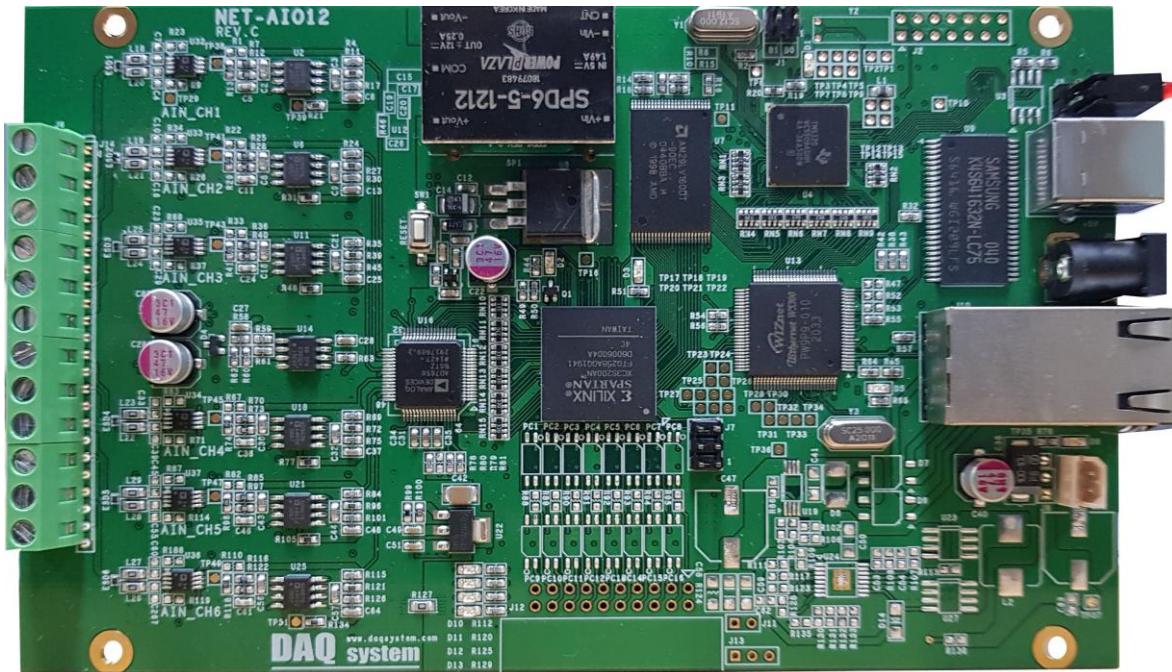


NET-AIO12

API Manual

Version 1.1



© 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

Memory Access API Functions

SYS_ReadyProgLoad	-----	2
SYS_ProgLoad	-----	2
SYS_FlashProgram	-----	3
SYS_FlashErase	-----	3
SYS_FlashEthernetConfig	-----	3
SYS_GetDeviceInfo	-----	4

Interface API Functions

OpenDAQDevice	-----	5
ClsoeDAQDevice	-----	5
SYS_SetInterface	-----	6
SYS_GetInterface	-----	6
SYS_OpenNetPort	-----	7
SYS_CloseNetPort	-----	7

AIO(Analog Input Output) API Functions

AIO_SetSamplerate	-----	8
AIO_SetConfig	-----	9
AIO_GetAdcDataEX	-----	9
AIO_StartGetFDataRead	-----	10
AIO_StopGetDataRead	-----	10
AIO_SetDelayTime	-----	10
AIO_GetDelayState	-----	11
AIO_StartDelayedDataRead	-----	11
AIO_StopDelayedDataRead	-----	11
AIO_GetPointer	-----	12

DIO(Digital Input/Output) API Functions

DIO_SetDOUT	-----	13
SDIO_GetDIN	-----	13

Memory Access API Functions

Overview

```

BOOL      SYS_ReadyProgLoad(void);
BOOL      SYS_ProgLoad(int nCount, unsigned char *byBuf);
BOOL      SYS_FlashProgram(void);
BOOL      SYS_FlashErase(void);
BOOL      SYS_FlashEthernetConfig(WORD * data);
BOOL      SYS_GetDeviceInfo(WORD *ipaddr, WORD *submask, WORD *gateway,
                           WORD *macaddr, WORD *portnum, WORD * firmware);

```

SYS_ReadyProgLoad

BOOL SYS_ReadyProgLoad(void)

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

Parameters:

Return Value:

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

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

SYS_ProgLoad

BOOL SYS_ProgLoad(int nCount, unsigned char *byBuf)

The board system image transmits via the USB interface.

Parameters:

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".

SYS_FlashProgram

BOOL **SYS_FlashProgram(void)**

Received system image is stored in flash memory.

Parameters: None

Return Value:

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

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

SYS_FlashErase

BOOL **SYS_FlashErase(void)**

Initialize to flash memory.

Parameters: None

Return Value:

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

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

SYS_FlashEthernetConfig

BOOL **SYS_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".

SYS_GetDeviceInfo

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

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

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".

Interface API Functions

Overview

BOOL	OpenDAQDevice(void)
BOOL	CloseDAQDevice(void)
BOOL	SYS_SetInterface(int if_type, char* strIP, int portnum);
BOOL	SYS_GetInterface(int * if_type, int *connected);
BOOL	SYS_OpenNetPort(char* strIP, int portnum)
BOOL	SYS_CloseNetPort(void)

OpenDAQDevice

BOOL **OpenDAQDevice(void)**

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

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

BOOL **CloseDAQDevice(void)**

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.

Parameters: None.

Return Value:

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

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

SYS_SetInterface

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

This function selects USB/Ethernet interface for connection between system and board and data input/output.

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".

SYS_GetInterface

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

This function gets the interface type and connection status of the system that is linked with the board.

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".

SYS_OpenNetPort

BOOL **SYS_OpenNetPort(char* strIP, int portnum)**

This function gets the IP and port number of the network interface that is linked with the board.

Parameters:

* **strIP**: Get the set IP.

* **portnum**: Get the corresponding port number.

Return Value:

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

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

SYS_CloseNetPort

BOOL **SYS_CloseNetPort(void)**

This function closes the network interface IP and port that is linked with the board.

Parameters:

* **strIP** : Get the set IP.

* **portnum** : Get the corresponding port number.

Return Value:

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

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

AIO(Analog Input Output) API Functions

Overview

BOOL	AIO_SetSamplerate(int sample);
BOOL	AIO_SetConfig(int dwChBit, int nRangeBit);
BOOL	AIO_GetAdcDataEX(short *ad_data);
BOOL	AIO_StartGetDataRead(int data_count, int trigger_en, int sync_en, int count);
BOOL	AIO_StopGetDataRead(void);
BOOL	AIO_SetDelayTime(int delay);
BOOL	AIO_GetDelayState(int *run_flag, int *mem_end);
BOOL	AIO_StartDelayedDataRead(int data_count);
BOOL	AIO_StopDelayedDataRead(void);
BOOL	AIO_GetPointer(int *rd_ptr, int *wr_ptr, int *rx_data_count,int *adc_len);

AIO_SetSamplerate

BOOL AIO_SetSamplerate(int sample)

This function sets the ADC Data Sampling Rate.

Parameters:

sample: Enter a sampling rate between 256 and 230,000spS.

The number of channel selections per interface and the maximum sampling rate are as follows

- Up to 230ksps when selecting TCP/IP 1 or 2 channels
 - Up to 220ksps with 3-channel selection
 - Up to 170ksps with 4-channel selection
 - Up to 140ksps with 5-channel selection
 - Up to 120ksps with 6-channel selection
- Up to 75ksps with USB 1-channel selection
 - Up to 70ksps with 2-channel selection
 - Up to 65ksps with 3-channel selection
 - Up to 60ksps with 4-channel selection
 - Up to 55ksps with 5-channel selection
 - Up to 50ksps with 6-channel selection

Return Value:

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

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

AIO_SetConfig

BOOL **AIO_SetConfig(int dwChBit, int nRangeBit)**

This function selects ADC channel and range value.

Parameters:

dwChBit : Channel Selection Bit Value.

Bit No.	7	6	5	4	3	2	1	0
Ch. No.	-	-	CH 6	CH 5	CH 4	CH 3	CH 2	CH 1

When 6-channel is selected, the bit value is '0x3F', and when channels 1, 3 and 5 are selected, the bit value is '0x2A'.

nRangeBit: ADC analog signal input range.

0: ±10V 1: ±5V

Return Value:

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

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

AIO_GetAdcDataEX

BOOL **AIO_GetAdcDataEX(short *ad_data)**

This function reads continuous mode ADC data.

Parameters:

* **ad_data** : It is a 16-bit short data buffer pointer. The number of data read is the same as the value of the first parameter of StartGetDataRead().

Return Value:

If there is no data to be read more than the value of the first parameter of StartGetDataRead(), "FALSE" is returned. Otherwise, "TRUE" is returned.

AIO_StartGetDataRead

BOOL **AIO_StartGetDataRead(int data_count, int trigger_en, int sync_en, int count)**

This function initiates continuous mode data collection as a start command and initializes the buffer pointer, etc.

Parameters:

data_count : This is the number of data to be read at one time. It supports up to 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".

AIO_StopGetDataRead

BOOL **AIO_StopGetDataRead(void)**

This function stops continuous mode data collection.

Parameters:

Return Value:

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

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

AIO_SetDelayTime

BOOL **AIO_SetDelayTime(int delay)**

This function sets the waiting time before sampling ADC data in delay mode.

Parameters:

delay : Set up to 255 seconds as a delay time in seconds.

Return Value:

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

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

AIO_GetDelayState

BOOL **AIO_GetDelayState(int *run_flag, int *mem_end)**

This function checks the delay wait state. Wait for the waiting time and check if data collection is finished.

Parameters:

***run_flag** : This flag indicates whether the collection of 524,288 data (2 seconds data in case of 32,768 sps) has been completed after waiting. TRUE if in progress, FALSE if terminated.

***mem_end** : It is a memory pointer that shows the number of collected data.

Return Value:

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

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

AIO_StartDelayedDataRead

BOOL **AIO_StartDelayedDataRead(int data_count)**

This function stops reading AD data in delay mode.

Parameters:

data_count : This is the number of data units read from the board to the application program.

Return Value:

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

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

AIO_StopDelayedDataRead

BOOL **AIO_StopDelayedDataRead(void)**

This function stops reading AD data in delay mode.

Parameters:

Return Value:

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

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

AIO_GetPointer

BOOL **AIO_GetPointer(DWORD *rd_ptr, int *wr_ptr, int *rx_data_count, int *adc_len);**

This function checks the number of USB/Ethernet data received and buffered memory pointers during data acquisition.

Parameters:

- ***rd_ptr** : This is the current memory pointer that reads the data buffer from the application program.
- ***wr_ptr** : This is the current memory pointer that the application program reads the data buffer.
- ***rx_data_count** : This is the number of all data input from USB/Ethernet (bytes).
- ***adc_len** : This is the number of ADC data (bytes) input from USB/Ethernet.

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 **DIO_SetDOUT(int dout_val);**
 BOOL **DIO_GetDIN(int *din_val);**

DIO_SetDOUT

BOOL **DIO_SetDOUT(int dout_val)**

This function sets the digital output.

Parameters:

dout_val : As an output setting value, only the 8-bit lower byte is valid.

When 1 is set, the corresponding DOUT pin and DOUT_COM are shorted.

Return Value:

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

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

DIO_GetDIN

BOOL **DIO_GetDIN (int *din_val)**

This function reads the digital input status.

Parameters:

***din_val** : Only the lower 8-bit is valid as an input pin value.

When DIN 1 is read, DIN_COM and DIN are looped and current flows.

bit	31	8	7	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

