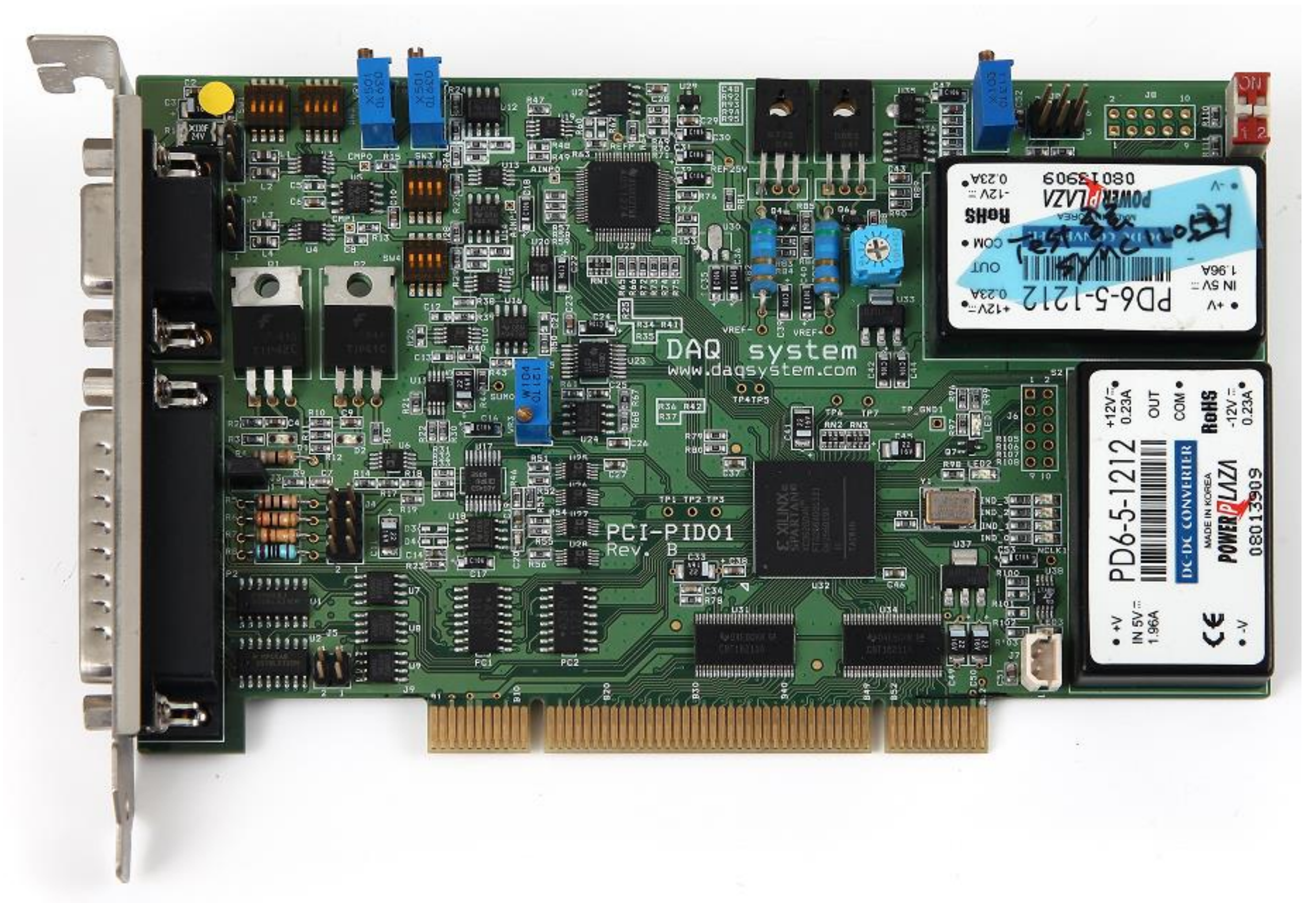
PCI-PID01

API Manual

Version 1.0e



Contents

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

This function checks whether the board is registered in the system.
Functions can be called only for normally registered boards.

BOOL **OpenDAQDevice (int nModel, int nBoard)**

Parameters:

nModel : Write down the model number. from 0 to 3
nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

Return Value:

If the function call fails, “FALSE” is returned.
If the function call succeeds, “TRUE” is returned.

CloseDAQDevice

This function cancels the use registration of the device. When the use of the device is finished, the corresponding function must be called so that other programs can use it.

BOOL **CloseDAQDevice (int nModel, int nBoard)**

Parameters:

nModel : Write down the model number. from 0 to 3
nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

Return Value:

If the function call fails, “FALSE” is returned.
If the function call succeeds, “TRUE” is returned.

GetBoardVersion

This function gets the hardware version information of the device.

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

Parameters:

nModel : Write down the model number. from 0 to 3

nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

*version : A pointer to a variable to receive version information.
In normal cases, it represents a positive integer value.

Return Value:

If the function call fails, "FALSE" is returned.

If the function call succeeds, "TRUE" is returned.

PLL API Functions

Overview

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

PLL_SetClock

This function sets the programmable clock generator output frequency for AD data acquisition.

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

Parameters:

nModel : Write down the model number. from 0 to 3

nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

dwVal : Enter the desired frequency value.
The range of values is 1,040~67,000,000hz.

Return Value:

If the function call fails, "FALSE" is returned.

If the function call succeeds, "TRUE" is returned.

PLL_GetClock

This function checks the programmable clock generator output frequency.

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

Parameters:

nModel : Write down the model number. from 0 to 3

nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

*dwVal : It is the set frequency value.

Return Value:

If the function call fails, "FALSE" is returned.

If the function call succeeds, "TRUE" is returned.

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

DAC_WaveGen

This function generates the DAC data output signal waveform.

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

Parameters:

nModel : Write down the model number. from 0 to 3

nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

nChannel : Since it has no meaning for the current device, "0" is set.

nMode : Determines the shape of the output waveform.
The waveform according to the applied value is as follows.

Value	Waveform
0	Sine wave
1	Sawtooth wave
2	Triangle wave
3	Square wave
4	DC
5	Custom

fFreq : Write the output frequency of the waveform. $0 < fFreq \le 1,000$.

peak : The peak value. $0 < peak \le 10$.

offset : DC signal component.

*dwVal : User-defined signal. 1,000 samples are required.

Return Value:

If the function call fails, "FALSE" is returned.

If the function call succeeds, "TRUE" is returned.

DAC_SetFrequency

This function sets the DAC output frequency.

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

Parameters:

nModel : Write down the model number. from 0 to 3

nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

nChannel : Since it has no meaning for the current device, "0" is set.

fFreq : Write the output frequency of the waveform.

Return Value:

If the function call fails, "FALSE" is returned.

If the function call succeeds, "TRUE" is returned.

DAC_GetCycle

This function checks how many cycles the DAC signal is output.

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

Parameters:

nModel : Write down the model number. from 0 to 3

nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

nChannel : Since it has no meaning for the current device, "0" is set.

*nCycle : A pointer to the buffer where the output count will be stored.
The maximum value is 16,777,215.

Return Value:

If the function call fails, "FALSE" is returned.

If the function call succeeds, "TRUE" is returned.

DAC_ClearCycle

This function initializes the DAC output cycle count stored value.

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

Parameters:

nModel : Write down the model number. from 0 to 3

nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

nChannel : Since it has no meaning for the current device, “0” is set.

Return Value:

If the function call fails, “FALSE” is returned.

If the function call succeeds, “TRUE” is returned.

ADC(Analog to Digital Convertor) 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

This function obtains AD-converted data.

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

Parameters:

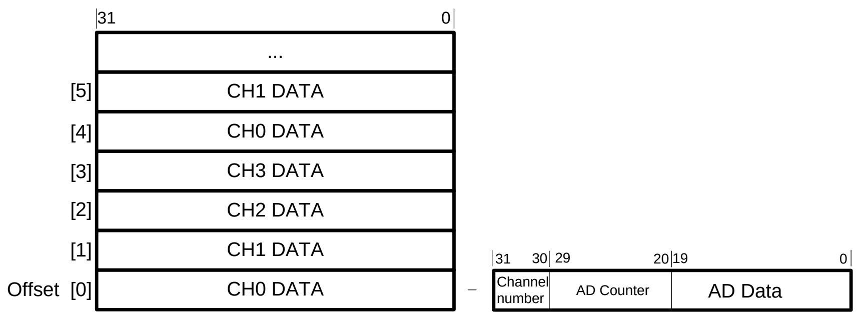
nModel : Write down the model number. from 0 to 3

nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

nChannel : Since it has no meaning for the current device, "0" is set.

nRead : Set the number of data to read.

*data : Variable pointer where AD converted data will be saved.



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

Return Value:

It returns the number of acquired data. The return value is less than or equal to the number of nReads.

ADC_Reset

This function initializes the AD conversion function.

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

Parameters:

nModel : Write down the model number. from 0 to 3

nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

nChannel : Since it has no meaning for the current device, “0” is set.

Return Value:

If the function call fails, “FALSE” is returned.

If the function call succeeds, “TRUE” is returned.

ADC_ClockSelect

This function selects the clock source of the AD converter.

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

Parameters:

nModel : Write down the model number. from 0 to 3

nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

nChannel : Since it has no meaning for the current device, “0” is set.

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

Return Value:

If the function call fails, “FALSE” is returned.

If the function call succeeds, “TRUE” is returned.

ADC_SetSampleRate

This function sets the sampling frequency of the AD converter.

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

Parameters:

nModel : Write down the model number. from 0 to 3

nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

nChannel : Since it has no meaning for the current device, “0” is set.

nSampleRate : Enter the sampling frequency to be used.
$5 \leq$ nSampleRate $\leq$ 1,000.

Return Value:

If the function call fails, “FALSE” is returned.

If the function call succeeds, “TRUE” is returned.

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

This function sets the gain of the signal that has been proportional, integrated, differentiated and dithered.

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

Parameters:

nModel : Write down the model number. from 0 to 3

nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

nChannel : Since it has no meaning for the current device, "0" is set.

nItem : Select a signal.
"0" proportional signal
"1" integral signal
"2" differential signal
"3" dithering signal

nGain : Set the gain value of Amp.
$0 \le nGain \le 255$.

Return Value:

If the function call fails, "FALSE" is returned.
If the function call succeeds, "TRUE" is returned.

AMP_SetEnable

This function chooses to use proportional, integral, derivative and dithering control.

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

Parameters:

nModel : Write down the model number. from 0 to 3

nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

nChannel : Since it has no meaning for the current device, "0" is set.

nItem : Select a signal.
"0" proportional signal
"1" integral signal
"2" differential signal
"3" dithering signal

bEnable : Control flag.
"0" is not used,
"1" can be used.

Return Value:

If the function call fails, "FALSE" is returned.
If the function call succeeds, "TRUE" is returned.

AMP_SetFeedback

This function selects an external analog input signal to be applied to the feedback control (AIN0, AIN1).

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

Parameters:

nModel : Write down the model number. from 0 to 3

nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

nChannel : Since it has no meaning for the current device, "0" is set.

nSelect : Select an external signal. "0" AIN0, "1" AIN1.

Return Value:

If the function call fails, "FALSE" is returned.
If the function call succeeds, "TRUE" is returned.

DIO(Digital Input Output) API Functions

Overview

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

DIO_Set

This function sets the digital output (DO).

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

Parameters:

nModel : Write down the model number. from 0 to 3

nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

nOutput: Set the output value.

Return Value:

If the function call fails, "FALSE" is returned.

If the function call succeeds, "TRUE" is returned.

DIO_Get

This function checks the digital input (DI).

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

Parameters:

nModel : Write down the model number. from 0 to 3

nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

*nInput: Get input value

Return Value:

If the function call fails, "FALSE" is returned.

If the function call succeeds, "TRUE" is returned.

ENCODER API Functions

Overview

BOOL	**ENC_Read (int nModel, int nBoard, int nChannel, int *nApulse, int *nBpulse, int *nZpulse)**
BOOL	**ENC_Run (int nModel, int nBoard, int nChannel, int nFlag)**
BOOL	**ENC_GetRpmCount (int nModel, int nBoard, int nChannel, int *nCnt)**
BOOL	**ENC_SetRpmClearCount (int nModel, int nBoard, int nChannel, int nCnt)**
BOOL	**ENC_GetDistanceCount (int nModel, int nBoard, int nChannel, int *nDir, int *nCnt)**

ENC_Read

This function checks the pulse count value for A, B, Z 3 phases of the encoder.

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

Parameters:

nModel : Write down the model number. from 0 to 3

nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

nChannel : Since it has no meaning for the current device, “0” is set.

*nApulse : A variable pointer with A-phase pulse count value.

*nBpulse : A variable pointer with the B-phase pulse count value.

*nZpulse : It is a variable pointer with Z-phase pulse count value.

Return Value:

If the function call fails, “FALSE” is returned.

If the function call succeeds, “TRUE” is returned.

ENC_Run

This function initializes the encoder counter and starts the function.

BOOL ENC_Run (int nModel, int nBoard, int nChannel, int nFlag)

Parameters:

nModel : Write down the model number. from 0 to 3

nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

nChannel : Since it has no meaning for the current device, “0” is set.

nFlag : This is a start/stop flag. If it is ‘0’, it stops, and if it is not ‘0’, the function is executed.

Return Value:

If the function call fails, “FALSE” is returned.

If the function call succeeds, “TRUE” is returned.

ENC_GetRpmCount

This function obtains the counter value it takes for the encoder to make one rotation.

BOOL ENC_GetRpmCount (int nModel, int nBoard, int nChannel, int *nCnt)

Parameters:

nModel : Write down the model number. from 0 to 3

nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

nChannel : Since it has no meaning for the current device, “0” is set.

*nCnt : 40Mhz clock counter value is 27-bit. If a new z-phase signal is not received within 10 seconds, the count value is initialized.

Return Value:

If the function call fails, “FALSE” is returned.

If the function call succeeds, “TRUE” is returned.

ENC_SetRpmClearCount

This function sets the counter value to wait for the z-phase signal when measuring encoder rpm.

BOOL ENC_GetRpmCount (int nModel, int nBoard, int nChannel, int *nCnt)

Parameters:

nModel : Write down the model number. from 0 to 3

nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

nChannel : Since it has no meaning for the current device, "0" is set.

*nCnt : 40Mhz clock counter value is 32-bit. After reset, the initial value is 10 seconds (0x17D78400), so change this value.

Return Value:

If the function call fails, "FALSE" is returned.

If the function call succeeds, "TRUE" is returned.

ENC_GetDistanceCount

This function checks the forward/reverse direction and the amount of counter change from the initial state through quadrature encoder input.

BOOL ENC_GetDistanceCount(int nModel, int nBoard, int nChannel, int *nDir, int *nCnt)

Parameters:

nModel : Write down the model number. from 0 to 3

nBoard : Shows the board number currently installed in the system. The board number is set using the DIP switch of the board.

nChannel : Since it has no meaning for the current device, "0" is set.

*nDir : This is the direction detection value by a/b signal. When it is '0', it is in the forward direction. When it is '1', it is in the reverse direction.

*nCnt : It is a 27-bit forward/reverse clock counter value according to a/b signals. The maximum value is 0xFFFFFFFF, and if it increases forward from the maximum value, it is initialized to 0x0 and increases again. The minimum (initial) value is 0x0, and when it increases in the reverse direction from the initial value, it decreases to 0xFFFFFFF.

Return Value:

If the function call fails, "FALSE" is returned.

If the function call succeeds, "TRUE" is returned.

PWM API Functions

Overview

```
BOOL          PWM_Enable (int nModel, int nBoard, int nChannel, int nDirection,
                          BOOL bEnable)
BOOL          PWM_CounterClear (int nModel, int nBoard, int nChannel,
                                int nDirection, BOOL bClear)
BOOL          PWM_SetSquare (int nModel, int nBoard, int nChannel, int nDirection,
                             int nTime)
BOOL          PWM_SetPattern (int nModel, int nBoard, int nChannel, int nDirection,
                              int nActTime, int nPeriod)
BOOL          PWM_SetUser (int nModel, int nBoard, int nChannel, int nDirection,
                           int Signal)
BOOL          PWM_SetShotCount (int nModel, int nBoard, int nChannel, int nDirection,
                                int nCount)
BOOL          PWM_GetShotCount (int nModel, int nBoard, int nChannel,
                                int nDirection, int *nCount)
```

PWM_Enable

This function checks the pulse count value for A, B, Z 3 phases of the encoder.

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

Parameters:

nModel : Write down the model number. from 0 to 3

nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

nChannel : Since it has no meaning for the current device, “0” is set.

nDirection : Set the direction.
“0” CW (clockwise)
“1” CCW (counterclockwise).

bEnable : A flag that controls signal generation.
“TRUE” during operation
“FALSE” during stop.

Return Value:

If the function call fails, “FALSE” is returned.

If the function call succeeds, “TRUE” is returned.

PWM_CounterClear

This function initializes the PWM time counter value.

BOOL PWM_CounterClear (int nModel, int nBoard, int nChannel, int nDirection, BOOL bClear)

Parameters:

nModel : Write down the model number. from 0 to 3

nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

nChannel : Since it has no meaning for the current device, “0” is set.

nDirection : Set the direction.
“0” CW (clockwise)
“1” CCW (counterclockwise).

bClear : Count reset flag. “1” Initialization performed, “0” Initialization not performed.

Return Value:

If the function call fails, “FALSE” is returned.

If the function call succeeds, “TRUE” is returned.

PWM_SetSquare

This function sets the set and reset times of the PWM square wave output.

BOOL PWM_SetSquare(int nModel, int nBoard, int nChannel, int nDirection, int nTime)

Parameters:

nModel : Write down the model number. from 0 to 3

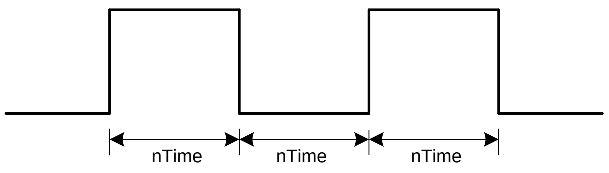
nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

nChannel : Since it has no meaning for the current device, "0" is set.

nDirection : Set the direction.
"0" CW (clockwise)
"1" CCW (counterclockwise).

nTime: This value is the count value of the 40Mhz clock that sets the Set and Reset maintenance time of the signal.

Holding time = 25nsec * nTime [nsec]



Return Value:

If the function call fails, "FALSE" is returned.

If the function call succeeds, "TRUE" is returned.

PWM_SetPattern

This function sets the set retention time and signal period of the PWM signal.

BOOL PWM_SetPattern (int nModel, int nBoard, int nChannel, int nDirection, int nActTime, int nPeriod)

Parameters:

nModel : Write down the model number. from 0 to 3

nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

nChannel : Since it has no meaning for the current device, "0" is set.

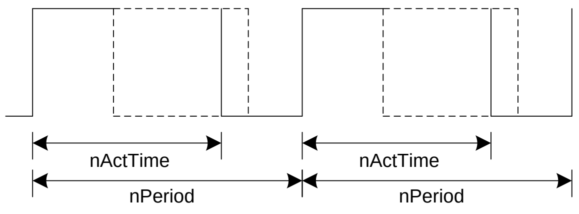
nDirection : Set the direction.
"0" CW (clockwise)
"1" CCW (counterclockwise).

nActTime : This value is the count value of the 40Mhz clock that sets the set maintenance time of the sig

Holding time = 25nsec * nActTime [nsec]

nPeriod : This value is the count value of the 40Mhz clock that sets the signal period.

Period = 25nsec * nPeriod [nsec]



Return Value:

If the function call fails, "FALSE" is returned.

If the function call succeeds, "TRUE" is returned.

PWM_SetUser

This function generates a PWM signal with a user-set value.

BOOL PWM_SetUser (int nModel, int nBoard, int nChannel, int nDirection, int nSignal)

Parameters:

nModel : Write down the model number. from 0 to 3

nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

nChannel : Since it has no meaning for the current device, "0" is set.

nDirection : Set the direction.
"0" CW (clockwise)
"1" CCW (counterclockwise).

nSignal : This is the PWM signal setting value. '1' – HI, '0' – LOW.

Return Value:

If the function call fails, "FALSE" is returned.

If the function call succeeds, "TRUE" is returned.

PWM_SetShotCount

This function sets the PWM signal count value to be output.

BOOL PWM_SetShotCount (int nModel, int nBoard, int nChannel, int nDirection, int nCount)

Parameters:

nModel : Write down the model number. from 0 to 3

nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

nChannel : Since it has no meaning for the current device, "0" is set.

nDirection : Set the direction.
"0" CW (clockwise)
"1" CCW (counterclockwise).

nCount : It is a 27-bit count value, and after outputting the corresponding number, the value is set to '0' (LO) and output is stopped.

Return Value:

If the function call fails, "FALSE" is returned.

If the function call succeeds, "TRUE" is returned.

PWM_GetShotCount

This function gets the PWM signal count value being output.

BOOL PWM_GetShotCount (int nModel, int nBoard, int nChannel, int nDirection, int *nCount)

Parameters:

nModel : Write down the model number. from 0 to 3

nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

nChannel : Since it has no meaning for the current device, "0" is set.

nDirection : Set the direction.
"0" CW (clockwise)
"1" CCW (counterclockwise).

*nCount : 27-bit Count value.

Return Value:

If the function call fails, "FALSE" is returned.

If the function call succeeds, "TRUE" is returned.

SYNC API Functions

Overview

BOOL	**SYNC_SetMode (int nModel, int nBoard, int mode)**
BOOL	**SYNC_SetRun (int nModel, int nBoard, int run)**
BOOL	**SYNC_GetParam (int nModel, int nBoard, int *mode, int *action)**
BOOL	**SYNC_SetDelayTime (int nModel, int nBoard, int delay)**
BOOL	**SYNC_GetDelayTime (int nModel, int nBoard, int *delay)**

SYNC_SetMode

This function sets the board as Master or Slave for each board.

The master board outputs the trigger signal, CLOCK_IO (fixed at 40Mhz), and CLOCK_PLL (variable) signals to the synchronization connector (J8). The slave board receives the above three signals.

The signal direction is determined by calling this function, and if it is the master, clock signals are output immediately.

BOOL SYNC_SetMode (int nModel, int nBoard, int mode)

Parameters:

nModel : Write down the model number. from 0 to 3

nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

mode : Setup value.
'1'- Master, '0'-Slave.

Return Value:

If the function call fails, "FALSE" is returned.

If the function call succeeds, "TRUE" is returned.

SYNC_SetRun

This function controls the Master board's trigger signal output.

BOOL SYNC_SetRun (int nModel, int nBoard, int run)

Parameters:

nModel : Write down the model number. from 0 to 3

nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

run : Setup value
'1'-Trigger HI('1'), '0'-Trigger LO('0').

Return Value:

If the function call fails, "FALSE" is returned.

If the function call succeeds, "TRUE" is returned.

SYNC_GetParam

This function can check whether the function operation has started after a delay using the mode setting status and trigger signal.

BOOL SYNC_GetParam (int nModel, int nBoard, int *mode, int *action)

Parameters:

nModel : Write down the model number. from 0 to 3

nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

*mode : Setup value.
'1'- Master, '0'-Slave.

*action : Setup value.
'1'- Executing function, '0'-Waiting for delay.

Return Value:

If the function call fails, "FALSE" is returned.

If the function call succeeds, "TRUE" is returned.

SYNC_SetDelayTime

This function sets the delay time for the board to operate after trigger signal assert input.

After calling SYNC_SetRun() on the master board, the slave board receives the trigger signal and executes the PID function with a delay of (25nsec*delay) time.

BOOL SYNC_SetDelayTime (int nModel, int nBoard, int delay)

Parameters:

nModel : Write down the model number. from 0 to 3

nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

delay: 0x00000000~0x7FFFFFF.

Return Value:

If the function call fails, "FALSE" is returned.

If the function call succeeds, "TRUE" is returned.

SYNC_GetDelayTime

This function reads the execution delay time from the trigger signal.

BOOL SYNC_GetDelayTime (int nModel, int nBoard, int *delay)

Parameters:

nModel : Write down the model number. from 0 to 3

nBoard : Shows the board number currently installed in the system.
The board number is set using the DIP switch of the board.

*delay: 0x00000000~0x7FFFFFF.

Return Value:

If the function call fails, "FALSE" is returned.

If the function call succeeds, "TRUE" is returned.

Memo

Contact Point

Web sit : https://www.daqsystem.com

Email : postmaster@daqsystem.com

DAQ SYSTEM
Measurement & Automation