DAQDriver Programmer's Guide

Version: 1.0.0

Issue Data: Oct 03, 2011

Contents

1 Preface

In addition to the easy-to-use data acquisition software FTezDAQ, Futek enables its users to communicate directly with the USB DAQ modules by calling the DAQDriver. In this way, users are free to write applications according to their particular requirements. This document provides the application programming interfaces (API) for the DAQDriver DLL function library.

If you are looking for examples demonstrating how to make use of these functions, refer to the Support page of the Futek Instruments website. Currently, demo codes for C, C++, C#, VB, LabVIEW and LabWindows/CVI are available.

2 Introduction

The DAQDriver DLL function library is a proprietary interface specifically for Futek USB DAQ devices. This document provides an explanation of the functions available to application developers via the DAQDriver library.

The USB DAQ modules have five types of components: ADC, DAC, Digital Input, Digital Output, and Counter. However, some of the devices don't implement all of them. As a result, some of the functions listed below are supported by all types of the devices, whereas others are supported by a subset of the devices.

All functions are supported on Windows® 2000 and later.

3 Functions

3.1 DAQ_Open

Description:

Open the device and return a handle which will be used for subsequent accesses.

Support:

All devices support this function.

Definition:

DAQ_STATUS DAQ_Open(DAQ_HANDLE *pHandle);

Parameter:

pHandle	Pointer to a variable of type DAQ_HANDLE where the handle will be stored. This handle must be used to access the device.

Return Value:

DAQ_OK if successful, otherwise the return value is a DAQ error code.

3.2 DAQ_SetADCMode

Description:

Set the input mode for ADC (all ADC channels).

Support:

All devices support this function.

Definition:

DAQ_STATUS DAQ_SetADCMode(DAQ_HANDLE handle, DAQ_ADCMODE nMode);

Parameters:

handle Handle of the device.

nMode MODE_DIFFERENTIAL for differential mode, and MODE_SINGLEENDED for single ended mode. The mode specified must be supported by the device.

Return Value:

DAQ_OK if successful, otherwise the return value is a DAQ error code.

Note:

Once the ADC mode has changed, all ADC channels are turned off right away. You need to turn them on before reading from them.

3.3 DAQ_ReadCalibrator

Description:

Get the gain and offset value of certain ADC channel under certain ADC mode for subsequent calibration.

Support:

All devices except DAQ-0311A support this function.

Definition:

DAQ_STATUS DAQ_ReadCalibrator(DAQ_HANDLE handle, DAQ_ADCMODE nMode, unsigned nIndex, double *pGain, double *pOffset);

Parameters:

handle Handle of the device.

nMode MODE_DIFFERENTIAL for differential mode, and MODE_SINGLEENDED for single ended mode. The mode specified must be supported by the device.

nIndex Zero-based index of the ADC channel. The index specified must be smaller than the number of ADC channels available under the ADC mode.

pGain Pointer to a variable of type double where the gain value will be stored.

pOffset Pointer to a variable of type double where the offset value will be stored.

Return Value:

DAQ_OK if successful, otherwise the return value is a DAQ error code.

Note:

The calibration parameters of the same ADC channel under different modes are different.

3.4 DAQ_SetADCConf

Description:

Set the configuration of certain ADC channel: on/off state, and range of the input voltage.

Support:

All devices support this function.

Definition:

DAQ_STATUS DAQ_SetADCConf(DAQ_HANDLE handle, unsigned nIndex, unsigned bOn, DAQ_ADCRANGE nRange);

Parameters:

handle Handle of the device.

nIndex Zero-based index of the ADC channel. The index specified must be smaller than the number of ADC channels available under the current ADC mode.

bOn Zero off; non-zero on.

nRange RANGE_N10000_P10000 for -10~+10V, RANGE_N5000_P5000 for -5~+5V, RANGE_N2500_P2500 for -2.5~+2.5V, and RANGE_N1250_P1250 for -1.25~+1.25V. The range specified must be supported by the device.

Return Value:

DAQ_OK if successful, otherwise the return value is a DAQ error code.

Note:

By default, all ADC channels are off. In order to read from an ADC channel, it must be turned on.

3.5 DAQ_SetADCRate

Description:

Set sampling rate of ADC (all ADC channels).

Support:

All devices support this function.

Definition:

DAQ_STATUS DAQ_SetADCRate(DAQ_HANDLE handle, unsigned long nRate);

Parameters:

handle Handle of the device.

nRate Data sampling rate. It should not exceed the maximum sampling rate of the device. That is 100 for DAQ-0311A, 320 for DAQ-1211(H), DAQ-1411(H) and DAQ-1811, 450 for MF-28, and 1000 for MF-126.

Return Value:

DAQ_OK if successful, otherwise the return value is a DAQ error code.

3.6 DAQ_SetCnterConf

Description:

Set the configuration of certain counter.

Support:

MF-28 and MF-126 support this function.

Definition:

DAQ_STATUS DAQ_SetCnterConf(DAQ_HANDLE handle, unsigned nIndex, unsigned bOn);

Parameters:

handle Handle of the device.

nIndex Zero-based index of the counter. The index specified must be smaller than the number of counters.

bOn Zero off; Non-zero on.

Return Value:

DAQ_OK if successful, otherwise the return value is a DAQ error code.

Note:

By default, all counters are off. In order to read from a counter, it must be turned on. By invoking this function, the number of failing edges counted by the counter is effectively reset to zero.

3.7 DAQ_ReadADC

Description:

Read voltage in volts from certain ADC channel.

Support:

All devices support this function.

Definition:

DAQ_STATUS DAQ_ReadADC(DAQ_HANDLE handle, unsigned nIndex, double *pVolt);

Parameters:

handle Handle of the device.

nIndex Zero-based index of the ADC channel. The index specified must be smaller than the number of ADC channels available under the current ADC mode. And the channel must be on.

pVolt Pointer to a variable of type double where the voltage value will be stored.

Return Value:

DAQ_OK if successful, otherwise the return value is a DAQ error code.

Note:

Make sure you have turned on the ADC channel, before reading from it.

3.8 DAQ_ReadADCEx

Description:

Read voltage values from certain ADC channel.

Support:

Only MF-126 supports this function.

Definition:

DAQ_STATUS WINAPI DAQ_ReadADCEx(DAQ_HANDLE handle, unsigned nIndex, unsigned bTrigger, unsigned long nCnt, unsigned long nItvl, unsigned long *pCnt, double *pVolts);

Parameters:

handle	Handle of the device.
nIndex	Zero-based index of the ADC channel. The index specified must be smaller than the number of ADC channels available under the current ADC mode. And the channel must be on.
bTrigger	Zero for waiting no trigger (sampling right away); Non-zero for waiting till a failing edge on counter 0 occurs.
nCnt	Number of data trying to read. It must be between 1 and 3808.
nItvl	Interval in micro-seconds between sampling. It must be between 25 and 4096. As a result, the sampling rate is between 40K and 244 SPS.
pCnt	A pointer to a variable of type unsigned long where the number of data successfully read will be stored.
pVolts	A pointer to a caller-allocated array of type double where the voltage values will be stored.

Return Value:

DAQ_OK if successful, otherwise the return value is a DAQ error code.

Note:

Make sure the ADC channel has been turned on before reading from it.

3.9 DAQ_ReadADCs

Description:

Read all ADC channels that are on.

Support:

All devices except DAQ-0311A support this function.

Definition:

DAQ_STATUS DAQ_ReadADCs(DAQ_HANDLE handle, unsigned *pIndexes, double *pVolts);

Parameters:

handle Handle of the device.

nIndex Zero-based index of the ADC channel. The index specified must be smaller than the number of ADC channels available under the current ADC mode. And the channel must be on.

pIndexes A pointer to a caller-allocated array of type unsigned int. The array will be used to store index information of the ADC channels corresponding to the voltage values. The first element of the array indicates the number of ADC channels successfully read. The others are indexes of the ADC channels to which the voltage values belong.

pVolts A pointer to a caller-allocated array of type double. The array will be used to store voltage values of all ADC channels that are on.

Return Value:

DAQ_OK if successful, otherwise the return value is a DAQ error code.

Note:

pVolts[0] is the voltage of ADC channel pIndexes[1], pVolts[1] is the voltage of ADC channel pIndexes[2], and so on.

3.10 DAQ_ReadDgt

Description:

Read status from certain digital input channel.

Support:

All devices support this function.

Definition:

DAQ_STATUS DAQ_ReadDgt(DAQ_HANDLE handle, unsigned nIndex, unsigned *pStatus);

Parameters

handle Handle of the device.

nIndex Zero-based index of the digital input channel. The index specified must be smaller than the number of digital input channels.

pStatus A pointer to a variable of type unsigned where the status (1 or 0) of the digital input channel will be stored.

Return Value:

DAQ_OK if successful, otherwise the return value is a DAQ error code.

Note:

The digital input channels are always on.

3.11 DAQ_ReadDgts

Description:

Read all digital input channels.

Support:

All devices except DAQ-0311A support this function.

Definition:

DAQ_STATUS DAQ_ReadDgts(DAQ_HANDLE handle, unsigned *pIndexes, unsigned *pStatuses);

Parameters:

handle Handle of the device.

pIndexes A pointer to a caller-allocated array of type unsigned int. The array will be used to store index information of the digital input channels corresponding to the status values. The first element of the array indicates the number of digital input channels successfully read. The others are indexes of the digital input channels to which the status values belong.

pStatuses A pointer to a caller-allocated array of type unsigned. The array will be used to store digital status values of all digital input channels.

Return Value:

DAQ_OK if successful, otherwise the return value is a DAQ error code.

Note:

pStatuses[0] is the status of digital input channel pIndexes[1], pStatuses[1] is the status of digital input channel pIndexes[2], and so on.

3.12 DAQ_ReadCnter

Description:

Read the number of counted failing edges from certain counter.

Support:

MF-28 and MF-126 support this function.

Definition:

DAQ_STATUS DAQ_ReadCnter(DAQ_HANDLE handle, unsigned nIndex, unsigned long *pCounted);

Parameters:

handle	Handle of the device.
nIndex	Zero-based index of the counter. The index specified must be smaller than the number of counters.
pCounted	A pointer to a variable of type unsigned long where the number of failing edges counted by the counter will be stored.

Return Value:

DAQ_OK if successful, otherwise the return value is a DAQ error code.

Note:

Make sure you have turned on the counter, before reading from it.

3.13 DAQ_ReadCnters

Description:

Read all counters that are on.

Support:

MF-28 and MF-126 support this function.

Definition:

DAQ_STATUS DAQ_ReadCnters(DAQ_HANDLE handle, unsigned *pIndexes, unsigned long *pCounteds);

Parameters:

handle Handle of the device.

pIndexes A pointer to a caller-allocated array of type unsigned int. The array will be used to store index information of the counters corresponding to the counted values. The first element of the array indicates the number of counters successfully read. The others are indexes of the counters to which the counted values belong.

pCounteds A pointer to a caller-allocated array of type unsigned long. The array will be used to store counted values of all counters that are on.

Return Value:

DAQ_OK if successful, otherwise the return value is a DAQ error code.

Note:

pCounteds[0] is the number of failing edges counted by counter pIndexes[1], pCounteds[1] is the number of failing edges counted by counter pIndexes[2], and so on.

3.14 DAQ_ReadAll

Description:

Read from all ADC channels that are on, all digital input channels and all counters that are on.

Support:

All devices support this function.

Definition:

DAQ_STATUS DAQ_ReadAll(DAQ_HANDLE handle, unsigned *pAIndexes, double *pVolts, unsigned *pDIndexes, unsigned *pStatuses, unsigned *pCIndexes, unsigned long *pCounteds);

Parameters:

handle Handle of the device.

pAIndexes A pointer to a caller-allocated array of type unsigned int. The array will be used to store index information of the ADC channels corresponding to the voltage values. The first element of the array indicates the number of ADC channels successfully

read. The others are indexes of the ADC channels to which the voltage values belong.

pVolts A pointer to a caller-allocated array of type double. The array will be used to store voltage values of all ADC channels that are on.

pDIndexes A pointer to a caller-allocated array of type unsigned int. The array will be used to store index information of the digital input channels corresponding to the status values. The first element of the array indicates the number of digital input channels successfully read. The others are indexes of the digital input channels to which the status values belong.

pStatuses A pointer to a caller-allocated array of type unsigned. The array will be used to store digital status values of all digital input channels.

pCIndexes A pointer to a caller-allocated array of type unsigned int. The array will be used to store index information of the counters corresponding to the counted values. The first element of the array indicates the number of counters successfully read. The others are indexes of the counters to which the counted values belong.

pCounteds A pointer to a caller-allocated array of type unsigned long. The array will be used to store counted values of all counters that are on.

Return Value:

DAQ_OK if successful, otherwise the return value is a DAQ error code.

Note

pVolts[0] is the voltage of ADC channel pIndexes[1], pVolts[1] is the voltage of ADC channel pIndexes[2], and so on.

pStatuses[0] is the status of digital input channel pIndexes[1], pStatuses[1] is the status of digital input channel pIndexes[2], and so on.

pCounteds[0] is the number of failing edges counted by counter pIndexes[1], pCounteds[1] is the number of failing edges counted by counter pIndexes[2], and so on.

3.15 DAQ_WriteDACVolt

Description:

Write voltage to certain DAC channel.

Support:

MF-28 and MF-126 support this function.

Definition:

DAQ_STATUS DAQ_WriteDACVolt(DAQ_HANDLE handle, unsigned nIndex, unsigned short nValue);

Parameters:

handle Handle of the device.

nIndex Zero-based index of the DAC channel. The index specified must be smaller than the number of DAC channels.

nValue 0 for 0 volts, and 65535 for max voltage.

Return Value:

DAQ_OK if successful, otherwise the return value is a DAQ error code.

Note:

For MF-28 the maximum output voltage is 4 volts. For MF-126, it's 2.5 volts.

3.16 DAQ_WriteDACWave

Description:

Write waveform to certain DAC channel.

Support:

MF-28 and MF-126 support this function.

Definition:

DAQ_STATUS WINAPI DAQ_WriteDACWave(DAQ_HANDLE handle, unsigned nIndex, unsigned long nItvl, DAQ_DACWAVE nWave);

Parameters:

handle Handle of the device.

nIndex Zero-based index of the DAC channel. The index specified must be smaller than the number of DAC channels.

nItvl Period of the wave in micro-seconds. It mustn't be smaller than 1000. As a result, the maximum frequency of DAC is 1000Hz.

nWave SINE_WAVE for sine wave, TRIANGLE_WAVE for triangle wave, and SAWTOOTH_WAVE for sawtooth wave.

Return Value:

DAQ_OK if successful, otherwise the return value is a DAQ error code.

3.17 DAQ_WriteDgtStatus

Description:

Write status to certain digital output channel.

Support:

All devices support this function.

Definition:

DAQ_STATUS DAQ_WriteDgtStatus(DAQ_HANDLE handle, unsigned nIndex, unsigned nStatus);

Parameters:

handle Handle of the device.

nIndex Zero-based index of the digital output channel. The index specified must be smaller than the number of digital output channels.

nStatus Zero for low; Non-zero for high.

Return Value:

DAQ_OK if successful, otherwise the return value is a DAQ error code.

3.18 DAQ_WriteDgtPWM

Description:

Write PWM wave to certain digital output channel.

Support:

DAQ-1411(H), DAQ-1811, MF-28 and MF-126 support this function.

Definition:

DAQ_STATUS DAQ_WriteDgtPWM(DAQ_HANDLE handle, unsigned nIndex, unsigned long nItvl, unsigned nDuty);

Parameters:

handle Handle of the device.

nIndex Zero-based index of the digital output channel. The index specified must be smaller than the number of digital output channels.

nItvl Period of the wave in micro-seconds. For DAQ-1411(H), DAQ-1811 and MF-28, it mustn't be smaller than 1000. As a result, the maximum frequency of PWM is 1000Hz for these devices. The above restriction applies to the second digital output channel of MF-126. However, for the other digital output channels, it must be between 1 and 4096.

nDuty Duty ratio of the frequency. It must be between 1 and 99.

Return Value:

DAQ_OK if successful, otherwise the return value is a DAQ error code.

3.19 DAQ_Close

Description:

Close an open device.

Support:

All devices support this function.

Definition:

DAQ_STATUS DAQ_Close(DAQ_HANDLE handle);

Parameter:

handle Handle of the device.

Return Value:

DAQ_OK if successful, otherwise the return value is a DAQ error code.

4 Functions Summary

The functions and supported devices are summarized below:

	Function	DAQ-0311A	DAQ-1211(H)	DAQ-1411(H)	DAQ-1811	MF-28	MF-126
1	DAQ_Open	√	√	√	√	√	√
2	DAQ_SetADCMode	√	√	√	√	√	√
3	DAQ_ReadCalibrator		√	√	√	√	√
4	DAQ_SetADCConf	√	√	√	√	√	√
5	DAQ_SetADCRate	√	√	√	√	√	√
6	DAQ_SetCnterConf					√	√
7	DAQ_ReadADC	√	√	√	√	√	√
8	DAQ_ReadADCEx						√
9	DAQ_ReadADCs		√	√	√	√	√
10	DAQ_ReadDgt	√	√	√	√	√	√
11	DAQ_ReadDgts		√	√	√	√	√
12	DAQ_ReadCnter					√	√
13	DAQ_ReadCnters					√	√
14	DAQ_ReadAll	√	√	√	√	√	√
15	DAQ_WriteDACVolt					√	√
16	DAQ_WriteDACWave					√	√
17	DAQ_WriteDgtStatus	√	√	√	√	√	√
18	DAQ_WriteDgtPWM			√	√	√	√
19	DAQ_Close	√	√	√	√	√	√