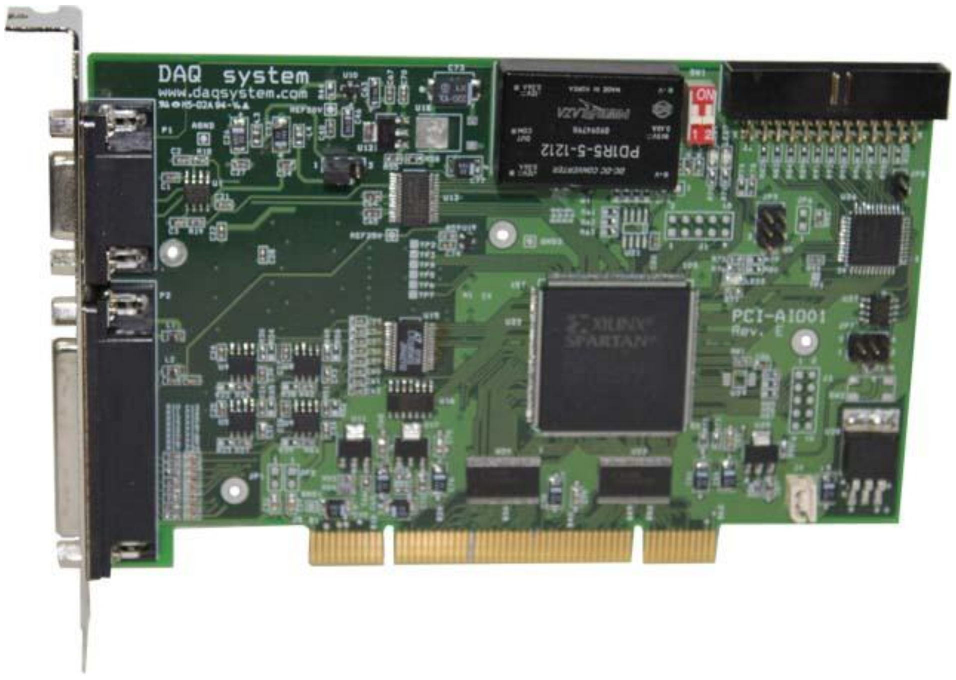
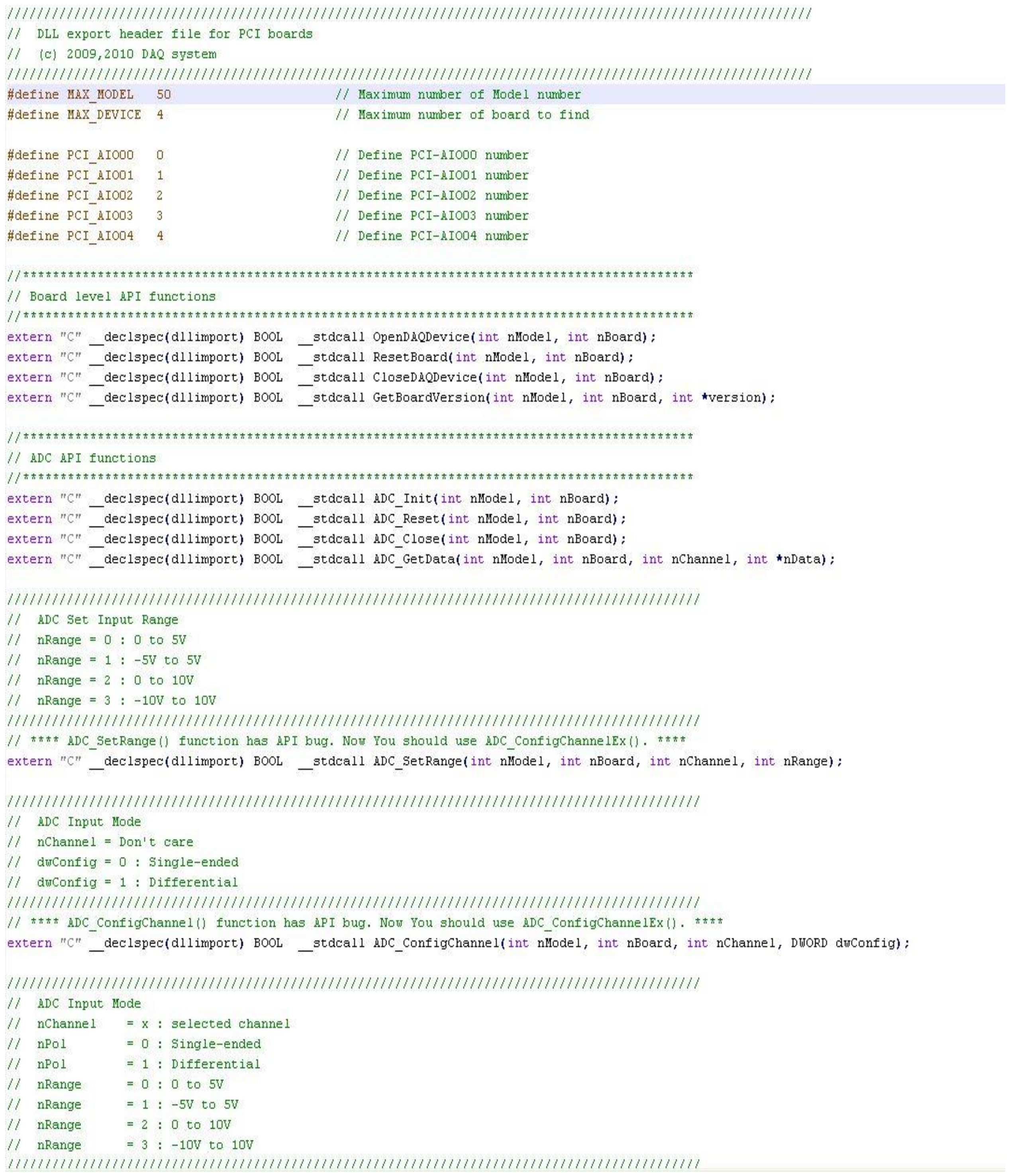
PCI-AIO01/02/04 API Programming (Rev 1.0)



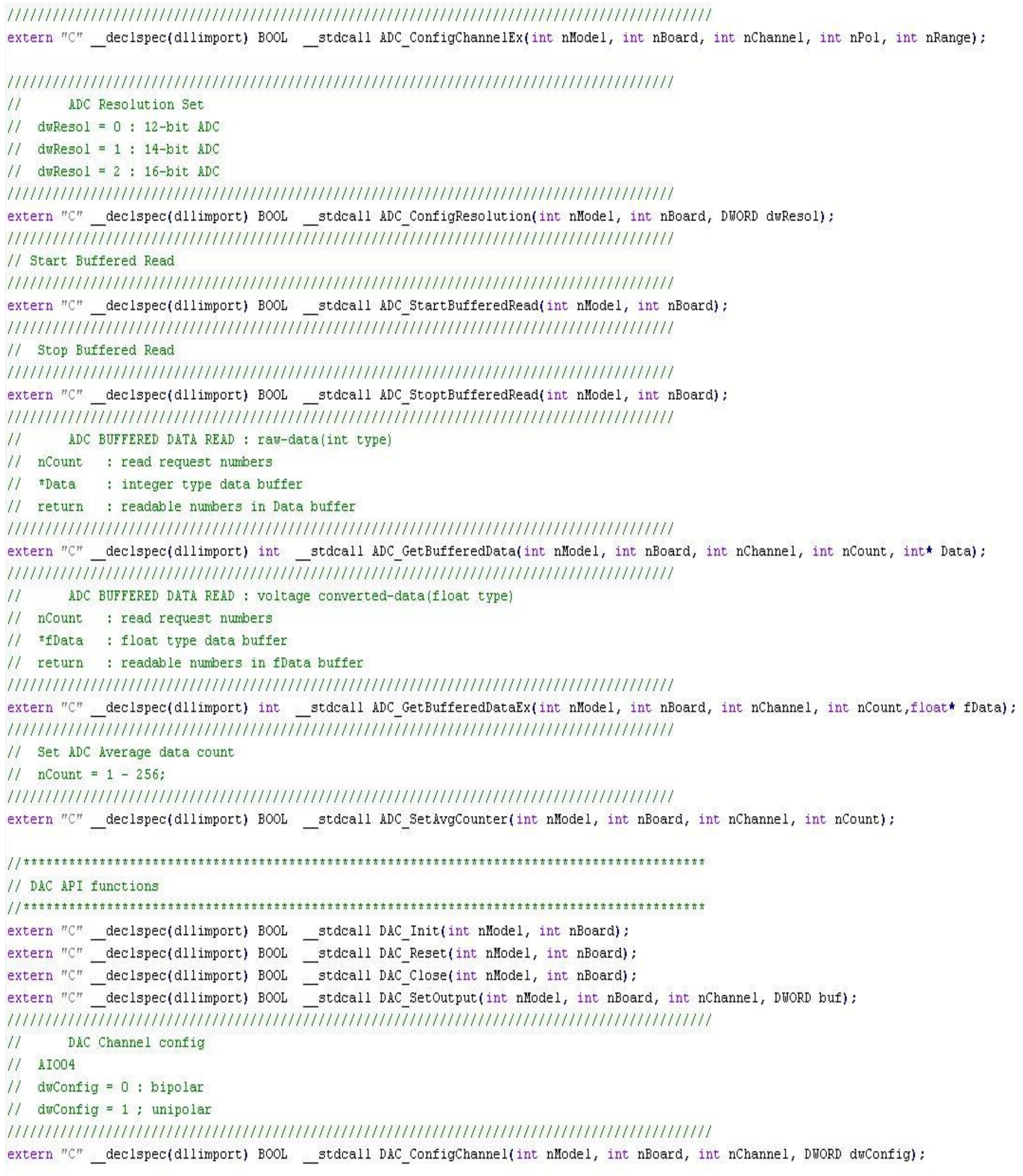
API

This chapter explains the API (Application Programming Interface) to use a PCI-AIO series boards.

This API support PCI_AIO01, PCI_AIO02, PCI_AIO04 products.



```
/////////////////////////////////////////////////////////////////////////////////////////////////////////
//  DLL export header file for PCI boards
//  (c) 2009,2010 DAQ system
/////////////////////////////////////////////////////////////////////////////////////////////////////////
#define MAX_MODEL   50                    // Maximum number of Model number
#define MAX_DEVICE  4                     // Maximum number of board to find

#define PCI_AIO00   0                     // Define PCI-AIO00 number
#define PCI_AIO01   1                     // Define PCI-AIO01 number
#define PCI_AIO02   2                     // Define PCI-AIO02 number
#define PCI_AIO03   3                     // Define PCI-AIO03 number
#define PCI_AIO04   4                     // Define PCI-AIO04 number

//*********************************************************************************************
// Board level API functions
//*********************************************************************************************
extern "C" __declspec(dllimport) BOOL  __stdcall OpenDAQDevice(int nModel, int nBoard);
extern "C" __declspec(dllimport) BOOL  __stdcall ResetBoard(int nModel, int nBoard);
extern "C" __declspec(dllimport) BOOL  __stdcall CloseDAQDevice(int nModel, int nBoard);
extern "C" __declspec(dllimport) BOOL  __stdcall GetBoardVersion(int nModel, int nBoard, int *version);

//*********************************************************************************************
// ADC API functions
//*********************************************************************************************
extern "C" __declspec(dllimport) BOOL  __stdcall ADC_Init(int nModel, int nBoard);
extern "C" __declspec(dllimport) BOOL  __stdcall ADC_Reset(int nModel, int nBoard);
extern "C" __declspec(dllimport) BOOL  __stdcall ADC_Close(int nModel, int nBoard);
extern "C" __declspec(dllimport) BOOL  __stdcall ADC_GetData(int nModel, int nBoard, int nChannel, int *nData);

/////////////////////////////////////////////////////////////////////////////////////////////
//  ADC Set Input Range
//  nRange = 0 : 0 to 5V
//  nRange = 1 : -5V to 5V
//  nRange = 2 : 0 to 10V
//  nRange = 3 : -10V to 10V
/////////////////////////////////////////////////////////////////////////////////////////////
// **** ADC_SetRange() function has API bug. Now You should use ADC_ConfigChannelEx(). ****
extern "C" __declspec(dllimport) BOOL  __stdcall ADC_SetRange(int nModel, int nBoard, int nChannel, int nRange);

/////////////////////////////////////////////////////////////////////////////////////////////
//  ADC Input Mode
//  nChannel = Don't care
//  dwConfig = 0 : Single-ended
//  dwConfig = 1 : Differential
/////////////////////////////////////////////////////////////////////////////////////////////
// **** ADC_ConfigChannel() function has API bug. Now You should use ADC_ConfigChannelEx(). ****
extern "C" __declspec(dllimport) BOOL  __stdcall ADC_ConfigChannel(int nModel, int nBoard, int nChannel, DWORD dwConfig);

/////////////////////////////////////////////////////////////////////////////////////////////
//  ADC Input Mode
//  nChannel    = x : selected channel
//  nPol        = 0 : Single-ended
//  nPol        = 1 : Differential
//  nRange      = 0 : 0 to 5V
//  nRange      = 1 : -5V to 5V
//  nRange      = 2 : 0 to 10V
//  nRange      = 3 : -10V to 10V
/////////////////////////////////////////////////////////////////////////////////////////////
```



```
//////////////////////////////////////////////////////////////////////////////////////////////
extern "C" __declspec(dllimport) BOOL  __stdcall ADC_ConfigChannelEx(int nModel, int nBoard, int nChannel, int nPol, int nRange);

/////////////////////////////////////////////////////////////////////////////////////////
//      ADC Resolution Set
//  dwResol = 0 : 12-bit ADC
//  dwResol = 1 : 14-bit ADC
//  dwResol = 2 : 16-bit ADC
/////////////////////////////////////////////////////////////////////////////////////////
extern "C" __declspec(dllimport) BOOL  __stdcall ADC_ConfigResolution(int nModel, int nBoard, DWORD dwResol);
/////////////////////////////////////////////////////////////////////////////////////////
// Start Buffered Read
/////////////////////////////////////////////////////////////////////////////////////////
extern "C" __declspec(dllimport) BOOL  __stdcall ADC_StartBufferedRead(int nModel, int nBoard);
/////////////////////////////////////////////////////////////////////////////////////////
//  Stop Buffered Read
/////////////////////////////////////////////////////////////////////////////////////////
extern "C" __declspec(dllimport) BOOL  __stdcall ADC_StoptBufferedRead(int nModel, int nBoard);
/////////////////////////////////////////////////////////////////////////////////////////
//      ADC BUFFERED DATA READ : raw-data(int type)
//  nCount   : read request numbers
//  *Data    : integer type data buffer
//  return   : readable numbers in Data buffer
/////////////////////////////////////////////////////////////////////////////////////////
extern "C" __declspec(dllimport) int  __stdcall ADC_GetBufferedData(int nModel, int nBoard, int nChannel, int nCount, int* Data);
/////////////////////////////////////////////////////////////////////////////////////////
//      ADC BUFFERED DATA READ : voltage converted-data(float type)
//  nCount   : read request numbers
//  *fData   : float type data buffer
//  return   : readable numbers in fData buffer
/////////////////////////////////////////////////////////////////////////////////////////
extern "C" __declspec(dllimport) int  __stdcall ADC_GetBufferedDataEx(int nModel, int nBoard, int nChannel, int nCount,float* fData);
/////////////////////////////////////////////////////////////////////////////////////////
//  Set ADC Average data count
//  nCount = 1 - 256;
/////////////////////////////////////////////////////////////////////////////////////////
extern "C" __declspec(dllimport) BOOL  __stdcall ADC_SetAvgCounter(int nModel, int nBoard, int nChannel, int nCount);

//*********************************************************************************************
// DAC API functions
//*********************************************************************************************
extern "C" __declspec(dllimport) BOOL  __stdcall DAC_Init(int nModel, int nBoard);
extern "C" __declspec(dllimport) BOOL  __stdcall DAC_Reset(int nModel, int nBoard);
extern "C" __declspec(dllimport) BOOL  __stdcall DAC_Close(int nModel, int nBoard);
extern "C" __declspec(dllimport) BOOL  __stdcall DAC_SetOutput(int nModel, int nBoard, int nChannel, DWORD buf);
//////////////////////////////////////////////////////////////////////////////////////////////
//      DAC Channel config
//  AIO04
//  dwConfig = 0 : bipolar
//  dwConfig = 1 ; unipolar
//////////////////////////////////////////////////////////////////////////////////////////////
extern "C" __declspec(dllimport) BOOL  __stdcall DAC_ConfigChannel(int nModel, int nBoard, int nChannel, DWORD dwConfig);
```

Board Level API Functions

Overview

BOOL	**OpenDAQDevice (int nModel, int nBoard)**
BOOL	**ResetBoard (int nModel, int nBoard)**
BOOL	**CloseDAQDevice (int nModel, int nBoard)**
BOOL	**GetBoardVersion (int nModel, int nBoard, int *version)**

OpenDAQDevice

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

It opens a device. You may call this function at the very first time you run the program and some suspicious operation.

Parameters:

nModel : It writes down the PCI-AIO model number. (0 ~ 3)

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

Return Value:

If the function succeeds, it returns the number of boards which were detected.

If the function fails, the return value is -1, it means there is no device in the system.

ResetBoard

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

It initializes a device at currently equipped system (PC).

Parameters:

nModel : It writes down the PCI-AIO model number. (0 ~ 3)

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

Return Value:

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

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

CloseDAQDevice

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

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.

Parameters:

nModel : It writes down the PCI-AIO model number. (0 ~ 3)

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

Return Value:

If the function call fails, it returns “FALSE”.

If the function call succeeds, it returns “TRUE”.

GetBoardVersion

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

Get the hardware version of PCI-AIO device.

Parameters:

nModel : It writes down the PCI-AIO model number. (0 ~ 3)

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

version : It is a pointer of variable to get from version. It is a positive integer in the normal state.

Return Value:

If the function call fails, it returns “FALSE”.

If the function call succeeds, it returns “TRUE”.

ADC(Analog to Digital Convertor) API Functions

Overview

BOOL	**ADC_Init (int nModel, int nBoard)**
BOOL	**ADC_Reset (int nModel, int nBoard)**
BOOL	**ADC_Close (int nModel, int nBoard)**
BOOL	**ADC_GetData (int nModel, int nBoard, int nChannel, int *nData)**
BOOL	**ADC_SetRange (int nModel, int nBoard, int nChannel, int nRange)**
BOOL	**ADC_ConfigChannel (int nModel, int nBoard, int nChannel, DWORD dwConfig)**
BOOL	**ADC_ConfigChannelEx (int nModel, int nBoard, int nChannel, int nPol, int nRange)**
BOOL	**ADC_ConfigResolution (int nModel, int nBoard, DWORD dwResol)**
BOOL	**ADC_StartBufferedRead (int nModel, int nBoard)**
BOOL	**ADC_StopBufferedRead (int nModel, int nBoard)**
int	**ADC_GetBufferedData (int nModel, int nBoard, int nChannel, int nCount, int *nData)**
int	**ADC_GetBufferedDataEx (int nModel, int nBoard, int nChannel, int nCount, float *fData)**
BOOL	**ADC_SetAvgCounter (int nModel, int nBoard, int nChannel, int nCount)**

ADC_Init

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

It initializes ADC setup.

Parameters:

nModel : It writes down the PCI-AIO model number. (0 ~ 3)

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

Return Value:

If the function call fails, it returns “FALSE”.

If the function call succeeds, it returns “TRUE”.

ADC_Reset

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

Reset the ADC function.

Parameters:

nModel : It writes down the PCI-AIO model number. (0 ~ 3)

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

Return Value:

If the function call fails, it returns “FALSE”.

If the function call succeeds, it returns “TRUE”.

ADC_Close

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

It closes the resource if the ADC function won’t use any more.

Parameters:

nModel : It writes down the PCI-AIO model number. (0 ~ 3)

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

Return Value:

If the function call fails, it returns “FALSE”.

If the function call succeeds, it returns “TRUE”.

ADC_GetData

BOOL **ADC_GetData (int nModel, int nBoard, int nChannel, int *nData)**

It reads one data regarding current ADC input.

The data which it will read are affected at range, accuracy and polarity so it will call function of ADC_ConfigResolution(), ADC_ConfigChannelEx().

The range of ADC following accuracy is according to polarity setting. [Refer Table 3]

[Table 3. AD value range following accuracy]

Accuracy	Polarity	
	Unipolar	Bipolar
12-bit	0 ~ +4095	-2048 ~ +2047
14-bit	0 ~ +16383	-8192 ~ +8191
16-bit	0 ~ +65535	-32768 ~ +32767

Actual voltage calculation is as follows.

Voltage = (Read value / Resolution Range) *(Range Max. – Min.)

Ex.) In case of 0 ~ 5V range, 12-bit setup

If it read 1024, actual voltage is (1024 /4095) * (5-0) = 1.25 [V].

In case of -10 ~ + 10V, 16-bit setup

If it read -16384, actual voltage is (-16384 /65536) * (10-(-10)) = -5 [V].

Parameters:

nModel : It writes down the PCI-AIO model number. (0 ~ 3)

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

The board number set up by DIP switch.

nChannel :Write the ADC channel value. The PCI-AIO01 channel number is from 0 to 7.

nData : It is a pointer of variable to get from an ADC value as reading.

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

ADC_SetRange

BOOL **ADC_SetRange (int nModel, int nBoard, int nChannel, int nRange)**

This function designates an analog input range of ADC.

Parameters:

nModel : It writes down the PCI-AIO model number. (0 ~ 3)

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

The board number set up by DIP switch.

nChannel : It is meaningless, and it set up '0'

It use an ADC_ ConfigChannelEx () function for setting by a channel because an

Input range of all channel is equally applied in this function.

nRange : It sets up input range of ADC. The value of default is from 0 to 10V.

0 : 0 ~ 5V

1 : -5 ~ 5V

2 : 0 ~ 10V

3 : -10 ~ 10V

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

ADC_ConfigChannel

BOOL **ADC_ConfigChannel (int nModel, int nBoard, int nChannel, DWORD dwConfig)**

This function set up an analog polarity of ADC channel.

Parameters:

nModel : It writes down the PCI-AIO model number. (0 ~ 3)

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

nChannel : It use an ADC_ ConfigChannelEx () function for setting by a channel because an
Input range of all channel is equally applied in this function.

dwConfig : Analog input polarity of Board.

0 : Single-Ended(SE)

1 : Differential(DI)

Return Value:

If the function call fails, it returns “FALSE”.

If the function call succeeds, it returns “TRUE”.

ADC_ConfigChannelEx

BOOL **ADC_ConfigChannelEx (int nModel, int nBoard, int nChannel, int nPol, int nRange)**

This function set up an input range and polarity by each ADC channels.

Parameters:

nModel : It writes down the PCI-AIO model number. (0 ~ 3)

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

nChannel : It designates channel value.

nPol : Analog InputPolarity.

0 : Single-Ended(SE)

1 : Differential(DI)

nRange : It sets up input range of ADC. The value of default is from 0 to 10V.

0 : 0 ~ 5V

1 : -5 ~ 5V

2 : 0 `10V

3 : -10 ~ 10V

Return Value:

If the function call fails, it returns “FALSE”.

If the function call succeeds, it returns “TRUE”.

ADC_ConfigResolution

BOOL **ADC_ConfigResolution (int nModel, int nBoard, DWORD dwResol)**

It set up an AD Data width (bit).

The AD converter of board support three resolution of 12,14,16 bit besides data accuracy.

Parameters:

nModel : It writes down the PCI-AIO model number. (0 ~ 3)

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

dwResol : Data bit setup value. It set up by data bits of AD converter.
If it is 12-bit ADC, it is '0' (default).
If it is 14-bit ADC, it is '1'.
If it is 16-bit ADC, it is '2'.

Return Value :

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

ADC_StartBufferedRead

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

It start to save the AD data to the buffer.

The 32,768 data buffer per channel is assigned to library. If it call this function, the AD data continues save a type of ring buffer to buffer. A user shall read the data are over written, and can get effective data.

Parameters:

nModel : It writes down the PCI-AIO model number. (0 ~ 3)

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

ADC_StopBufferedRead

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

It stops to save AD data to the buffer.

Parameters:

nModel : It writes down the PCI-AIO model number. (0 ~ 3)

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

Return Value:

If the function call fails, it returns “FALSE”.

If the function call succeeds, it returns “TRUE”.

ADC_GetBufferedData

int **ADC_GetBufferedData (int nModel, int nBoard, int nChannel, int nCount, int *nData)**

It reads AD data which stored buffer.

An accuracy, input polarity and range like a function RageADC_GetDtat() set up as if use function of ADC_ConfigResolution(), ADC_ConfigChannelEx().

The range of ADC following accuracy is according to polarity setting. [Refer Table 3]

Actual voltage calculation is as follows.

Voltage = (Read value / Resolution Range) *(Range Max. – Min.)

ex) In case of 0 ~ 5V range, 12-bit setup

If it read 1024, actual voltage is (1024 /4095) * (5-0) = 1.25 [V].

In case of -10 ~ + 10V, 16-bit setup

If it read -16384, actual voltage is (-16384 /65536) * (10-(-10)) = -5 [V].

Parameters:

nModel : It writes down the PCI-AIO model number. (0 ~ 3)

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

nChannel : It writes down ADC channel value. The PCI-AIO01 channel number is from 0 to 7.

nCount : It sets up a data number reading within the maximum buffer numbers (32,768).

nData : It is a pointer of the variable that AD data are stored.
It sets the buffer size up over data count.

Return Value:

It is a data number which stored of read nData.

ADC_GetBufferedDataEx

int **ADC_GetBufferedDataEx (int nModel, int nBoard, int nChannel, int nCount, float *fData)**

It reads AD data which stored buffer.

It acquires data with voltage level float type to stored buffer. As data value got at an accuracy, input polarity and input is affected, it have to certainly call a function of ADC_ConfigResolution() and ADC_ConfigChannelEx().

Parameters:

nModel : It writes down the PCI-AIO model number. (0 ~ 3)

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

nChannel : It writes down ADC channel value. The PCI-AIO01 channel number is from 0 to 7.

nCount : It sets up a data number reading within the maximum buffer numbers (32,768).

nData : It is a pointer of the variable that voltage level data are stored.
It sets the buffer size up over data count.

Return Value:

It is a data number which stored of read nData.

ADC_SetAvgCounter

BOOL **ADC_SetAvgCounter (int nModel, int nBoard, int nChannel, int nCount)**

It set up data number to use to a moving average.

If data number is ‘1’, it don’tapply a moving average, a function of ADC_GetBufferedData() and ADC_GetBufferedDataEx() get an average data.

Parameters:

nModel : It writes down the PCI-AIO model number. (0 ~ 3)

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

nChannel : It writes down ADC channel value. The PCI-AIO01 channel number is from 0 to 7.

nCount : It sets up the data number that a moving average is applied. It set up a number 1 ~ 255.

Return Value:

If the function call fails, it returns “FALSE”.

If the function call succeeds, it returns “TRUE”.

DAC(Digital to Analog Convertor) API Functions

Overview

BOOL	**DAC_Init (int nModel, int nBoard)**
BOOL	**DAC_Reset (int nModel, int nBoard)**
BOOL	**DAC_Close (int nModel, int nBoard)**
BOOL	**DAC_SetOutput (int nModel, int nBoard, int nChannel, DWORD dwData)**
BOOL	**DAC_ConfigChannel (int nModel, int nBoard, int nChannel, DWORD dwConfig)**

DAC_Init

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

Initialize a DAC setup.

Parameters:

nModel : It writes down the PCI-AIO model number. (0 ~ 3)

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

Return Value:

If the function call fails, it returns “FALSE”.

If the function call succeeds, it returns “TRUE”.

DAC_Reset

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

Reset a DAC function.

Parameters:

nModel : It writes down the PCI-AIO model number. (0 ~ 3)

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

Return Value:

If the function call fails, it returns “FALSE”.

If the function call succeeds, it returns “TRUE”.

DAC_Close

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

It closes the resources which it used if the DAC function won't use any more.

Parameters:

nModel : It writes down the PCI-AIO model number. (0 ~ 3)

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

DAC_SetOutput

BOOL **DAC_SetOutput(int nModel, int nBoard, int nChannel, DWORD dwData)**

This function output a value of DAC.

Return output range of DAC is from 0 to 10V in case of Unipolar, from -10 to 10V in case of Bipolar. At this time range of setup is from 0 to 4095, calculation is as follows.

Ex) In case of 0 ~ 10V range setup

If output voltage is 5V, setup value is (5V/ 10V) * 4096 = 2048.

In case of 0 ~ 10V range setup

If output voltage is 5V, setup value is (10+5V/20V) * 4096 = 3072.

If output voltage is -5V, setup value is (10-5V/20V) * 4096 = 1024.

Parameters:

nModel : It writes down the PCI-AIO model number. (0 ~ 3)

nBoard : It informs a board number at currently equipped system.
The board number set up by DIP switch.

nChannel : It writes down DAC channel value. The PCI-AIO01 channel number is from 0 to 1.

dwData : It set up a value to be displayed to DAC.

Return Value:

If the function call fails, it returns "FALSE".

If the function call succeeds, it returns "TRUE".

DAC_ConfigChannel

BOOL **DAC_ConfigChannel(int nModel, int nBoard, int nChannel, DWORD dwConfig)**

It sets up of output range each DAC channel.

Parameters:

nModel : It writes down the PCI-AIO model number. (0 ~ 3)

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

The board number set up by DIP switch.

nChannel : It writes down DAC channel value. The PCI-AIO01 channel number is from 0 to 1.

dwConfig :	0 : Bipolar (-10V ~ + 10V)
	1: Unipolar (0V ~ + 10V) default value

Return Value:

If the function call fails, it returns “FALSE”.

If the function call succeeds, it returns “TRUE”.