PCI-PID01 API Manual

Windows, **Windows2000**, **Windows NT** and **Windows XP** are trademarks of **Microsoft**. We acknowledge that the trademarks or service names of all other organizations mentioned in this document as their own property.

Information furnished by DAQ system is believed to be accurate and reliable. However, no responsibility is assumed by DAQ system for its use, nor for any infringements of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or copyrights of DAQ system.

The information in this document is subject to change without notice and no part of this document may be copied or reproduced without the prior written consent.

Board Level API Functions

Overview

BOOL OpenDAQDevice (int nModel, int nBoard)

BOOL CloseDAQDevice (int nModel, int nBoard)

BOOL GetBoardVersion (int nModel, int nBoard, int *version)

OpenDAQDevice

The board should make sure to register on the system. Function can be called only on board the normal registration.

BOOL OpenDAQDevice (int nModel, int nBoard)

Parameters:

nModel : Select the model number.

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch. (0 ~ 3)

Return Value:

If the function fail to reset, it returns “FALSE”. If the function succeed to reset, it returns “TRUE”.

CloseDAQDevice

The CloseDAQDevice function closes all opened devices (PCI-AIO series boards). If use of device is finished, it can certainly close a device for making it other programs so as usable.

BOOL CloseDAQDevice (int nModel, int nBoard)

Parameters:

nModel : Select the model number.

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch. (0 ~ 3)

Return Value:

If the function fail to reset, it returns “FALSE”. If the function succeed to reset, it returns “TRUE”.

GetBoardVersion

Get the hardware version of the device.

BOOL GetBoardVersion (int nModel, int nBoard, int *version)

Parameters:

nModel : Select the model number.

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch. (0 ~ 3)

*version : It's a pointer of variable to receive the version information. Norma case represents a positive integer value.

Return Value:

If the function fail to reset, it returns "FALSE". If the function succeed to reset, it returns "TRUE".

PLL API Functions

Overview

BOOL PLL_SetClock (int nModel, int nBoard, int dwVal)

BOOL PLL_GetClock (int nModel, int nBoard, int* dwVal)

PLL_SetClock

Set up the programmable clock generator output frequency for AD data acquisition.

BOOL PLL_SetClock (int nModel, int nBoard, int dwVal)

Parameters:

nModel : Select the model number.

nBoard : It informs a board number at currently equipped system.

The board number set up by DIP switch. (0 ~ 3)

dwVal : Fill in the desired frequency value. The range is 1,040~67,000,000hz.

Return Value:

If the function fail to reset, it returns “FALSE”. If the function succeed to reset, it returns “TRUE”.

PLL_GetClock

Heck the programmable clock generator output frequency.

BOOL PLL_GetClock (int nModel, int nBoard, int* dwVal)

Parameters:

nModel : Select the model number.

nBoard : It informs a board number at currently equipped system.

The board number set up by DIP switch. (0 ~ 3)

*dwVal : Frequency value is set.

Return Value:

If the function fail to reset, it returns “FALSE”. If the function succeed to reset, it returns “TRUE”.

DAC API Functions

Overview

BOOL DAC_WaveGen (int nModel, int nBoard, int nChannel, int nMode, float fFreq, float peak, float offset, int *dwBuf)

BOOL DAC_SetFrequency (int nModel, int nBoard, int nChannel, float fFreq)

DAC_WaveGen

DAC output signal waveform data is generated.

BOOL DAC_WaveGen (int nModel, int nBoard, int nChannel, int nMode, float fFreq, float peak, float offset,int *dwBuf)

Parameters:

nModel : Select the model number.

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch. (0 ~ 3)

nChannel : Current device does not make sense, “0” is set.

nMode : Determine the shape of the output waveform. According to the value of the waveform is as follows.

Value	Waveform
0	Sign wave
1	Saw Wavw
2	Triangular Wave
3	Square Wavw
4	DC
5	User-defined

fFreq : Fill in the output frequency of waveform. 0 < fFreq ≤ 1,000.

peak : Peak Value. 0 < peak ≤ 10.

offset : DC Signal Components.

*dwVal : User-defined signal. Requires a sample of 1,000.

Return Value:

If the function fail to reset, it returns “FALSE”. If the function succeed to reset, it returns “TRUE”.

DAC_SetFrequency

Set the DAC output frequency.

BOOL DAC_SetFrequency (int nModel, int nBoard, int nChannel, float fFreq)

Parameters:

nModel : Select the model number.

nBoard : It informs a board number at currently equipped system.

The board number set up by DIP switch. (0 ~ 3)

nChannel : Current device does not make sense, “0” is set..

fFreq : Writes the waveform of the output frequency.

Return Value:

If the function fail to reset, it returns “FALSE”. If the function succeed to reset, it returns “TRUE”.

DAC_GetCycle

Check the how many cycles DAC signal output.

BOOL DAC_GetCycle (int nModel, int nBoard, int nChannel, int *nCycle)

Parameters:

nModel : Select the model number.

nBoard : It informs a board number at currently equipped system.

The board number set up by DIP switch. (0 ~ 3)

nChannel : Current device does not make sense, “0” is set.

*nCycle : Buffer pointer that output the number to be stored. The maximum value is 16,777,215.

Return Value:

If the function fail to reset, it returns “FALSE”. If the function succeed to reset, it returns “TRUE”.

DAC_ClearCycle

Initialize the saved value of DAC output cycle numbers.

BOOL DAC_ClearCycle (int nModel, int nBoard, int nChannel)

Parameters:

nModel : Select the model number.

nBoard : It informs a board number at currently equipped system.

The board number set up by DIP switch. (0 ~ 3)

nChannel : Current device does not make sense, “0” is set.

Return Value:

If the function fail to reset, it returns “FALSE”. If the function succeed to reset, it returns “TRUE”.

ADC API Functions

Overview

Int **ADC_Read (int nModel, int nBoard, int nChannel, int nRead, int *data)**

BOOL **ADC_Reset (int nModel, int nBoard, int nChannel)**

BOOL **ADC_ClockSelect (int nModel, int nBoard, int nChannel, int nSelect)**

BOOL **ADC_SetSampleRate (int nModel, int nBoard, int nChannel, int nSampleRate)**

ADC_Read

Get the AD converted data.

Int **ADC_Read (int nModel, int nBoard, int nChannel, int nRead, int *data)**

Parameters:

nModel : Select the model number.

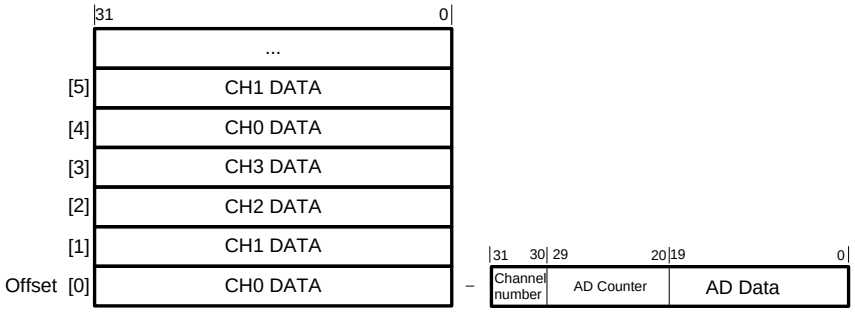
nBoard : It informs a board number at currently equipped system.

The board number set up by DIP switch. (0 ~ 3)

nChannel : Current device does not make sense, “0” is set.

nRead : Set the number of data to be read.

*data : Buffer pointer that AD converted data to be stored.



[Data Order] **[Data Format]**

Return Value:

Return the number of acquired data. Return value is less than or equal to the nRead number.

ADC_Reset

Initialize AD conversion function.

BOOL ADC_Reset (int nModel, int nBoard, int nChannel)

Parameters:

nModel : Select the model number.

nBoard : It informs a board number at currently equipped system.

The board number set up by DIP switch. (0 ~ 3)

nChannel : Current device does not make sense, “0” is set.

Return Value:

If the function fail to reset, it returns “FALSE”. If the function succeed to reset, it returns “TRUE”.

ADC_ClockSelect

Select the clock source of AD converter.

BOOL ADC_ClockSelect (int nModel, int nBoard, int nChannel, int nSelect)

Parameters:

nModel : Select the model number.

nBoard : It informs a board number at currently equipped system.

The board number set up by DIP switch. (0 ~ 3)

nChannel : Current device does not make sense, “0” is set.

nSelect : If the value is “0”, use the 40Mhz OSC. If the value is “1”, use the programmable clock.

Return Value:

If the function fail to reset, it returns “FALSE”. If the function succeed to reset, it returns “TRUE”.

ADC_SetSampleRate

Set the sampling frequency of AD converter.

BOOL ADC_SetSampleRate (int nModel, int nBoard, int nChannel, int nSampleRate)

Parameters:

nModel : Select the model number.

nBoard : It informs a board number at currently equipped system.

The board number set up by DIP switch. (0 ~ 3)

nChannel : Current device does not make sense, “0” is set.

nSampleRate : Enter the sampling frequency used. 5 ≤ nSampleRate ≤ 1,000.

Return Value:

If the function fail to reset, it returns “FALSE”. If the function succeed to reset, it returns “TRUE”.

AMP API Functions

Overview

BOOL AMP_SetGain(int nModel, int nBoard, int nChannel, int nItem, int nGain)

BOOL AMP_SetEnable(int nModel, int nBoard, int nChannel,
int nItem, BOOL bEnable)

BOOL AMP_SetFeedback(int nModel, int nBoard, int nChannel, int nSelect)

AMP_SetGain

Set the Gain of Proportional, Integral, Differential, and dithering signal.

BOOL AMP_SetGain (int nModel, int nBoard, int nChannel, int nItem, int nGain)

Parameters:

nModel : Select the model number.

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch. (0 ~ 3)

nChannel : Current device does not make sense, “0” is set.

nItem : Select the signal. “0” is Proportional signal, “1” is Integral signal, “2” is Differential signal, “3” is Dithering signal.

nGain : Set the Gain of Amp. 0 ≤ nGain ≤ 255.

Return Value:

If the function fail to reset, it returns “FALSE”. If the function succeed to reset, it returns “TRUE”.

AMP_SetEnable

Select the Proportional, Integral, Derivative, and the use of dithering control.

BOOL AMP_SetEnable (int nModel, int nBoard, int nChannel, int nItem, BOOL bEnable)

Parameters:

nModel : Select the model number.

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch. (0 ~ 3)

nChannel : Current device does not make sense, “0” is set.

nItem : Select the signal. “0” is Proportional signal, “1” is Integral signal, “2” is Differential signal, “3” is Dithering signal.

bEnable: Control Flag. “0” : prohibited use, “1” : use available.

Return Value:

If the function fail to reset, it returns “FALSE”. If the function succeed to reset, it returns “TRUE”.

AMP_SetFeedback

Select the external analog input signal that applied to the feedback control.(AIN0, AIN1)

BOOL AMP_SetFeedback (int nModel, int nBoard, int nChannel, int nSelect)

Parameters:

nModel : Select the model number.

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch. (0 ~ 3)

nChannel : Current device does not make sense, “0” is set.

nSelect : Select the external signal. “0” : AIN0, “1” : AIN1.

Return Value:

If the function fail to reset, it returns “FALSE”. If the function succeed to reset, it returns “TRUE”.

DIO API Functions

Overview

BOOL DIO_Set(int nModel, int nBoard, int nOutput)

BOOL DIO_Get(int nModel, int nBoard, int *nInput)

DIO_Set

Set the Digital Output(DO).

BOOL DIO_Set (int nModel, int nBoard, int nOutput)

Parameters:

nModel : Select the model number.

nBoard : It informs a board number at currently equipped system.

The board number set up by DIP switch. (0 ~ 3)

nOutput: Set the output value.

Return Value:

If the function fail to reset, it returns “FALSE”. If the function succeed to reset, it returns “TRUE”.

DIO_Get

Heck the Digital Input(DI).

BOOL DIO_Get (int nModel, int nBoard, int *nInput)

Parameters:

nModel : Select the model number.

nBoard : It informs a board number at currently equipped system.

The board number set up by DIP switch. (0 ~ 3)

*nInput: Get the input value.

Return Value:

If the function fail to reset, it returns “FALSE”. If the function succeed to reset, it returns “TRUE”.

ENCODER API Functions

Overview

BOOL ENC_Read(int nModel, int nBoard, int nChannel, int *nApulse, int *nBpulse, int *nZpulse)

ENC_Read

Check the value of pulse counter for A, B, Z 3-phase on the Encoder.

BOOL ENC_Read (int nModel, int nBoard, int nChannel, int *nApulse, int *nBpulse, int *nZpulse)

Parameters:

nModel : Select the model number.

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch. (0 ~ 3)

nChannel : Current device does not make sense, “0” is set.

*nApulse : With the A-phase pulse count value is a pointer variable.

*nBpulse : With the B-phase pulse count value is a pointer variable.

*nZpulse : With the Z-phase pulse count value is a pointer variable.

Return Value:

If the function fail to reset, it returns “FALSE”. If the function succeed to reset, it returns “TRUE”.

PWM API Functions

Overview

BOOL PWM_Enable (int nModel, int nBoard, int nChannel, int nDirection, BOOL bEnable)

PWM_Enable

Check the value of pulse counter for A, B, Z 3-phase on the Encoder.

BOOL PWM_Enable (int nModel, int nBoard, int nChannel, int nDirection, BOOL bEnable)

Parameters:

nModel : Select the model number.

nBoard : It informs a board number at currently equipped system.

The board number set up by DIP switch. (0 ~ 3)

nChannel : Current device does not make sense, “0” is set.

nDirection : Set the direction. “0” : CW(ClockWise), “1” CCW(CounterClockWise).

bEnable : Flag to control the signal generator. During operation is “TRUE”, When Stop is “FALSE”.

Return Value:

If the function fail to reset, it returns “FALSE”. If the function succeed to reset, it returns “TRUE”.

