad (1)$$

It is useful for a variety of applications, such as determining the frequency content of a noisy signal. This is known as spectrum analysis or power spectrum estimation. It is only a power spectrum *estimate* because the input data sequence is finite in length. A typical spectrum analysis application attempts to measure the power in the signal at each frequency. As can be seen from equation 1,  $X(k)$  is the value of the DFT at the normalized frequency  $k$  over the range  $[0, 2\pi)$ . This maps into actual frequency by the equation

$$f = k * F_s / N \quad (2)$$

where:  $F_s$  is the sampling frequency

$X(k)$  is, in general, a complex number. To estimate the power at frequency  $k$ , the magnitude squared of  $X(k)$  must be computed and scaled by the sequence length:

$$P(k) = (1/N) * |X(k)|^2 = (1/N) * [\text{Re}\{X(k)\}^2 + \text{Im}\{X(k)\}^2] \quad (3)$$

where:  $\text{Re}\{x\}$  denotes the real part of the complex number  $x$ , and  
 $\text{Im}\{x\}$  denotes the imaginary part of the complex number  $x$ .

This results in an estimate of the frequency spectrum called the “periodogram”, which is the simplest method of estimating the spectrum from a finite set of data. The factor  $(1/N)$  equates the energy in the time domain with the power in the frequency domain by Parseval’s Theorem:

$$\sum_{n=0}^{N-1} |x(n)|^2 = 1/N * \sum_{k=0}^{N-1} |X(k)|^2 \quad (4)$$

Normalizing (4) by the sequence length yields mean-square power:

$$x_{ms} = 1/N * \sum_{n=0}^{N-1} |x(n)|^2 = 1/N^2 * \sum_{k=0}^{N-1} |X(k)|^2 \quad (5)$$

One of the problems with the basic periodogram results from the discontinuity at the beginning and end of the data sequence. The discontinuity leads to a phenomenon known as “spectral leakage”, in which energy from one frequency appears in the periodogram at other frequencies. This effect can be drastically reduced by the use of window functions. The acquired data is scaled by the window function prior to computing the periodogram, and the result is called a “windowed periodogram”. Examples of the window functions implemented in the XDADAQ SDK are shown in figure 1. Two observations to be made from this figure are: 1) Using a rectangular window is equivalent to working with an unmodified finite length data sequence; 2) The shape of all other window functions is an attempt to minimize the discontinuity at the beginning and end of the data sequence.

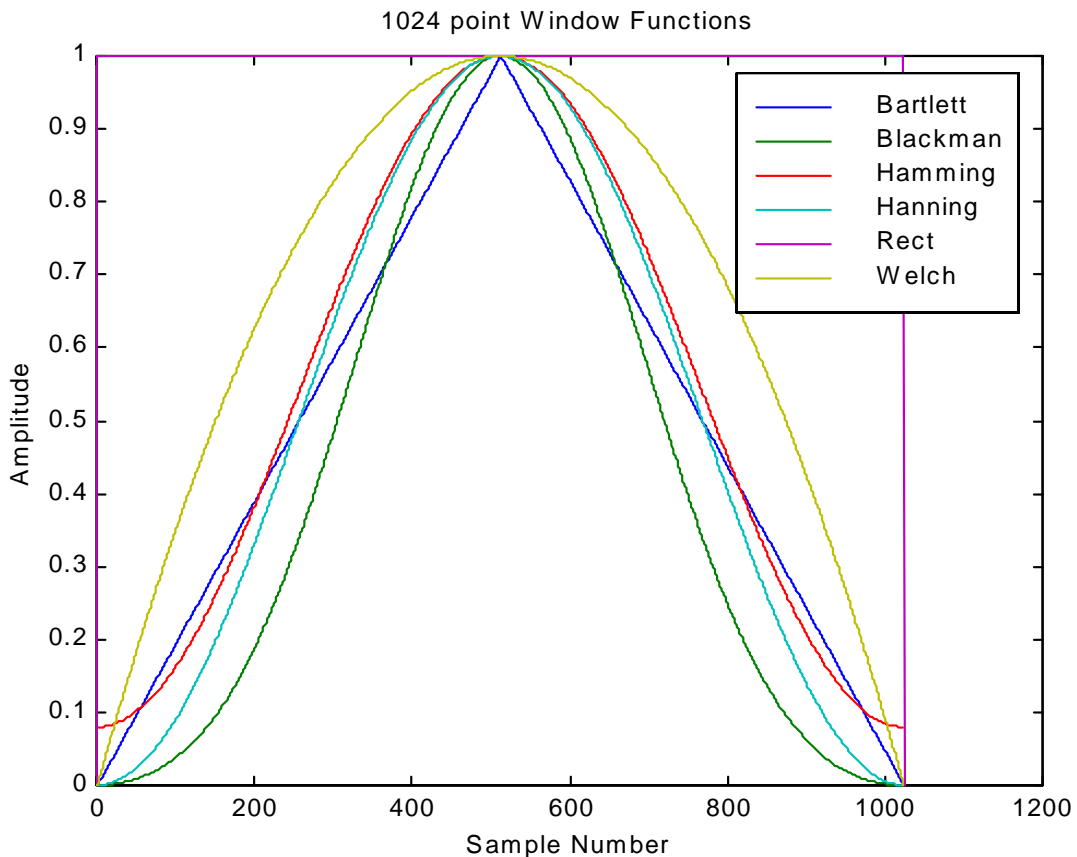


Figure 1 – Window Functions Implemented in XDADAQ SDK

Evaluation of the effectiveness of a given window function is usually done in the frequency domain. The two most common metrics are main lobe width and sideband height. The frequency responses of the window functions depicted in Figure 1 are given below in Figure 2. Note that the frequency response of 32 point windows is shown for viewing ease, but the relative characteristics of the windows are unchanged for any given length. Mathematics tells us that because the data sequence is multiplied by the window in the time domain, the ideal data frequency spectrum is convolved by the window frequency response in the frequency domain. This convolution will distort the ideal spectrum. The narrower the window's frequency domain main lobe, the better the window's resolution or ability to distinguish closely spaced frequencies. The lower the sidebands, the smaller the effects from noise or interference outside the band of interest.

As can be seen from Figure 2, the rectangular window has the narrowest main lobe, but also the highest sidelobes. The sidelobes are the cause of the "spectral leakage" mentioned above in the discussion of the periodogram (unwindowed) spectrum estimator. The Hanning window is a reasonable compromise between main lobe width and out of band rejection. It is the window used by the Acquittek Bench spectrum analyzer.

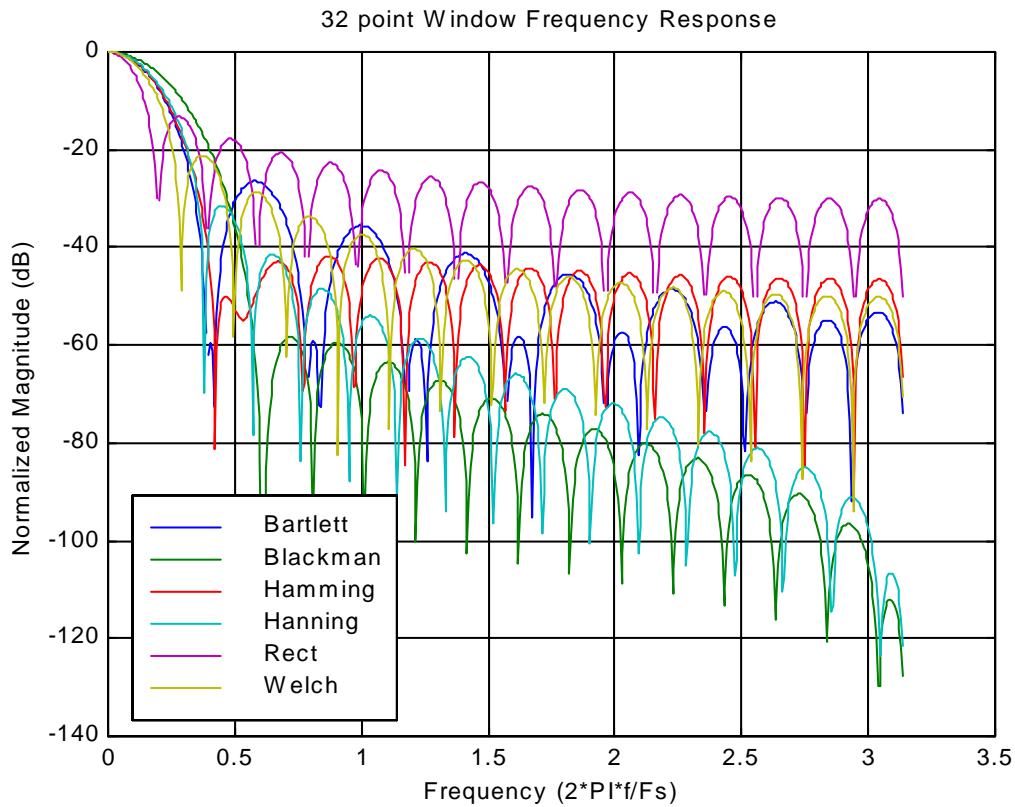


Figure 2 – Frequency Response of XDADAQ SDK Window Functions

Finally, in looking at Figure 1, it is apparent that scaling by the window function will reduce the power in the acquired signal. Therefore, the resultant windowed periodogram must be scaled in order to obtain a calibrated power spectrum estimate. The scale factor is computed as follows:

$$U = N / \sum_{n=0}^{N-1} [w(n)]^2 \quad (6)$$

where: N is the window length, and  
w(n) is the window coefficient at point n.

The scale factors for several windows are computed in Table 1.

	N=16	N=32	N=64	N=128	N=256	N=512	N=1024	N=2048	N=4096
Bartlett	3.2143	3.1000	3.0484	3.0238	3.0118	3.0059	3.0029	3.0015	3.0007
Blackman	3.5019	3.3889	3.3351	3.3088	3.2959	3.2894	3.2862	3.2846	3.2838
Hamming	2.6812	2.5962	2.5556	2.5358	2.5261	2.5212	2.5188	2.5176	2.5170
Hanning	2.5098	2.5859	2.6256	2.6460	2.6563	2.6615	2.6641	2.6654	2.6660
Rectangle	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Welch	2.0000	1.9355	1.9048	1.8898	1.8824	1.8787	1.8768	1.8759	1.8755

Table 1 – Windowed Periodogram Scale Factors

This is a very brief treatment on the usage of the FFT on an Acquitek Technologies data acquisition board. Please refer to a digital signal processing text for additional details on the DFT, computation of the FFT, power spectrum estimation, and windowing. One such text is “Introduction to Digital Signal Processing” by Proakis and Manolakis and published by MacMillan Publishing Company.

## Usage

On applicable Acquitek Technologies data acquisition boards (CM, CH, XM, XH series boards), the DFT of the acquired signal can be computed by the on-board DSP. The DSP uses a Fast Fourier Transform (FFT) algorithm to compute the DFT of a signal. By nature, the FFT algorithm operates on a data size which is a power of 2. Presently, the XDADAQ SDK allows FFTs on blocks of data ranging in size from 16 to 4096 samples ( $2^4$  to  $2^{12}$ ). Requesting FFT sizes outside of this range or which are not a power of 2 will result in an error.

Block FFT mode usage is similar to normal time-domain data capture using the XDADAQ SDK. In both modes, the following commands are used to set up and start the capture. See the documentation on each individual function for details on the calling parameters.

XDA\_Ain\_SetSequence() – Sets up number of channels and voltage range of channels  
 XDA\_Ain\_SetClock() – Sets sampling clock  
 XDA\_Ain\_SetBuffer() – Configures buffer for scatter-gather transfers  
 XDA\_Ain\_Start() – Starts data capture

For Block FFT mode, there are a few additional setup steps and subtleties to note. First, the parameters shown below must be set prior to the call to XDA\_Ain\_Start() in order for Block FFT mode to function as desired:

```
XDA_Ain_SetParm(device, 0, XDA_AIN_PARM_FFT_BLOCK_MODE, 1);
XDA_Ain_SetParm(device, 0, XDA_AIN_PARM_FFT_SIZE, N);
XDA_Ain_SetParm(device, 0, XDA_AIN_PARM_FFT_WINDOW_TYPE, /
  FFT_WINDOW_TYPE_HANNING );
...
XDA_Ain_Start(device, XDA_SYNC, N);
```

See the documentation on the XDA\_Ain\_SetParm() function for a list of the currently defined FFT window types.

Next, the size of the data returned from the FFT calculation must be considered. There are two differences from time-domain capture. The FFT returns 32 bit floating point numbers, designated the f32 type. This maximizes the dynamic range of the resulting transform data. Also, for an N point real input FFT, N complex points are returned as N floating point pairs (i.e. 2N f32 values are returned for an N point FFT). The following pseudocode illustrates this point. For a time-domain capture, the following code would be appropriate for capture of N samples:

```
...
i16    *captureBuffer; // i16 is a 16 bit integer data type
...
captureBuffer = (i16*)malloc(N*sizeof(i16));
XDA_Ain_SetBuffer(dev, N, captureBuffer);
...
XDA_Ain_Start(dev, XDA_SYNC, N);
...
```

In dual channel mode, the N samples would be split between both channels, resulting in a capture of N/2 samples from each channel. For a Block FFT capture, the code is somewhat different:

```
...
f32    *fftData;      // f32 is a 32 bit float
...
XDA_Ain_SetParm(device, 0, XDA_AIN_PARM_FFT_SIZE, N);
fftData = (f32*)malloc(2*N*numberOfChan*sizeof(f32));
XDA_Ain_SetBuffer(dev, 2*N*numberOfChan*sizeof(f32)/sizeof(i16), fftData);
...
XDA_Ain_Start(dev, XDA_SYNC, N*numberOfChan);
...
```

For single channel operation, the code above will cause an N-point FFT to be computed. In dual channel mode, the code above will also cause an N-point FFT to be computed for each channel; therefore, twice as much data space is required.

The underlying math of the FFT is based upon a complex time domain input signal. In both single and dual channel mode, the XDADAQ SDK assumes that the input data on each channel is entirely real and the imaginary part is all zeros for both channels. Therefore, the FFT result is known to be symmetric about the Nyquist frequency  $F_s/2$ . As a result, for performance optimization, the second half of the data (frequencies  $F_s/2$  ..  $F_s$ ) may not contain valid return data. Only the non-redundant portion of the FFT data buffer (the first  $N/2+1$  complex frequency bins from frequencies 0 ..  $F_s/2$ ) is guaranteed to contain valid data.

The format of the returned FFT data buffer after a single channel 1024 point FFT is as follows.

```
f32 fftData[1024*2]; // *2 is because returned data is complex

fftData[0] // Re{X(0)}, where X(n) is DFT of input data sequence
fftData[1] // Im{X(0)}, always zero for real-valued input sequence
fftData[2] // Re{X(1)}
fftData[3] // Im{X(1)}
```

```
...
fftData[1022] // Re{X(511)}
fftData[1023] // Im{X(511)}
fftData[1024] // Re{X(512)}
fftData[1025] // Im{X(512)}, always zero for real-valued input sequence
fftData[1026] // Invalid
...
fftData[2047] // Invalid
```

The format of the returned FFT data buffer after a dual channel 1024 point FFT is as follows.

```
#define NUMCHAN 2
f32 fftData[1024*2* NUMCHAN]; // *2 is because returned data is complex

fftData[0] // Re{X(0)}, where X(n) is DFT of Ch0 input data sequence
fftData[1] // Im{X(0)}, always zero for real-valued input sequence
fftData[2] // Re{X(1)}
fftData[3] // Im{X(1)}
...
fftData[1022] // Re{X(511)}
fftData[1023] // Im{X(511)}
fftData[1024] // Re{X(512)}
fftData[1025] // Im{X(512)}, always zero for real-valued input sequence
fftData[1026] // Invalid
...
fftData[2047] // Invalid
fftData[2048] // Re{Y(0)}, where Y(n) is DFT of Ch1 input data sequence
fftData[2049] // Im{Y(0)}
fftData[2050] // Re{Y(1)}
fftData[2051] // Im{Y(1)}
...
fftData[3070] // Re{Y(511)}
fftData[3071] // Im{Y(511)}
fftData[3072] // Re{Y(512)}
fftData[3073] // Im{Y(512)}, always zero for real-valued input sequence
fftData[3074] // Invalid
...
fftData[4095] // Invalid
```

For an FFT of more than two channels, the format of the returned data is an extension of the dual channel case.

Finally, a note on amplitude scaling of the FFT data for spectrum analysis. As discussed in equations (3) through (6) in the previous section, the FFT magnitude squared can be scaled by the FFT length and the window function scale factor in order to yield a calibrated power spectrum estimate. A final scaling depending upon the voltage range at which the signal was captured must also be performed to achieve calibrated results.

An example best illustrates the point. Consider the case where we input a sine wave with amplitude 3V and frequency 10kHz. We capture a single channel on the +/-5V scale at a 100kHz sampling frequency, and perform a 1024 point FFT using the Hanning window. First, determine the FFT frequency point of interest. Rearranging equation (2) yields:

$$k = N * f / F_s \quad (7)$$

where:  $f = 10\text{kHz}$  (input frequency),  
 $F_s = 100\text{kHz}$  (sampling frequency),  
 $N = 1024$  (points in FFT)

Thus,  $k = 102$

Next, find the power spectrum estimate at that frequency:

$$P(k) = U/N^2 * |X(k)|^2 = U/N^2 * (\text{fftData}[2*k]^2 + \text{fftData}[2*k+1]^2) \quad (8)$$

where:  $k = 102$  (from above),  
 $N = 1024$  (points in FFT),  
 $U = 2.6641$  (from Table 1 for Hanning window of length  $N=1024$ ),  
 $X(k)$  is the FFT result

Thus,  $P(102) = 2.541\text{e-}6 * (\text{fftData}[204]^2 + \text{fftData}[205]^2)$

Finally, find the voltage range normalization factor. It will be assumed that the FFT input is captured on a bipolar range. The A/D converter output (i.e the FFT time domain input) is given by:

$$x(n) = 2^{(B-1)}/V_{\text{max}} * v_{\text{in}}(n\Delta t) \quad (9)$$

where:  $B =$  number of A/D bits  
 $V_{\text{max}} =$  maximum positive input voltage for the selected range

Since the FFT is a linear operation,

$$X(k) = 2^{(B-1)}/V_{\text{max}} * V_{\text{in}}(k) \quad (10)$$

On a CH,  $B=12$ , so the scaling factor on the +/-5V range is  $2^{11}/5 = 409.6$ .

Now, we're nearing the end. The desired result is RMS voltage at a given frequency.

$$\begin{aligned} V'_{\text{rms}}(k) &= (V_{\text{max}}/2^{(B-1)}) * \text{sqrt}(P(k)) \\ &= (V_{\text{max}}/2^{(B-1)}) * (1/N) * \text{sqrt}(U * (\text{fftData}[2*k]^2 + \text{fftData}[2*k+1]^2)) \\ &= 3.89\text{e-}6 * \text{sqrt}(\text{fftData}[2*k]^2 + \text{fftData}[2*k+1]^2) \text{ from the example above.} \end{aligned}$$

Now for a little housekeeping - since this is a real signal, the frequency content is symmetric about DC (0 Hz). The above equation only looked at the positive frequency component, and thus contains only half the actual signal power. Also, due to spectral spreading by the window function main lobe, the power in adjacent frequency points must be summed to yield an accurate result. These corrections can be implemented as follows:

$$V_{\text{rms}}(k) = \text{sqrt}( 2*[(V'_{\text{rms}}[k-1])^2 + (V'_{\text{rms}}[k])^2 + (V'_{\text{rms}}[k+1])^2 ] )$$

## Benchmarks

With XDADAQ SDK Version 1.1 software, the DSP computes a single channel, 1024 point FFT in approximately 750us. This number scales nearly linearly with FFT length, so a 512 point single channel FFT takes approximately 375us, and a 2048 point FFT takes 1500us. The FFT computation time also scales linearly with the number of channels, so a two channel, 1024 point FFT is computed in approximately 1500us.

The window function is computed only once when the window type and/or FFT length are specified. It can take a not insignificant amount of time depending upon the type and length. A 1024 point Hanning window takes 1380us to compute, while a Blackman window takes longer due to the two cosine terms, and a rectangular window takes almost no time. Since the window is only computed once, this computation time is not critical. The time required to scale the input data sequence by the window is included in the FFT computation time benchmarks.

<b>Value</b>	<b>Constant Name</b>	<b>Description</b>
0	XDA_SUCCESS	The function executed successfully.
-1	XDA_ERR_UNSPECIFIED	An unspecified error has occurred.
-9000	XDA_ERR_DEVICE_NUMBER	An invalid device number was specified.
-9001	XDA_ERR_CHANNEL_NUMBER	An invalid channel number was specified.
-9002	XDA_ERR_PORT_NUMBER	An invalid port number was specified.
-9003	XDA_ERR_LINE_NUMBER	An invalid line number was specified.
-9004	XDA_ERR_DIRECTION	An invalid direction value was specified.
-9005	XDA_ERR_INIT	An error occurred during device initialization.
-9006	XDA_ERR_CONFIG_CHANGED	Hardware configuration has changed. Run Acquitek Control Center.
-9007	XDA_ERR_VALUE	An invalid value was specified. This could result from a reserved variable being set to a nonzero value.
-9008	XDA_ERR_MEMORY_FULL	Memory was full, so allocation failed.
-9009	XDA_ERR_UNSUPPORTED	The specified device is not supported by this AcquitekDA version.
-9010	XDA_ERR_RATE	An invalid rate was specified.
-9011	XDA_ERR_RANGE	An invalid range was specified.
-9012	XDA_ERR_INDEX	An invalid index number was specified.
-9013	XDA_ERR_DEVICE_BUSY	The device is in use by another program.
-9014	XDA_ERR_OVERFLOW	An on-board buffer overflow has occurred.
-9015	XDA_ERR_MB_DISABLED	Multi-buffering has been disabled.
-9016	XDA_ERR_COUNT	An invalid count was specified.
-9017	XDA_ERR_NULL_POINTER	A null pointer was passed to the function.
-9018	XDA_ERR_ALIGNMENT	A pointer passed to the function was not aligned on a 32-bit boundary.
-9019	XDA_ERR_MEMORY_LOCK	Memory lock for DMA transfer failed.
-9020	XDA_ERR_SEQUENCE_ORDER	An invalid sequence order was specified.

-9021	XDA_ERR_MODE	An invalid mode was specified.
-9022	XDA_ERR_TIMEOUT	The device timed out.
-9023	XDA_ERR_OVERWRITE	A host buffer for multi-buffer mode was overwritten. Decrease the amount of processing time spent in XDA_Ain_MB_Proc() or after XDA_Ain_MB_Copy().
-9024	XDA_ERR_CT_NUMBER	An invalid counter/timer number was specified.
-9025	XDA_ERR_SA_LOCK	SafeArrayLock or SafeArrayUnlock returned an error. This is returned by XDA_Ain_SetBuffer, XDA_Aout_SetBuffer, XDA_Din_SetBuffer, and XDA_Dout_SetBuffer.
-9026	XDA_ERR_FLAGS	An invalid flags value was specified.
-9027	XDA_ERR_PXI_NOT_ENABLED	The function cannot complete successfully because outputs are not enabled for the necessary lines. You can enable them in Acquitek Control Center.
-9028	XDA_ERR_MEMORY_BUSY	On board memory allocation can not be changed while input or output is active.
-9029	XDA_ERR_MEMORY_SIZE	On board memory allocation size is incorrect.
-9030	XDA_ERR_PCI_THROUGHPUT	In normal capture mode, the PCI bus throughput cannot empty the on-board RAM FIFOs quickly enough, and the FIFO filled. Increasing the RAM FIFO using the XDA_BOARD_PARM_AIN_MEMORY parameter may help.
		In FFT mode, the DSP cannot keep up with the input sample rate, and input data FIFO filled. Lower the sample rate to enable the DSP to keep up.
-9999	XDA_ERR_UNIMPLEMENTED	The function is not implemented.

## Technical Support

Acquitek is committed to providing exceptional technical and engineering support. When you need help with your Acquitek Data Acquisition product, please have the following information available:

- A complete description of the problem, including any error messages or instructions on re-creating the error.
- Your computer configuration, including brand, processor, speed, memory, and other hardware installed.
- Description of any connections to the Acquitek Data Acquisition product.
- Operating System Environment (Windows, Linux, etc).
- Information on the compiler you are using, if applicable.
- Sample code, if applicable.

Technical support can be contacted as follows:

**Acquitek , SAS.**

1 bis rue Marcel Paul

91300 Massy

France

Phone: +33 1 60 13 52 73

Fax: +33 1 60 13 03 68

e-mail: [support@acquitek.com](mailto:support@acquitek.com)

Web: <http://www.acquitek.com>

**Technical Support Hours**

Monday – Friday: 9:00 am – 6:00 pm (GMT +1)

Saturday, Sunday & Holidays: Closed