PCI-AIO05 API Programming (Rev 1.0)

API 설명

PCI-AIO05 보드를 사용하기 위한 API(Application Programming Interface)를 설명한다.

현재 지원하는 API는 다음과 같다.

```
////////////////////////////////////////////////////////////////////////////////////////////////////
//  DLL export header file for PCI boards
//  (c) 2010 DAQ system
////////////////////////////////////////////////////////////////////////////////////////////////////
#define MAX_MODEL   20                      // Maximum number of model to find
#define MAX_DEVICE  4                       // Maximum number of board to find
#define MAX_CHANNEL 16                      // Maximum number of channel to find
#define MIN_ADC_AVG 1                       // Minimum number of adc data average
#define MAX_ADC_AVG 256                     // Maximum number of adc data average
#define MAX_BUFFER_LENGTH   1024            // Maximum number of adc data average;auto buffer mode

#define PCI_AIO05   5                       // Define PCI-AIO05 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_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 nResol, int *nData);
extern "C" __declspec(dllimport) BOOL  __stdcall ADC_SetRange(int nModel, int nBoard, int nChannel,
                                                              int nRange);
extern "C" __declspec(dllimport) BOOL  __stdcall ADC_GetChannelState(int nModel, int nBoard,
                                                                     int nChannel, BYTE *status);
extern "C" __declspec(dllimport) BOOL  __stdcall ADC_SetConversionTime(int nModel, int nBoard, int nChannel,
                                                                       int time);
extern "C" __declspec(dllimport) BOOL  __stdcall ADC_SetAvgCounter(int nModel, int nBoard, int nChannel,
                                                                   int nCount);
extern "C" __declspec(dllimport) BOOL  __stdcall ADC_SetResolution(int nModel, int nBoard, int nChannel,
                                                                   int nResol);
extern "C" __declspec(dllimport) BOOL  __stdcall ADC_StartBufferedRead(int nModel, int nBoard);
extern "C" __declspec(dllimport) BOOL  __stdcall ADC_StoptBufferedRead(int nModel, int nBoard);
extern "C" __declspec(dllimport) int   __stdcall ADC_GetBufferedData(int nModel, int nBoard, int nChannel,
                                                                     int nCount, int* Data);
//*************************************************************************************
// DIO API functions
//*************************************************************************************
extern "C" __declspec(dllimport) BOOL  __stdcall DIO_Read(int nModel, int nBoard, DWORD *nData);
extern "C" __declspec(dllimport) BOOL  __stdcall DIO_Write(int nModel, int nBoard, DWORD nData);
extern "C" __declspec(dllimport) BOOL  __stdcall DIO_WriteBack(int nModel, int nBoard, DWORD *nData);
```

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

디바이스를 Open한다. PCI-AIO 시리즈 보드를 이용하는 프로그램에서 초기에 반드시 한번 함수를 호출하여 디바이스를 Open 하여야 한다.

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

Parameters:

nModel : PCI-AIO 모델번호를 적는다.

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

Return Value:

함수 호출에 실패할 경우 “FALSE” 성공일 경우 “TRUE”을 리턴함.

ResetBoard

현재 시스템(PC)에 장착된 디바이스를 초기화 한다.

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

Parameters:

nModel : PCI-AIO 모델번호를 적는다.

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

Return Value:

함수 호출에 실패할 경우 “FALSE” 성공일 경우 “TRUE”을 리턴함.

CloseDAQDevice

오픈된 모든 PCI-AIO 시리즈 디바이스를 Close한다. 장치의 사용이 끝나게 되면, 반드시 장치를 Close 하여 다른 프로그램에서 사용할 수 있도록 한다.

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

Parameters:

nModel : PCI_ AIO 모델번호를 적는다.

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

Return Value:

함수 호출에 실패할 경우 “FALSE” 성공일 경우 “TRUE”을 리턴함.

GetBoardVersion

PCI-AIO 디바이스의 하드웨어 버전 정보를 얻는다. .

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

Parameters:

nModel : PCI-AIO 모델번호를 적는다.

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

version : 버전 정보를 받을 변수의 포인터이다. 정상적인 경우 양의 정수 값을 나타낸다.

Return Value:

함수 호출에 실패할 경우 “FALSE” 성공일 경우 “TRUE”을 리턴함.

ADC(Analog to Digital Convertor) API Functions

Overview

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

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

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

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

BOOL **ADC_GetChannelState (int nModel,int nBoard,int nChannel,BYTE *status)**

BOOL **ADC_SetConversionTime (int nModel, int nBoard, int nChannel, int time)**

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

BOOL **ADC_SetResolution (int nModel, int nBoard, int nChannel, int nResol)**

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)**

ADC_Reset

AD Converter 기능을 초기화한다.

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

Parameters:

nModel : PCI_ AIO05 모델번호를 적는다.

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

Return Value:

함수 호출에 실패할 경우 “FALSE” 성공일 경우 “TRUE”을 리턴함.

ADC_Close

ADC 기능을 더 이상 사용하지 않을 경우 호출하여 사용한 리소스를 해제한다.

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

Parameters:

nModel : PCI_ AIO05 모델번호를 적는다.

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

Return Value:

함수 호출에 실패할 경우 “FALSE” 성공일 경우 “TRUE”을 리턴함.

ADC_GetData

AD 변환된 데이터를 보드로부터 읽는다.

16 ~ 24비트 분해능(Resolution)과 ±10V, +10V, ±5V 및 +5V 입력범위를 지원한다. 입력신호의 레벨이 정해진 입력범위를 벗어날 경우, 최대 범위 이상 값에서는 최대값, 최소 범위 이하에 대해서는 최소값이 표시된다.

[표 4. 정밀도에 따른 AD 값의 범위]

정밀도	극성	
	Unipolar	Bipolar
16-비트	0 ~ +65535	-32768 ~ +32767
17-비트	0 ~ +131071	-65536 ~ +65535
18-비트	0 ~ +262143	-131072 ~ +131071
19-비트	0 ~ +524287	-262144 ~ +262143
20-비트	0 ~ +1048575	-524288 ~ +524287
21-비트	0 ~ +2097151	-1048576 ~ 1048575
22-비트	0 ~ +4194303	-2097152 ~ 2097151
23-비트	0 ~ +8388607	-4194304 ~ 4194303
24-비트	0 ~ +16777215	-8388608 ~ 8388607

이때 실제 전압의 계산은 다음과 같다.

전압 = (읽은 값/Resolution 범위) *(Range 최대값-최소값)

예) 분해능 16 비트, ±10V 범위에서 (-16384)를 읽었을 경우,

(읽은 값)/65,536 * (전압변동범위)

= (-16384)/65,536 * (10 - (-10)) = (-0.25) * 20

= (-5.0) [V]

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

Parameters:

nModel : PCI_ AIO05 모델번호를 적는다.

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

nChannel : ADC 채널 값을 적는다. PCI_ AIO05의 채널 번호는 0에서 3까지 이다.

nResol : 변환 데이터의 분해능을 적는다.

nData : 읽어올 ADC값을 받을 변수의 포인터이다. 값의 범위는 위 [표 4]와 같다.

Return Value:

함수 호출에 실패할 경우 “FALSE” 성공일 경우 “TRUE”을 리턴함.

ADC_SetRange

ADC 채널의 입력 범위를 지정한다.

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

Parameters:

nModel : PCI_ AIO05 모델번호를 적는다.

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

nChannel : ADC 채널 값을 적는다. PCI_ AIO05의 채널 번호는 0에서 3까지 이다.

nRange : ADC의 입력 범위를 설정한다.

0 : ±10V(default)

1 : +10V

2 : ±5V

3 : +5V

Return Value:

함수 호출에 실패할 경우 “FALSE” 성공일 경우 “TRUE”을 리턴함.

ADC_GetChannelState

ADC 채널의 상태를 확인한다.

BOOL **ADC_GetChannelState (int nModel,int nBoard,int nChannel,BYTE *status)**

Parameters:

nModel : PCI_ AIO05 모델번호를 적는다.

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

nChannel : ADC 채널 값을 적는다. PCI_ AIO05의 채널 번호는 0에서 3까지 이다.

*status : 채널 상태를 얻는 데이터.

비트 6-5 – 상태 값을 읽은 채널 번호

비트 1 – 아날로그 입력 신호의 극성을 표시
0 : 양전압 상태, 1 : 음전압 상태

비트 0 – 아날로그 입력 신호의 범위 상태
0 : 신호가 입력범위에 존재하는 정상 상태, 1 : over-range 상태

Return Value:

함수 호출에 실패할 경우 “FALSE” 성공일 경우 “TRUE”을 리턴함.

ADC_SetConversionTime

ADC의 필터워드(FW) 값을 설정하여 변환시간을 조절한다.

BOOL **ADC_SetConversionTime (int nModel, int nBoard, int nChannel, int time)**

Parameters:

nModel : PCI_ AIO05 모델번호를 적는다.

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

nChannel : ADC 채널 값을 적는다. PCI_ AIO05의 채널 번호는 0에서 3까지 이다.

Time : FW 값. 표 3)을 참조하여 설정 할 것.

Return Value:

함수 호출에 실패할 경우 “FALSE” 성공일 경우 “TRUE”을 리턴함.

ADC_SetAvgCounter

이동평균에 사용할 AD 데이터 수를 설정한다.

데이터 수가 1일 경우에는 이동평균이 적용되지 않으며, ADC_GetBufferedData(), 함수는 평균화된 데이터를 취득한다.

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

Parameters:

nModel : PCI-AIO05 모델번호를 적는다.

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

nChannel : ADC 채널 값을 적는다. PCI-AIO05의 채널 번호는 0에서 3까지 이다.

nCount : 이동평균이 적용되는 데이터 수를 지정한다. 1 ~ 255의 수를 지정한다.

Return Value:

함수 호출에 실패할 경우 “FALSE” 성공일 경우 “TRUE”을 리턴함.

ADC_SetResolution

ADC의 Resolution 입력 범위를 지정한다.

BOOL **ADC_SetResolution (int nModel, int nBoard, int nChannel, int nResol)**

Parameters:

nModel : PCI-AIO05 모델번호를 적는다.

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

nChannel : ADC 채널 값을 적는다. PCI-AIO05의 채널 번호는 0에서 3까지 이다.

nResolution : ADC의 Resolution 입력 범위를 설정한다. (16bit ~ 24bit)

Return Value:

함수 호출에 실패할 경우 “FALSE” 성공일 경우 “TRUE”을 리턴함.

ADC_StartBufferedRead

AD 데이터를 버퍼에 저장을 시작한다.

라이브러리에 채널당 32,768개의 데이터 버퍼가 할당된다. 이 함수를 호출하게 되면 버퍼에 AD 데이터가 연속으로 링 버퍼 형식으로 연속 저장된다. 사용자는 데이터가 덮어써지기 전에 읽어야만 유효한 데이터를 취득할 수 있다.

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

Parameters:

nModel : PCI-AIO05 모델번호를 적는다.

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

Return Value:

함수 호출에 실패할 경우 “FALSE” 성공일 경우 “TRUE”을 리턴함.

ADC_StopBufferedRead

AD 데이터를 버퍼에 저장을 중지한다.

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

Parameters:

nModel : PCI-AIO05 모델번호를 적는다.

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

Return Value:

함수 호출에 실패할 경우 “FALSE” 성공일 경우 “TRUE”을 리턴함.

ADC_GetBufferedData

버퍼에 저장된 AD 데이터를 읽어 온다.
정밀도에 따른 ADC의 값의 범위는 극성 설정에 따라 [표 4]와 같다.

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

Parameters:

nModel : PCI-AIO05 모델번호를 적는다.

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

nChannel : ADC 채널 값을 적는다. PCI-AIO05의 채널 번호는 0에서 3까지 이다.

nCount : 읽어 올 데이터 수를 지정한다. 최대 버퍼 수(32,768) 이내의 수를 지정한다.

nData : AD 데이터가 저장될 변수의 포인터이다. 버퍼 크기를 읽어 올 데이터 수 이상으로 할당한다.

Return Value:

실제 nData에 저장된 데이터 수이다.

DIO(Digital Input Output) API Functions

Overview

BOOL **DIO_Read (int nModel, int nBoard, DWORD *nData)**

BOOL **DIO_Write (int nModel, int nBoard, DWORD nData)**

BOOL **DIO_WriteBack (int nModel, int nBoard, DWORD *nData)**

DIO_Read

Digital Input 의 상태를 읽는다.

BOOL **DIO_Read (int nModel, int nBoard, DWORD *nData)**

Parameters:

nModel : PCI_ AIO05 모델번호를 적는다.

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

nData :읽어올 DI값을 받을 변수의 포인터이다.
데이터 비트와 입력 핀과의 관계는 비트 0 – EIN0, 비트 1 – EIN1, 비트 2 – EIN2, 비트 3 – EIN3을 의미한다.

Return Value:

함수 호출에 실패할 경우 “FALSE” 성공일 경우 “TRUE”을 리턴함.

DIO_Write

Digital Output 핀의 상태를 설정한다.

BOOL **DIO_Write (int nModel, int nBoard, DWORD nData)**

Parameters:

nModel : PCI_ AIO05 모델번호를 적는다.

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

nData : 출력 DO값으로 데이터 비트와 입력 핀과의 관계는 비트 0 – EOUT0, 비트 1 – EOUT1, 비트 2 – EOUT2, 비트 3 – EOUT3을 의미한다.
EOUT 핀은 출력 값이 ‘1’ 이면 접지에 연결되고, ‘0’이면 float 상태가 된다.

Return Value:

함수 호출에 실패할 경우 “FALSE” 성공일 경우 “TRUE”을 리턴함.

DIO_WriteBack

Digital Output 레지스터의 값을 얻는다.

BOOL **DIO_WriteBack (int nModel, int nBoard, DWORD *nData)**

Parameters:

nModel : PCI_ AIO05 모델번호를 적는다.

nBoard : 현재 시스템에 장착되어 있는 보드 번호를 알려준다.
보드 번호는 보드의 DIP 스위치를 이용하여 설정한다.

nData : DO 레지스터 값으로 출력상태를 나타낸다.

Return Value:

함수 호출에 실패할 경우 “FALSE” 성공일 경우 “TRUE”을 리턴함.