

NET-AIO11

API Manual

Version 1.0



© 2005 DAQ SYSTEM Co., Ltd. All rights reserved.

Microsoft® is a registered trademark; Windows®, Windows NT®, Windows XP®, Windows 7®, Windows 8®, Windows 10®
All other trademarks or intellectual property mentioned herein belongs to their respective owners.

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.

Contents

Board Level API Functions

OpenDAQDevice	-----	2
ClsoeDAQDevice	-----	2

Board Programming API Functions

ReadyProgLoad	-----	3
ProgLoad	-----	3
FlashProgram	-----	4
FlashErase	-----	4
FlashEthernetConfig	-----	4

Board Control API Functions

SetInterface	-----	5
GetInterface	-----	5
GetDeviceInfo	-----	6

ADC(Analog to Digital Convertor) API Functions

SetSampleRate	-----	7
GetAdcDataEX	-----	8
StartGetDataRead	-----	8
StopGetDataRead	-----	8
GetPointer	-----	9
SetDelayTime	-----	9
GetDelayState	-----	10
StartDelayedDataRead	-----	10
StopDelayedDataRead	-----	10

DIO(Digital Input/Output) API Functions

SetDOUT	-----	11
GetDIN	-----	11

Board Level API Functions

Overview

BOOL OpenDAQDevice (void)
BOOL CloseDAQDevice (void)

[OpenDAQDevice](#)

Make sure the system board is registered. Normal registered on the board can only call the function.

BOOL OpenDAQDevice (void)

Parameters: None

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.

[CloseDAQDevice](#)

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

BOOL CloseDAQDevice (void)

Parameters: None.

Return Value:

If the function fail to close, it returns "FALSE".

If the function succeed to close, it returns "TRUE".

Board Programming API Functions

Overview

BOOL **ReadyProgLoad (void);**
BOOL **ProgLoad (int nCount, unsigned char *byBuf);**
BOOL **FlashProgram (void);**
BOOL **FlashErase (void);**
BOOL **FlashEthernetConfig (WORD * data);**

ReadyProgLoad

Before the board image store in Flash memory via the USB interface, the command is ready to receive data.

BOOL **ReadyProgLoad (void)**

Parameters: None

Return Value:

If the function fail to close, it returns "FALSE".

If the function succeed to close, it returns "TRUE".

ProgLoad

The board system image transmits via the USB interface.

BOOL **ProgLoad (int nCount, unsigned char *byBuf)**

Parameters: None

nCount : Bytes of data that will be transferred

***byBuf** : Transmit Data Buffer Pointer

Return Value:

If the function fail to close, it returns "FALSE".

If the function succeed to close, it returns "TRUE".

FlashProgram

Received system image is stored in flash memory.

BOOL **FlashProgram (void)**

Parameters: None

Return Value:

If the function fail to close, it returns "FALSE".

If the function succeed to close, it returns "TRUE".

FlashErase

Initialize to flash memory.

BOOL **FlashErase (void)**

Parameters: None

Return Value:

If the function fail to close, it returns "FALSE".

If the function succeed to close, it returns "TRUE".

FlashEthernetConfig

TCP/IP Ethernet Link information stores in flash memory. Stored information is applied when the board is booted.

BOOL **FlashEthernetConfig (WORD * data)**

Parameters:

***data** : Pointer of Data included IP, MAC, GateWay, Masking data

Return Value:

If the function fail to close, it returns "FALSE".

If the function succeed to close, it returns "TRUE".

Board Control API Functions

Overview

BOOL **SetInterface (int if_type, char* strIP, int portnum);**
BOOL **GetInterface (int * if_type, int *connected);**
BOOL **GetDeviceInfo (WORD *ipaddr, WORD *submask, WORD *gateway,
 WORD *macaddr, WORD *portnum, WORD * firmware);**

SetInterface

Set the USB/Ethernet interface for data input/output.

BOOL **SetInterface (int if_type, char* strIP, int portnum)**

Parameters:

If_type : Select the Interface. 0 : USB, 1 : Ethernet
*** strip** : When Ethernet set, it is an IP address of the other boards.
portnum : Port number for using TCP/IP setup

Return Value:

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

GetInterface

Get the status of interface type and connection on board and linked system.

BOOL **GetInterface (int * if_type, int *connected)**

Parameters:

*** if_type** : Get the selected Interface
0 : USB
1 : Ethernet
*** connected** : Get the board connection status on interface
0 : Not Connection
1 : Connection

Return Value:

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

GetDeviceInfo

Get the firmware version and Ethernet setup information on the board.

```
BOOL GetDeviceInfo (WORD *ipaddr, WORD *submask, WORD *gateway,  
                   WORD *macaddr, WORD *portnum, WORD * firmware);
```

Parameters:

- ***ipaddr** : IP address is stored.
- ***submask** : Sub-net Mask information is stored.
- ***gateway** : Gateway Address is stored.
- ***macaddr** : MAC address information is stored
- ***portnum** : TCP/IP port information is stored.
- ***firmware** : Firmware version information is stored.

Return Value:

If the function fail to close, it returns "FALSE".

If the function succeed to close, it returns "TRUE".

ADC(Analog to Digital Convertor) API Functions

Overview

```
BOOL      SetSamplerate (DWORD sample);
BOOL      GetAdcDataEX (int *ad_data);
BOOL      StartGetDataRead (DWORD data_count, DWORD trigger_en,
                           DWORD sync_en, DWORD count);
BOOL      StopGetDataRead (void);
BOOL      GetPointer (DWORD *rd_ptr, DWORD *wr_ptr, DWORD *rx_data_count,
                     DWORD *adc_len);
BOOL      SetDelayTime (DWORD delay);
BOOL      GetDelayState (DWORD *run_flag, DWORD *mem_end);
BOOL      StartDelayedDataRead (DWORD data_count);
BOOL      StopDelayedDataRead (void);
```

SetSamplerate

Set the ADC Data Sampling Rate.

BOOL SetSamplerate (DWORD sample)

Parameters:

sample : Enter the Sampling Rate Code.

0 -> 65,536sps, 1 -> 32,768sps

Sampling Rate[sps] = 65,536/(2^{Setup Value})

Return Value:

If the function fail to close, it returns "FALSE".

If the function succeed to close, it returns "TRUE".

GetAdcDataEX

Read continuous mode ADC data.

BOOL GetAdcDataEX (int *ad_data)

Parameters:

* **ad_data** : Data Buffer Pointer. The number of stored data is equal to the value of StartGetDataRead() first argument.

Return Value:

StartGetDataRead () the first argument does not exist, "FALSE", otherwise "TRUE" is returned.

StartGetDataRead

Start the continuous mode data collection.

BOOL StartGetDataRead (DWORD data_count, DWORD trigger_en, DWORD sync_en, DWORD count)

Parameters:

data_count : The number of data is to be read at a time. Support to max. 4,194,034(4M).

trigger_en : Reserved

sync_en : Reserved

count : Reserved

Return Value:

If the function fail to close, it returns "FALSE".

If the function succeed to close, it returns "TRUE".

StopGetDataRead

Stop the continuous mode data collection.

BOOL StopGetDataRead (void)

Parameters:

Return Value:

If the function fail to close, it returns "FALSE".

If the function succeed to close, it returns "TRUE".

GetPointer

Make sure the buffering memory pointer and the number of USB/Ethernet data.

BOOL **GetPointer (DWORD *rd_ptr, DWORD *wr_ptr, DWORD *rx_data_count,
 DWORD *adc_len);**

Parameters:

***rd_ptr** : Current Memory Pointer to read data buffer from the Application

***wr_ptr** : Current Memory Pointer to write data buffer

***rx_data_count** : Number of all data(byte) from USB/Ethernet

***adc_len** : Number of ADC data(16-bit) from USB/Ethernet

Return Value:

If the function fail to close, it returns "FALSE".

If the function succeed to close, it returns "TRUE".

SetDelayTime

Set the waiting time before ADC data sampling of continuous mode.

BOOL **SetDelayTime (DWORD delay)**

Parameters:

delay : Delay time can be set to 255 seconds.

Return Value:

If the function fail to close, it returns "FALSE".

If the function succeed to close, it returns "TRUE".

GetDelayState

Check the standby delay.

After waiting standby delay time, check the data collection finished.

BOOL GetDelayState (DWORD *run_flag, DWORD *mem_end)

Parameters:

***run_flag** : It is a Flag to inform that more 524,288 data(2seconds data of 32,768sp) collection finished, after standby. If progress is TRUE, the end is FALSE.

***mem_end** : Memory Pointer to know the number of the collected data

Return Value:

If the function fail to close, it returns "FALSE".

If the function succeed to close, it returns "TRUE".

StartDelayedDataRead

Command to read collected AD data in delay mode.

BOOL StartDelayedDataRead (DWORD data_count)

Parameters:

data_count : Number of data unit to get the data from application

Return Value:

If the function fail to close, it returns "FALSE".

If the function succeed to close, it returns "TRUE".

StopDelayedDataRead

Stop the reading continuous mode AD data.

BOOL StopDelayedDataRead (void)

Parameters:

Return Value:

If the function fail to close, it returns "FALSE".

If the function succeed to close, it returns "TRUE".

DIO(Digital Input/Output) API Functions

Overview

BOOL **SetDOUT (DWORD dout_val);**

BOOL **GetDIN(DWORD *din_val);**

SetDOUT

Set the Digital Output.

BOOL **SetDOUT (DWORD dout_val)**

Parameters:

dout_val : It is value of the output setting, only 8-bit low byte is valid.

When set to 1, the DOUT pin and DOUT_COM will be disconnected (short).

Return Value:

If the function fail to close, it returns "FALSE".

If the function succeed to close, it returns "TRUE".

GetDIN

Reads the digital input states.

BOOL **GetDIN (DWORD *din_val)**

Parameters:

***din_val** : The lower 8-bit value of the input pin is valid.

If DIN is "1", the current is flowing state because DIN_COM and DIN loop is formed.

bit	31	7	
		8	0
data	Not Use		DIN[7:0]

Return Value:

If the function fail to close, it returns "FALSE".

If the function succeed to close, it returns "TRUE".

Memo

Contact Point

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

Email : postmaster@daqsystem.com

