General information on PCI board API

Board Outline and Functional sub-systems

We can separate the board function into four functional sub-systems in the software control viewpoint.

**Board Control Sub-system**
- Controls over-all board operation.

**Analog Input Sub-system**
- Controls the functions related with analog input operation.

**Analog Output Sub-system**
- Controls the functions related with analog output operation.

**Digital Input/Output Sub-system**
- Controls the functions related with digital input/output operation.

[note] To get information on each sub-systems capability, please refer to each board manual.
Board Level API Functions

**Overview**

```c
int OpenDAQDevice(void)

BOOL ResetBoard(int nBoard)

BOOL CloseDAQDevice(void)
```

**OpenDAQDevice**

```c
int OpenDAQDevice(void)
```

**Return Value:**

- If the function succeeds, it returns the number of boards which were detected.
- If the function fails, the return value is zero, it means there is no device in the system.

**ResetBoard**

```c
BOOL ResetBoard(int nBoard)
```

**Parameters:**

- `nBoard`: The ordinal number of board which you want to initialize. You can get how many boards are installed in this system using `OpenDAQDevice` function.

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

**Return Value:**

- It returns TRUE in case of the success of reset and initialization.
- If you get FALSE you should not use any API call functions with the board and call `CloseDAQDevice` the board.

**CloseDAQDevice**

```c
BOOL CloseDAQDevice(void)
```

The CloseDAQDevice function closes all opened devices(boards).

**Return Value:**

- If the function succeeds, it returns TRUE.
Analog Input API Functions

Overview

BOOL Al_Init(int nBoard)
BOOL Al_Reset(int nBoard)
BOOL Al_Close(int nBoard)
BOOL Al_SetInterrupt(int nBoard, BOOL bEnable, IntHandler lpISR)
DWORD Al_ReadData(int nBoard, DWORD Offset, DWORD nCount, DWORD *buf)
DWORD Al_WriteData(int nBoard, DWORD Offset, DWORD nCount, DWORD *buf)
DWORD Al_ReadReg(int nBoard, DWORD Offset)
DWORD Al_WriteRegMul(int nBoard, DWORD Offset, DWORD nCount, DWORD *buf)
DWORD Al_WriteReg(int nBoard, DWORD Offset)
DWORD Al_WriteReg(int nBoard, DWORD Offset, DWORD nCount, DWORD *buf)

Al_Init
Initialize analog input sub-system
BOOL Al_Init(int nBoard)

Parameters:
nBoard :
The ordinal number of board which you want to initialize the analog input sub-system.

Return Value:
If the function succeeds, it returns TRUE.

Al_Reset
Reset analog input sub-system
BOOL Al_Reset(int nBoard)

Parameters:
nBoard :
The ordinal number of board which you want to reset.

Return Value:
If the function succeeds, it returns TRUE.

Al_Close
Close analog input sub-system, this function will stop all analog input operation
BOOL  AI_Close(int nBoard)

Parameters:

nBoard :

The ordinal number of board which you want to close.

Return Value:

If the function succeeds, it returns TRUE.

---

**AI_SetInterrupt**

Set handler function which will be called automatically by DLL in case interrupt service is required.

BOOL  AI_SetInterrupt(int nBoard, BOOL bEnable, IntHandler lpISR)

Parameters:

nBoard :

The board number which you want to set up the interrupt handler.

bEnable :

If TRUE, enables analog output interrupt, FALSE disables the interrupt.

lpISR :

Interrupt handler callback function.

Return Value:

If the function succeeds, it returns TRUE.

---

Interrupt handler function definition

typedef DWORD (CALLBACK *IntHandler)(DWORD dwStatus)

in application program, the callback function need to be defined as shown below

DWORD testHandler(DWORD dwIntStatus)
{
    ....,
}

and call the AI_SetInterrupt like this

    AI_SetInterrupt(1, TRUE, testHandler);

---

**AI_ReadData**

Read analog input data from the buffer or fifo mapped in memory space.

DWORD  AI_ReadData(int nBoard, DWORD Offset, DWORD nCount, DWORD *buf)

Parameters:
nBoard:
The ordinal number of board.

Offset:
The offset from the base buffer address in BYTE size.

nCount:
The number of DWORD to be read.

buf:
The buffer address to receive the analog data.

Return Value:
The number of data received in DWORD size.

**AI_WriteData**
Write data to analog-input subsystem.

DWORD **AI_WriteData**(int nBoard, DWORD Offset, DWORD nCount, DWORD *buf)

**Parameters:**
nBoard:
The ordinal number of board.

Offset:
The offset from the base buffer address in BYTE size.

nCount:
The number of DWORD to be sent.

buf:
The buffer address to send to analog data.

Return Value:
The number of data be sent in DWORD size.

**AI_ReadReg**
Read register value which is related with analog input operation.

DWORD **AI_ReadReg**(int nBoard, DWORD Offset)

**Parameters:**
nBoard:
The ordinal number of board.

Offset:
The offset from the base buffer address in BYTE size.
Return Value:
  The value of register.

**AI_ReadRegMul**
Read register multiple, this function reads multiple data from IO space register. Some fifo are mapped to IO space.

*note* In reading multiple data from IO space, there is no address auto-increment operation.

DWORD **AI_ReadRegMul**(int nBoard, DWORD Offset, DWORD nCount, DWORD *buf)

**Parameters:**
- nBoard :
  The ordinal number of board.
- Offset :
  The offset from the base buffer address in BYTE size.
- nCount :
  The number of DWORD to be read.
- buf :
  The buffer address to receive the data.

**Return Value:**
The number of data received in DWORD size.

**AI_WriteReg**
Write register value which is related with analog input operation.

DWORD **AI_WriteReg**(int nBoard, DWORD Offset)

**Parameters:**
- nBoard :
  The ordinal number of board.
- Offset :
  The offset from the base buffer address in BYTE size.

**Return Value:**
The value of register after writing.
AI_WriteRegMul
Write register multiple, this function writes multiple data to IO space. Some fifo are mapped to IO space.

[note] In writing multiple data to IO space, there is no address auto-increment operation.

DWORD AI_WriteRegMul(int nBoard, DWORD Offset, DWORD nCount, DWORD *buf)

Parameters:
  nBoard :
    The board number.
  Offset :
    The offset from the base buffer address in BYTE size.
  nCount :
    The number of DWORD to be written.
  buf :
    The buffer address to send to AI register.

Return Value:
  The number of data written to in DWORD size.
### Analog Output API Functions

**Overview**

**BOOL** `AO_Init(int nBoard)`

**BOOL** `AO_Reset(int nBoard)`

**BOOL** `AO_Close(int nBoard)`

**BOOL** `AO_SetInterrupt(int nBoard, BOOL bEnable, IntHandler lpISR)`

**DWORD** `AO_ReadData(int nBoard, DWORD Offset, DWORD nCount, DWORD *buf)`

**DWORD** `AO_WriteData(int nBoard, DWORD Offset, DWORD nCount, DWORD *buf)`

**DWORD** `AO_ReadReg(int nBoard, DWORD Offset)`

**DWORD** `AO_ReadRegMul(int nBoard, DWORD Offset, DWORD nCount, DWORD *buf)`

**DWORD** `AO_WriteReg(int nBoard, DWORD Offset)`

**DWORD** `AO_WriteRegMul(int nBoard, DWORD Offset, DWORD nCount, DWORD *buf)`

---

**AO_Init**

Initialize analog output sub-system

**BOOL** `AO_Init(int nBoard)`

**Parameters:**

- **nBoard:**
  
  The ordinal number of board which you want to initialize the analog output sub-system.

**Return Value:**

- If the function succeeds, it returns TRUE.

---

**AO_Reset**

Reset analog output sub-system

**BOOL** `AO_Reset(int nBoard)`

**Parameters:**

- **nBoard:**
  
  The ordinal number of board which you want to reset.

**Return Value:**

- If the function succeeds, it returns TRUE.

---

**AO_Close**

Close analog output sub-system, this function will stop all analog output operation
BOOL **AO_Close**(int nBoard)

**Parameters:**

- **nBoard:**
  The ordinal number of board which you want to close.

**Return Value:**

- If the function succeeds, it returns TRUE.

**AO_SetInterrupt**

Set handler function which will be called automatically by DLL in case interrupt service is required.

BOOL **AO_SetInterrupt**(int nBoard, BOOL bEnable, IntHandler lpISR)

**Parameters:**

- **nBoard:**
  The board number which you want to set up the interrupt handler.

- **bEnable:**
  If TRUE, enables analog output interrupt, FALSE disables the interrupt.

- **lpISR:**
  Interrupt handler callback function.

**Return Value:**

- If the function succeeds, it returns TRUE.

Interrupt handler function definition

```c
typedef DWORD (CALLBACK *IntHandler)(DWORD dwStatus)
```

in application program, the callback function need to be defined as shown below

```c
DWORD testHandler(DWORD dwIntStatus)
{
    ....,
}
```

and call the **AO_SetInterrupt** like this

```c
AO_SetInterrupt(1, TRUE, testHandler);
```

**AO_ReadData**

Read analog output data from the buffer or fifo mapped in memory space.

DWORD **AO_ReadData**(int nBoard, DWORD Offset, DWORD nCount, DWORD *buf)

**Parameters:**

- **nBoard:**
- **Offset:**
- **nCount:**
- **buf:**
nBoard :
   The ordinal number of board.
Offset :
   The offset from the base buffer address in BYTE size.
nCount :
   The number of DWORD to be read.
buf :
   The buffer address to receive the analog data.

Return Value:
   The number of data received in DWORD size.

AO_WriteData
Write data to analog-input subsystem.
DWORD AO_WriteData(int nBoard, DWORD Offset, DWORD nCount, DWORD *buf)

Parameters:
   nBoard :
      The ordinal number of board.
   Offset :
      The offset from the base buffer address in BYTE size.
   nCount :
      The number of DWORD to be sent.
   buf :
      The buffer address to send to analog data.

Return Value:
   The number of data be sent in DWORD size.

AO_ReadReg
Read register value which is related with analog output operation.
DWORD   AO_ReadReg(int nBoard, DWORD Offset)

Parameters:
   nBoard :
      The ordinal number of board.
   Offset :
      The offset from the base buffer address in BYTE size.
Return Value:
   The value of register.

**AO_ReadRegMul**

Read register multiple, this function reads multiple data from IO space register. Some fifo are mapped to IO space.

[note] In reading multiple data from IO space, there is no address auto-increment operation.

DWORD **AO_ReadRegMul**(int nBoard, DWORD Offset, DWORD nCount, DWORD *buf)

Parameters:

  nBoard :
       The ordinal number of board.

  Offset :
       The offset from the base buffer address in BYTE size.

  nCount :
       The number of DWORD to be read.

  buf :
       The buffer address to receive the data.

Return Value:
   The number of data received in DWORD size.

**AO_WriteReg**

Write register value which is related with analog output operation.

DWORD **AO_WriteReg**(int nBoard, DWORD Offset)

Parameters:

  nBoard :
       The ordinal number of board.

  Offset :
       The offset from the base buffer address in BYTE size.

Return Value:
   The value of register after writing.
AO_WriteRegMul

Write register multiple, this function writes multiple data to IO space. Some fifo are mapped to IO space. 

[note] In writing multiple data to IO space, there is no address auto-increment operation.

DWORD    AO_WriteRegMul(int nBoard, DWORD Offset, DWORD nCount, DWORD *buf)

Parameters:

    nBoard : 
        The board number.

    Offset : 
        The offset from the base buffer address in BYTE size.

    nCount : 
        The number of DWORD to be written.

    buf : 
        The buffer address to send to AI register.

Return Value:

    The number of data written to in DWORD size.
Digital Input Output API Functions

**Overview**

BOOL   DIO_Init(int nBoard)
BOOL   DIO_Reset(int nBoard)
BOOL   DIO_Close(int nBoard)
BOOL   DIO_SetInterrupt(int nBoard, BOOL bEnable, IntHandler lpISR)
DWORD DIO_ReadData(int nBoard, DWORD Offset, DWORD nCount, DWORD *buf)
DWORD DIO_WriteData(int nBoard, DWORD Offset, DWORD nCount, DWORD *buf)
DWORD DIO_ReadReg(DWORD Offset)
DWORD DIO_ReadRegMul(DWORD Offset, DWORD nCount, DWORD *buf)
DWORD DIO_WriteReg(DWORD Offset)
DWORD DIO_WriteRegMul(DWORD Offset, DWORD nCount, DWORD *buf)

**DIO_Init**

Initialize digital input output sub-system

BOOL   DIO_Init(int nBoard)

**Parameters:**

nBoard :

The ordinal number of board which you want to initialize the digital input/output sub-system.

**Return Value:**

If the function succeeds, it returns TRUE.

**DIO_Reset**

Reset digital input/output sub-system

BOOL   DIO_Reset(int nBoard)

**Parameters:**

nBoard :

The ordinal number of board which you want to reset.

**Return Value:**

If the function succeeds, it returns TRUE.

**DIO_Close**

Close digital input/output sub-system, this function will stop all digital input/output operation
BOOL  DIO_Close(int nBoard)
Parameters:
nBoard :
The ordinal number of board which you want to close.
Return Value:
If the function succeeds, it returns TRUE.

DIO_SetInterrupt
Set hander function which will be called automatically by DLL in case interrupt service is required.
BOOL  DIO_SetInterrupt(int nBoard, BOOL bEnable, IntHandler lpISR)
Parameters:
nBoard :
The board number which you want to set up the interrupt handler.
bEnable :
If TRUE, enables analog output interrupt, FALSE disables the interrupt.
lpISR :
Interrupt handler callback function.
Return Value:
If the function succeeds, it returns TRUE.

Interrupt handler function definition
typedef DWORD (CALLBACK *IntHandler)(DWORD dwStatus)
in application program, the callback function need to be defined as shown below
DWORD testHandler(DWORD dwIntStatus)
{
    ....,
}
and call the DIO_SetInterrupt like this
DIO_SetInterrupt(1, TRUE, testHandler);

DIO_ReadData
Read digital input/output data from the buffer or fifo mapped in memory space.
DWORD  DIO_ReadData(int nBoard, DWORD Offset, DWORD nCount, DWORD *buf)
Parameters:
nBoard:
The ordinal number of board.

Offset:
The offset from the base buffer address in BYTE size.

nCount:
The number of DWORD to be read.

buf:
The buffer address to receive the analog data.

Return Value:
The number of data received in DWORD size.

**DIO_WriteData**
Write data to digital input/output subsystem.

DWORD **DIO_WriteData**(int nBoard, DWORD Offset, DWORD nCount, DWORD *buf)

**Parameters:**

nBoard:
The ordinal number of board.

Offset:
The offset from the base buffer address in BYTE size.

nCount:
The number of DWORD to be sent.

buf:
The buffer address to send to digital data.

Return Value:
The number of data be sent in DWORD size.

**DIO_ReadReg**
Read register value which is related with digital input/output operation.

DWORD **DIO_ReadReg**(int nBoard, DWORD Offset)

**Parameters:**

nBoard:
The ordinal number of board.

Offset:
The offset from the base buffer address in BYTE size.
Return Value:
The value of register.

**DIO_ReadRegMul**
Read register multiple, this function reads multiple data from IO space register. Some fifo are mapped to IO space.

[note] In reading multiple data from IO space, there is no address auto-increment operation.

DWORD **DIO_ReadRegMul**(int nBoard, DWORD Offset, DWORD nCount, DWORD *buf)

**Parameters:**
- nBoard:
  The ordinal number of board.
- Offset:
  The offset from the base buffer address in BYTE size.
- nCount:
  The number of DWORD to be read.
- buf:
  The buffer address to receive the data.

**Return Value:**
The number of data received in DWORD size.

**DIO_WriteReg**
Write register value which is related with digital input/output operation.

DWORD **DIO_WriteReg**(int nBoard, DWORD Offset)

**Parameters:**
- nBoard:
  The ordinal number of board.
- Offset:
  The offset from the base buffer address in BYTE size.

**Return Value:**
The value of register after writing.
DIO_WriteRegMul

Write register multiple, this function writes multiple data to IO space. Some fifo are mapped to IO space. [note] In writing multiple data to IO space, there is no address auto-increment operation.

DWORD DIO_WriteRegMul(int nBoard, DWORD Offset, DWORD nCount, DWORD *buf)

Parameters:

    nBoard :
            The board number.

    Offset :
            The offset from the base buffer address in BYTE size.

    nCount :
            The number of DWORD to be written.

    buf :
            The buffer address to send to AI register.

Return Value:

    The number of data written to in DWORD size.