

**AIO-3322/3323**

**Isolated Analog I/O and  
Digital I/O Card**

**Software Manual (V1.0)**

**健昇科技股份有限公司**

**JS AUTOMATION CORP.**

台北縣汐止市中興路 100 號 6 樓

6F, No. 100, Chungshin Rd.

Shitsu, Taipei, Taiwan, R.O.C.

TEL : 886-2-2647-6936

FAX : 886-2-2647-6940

<http://www.automation.com.tw>

E-mail : [control.cards@automation.com.tw](mailto:control.cards@automation.com.tw)

# Correction record

| Manual Version | Record             |
|----------------|--------------------|
| 1.0            | for driver V1.0 up |

# Contents

|     |   |    |
|-----|---|----|
| 1.  | How to install the software of AIO3322 .....                            | 5  |
| 1.1 | Install the PCI driver.....   | 5  |
| 2.  | Where to find the file you need.....                                    | 6  |
| 3.  | About the AIO-3322 software.....  | 7  |
| 3.1 | What you need to get started.....                                       | 7  |
| 3.2 | Software programming choices .....                                      | 7  |
| 4.  | AIO3322 Language support.....   | 8  |
| 4.1 | Building applications with the AIO3322 software library.....            | 8  |
| 4.2 | AIO3322 Windows libraries .....   | 8  |
| 5.  | Software overview .....   | 9  |
| 5.1 | Initialization and close.....   | 9  |
| 5.2 | Analog input .....  | 9  |
| 5.3 | Auto data acquisition .....   | 9  |
| 5.4 | Analog output .....   | 10 |
| 5.5 | Digital I/O.....  | 11 |
| 5.6 | Counter / Timer function .....  | 11 |
| 5.7 | Interrupt function .....  | 12 |
| 5.8 | Security function.....  | 12 |
| 5.9 | Error conditions .....  | 12 |
| 6.  | Flow chart of implement an application.....                             | 13 |
| 6.1 | AIO3322/3323 Flow chart of implementation.....                          | 13 |
| 6.2 | AIO3322/3323 Flow chart of auto data acquisition (Digital trigger)..... | 14 |
| 6.3 | AIO3322/3323 Flow chart of auto data acquisition (Analog trigger) ..... | 15 |
| 6.4 | AIO3322/3323 Flow chart of Timer / Counter / PWM application.....       | 16 |
| 6.5 | Gated counter application .....   | 17 |
| 6.6 | AIO3322/3323 Flow chart of interrupt.....                               | 18 |
| 7.  | Function reference.....   | 19 |
| 7.1 | Error codes and CardID .....  | 19 |
| 7.2 | Variable data types .....   | 20 |
| 7.3 | Programming language considerations .....                               | 21 |
| 7.4 | AIO3322 Functions .....   | 23 |
|     | Initialization and close.....   | 23 |
|     | aio3322_initial .....   | 23 |
|     | aio3322_close .....   | 23 |
|     | aio3322_info .....  | 23 |
|     | aio3322_get_DeviceHandle.....   | 23 |
|     | Analog input .....  | 24 |
|     | aio3322_smart_AtoD.....   | 24 |
|     | aio3322_set_AD_command .....  | 24 |

|   |    |
|---|----|
| aio3322_start_AD_conversion .....             | 25 |
| aio3322_read_AD_status.....                   | 25 |
| aio3322_read_AD_data .....                    | 25 |
| Auto data acquisition .....                   | 26 |
| aio3322_set_DAQ_sampling_rate.....            | 26 |
| aio3322_set_Digital_trigger_DAQ.....          | 26 |
| aio3322_set_Analog_trigger_DAQ.....           | 27 |
| aio3322_set_Software_trigger_DAQ .....        | 28 |
| aio3322_read_DAQ_mode .....                   | 28 |
| aio3322_config_Analog_Trigger_condition.....  | 29 |
| aio3322_set_Trigger_IN_polarity.....          | 29 |
| aio3322_read_Trigger_IN_status .....          | 30 |
| aio3322_config_Trigger_OUT_mode .....         | 30 |
| aio3322_set_Trigger_OUT_polarity.....         | 31 |
| aio3322_load_Trigger_OUT_pulse_duration.....  | 31 |
| aio3322_write_Trigger_OUT .....               | 31 |
| aio3322_readback_Trigger_OUT_status.....      | 32 |
| aio3322_set_Trigger_DAQ_retaining_count ..... | 32 |
| aio3322_enable_Digital_trigger_DAQ.....       | 33 |
| aio3322_enable_Analog_trigger_DAQ.....        | 33 |
| aio3322_start_Software_trigger_DAQ.....       | 33 |
| aio3322_release_trigger_DAQ.....              | 34 |
| aio3322_read_DAQ_finish_flag.....             | 34 |
| aio3322_read_DAQ_value .....                  | 34 |
| aio3322_read_DAQ_RAM_data .....               | 35 |
| aio3322_read_DAQ_cycling_flag.....            | 35 |
| Analog output .....                           | 36 |
| aio3322_out_DA_data.....                      | 36 |
| Digital I/O.....                              | 37 |
| aio3322_read_port .....                       | 37 |
| aio3322_read_point .....                      | 37 |
| aio3322_set_port.....                         | 38 |
| aio3322_set_point.....                        | 38 |
| aio3322_set_dedicate_IO.....                  | 38 |
| aio3322_readback_dedicate_IO_status .....     | 39 |
| Counter / Timer function .....                | 40 |
| aio3322_set_timer.....                        | 40 |
| aio3322_set_counter .....                     | 41 |
| aio3322_set_pwm .....                         | 42 |
| aio3322_set_clock_frequency .....             | 42 |
| aio3322_load_GPT .....                        | 43 |

|   |    |
|---|----|
| aio3322_GPT_enable.....                   | 43 |
| aio3322_read_GPT .....                    | 44 |
| aio3322_one_shot_command .....            | 44 |
| aio3322_read_parameter.....               | 45 |
| Interrupt function .....                  | 46 |
| aio3322_enable_IRQ .....                  | 46 |
| aio3322_link_IRQ_process .....            | 46 |
| aio3322_disable_IRQ .....                 | 46 |
| aio3322_read_IRQ_status.....              | 47 |
| Security function.....                    | 48 |
| aio3322_set_password.....                 | 48 |
| aio3322_change_password .....             | 48 |
| aio3322_clear_password.....               | 48 |
| aio3322_unlock_security .....             | 49 |
| aio3322_read_security_status.....         | 49 |
| 7.5 Dll list .....                        | 50 |
| 8. AIO-3322/3323 Error codes summary..... | 52 |
| 8.1 AIO3322/3323 Error codes table.....   | 52 |

# 1. **How to install the software of AIO3322**

## 1.1 Install the PCI driver

The PCI card is a plug and play card, once you add a new card, the window system will detect while it is booting. Please follow the following steps to install your new card.

In Windows 98/2000/XP system you should: (take win98 as example)

1. Make sure the power is off
2. Plug in the interface card
3. Power on
4. A hardware install wizard will appear and tell you it finds a new PCI card
5. Tell the wizard the directory of the driver files (..\aio3322\software\win98\_2k\_me\driver or if you download from website, please execute the self-unzip file aio3322\_driver.exe to get the file), then it will automatically setup the driver
6. After installation, power off
7. Power on, it's ready to use

**Note: AIO3322, AIO3323 use the same driver and dll.**

For more detail of step by step installation guide, please refer the file "installation.pdf" on the CD come with the product or register as a member of our user's club at:

<http://automation.com.tw/>

to download the complementary documents.

## 2. **Where to find the file you need**

### **Windows98, 2000, XP**

In Windows 98, 2000,XP system, the demo program can be setup by

\aio3322\software\win98\_2k\_me\install\Setup3322VBdemo.exe and header files setup by

\aio3322\software\win98\_2k\_me\install\Setup3322API.exe

The directory will be located at

**../ JS Automation /AIO3322/Api** (header files and VB,VC lib files)

**../ JS Automation /AIO3322/Example** (easy source code of initial user)

**../ JS Automation /AIO3322/exe** (demo program and source code)

The system driver is located at ../system32/Driver and the DLL is located at ../system.

For your easy startup, the demo program of the cards function and help file can be setup by

\aio3322\software\nt\install\Setup3322VBdemo.exe

### **3. About the AIO-3322 software**

AIO3322 software includes a set of dynamic link library (DLL) and system driver that you can utilize to control the I/O card's function separately.

Your AIO3322 software package includes setup driver, tutorial example and test program that help you how to setup and run appropriately, as well as an executable file which you can use to test each of the AIO3322 functions within Windows' operation system environment.

#### 3.1 What you need to get started

To set up and use your AIO3322 software, you need the following:

- AIO3322 software
- AIO3322 hardware
  - Main board
  - Wiring board (Option)

#### 3.2 Software programming choices

You have several options to choose from when you are programming AIO3322 software. You can use Borland C/C++, Microsoft Visual C/C++, Microsoft Visual Basic, or any other Windows-based compiler that can call into Windows dynamic link libraries (DLLs) for use with the AIO3322 software.

## 4. **AIO3322 Language support**

The AIO3322 software library is a DLL used with Windows 98/2000/XP. You can use these DLL with any Windows integrating development environment that can call Windows DLLs.

### 4.1 Building applications with the AIO3322 software library

The AIO3322 function reference topic contains general information about building AIO3322 applications, describes the nature of the AIO3322 files used in building AIO3322 applications, and explains the basics of making applications using the following tools:

#### **Applications tools**

- ◆ **Borland C/C++**
- ◆ **Microsoft Visual C/C++**
- ◆ **Microsoft Visual Basic**

If you are not using one of the tools listed, consult your development tool reference manual for details on creating applications that call DLLs.

### 4.2 AIO3322 Windows libraries

The AIO3322 for Windows function library is a DLL called **aio3322.dll**. Since a DLL is used, AIO3322 functions are not linked into the executable files of applications. Only the information about the AIO3322 functions in the AIO3322 import libraries is stored in the executable files. Import libraries contain information about their DLL-exported functions. They indicate the presence and location of the DLL routines. Depending on the development tools you are using, you can make your compiler and linker aware of the DLL functions through import libraries or through function declarations.

Refer to **Table 1** to determine to which files you need to link and which to include in your development to use the AIO3322 functions in aio3322.dll.

| Header Files and Import Libraries for Different Development Environments |                    |                       |
|--|--------------------|-----------------------|
| <b>Development Environment</b>   | <b>Header File</b> | <b>Import Library</b> |
| <b>Microsoft C/C++</b>   | aio3322.h          | aio3322vc.lib         |
| <b>Borland C/C++</b>   | aio3322.h          | aio3322bc.lib         |
| <b>Microsoft Visual Basic</b>  | aio3322.bas        |                       |

**Table 1**

## 5. Software overview

These topics describe the features and functionality of the AIO3322 boards and briefly describes the AIO3322 functions.

### 5.1 Initialization and close

You need to initialize system resource each time you run your application.

[\*aiio3322 initial\(\)\*](#) will do.

Once you want to close your application, call

[\*aiio3322 close\(\)\*](#) to release all the resource.

If you want to know the physical address assigned by OS. use

[\*aiio3322 info\(\)\*](#) to get the address.

If you want to get the device handle you just opened use

[\*aiio3322 get DeviceHandle\(\)\*](#)

### 5.2 Analog input

If you are a green hand of A/D converter,

[\*aiio3322 smart AtoD\(\)\*](#) is a good selection to read your A/D. It will return the scaled conversion data according your selection range.

Expert as you are, first apply

[\*aiio3322 set AD command\(\)\*](#) to select channel , range.

[\*aiio3322 start AD conversion\(\)\*](#) to start A/D conversion.

Then check the A/D status to see if it is conversion ready,

[\*aiio3322 read AD status\(\)\*](#) will do.

If it is ready , read the conversion data for your application.

[\*aiio3322 read AD data\(\)\*](#).

### 5.3 Auto data acquisition

For some applications, you want to get data at certain period while certain event occurs, the hardware implemented data acquisition mode is a good choice. AIO3322/3323 has build-in 1024 hardware data buffers and programmable sampling rate.

[\*aiio3322 set DAO sampling rate\(\)\*](#) to setup the data acquisition rate

To start the auto data acquisition operation, there are 3 trigger mode

[\*aiio3322 set Digital trigger DAO\(\)\*](#) to setup digital trigger mode

[\*aiio3322 set Analog trigger DAO\(\)\*](#) to setup analog trigger mode

[\*aiio3322 set Software trigger DAO\(\)\*](#) to setup software trigger mode.

[\*aiio3322 read DAO mode\(\)\*](#) to retrieve the mode set.

If you will use analog trigger mode, use

[\*aiio3322 config Analog Trigger condition\(\)\*](#) to setup trigger condition.

If you will use digital trigger mode, use

[\*ai03322 set Trigger IN polarity\(\)\*](#) to set up the trigger polarity.

You can also read the digital trigger input status by

[\*ai03322 read Trigger IN status\(\)\*](#).

No matter what trigger mode you use, the trigger output is available for you to signal complete of records or analog trigger event occurs, it can also configured as general output

[\*ai03322 config Trigger OUT mode\(\)\*](#) will do.

The trigger out polarity can also be configured by using

[\*ai03322 set Trigger OUT polarity\(\)\*](#).

The output pulse duration can be configured by

[\*ai03322 load Trigger OUT pulse duration\(\)\*](#).

Trigger output can be used as general output, and control its output state by

[\*ai03322 write Trigger OUT\(\)\*](#).

The output status can be read back, no matter what mode it is by

[\*ai03322 readback Trigger OUT status\(\)\*](#).

To configure how many buffers will record the data after trigger, use

[\*ai03322 set Trigger DAO retaining count\(\)\*](#) to set the buffer size and the remainders are used to record the data after the trigger event.

After all the preparatory work is done, use

[\*ai03322 enable Digital trigger DAO\(\)\*](#) to start waiting digital trigger

[\*ai03322 enable Analog trigger DAO\(\)\*](#) to start waiting analog trigger, and

[\*ai03322 start Software trigger DAO\(\)\*](#) to start software trigger (start DAQ immediately).

At ant condition if you will reset the auto DAQ mode to normal mode, use

[\*ai03322 release trigger DAO\(\)\*](#).

To check if the data buffers are full (DAQ complete),

[\*ai03322 read DAO finish flag\(\)\*](#) will report the result.

To read the converted to scale data, use

[\*ai03322 read DAO value\(\)\*](#).

To read the raw data DAQ acquired,

[\*ai03322 read DAO RAM data\(\)\*](#) will do.

If the sampling rate is low, to fill the buffers will take several seconds. To change DAQ mode is not allowed while DAQ is an action, use

[\*ai03322 read DAO cycling flag\(\)\*](#) to read the cycling flag, if it is active, the hard ware is in active state of DAQ, you can not change operation mode.

#### 5.4 Analog output

The analog output is as simple as digital output

[\*ai03322 out DA data\(\)\*](#) will output the data to selected channel.

## 5.5 Digital I/O

There are one byte isolated digital input and one byte digital output on card. To read a port data, use

[\*aio3322 read port\(\)\*](#)

and to read one bit use

[\*aio3322 read point\(\)\*](#).

To output a port data, use

[\*aio3322 set port\(\)\*](#)

and output bit data

[\*aio3322 set point\(\)\*](#) will do.

For the timer/counter applications, the digital I/O also provide the timer/counter I/O function.

[\*aio3322 set dedicate IO\(\)\*](#) can be used to change the IN0,IN1 as general or timer/counter dedicated input,OUT0,OUT1 as general or timer/counter dedicated output. To read back status of dedicated I/O use:

[\*aio3322 readback dedicate IO status\(\)\*](#)

## 5.6 Counter / Timer function

There are 3 major functions of counter/timer function—to work as timer, as counter or as PWM generator. To configure the working mode use

[\*aio3322 set timer\(\)\*](#) to configure as timer and its output mode

[\*aio3322 set counter\(\)\*](#) to configure as counter and its input and output mode

[\*aio3322 set pwm\(\)\*](#) to configure as PWM generator.

Since the timer /counter has multiple clock source to choose,

[\*aio3322 set clock frequency\(\)\*](#) can be used to choose the 4MHz or 33MHz clock source.

After you configure the working mode, a constant must load into counter/timer register and the pre-load register use

[\*aio3322 load GPT\(\)\*](#) will do.

Once you have setup the data

[\*aio3322 GPT enable\(\)\*](#) will enable or disable the counter/timer function.

To check the current counter/timer value

[\*aio3322 read GPT\(\)\*](#) will do.

The special function of one shot mode can be applied by

[\*aio3322 one shot command\(\)\*](#).

To read back the parameters you set

[\*aio3322 read parameter\(\)\*](#) will response the current working mode and the parameters you set for individual timer, counter.

## 5.7 Interrupt function

Interrupt is an efficient method to quick response without occupy too much system resource.

AIO3322 provide timer/counter interrupts, to use interrupt function use

[\*ai03322 enable IRQ\(\)\*](#) to enable it and

[\*ai03322 link IRQ process\(\)\*](#) to link your irq service routine,

[\*ai03322 disable IRQ\(\)\*](#) to disable it.

After you enable and link interrupt, you can enable/disable timer/counter function or enable/disable interrupt function as you need.

To check the irq status

[\*ai03322 read IRQ status\(\)\*](#) will do.

## 5.8 Security function

Since AIO3322 is a general purpose card, anyone who can buy from JS Automation Corp. or her distributors. Your program is the fruit of your intelligence, un-authorized copy maybe prevent by the security function enabled.

You can use

[\*ai03322 set password\(\)\*](#) to set password and start the security function. Use

[\*ai03322 change password\(\)\*](#) to change it.

If you don't want to use security function after the password being setup,

[\*ai03322 clear password\(\)\*](#) will reset to the virgin state.

Once the password is set, any function call of the dll's (except for the security functions) will be blocked until the

[\*ai03322 unlock security\(\)\*](#) unlock the security.

You can also use

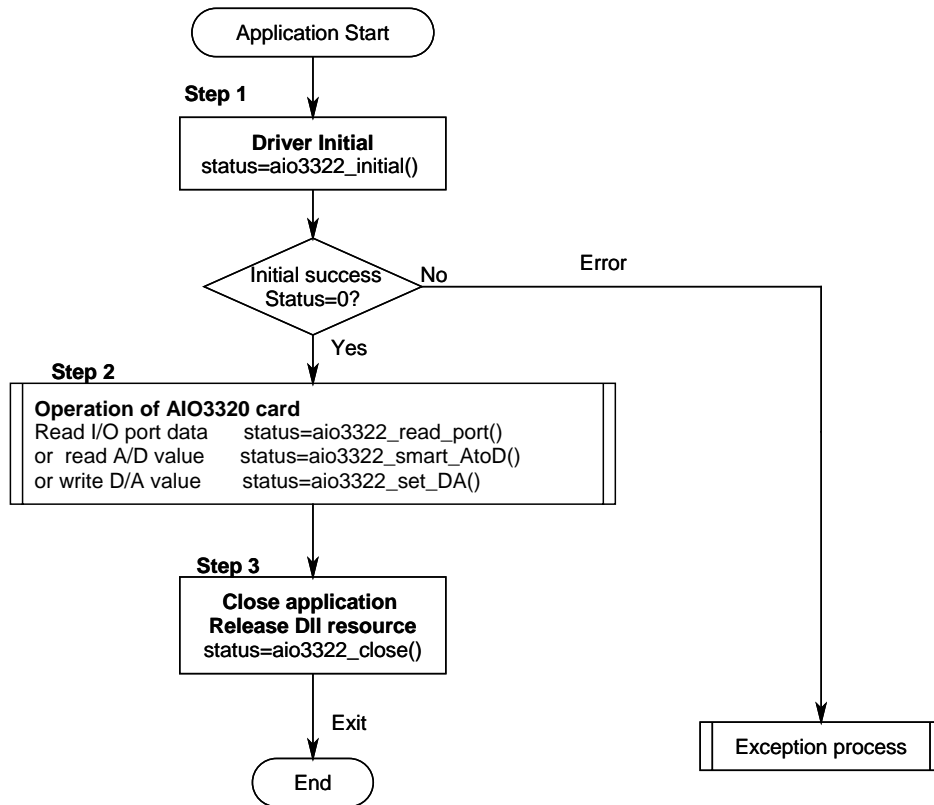
[\*ai03322 read security status\(\)\*](#) to check the current status of security.

## 5.9 Error conditions

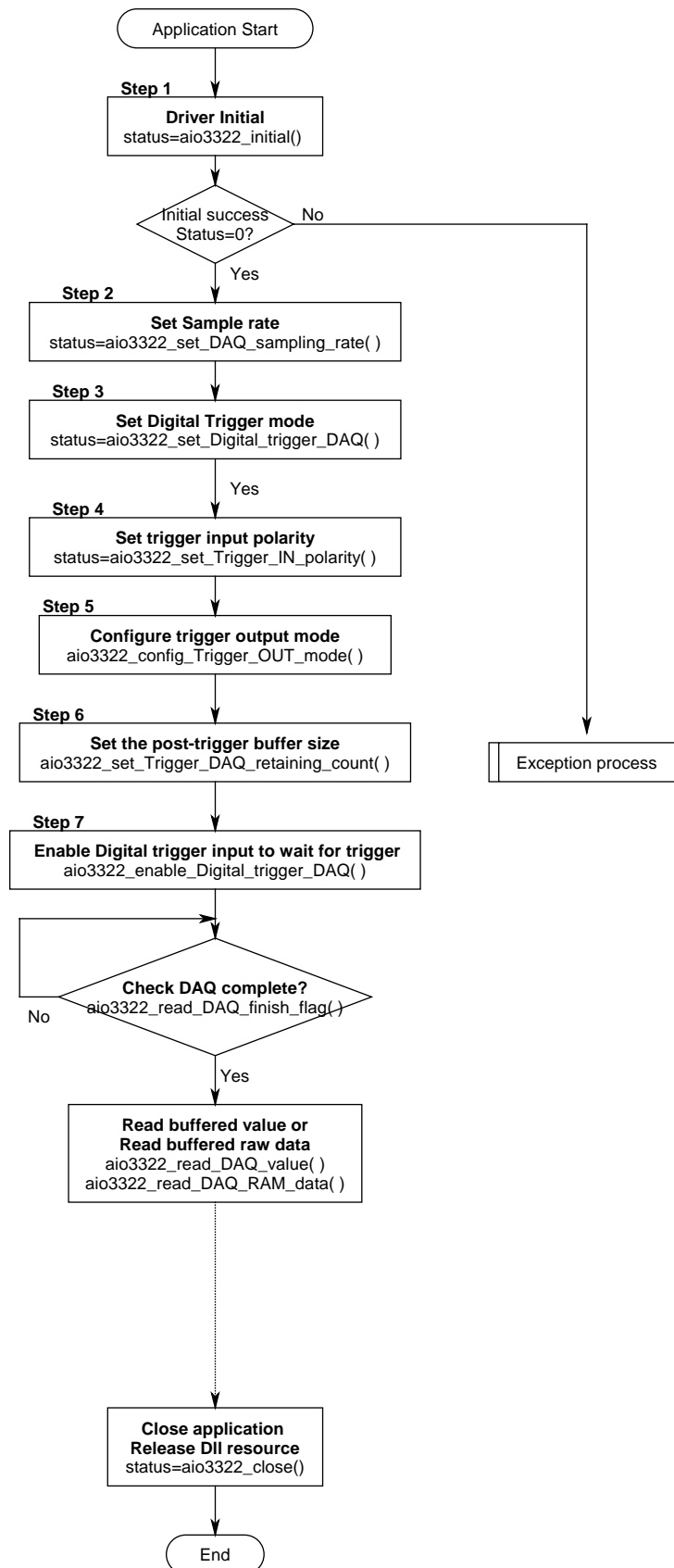
These error types may indicate an internal hardware problem on the board. Error Codes summary contains a detailed listing of the error status returned by AIO3322 functions.

## 6. Flow chart of implement an application

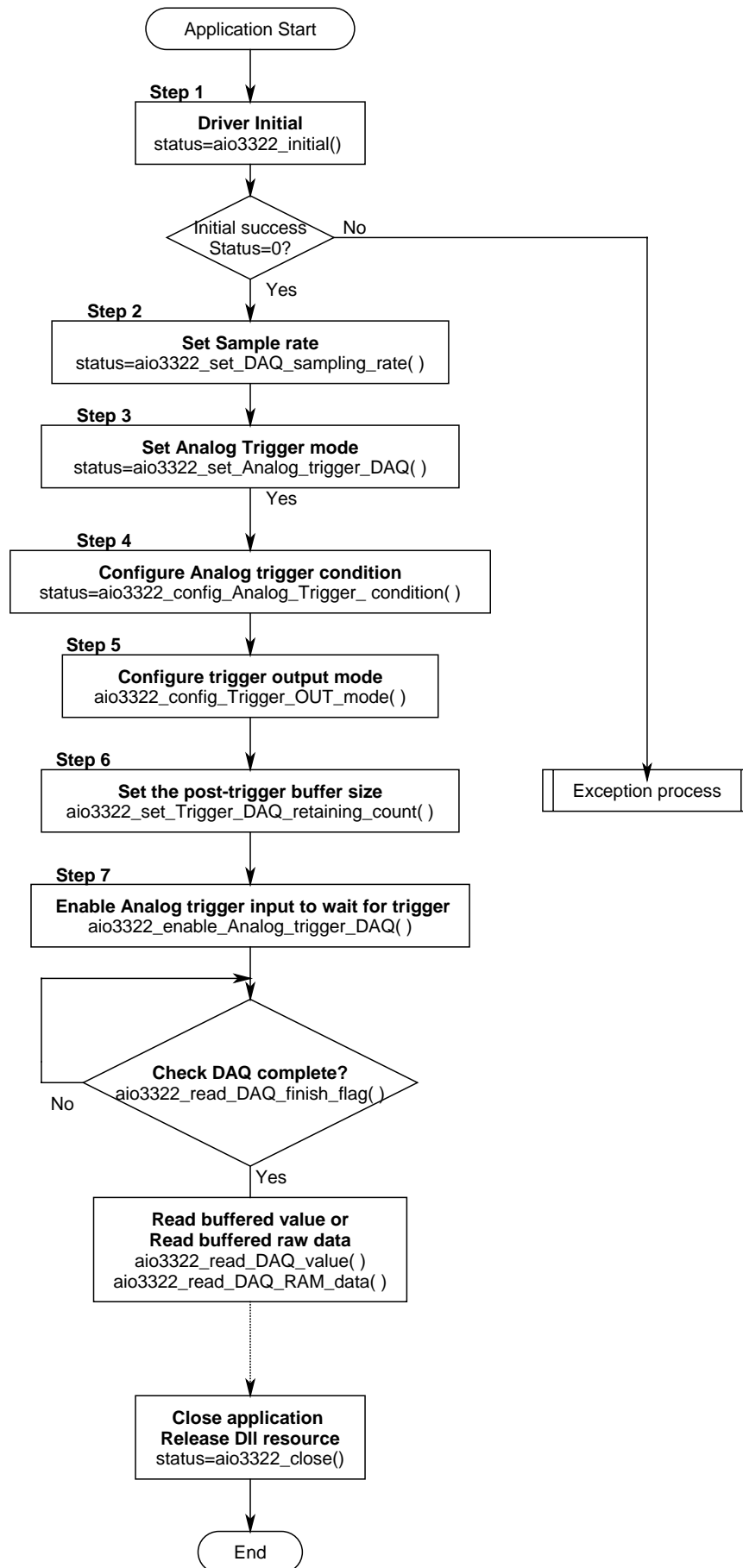
### 6.1 AIO3322/3323 Flow chart of implementation



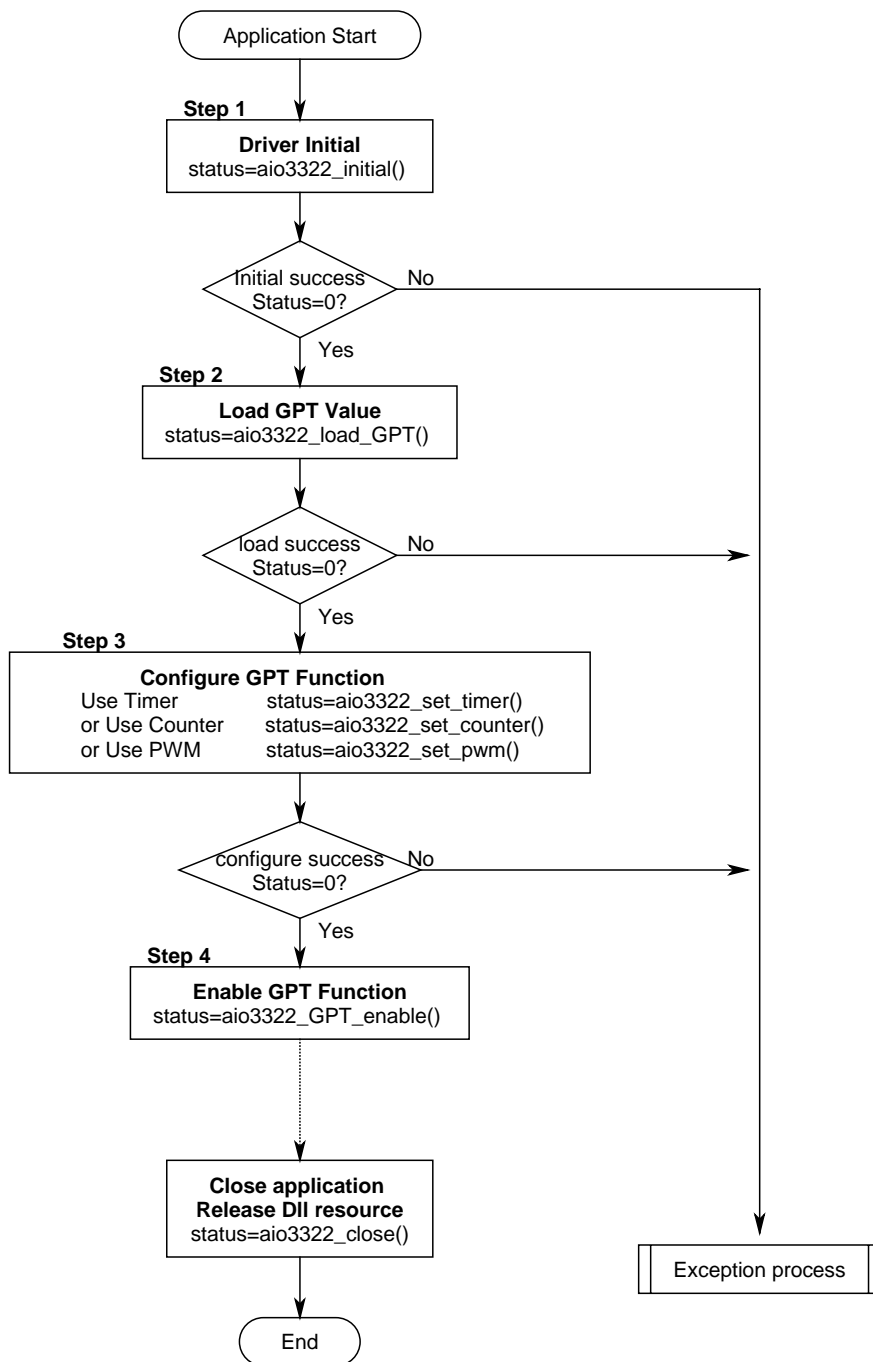
## 6.2 AIO3322/3323 Flow chart of auto data acquisition (Digital trigger)



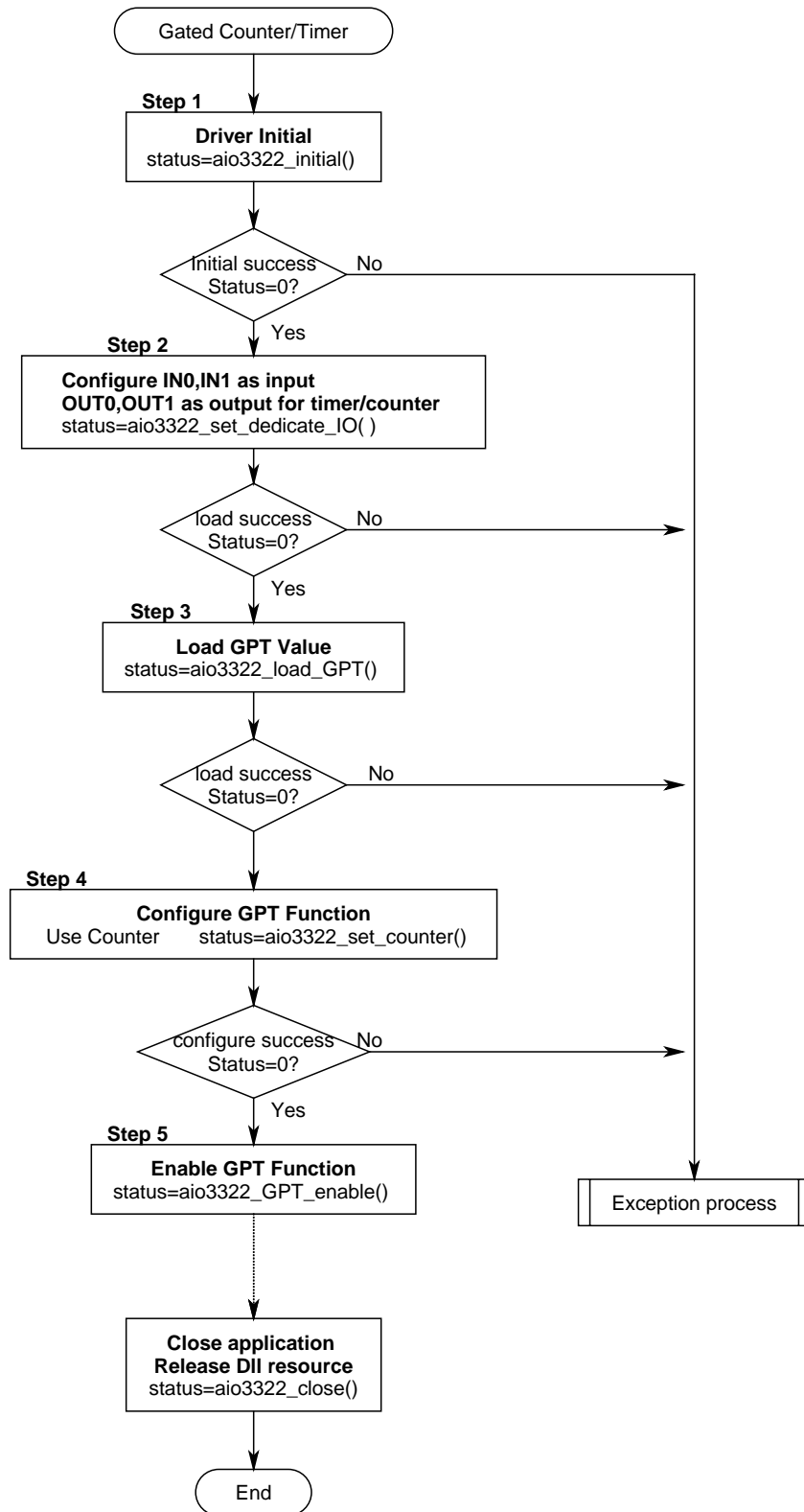
### 6.3 AIO3322/3323 Flow chart of auto data acquisition (Analog trigger)



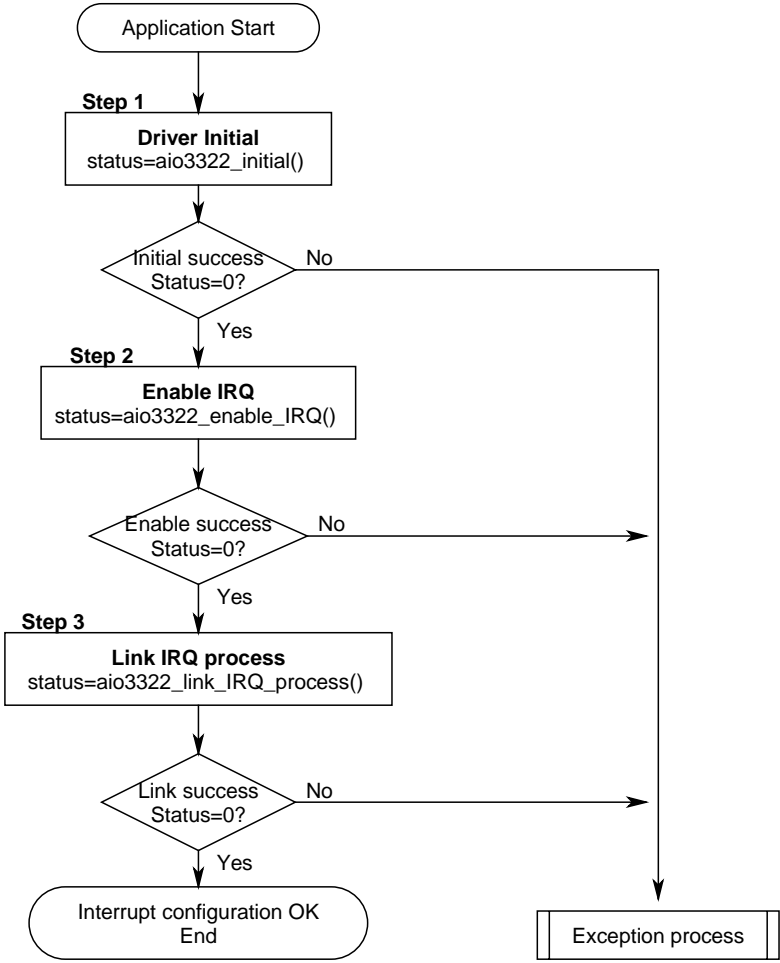
## 6.4 AIO3322/3323 Flow chart of Timer / Counter / PWM application



## 6.5 Gated counter application



6.6 AIO3322/3323 Flow chart of interrupt



## 7. **Function reference**

### 7.1 Error codes and CardID

Every AIO3322 function is consist of the following format:

**Status = function\_name (parameter 1, parameter 2, ... parameter n)**

Each function returns a value in the **Status** global variable that indicates the success or failure of the function. A returned **Status** equal to zero that indicates the function executed successfully. A non-zero status indicates failure that the function did not execute successfully because of an error, or executed with an error.

**Note** : **Status** is a 32-bit unsigned integer.

The first parameter to almost every AIO3322 function is the parameter **CardID** which is located the driver of AIO3322 board you want to use those given operation. The **CardID** is assigned by DIP/ROTARY switch. You can utilize multiple devices with different card CardID within one application; to do so, simply pass the appropriate **CardID** to each function.

**Note:** **CardID** is set by DIP/ROTARY switch (**0x0-0xF**)

## 7.2 Variable data types

Every function description has a parameter table that lists the data types for each parameter. The following sections describe the notation used in those parameter tables and throughout the manual for variable data types.

| Primary Type Names |  |   |                                     |   |   |
|--------------------|--|---|-------------------------------------|---|---|
| Name               | Description                                  | Range   | C/C++                               | Visual BASIC  | Pascal (Borland Delphi)   |
| <b>u8</b>          | 8-bit ASCII character                        | 0 to 255  | char                                | Not supported by BASIC. For functions that require character arrays, use string types instead.  | Byte  |
| <b>I16</b>         | 16-bit signed integer                        | -32,768 to 32,767                                 | short                               | Integer (for example: deviceNum%)   | SmallInt  |
| <b>U16</b>         | 16-bit unsigned integer                      | 0 to 65,535                                       | unsigned short for 32-bit compilers | Not supported by BASIC. For functions that require unsigned integers, use the signed integer type instead. See the i16 description.           | Word  |
| <b>I32</b>         | 32-bit signed integer                        | -2,147,483,648 to 2,147,483,647                   | long                                | Long (for example: count&)  | LongInt   |
| <b>U32</b>         | 32-bit unsigned integer                      | 0 to 4,294,967,295                                | unsigned long                       | Not supported by BASIC. For functions that require unsigned long integers, use the signed long integer type instead. See the i32 description. | Cardinal (in 32-bit operating systems). Refer to the i32 description. |
| <b>F32</b>         | 32-bit single-precision floating-point value | -3.402823E+38 to 3.402823E+38                     | float                               | Single (for example: num!)  | Single  |
| <b>F64</b>         | 64-bit double-precision floating-point value | -1.797683134862315E+308 to 1.797683134862315E+308 | double                              | Double (for example: voltage Number)  | Double  |

**Table 2**

### 7.3 Programming language considerations

Apart from the data type differences, there are a few language-dependent considerations you need to be aware of when you use the AIO3322 API. Read the following sections that apply to your programming language.

**Note :** Be sure to include the declaration functions of AIO3322 prototypes by including the appropriate AIO3322 header file in your source code. Refer to section 4.1 Building Applications with the AIO3322 Software Library for the header file appropriate to your compiler.

#### 7.3.1 C/C++

For C or C++ programmers, parameters listed as Input/Output parameters or Output parameters are pass-by-reference parameters, which means a pointer points to the destination variable should be passed into the function. For example, the Read Port function has the following format:

```
Status = aio3322_read_port(u8 CardID, u8* data);
```

where **CardID** is input parameters, and **data** is an output parameter. Consider the following example:

```
u8 CardID;
```

```
u8 data,
```

```
u32 Status;
```

```
Status = aio3322_read_port (CardID, &data);
```

#### 7.3.2 Visual basic

The file aio3322.bas contains definitions for constants required for obtaining AIO3322 card information and declared functions and variable as global variables. You should use these constants symbols in the aio3322.bas, do not use the numerical values.

In Visual Basic, you can add the entire aio3322.bas file into your project. Then you can use any of the constants defined in this file and call these constants in any module of your program. To add the aio3322.bas file for your project in Visual Basic 4.0, go to the **File** menu and select the **Add File... option**. Select aio3322.bas, which is browsed in the aio3322 \ api directory. Then, select **Open** to add the file to the project.

To add the aio3322.bas file to your project in Visual Basic , go to the **Project** menu and select **Add Module**. Click on the Existing tab page. **Select** aio3322.bas, which is in the aio3322 \ api directory. Then, select **Open** to add the file to the project.

### 7.3.3 Borland C++ builder

To use Borland C++ builder as development tool, you should generate a .lib file from the .dll file by implib.exe.

```
implib aio3322bc.lib aio3322.dll
```

Then add the **aio3322bc.lib** to your project and add

```
#include "aio3322.h" to main program.
```

Now you may use the dll functions in your program. For example, the Read Port function has the following format:

```
Status = aio3322_read_port(u8 CardID, u8* data);
```

where **CardID** and **port** are input parameters, and **data** is an output parameter. Consider the following example:

```
u8 CardID;
```

```
u8 data;
```

```
u32 Status;
```

```
Status = aio3322_read_port (CardID, &data);
```

## 7.4 AIO3322 Functions

These topics contain detailed descriptions of each AIO3322 function. The functions are arranged by dll function block.

### Initialization and close

#### ● **aio3322\_initial**

**Format :** u32 Status =aio3322\_initial (void)

**Purpose:** Initial the AIO3322 resource when start the Windows applications.

#### ● **aio3322\_close**

**Format :** u32 Status =aio3322\_close (void);

**Purpose:** Release the AIO3322 resource when close the Windows applications.

#### ● **aio3322\_info**

**Format :** u32 status =aio3322\_info(u8 CardID,u16 \*address)

**Purpose:** Read the physical I/O address assigned by O.S.

**Parameters:**

**Input:**

| Name   | Type | Description                   |
|--------|------|-------------------------------|
| CardID | u8   | assigned by DIP/ROTARY switch |

**Output:**

| Name    | Type | Description                         |
|---------|------|-------------------------------------|
| address | u16  | physical I/O address assigned by OS |

#### ● **aio3322\_get\_DeviceHandle**

**Format :** u32 status =aio3322\_get\_DeviceHandle(u8 CardID,HANDLE \*DeviceHandle)

**Purpose:** Read the device handle assigned by O.S..

**Parameters:**

**Input:**

| Name   | Type | Description                   |
|--------|------|-------------------------------|
| CardID | u8   | assigned by DIP/ROTARY switch |

**Output:**

| Name         | Type   | Description             |
|--------------|--------|-------------------------|
| DeviceHandle | HANDLE | Handle assigned by O.S. |

## Analog input

### ● aio3322 smart AtoD

**Format :** u32 status = aio3322\_smart\_AtoD(u8 CardID, u8 channel, u8 AD\_scale, f32 \*value)

**Purpose:** Read A/D value then convert to scale.

**Parameters:**

**Input:**

| Name     | Type | Description   |
|----------|------|---|
| CardID   | u8   | assigned by DIP/ROTARY switch   |
| channel  | u8   | A/D channel number (0~7)  |
| AD_scale | u8   | scale range:<br>0: 0V ~ 5V<br>1: -5V ~ +5V<br>2: 0V ~ 10V<br>3: -10V ~ +10V |

**Output:**

| Name  | Type | Description      |
|-------|------|------------------|
| value | f32  | data (converted) |

### ● aio3322 set AD command

**Format :** u32 status = aio3322\_set\_AD\_command(u8 CardID,u8 channel,u8 AD\_scale)

**Purpose:** Set channel selection and scale to A/D converter.

**Parameters:**

**Input:**

| Name     | Type | Description   |
|----------|------|---|
| CardID   | u8   | assigned by DIP/ROTARY switch   |
| channel  | u8   | select A/D channel number (0~7)   |
| AD_scale | u8   | scale range:<br>0: 0V ~ 5V<br>1: -5V ~ +5V<br>2: 0V ~ 10V<br>3: -10V ~ +10V |

- **aio3322 start AD conversion**

**Format :** u32 status = aio3322\_set\_AD\_conversion(u8 CardID )

**Purpose:** start A/D conversion on previously selected channel .

**Parameters:**

**Input:**

| Name   | Type | Description                   |
|--------|------|-------------------------------|
| CardID | u8   | assigned by DIP/ROTARY switch |

- **aio3322 read AD status**

**Format :** u32 status = aio3322\_read\_AD\_status( u8 CardID,u8 \*status)

**Purpose:** Read A/D status (on previous selected channel) for checking if it is ready

**Parameters:**

**Input:**

| Name   | Type | Description                   |
|--------|------|-------------------------------|
| CardID | u8   | assigned by DIP/ROTARY switch |

**Output:**

| Name   | Type | Description  |
|--------|------|--|
| status | u8   | 0: busy, not ready<br>1: ready (end of conversion) |

- **aio3322 read AD data**

**Format :** u32 status = aio3322\_read\_AD\_data(u8 CardID,u8 channel,i16 \*data )

**Purpose:** Read A/D conversion result (data).

**Parameters:**

**Input:**

| Name    | Type | Description                         |
|---------|------|-------------------------------------|
| CardID  | u8   | assigned by DIP/ROTARY switch       |
| channel | u8   | A/D channel number to be read (0~7) |

**Output:**

| Name | Type | Description   |
|------|------|---|
| data | i16  | For 0~5V or 0~10V mode<br>data will be in 0~0xFFF.<br>For -5V~+5V or -10V ~+10V mode<br>data will be in 2's complement<br>format. |

## Auto data acquisition

### ● **aio3322\_set\_DAQ\_sampling\_rate**

**Format :** u32 status = aio3322\_set\_DAQ\_sampling\_rate(u8 CardID,u8 speed\_mode)

**Purpose:** To setup the auto data acquisition sampling rate

**Parameters:**

**Input:**

| Name       | Type | Description  |
|------------|------|--|
| CardID     | u8   | assigned by DIP/ROTARY switch  |
| speed_mode | u8   | speed_mode:<br>0: 110kHz sampling rate<br>1: 10 kHz<br>2: 1 kHz<br>3: 100 Hz |

### ● **aio3322\_set\_Digital\_trigger\_DAQ**

**Format :** u32 status = aio3322\_set\_Digital\_trigger\_DAQ(u8 CardID,u8 channel,  
u8 AD\_scale)

**Purpose:** To setup trigger mode as digital trigger

**Parameters:**

**Input:**

| Name     | Type | Description   |
|----------|------|---|
| CardID   | u8   | assigned by DIP/ROTARY switch   |
| channel  | u8   | select A/D channel number (0~7)   |
| AD_scale | u8   | scale range:<br>0: 0V ~ 5V<br>1: -5V ~ +5V<br>2: 0V ~ 10V<br>3: -10V ~ +10V |

**Note:**

This function also assign digital trigger input to trigger I0, else it can be used as general purpose input.

This function must used first to setup DAQ mode, the procedure to setup digital trigger

1. *aio3322\_set\_Digital\_trigger\_DAQ*
2. *aio3322\_set\_Trigger\_IN\_polarity*
3. *aio3322\_config\_Trigger\_OUT\_mode* (if use trigger out)
4. *aio3322\_set\_Trigger\_OUT\_polarity* (if use trigger out)
5. *aio3322\_load\_Trigger\_OUT\_pulse\_duration* to setup trigger duration (if use trigger out)
6. *aio3322\_set\_Trigger\_DAQ\_retaining\_count* to setup buffer size after trigger
7. *aio3322\_enable\_Digital\_trigger\_DAQ* to start waiting DAQ trigger

● **aio3322\_set Analog trigger DAQ**

**Format :** u32 status = aio3322\_set\_Analog\_trigger\_DAQ(u8 CardID,u8 channel,  
u8 AD\_scale)

**Purpose:** To setup trigger mode as analog trigger

**Parameters:**

**Input:**

| Name     | Type | Description   |
|----------|------|---|
| CardID   | u8   | assigned by DIP/ROTARY switch   |
| channel  | u8   | select A/D channel number (0~7)   |
| AD_scale | u8   | scale range:<br>0: 0V ~ 5V<br>1: -5V ~ +5V<br>2: 0V ~ 10V<br>3: -10V ~ +10V |

**Note:**

This function must used first to setup DAQ mode, the procedure to setup digital trigger

1. *aio3322\_set\_Analog\_trigger\_DAQ*
2. *aio3322\_config\_Analog\_Trigger\_condition*
2. *aio3322\_config\_Trigger\_OUT\_mode* (if use trigger out)
3. *aio3322\_set\_Trigger\_OUT\_polarity* (if use trigger out)
4. *aio3322\_load\_Trigger\_OUT\_pulse\_duration* to setup trigger duration (if use trigger out)
5. *aio3322\_set\_Trigger\_DAQ\_retaining\_count* to setup buffer size after trigger
6. *aio3322\_enable\_Analog\_trigger\_DAQ* to start waiting DAQ trigger

● **aio3322 set Software trigger DAQ**

**Format :** u32 status = aio3322\_set\_Software\_trigger\_DAQ(u8 CardID,u8 channel,  
u8 AD\_scale)

**Purpose:** To setup trigger mode as software trigger

**Parameters:**

**Input:**

| Name     | Type | Description   |
|----------|------|---|
| CardID   | u8   | assigned by DIP/ROTARY switch   |
| channel  | u8   | select A/D channel number (0~7)   |
| AD_scale | u8   | scale range:<br>0: 0V ~ 5V<br>1: -5V ~ +5V<br>2: 0V ~ 10V<br>3: -10V ~ +10V |

**Note:**

This function must used first to setup DAQ mode, the procedure to setup digital trigger

1. *aio3322\_set\_Software\_trigger\_DAQ*
2. *aio3322\_config\_Trigger\_OUT\_mode* (if use trigger out)
3. *aio3322\_set\_Trigger\_OUT\_polarity* (if use trigger out)
4. *aio3322\_load\_Trigger\_OUT\_pulse\_duration* to setup trigger duration (if use trigger out)
5. *aio3322\_set\_Trigger\_DAQ\_retaining\_count* to setup buffer size after trigger
6. *aio3322\_start\_Software DAQ* to start DAQ.

● **aio3322 read DAQ mode**

**Format :** u32 status = aio3322\_read\_DAQ\_mode(u8 CardID,u8 \*mode)

**Purpose:** To setup the auto data acquisition sampling rate

**Parameters:**

**Input:**

| Name   | Type | Description                   |
|--------|------|-------------------------------|
| CardID | u8   | assigned by DIP/ROTARY switch |

**Output:**

| Name | Type | Description  |
|------|------|--|
| mode | u8   | DAQ_mode:<br>0: normal mode, not DAQ mode<br>1: Digital trigger DAQ mode<br>2: Software trigger DAQ mode<br>3: Analog trigger DAQ mode |

- **aio3322 config Analog Trigger condition**

**Format :** u32 status = aio3322\_config\_Analog\_Trigger\_condition(u8 CardID,u8 AD\_scale, u8 method, f32 compare\_voltage)

**Purpose:** To setup analog trigger condition

**Parameters:**

**Input:**

| Name            | Type | Description   |
|-----------------|------|---|
| CardID          | u8   | assigned by DIP/ROTARY switch   |
| AD_scale        | u8   | scale range:<br>0: 0V ~ 5V<br>1: -5V ~ +5V<br>2: 0V ~ 10V<br>3: -10V ~ +10V   |
| method          | u8   | trigger method:<br>0 : signed greater than compare_voltage<br>1 : signed less than compare_voltage<br>2 : unsigned greater than compare_voltage<br>3 : unsigned less than compare_voltage |
| compare_voltage | f32  | voltage value to be compared with<br>for example:<br>0~5V range,<br>compare_voltage should be in 0~5  |

- **aio3322 set Trigger IN polarity**

**Format :** u32 status = aio3322\_set\_Trigger\_IN\_polarity(u8 CardID,u8 point,u8 polarity)

**Purpose:** To setup digital trigger input polarity

**Parameters:**

**Input:**

| Name     | Type | Description  |
|----------|------|--|
| CardID   | u8   | assigned by DIP/ROTARY switch                            |
| point    | u8   | trigger input point:<br>0 : trigger I0<br>1 : trigger I1 |
| polarity | u8   | trigger active state:<br>0 : normal<br>1 : inverse       |

**Note:**

Only trigger I0 can be used as digital trigger input, trigger I1 should used as general purpose high speed input. **(Please notice that trigger I0, trigger I1 are not IN0, IN1)**

- **aio3322 read Trigger IN status**

**Format :** u32 status = aio3322\_read\_Trigger\_IN\_status(u8 CardID,u8 point,u8 \*state)

**Purpose:** To read digital trigger input status

**Parameters:**

**Input:**

| Name   | Type | Description  |
|--------|------|--|
| CardID | u8   | assigned by DIP/ROTARY switch                            |
| point  | u8   | trigger input point:<br>0 : trigger I0<br>1 : trigger I1 |

**Output:**

| Name  | Type | Description   |
|-------|------|---|
| state | u8   | trigger input status:<br>0 : inactive<br>1 : active |

**Note:**

Only trigger I0 can be used as digital trigger input, trigger I1 should used as general purpose high speed input. **(Please notice that trigger I0, trigger I1 are not IN0, IN1)**

- **aio3322 config Trigger OUT mode**

**Format :** u32 status = aio3322\_config\_Trigger\_OUT\_mode(u8 CardID,u8 point,u8 mode)

**Purpose:** To configure trigger output mode

**Parameters:**

**Input:**

| Name   | Type | Description   |
|--------|------|---|
| CardID | u8   | assigned by DIP/ROTARY switch   |
| point  | u8   | trigger output point:<br>0 : trigger out O0<br>1 : trigger out O1   |
| mode   | u8   | trigger output mode:<br>0 : use trigger out as general output<br>1 : use trigger out as dedicate pulse output<br>trigger out O0 will be trigger DAQ finish output,<br>trigger out O1 will be analog trigger DAQ event.<br>2 : use trigger out as dedicate toggle output |

- **aio3322 set Trigger OUT polarity**

**Format :** u32 status = aio3322\_set\_Trigger\_OUT\_polarity(u8 CardID,u8 point,u8 polarity)

**Purpose:** To configure trigger output polarity

**Parameters:**

**Input:**

| Name     | Type | Description   |
|----------|------|---|
| CardID   | u8   | assigned by DIP/ROTARY switch                                     |
| point    | u8   | trigger output point:<br>0 : trigger out O0<br>1 : trigger out O1 |
| polarity | u8   | trigger output polarity:<br>0 : normal<br>1 : inverse             |

- **aio3322 load Trigger OUT pulse duration**

**Format :** u32 status = aio3322\_load\_Trigger\_OUT\_pulse\_duration(u8 CardID,  
u32 duration\_time)

**Purpose:** To load the trigger out pulse duration constant

**Parameters:**

**Input:**

| Name          | Type | Description                            |
|---------------|------|--|
| CardID        | u8   | assigned by DIP/ROTARY switch          |
| duration_time | u32  | duration constant (0 ~ 16777215) in us |

- **aio3322 write Trigger OUT**

**Format :** u32 status = aio3322\_write\_Trigger\_OUT(u8 CardID,u8 point,u8 state)

**Purpose:** To set/reset trigger out O0 or O1 (while use as general purpose output)

**Parameters:**

**Input:**

| Name   | Type | Description   |
|--------|------|---|
| CardID | u8   | assigned by DIP/ROTARY switch                                     |
| point  | u8   | trigger output point:<br>0 : trigger out O0<br>1 : trigger out O1 |
| state  | u8   | 0: inactive<br>1: active  |

● **aio3322 readback Trigger OUT status**

**Format :** u32 status = aio3322\_readback\_Trigger\_OUT\_status(u8 CardID,u8 point,  
u8 \*state)

**Purpose:** To read back the status of trigger out O0 or O1

**Parameters:**

**Input:**

| Name   | Type | Description   |
|--------|------|---|
| CardID | u8   | assigned by DIP/ROTARY switch                                     |
| point  | u8   | trigger output point:<br>0 : trigger out O0<br>1 : trigger out O1 |

**Output:**

| Name  | Type | Description              |
|-------|------|--------------------------|
| state | u8   | 0: inactive<br>1: active |

● **aio3322 set Trigger DAQ retaining count**

**Format :** u32 status = aio3322\_set\_Trigger\_DAQ\_retaining\_count(u8 CardID,u16 number)

**Purpose:** To setup the data buffer length after trigger event

**Parameters:**

**Input:**

| Name   | Type | Description   |
|--------|------|---|
| CardID | u8   | assigned by DIP/ROTARY switch   |
| number | u16  | data buffer length for post-trigger<br>$0 \leq \text{number} \leq 1023$ |

**Note:**

1. maximum data buffer length for pre-trigger will be 1024 .
2. you may configure the trigger buffer at you will and get the conversion data before trigger event or after trigger event.

For example, if you set number=1023, then you will get the DAQ data stored in all data buffers after trigger event.

If you set number=0 then you will get the DAQ data stored in all data buffers before trigger event.

- **aio3322 enable Digital trigger DAQ**

**Format :** u32 status = aio3322\_enable\_Digital\_trigger\_DAQ(u8 CardID,u8 enable)

**Purpose:** To enable (start) and waiting the digital trigger IO input

**Parameters:**

**Input:**

| Name   | Type | Description                               |
|--------|------|---|
| CardID | u8   | assigned by DIP/ROTARY switch             |
| enable | u8   | 0: disable                      1: enable |

- **aio3322 enable Analog trigger DAQ**

**Format :** u32 status = aio3322\_enable\_Analog\_trigger\_DAQ(u8 CardID,u8 enable)

**Purpose:** To enable (start) and waiting the analog trigger condition occurs.

**Parameters:**

**Input:**

| Name   | Type | Description                               |
|--------|------|---|
| CardID | u8   | assigned by DIP/ROTARY switch             |
| enable | u8   | 0: disable                      1: enable |

- **aio3322 start Software trigger DAQ**

**Format :** u32 status = aio3322\_start\_Software\_trigger\_DAQ(u8 CardID,u8 enable)

**Purpose:** To start auto data acquisition immediately

**Parameters:**

**Input:**

| Name   | Type | Description                               |
|--------|------|---|
| CardID | u8   | assigned by DIP/ROTARY switch             |
| enable | u8   | 0: disable                      1: enable |

- **aio3322 release trigger DAQ**

**Format :** u32 status = aio3322\_release\_trigger\_DAQ(u8 CardID)

**Purpose:** To reset the DAQ mode to normal

**Parameters:**

**Input:**

| Name   | Type | Description                   |
|--------|------|-------------------------------|
| CardID | u8   | assigned by DIP/ROTARY switch |

**Note:**

Trigger mode can only release at any of the following conditions:

1. the current active DAQ mode is in waiting state
2. the current active DAQ mode is completed

The mode can not release until any of the above state reached.

- **aio3322 read DAQ finish flag**

**Format :** u32 status = aio3322\_read\_DAQ\_finish\_flag(u8 CardID,u8 \*flag)

**Purpose:** To read DAQ finish (complete) flag

**Parameters:**

**Input:**

| Name   | Type | Description                   |
|--------|------|-------------------------------|
| CardID | u8   | assigned by DIP/ROTARY switch |

**Output:**

| Name | Type | Description                       |
|------|------|-----------------------------------|
| flag | u8   | 0: not completed      1: finished |

- **aio3322 read DAQ value**

**Format :** u32 status = aio3322\_read\_DAQ\_value(u8 CardID,u16 pointer,f32 \*value)

**Purpose:** To read the data in the buffer

**Parameters:**

**Input:**

| Name    | Type | Description                   |
|---------|------|-------------------------------|
| CardID  | u8   | assigned by DIP/ROTARY switch |
| pointer | u16  | data buffer pointer (0~1023)  |

**Output:**

| Name  | Type | Description  |
|-------|------|--|
| value | f32  | converted value of data stored in designate buffer |

- **aio3322 read DAQ RAM data**

**Format :** u32 status = aio3322\_read\_DAQ\_RAM\_data(u8 CardID,u16 pointer,i16 \*data)

**Purpose:** To read the data in the buffer

**Parameters:**

**Input:**

| Name    | Type | Description                   |
|---------|------|-------------------------------|
| CardID  | u8   | assigned by DIP/ROTARY switch |
| pointer | u16  | data buffer pointer (0~1023)  |

**Output:**

| Name | Type | Description                     |
|------|------|---------------------------------|
| data | i16  | data stored in designate buffer |

- **aio3322 read DAQ cycling flag**

**Format :** u32 status = aio3322\_read\_DAQ\_cycling\_flag(u8 CardID,u8 \*flag)

**Purpose:** To read DAQ in action flag

**Parameters:**

**Input:**

| Name   | Type | Description                   |
|--------|------|-------------------------------|
| CardID | u8   | assigned by DIP/ROTARY switch |

**Output:**

| Name | Type | Description  |
|------|------|--|
| flag | u8   | 0: DAQ in wait or complete state<br>1: DAQ in data acquisition state |

## Analog output

- **aio3322\_out\_DA\_data**

**Format :** u32 status = aio3322\_out\_DA\_data(u8 CardID, u8 channel, i16 data)

**Purpose:** To output DA data

**Parameters:**

**Input:**

| Name    | Type | Description  |
|---------|------|--|
| CardID  | u8   | assigned by DIP/ROTARY switch  |
| channel | u8   | DA channel no.   |
| data    | i16  | data to be output (-32768 ~ 32768)<br>0 : 0V<br>32767(7FFFH): 10V<br>-32767(FFFFH): -10V |

## Digital I/O

### ● aio3322\_read\_port

**Format :** u32 status = aio3322\_read\_port(u8 CardID, u8 port, u8 \*data )

**Purpose:** To read data input or read back the output data

**Parameters:**

**Input:**

| Name   | Type | Description                     |
|--------|------|---------------------------------|
| CardID | u8   | assigned by DIP/ROTARY switch   |
| port   | u8   | 0: input port<br>1: output port |

**Output:**

| Name | Type | Description |
|------|------|-------------|
| data | u8   | port data   |

### ● aio3322\_read\_point

**Format :** u32 status = aio3322\_read\_point(u8 CardID, u8 port, u8 point, u8 \*state )

**Purpose:** To read data input or read back the output data of a specific point

**Parameters:**

**Input:**

| Name   | Type | Description  |
|--------|------|--|
| CardID | u8   | assigned by DIP/ROTARY switch  |
| port   | u8   | 0: input port<br>1: output port  |
| point  | u8   | bit designation<br>0: bit0                      1: bit 1<br>2: bit 2                      3: bit 3<br>4: bit 4                      5: bit 5<br>6: bit 6                      7: bit 7 |

**Output:**

| Name  | Type | Description |
|-------|------|-------------|
| state | u8   | bit data    |

- **aio3322\_set\_port**

**Format :** u32 status = aio3322\_set\_port(u8 CardID, u8 data )

**Purpose:** To output data

**Parameters:**

**Input:**

| Name   | Type | Description                   |
|--------|------|-------------------------------|
| CardID | u8   | assigned by DIP/ROTARY switch |
| data   | u8   | data to be output             |

- **aio3322\_set\_point**

**Format :** u32 status = aio3322\_set\_point(u8 CardID,u8 point,u8 state)

**Purpose:** To set/reset specific bit of output port

**Parameters:**

**Input:**

| Name   | Type | Description  |
|--------|------|--|
| CardID | u8   | assigned by DIP/ROTARY switch  |
| point  | u8   | bit designation<br>0: bit0                      1: bit 1<br>2: bit 2                      3: bit 3<br>4: bit 4                      5: bit 5<br>6: bit 6                      7: bit 7 |
| state  | u8   | bit data to be output  |

- **aio3322\_set\_dedicate\_IO**

**Format :** u32 status = aio3322\_set\_dedicate\_IO(u8 CardID,u8 enable)

**Purpose:** To set IN0, IN1, OUT0, OUT1 as dedicate I/O or general I/O

**Parameters:**

**Input:**

| Name   | Type | Description   |
|--------|------|---|
| CardID | u8   | assigned by DIP/ROTARY switch   |
| enable | u8   | 0: general I/O<br>1: dedicate I/O for timer/counter<br>IN0,OUT0 for timer/counter0<br>IN1,OUT1 for timer/counter1 |

- **aio3322\_readback\_dedicate\_IO\_status**

**Format :** u32 status = aio3322\_readback\_dedicate\_IO\_status(u8 CardID, u8 \*enable)

**Purpose:** To read back configuration of dedicate I/O

**Parameters:**

**Input:**

| Name   | Type | Description                   |
|--------|------|-------------------------------|
| CardID | u8   | assigned by DIP/ROTARY switch |

**Output:**

| Name   | Type | Description   |
|--------|------|---|
| enable | u8   | 0: general I/O<br>1: dedicate I/O for timer/counter<br>IN0,OUT0 for timer/counter0<br>IN1,OUT1 for timer/counter1 |

## Counter / Timer function

- **aio3322 set timer**

**Format :** u32 status = aio3322\_set\_timer(u8 CardID, u8 TimerID, u8 To\_mode )

**Purpose:** To set timer / counter at timer mode and configure its timer output function.

**Parameters:**

**Input:**

| Name    | Type | Description  |
|---------|------|--|
| CardID  | u8   | assigned by DIP/ROTARY switch  |
| TimerID | u8   | timer/counter designation<br>0: timer/counter0<br>1: timer/counter1  |
| To_mode | u8   | designation of timer output mode:<br>0: To will pulse low for one clock cycle when terminal count is reached<br>1: To will pulse low for 8 clock cycle when terminal count is reached<br>2: To will toggle whenever terminal count is reached<br>3: no output function |

● **aio3322\_set\_counter**

**Format :** u32 status = aio3322\_set\_counter(u8 CardID, u8 TimerID, u8 To\_mode, u8 Ti\_mode )

**Purpose:** To set timer / counter at counter mode and configure its counter input and counter output function.

**Parameters:**

**Input:**

| Name    | Type | Description  |
|---------|------|--|
| CardID  | u8   | assigned by DIP/ROTARY switch  |
| TimerID | u8   | timer/counter designation<br>0: timer/counter0<br>1: timer/counter1  |
| To_mode | u8   | counter output mode designation:<br>0: To will pulse low for one clock cycle when terminal count is reached<br>1: To will pulse low for 8 clock cycle when terminal count is reached<br>2: To will toggle whenever terminal count is reached<br>3: no output function                          |
| Ti_mode | u8   | counter input mode designation:<br>0: Counter decrease with system clock (4/33MHz) while Ti is low<br>1: Counter decrease with system clock (4/33MHz) while Ti is high<br>2: Counter decrease 1 while Ti is high to low transition<br>3: Counter decrease 1 while Ti is low to high transition |

**Notes:**

1. Counter/Timer0 input is at IN0, Counter/Timer1 input is at IN1.
2. Counter/Timer0 output is at OUT0, Counter/Timer1 output is at OUT1, and before using this function, port should be configured as dedicated port by aio3322\_set\_dedicate\_IO( ).

● **aio3322\_set\_pwm**

**Format :** u32 status = aio3322\_set\_pwm(u8 CardID,u8 TimerID )

**Purpose:** To set timer/counter at PWM mode and map the PWM output to port1 bit n for timer/counter n.

**Parameters:**

**Input:**

| Name    | Type | Description   |
|---------|------|---|
| CardID  | u8   | assigned by DIP/ROTARY switch                                       |
| TimerID | u8   | timer/counter designation<br>0: timer/counter0<br>1: timer/counter1 |

**Note:**

1. You must configure digital i/o port1 as output to get PWM out from OUT0(timer/counter0) ,OUT1(timer/counter1).
2. Each timer/counter has 32 bit register length, if you program as PWM mode, the register is divided as 2 16 bit width, the upper 16 bit work as the pulse high width and the lower 16 bit work as PWM frequency register.

For example: if you program timer/counter0 work at PWM mode and load its constant with 0x0005000A, you will get a PWM output from OUT0. Its working frequency is 400KHz (system frequency is 4MHz and divide by 0xA,ie. 10 system clock period) with 5 system clock period high (the high word is 0x5).

● **aio3322\_set\_clock\_frequency**

**Format :** u32 status = aio3322\_set\_clock\_frequency(u8 CardID, u8 freq\_mode )

**Purpose:** To set the time base frequency of timer/counter0 and timer/counter1.

**Parameters:**

**Input:**

| Name      | Type | Description  |
|-----------|------|--|
| CardID    | u8   | assigned by DIP/ROTARY switch                            |
| freq_mode | u8   | timer/counter time base frequency<br>0: 4MHz<br>1: 33MHz |

- **aio3322 load GPT**

**Format :** u32 status = aio3322\_load\_GPT (u8 CardID,u8 TimerID,u32 value,u8 preset)

**Purpose:** To load reload register or current register of timer/counter

**Parameters:**

**Input:**

| Name    | Type | Description  |
|---------|------|--|
| CardID  | u8   | assigned by DIP/ROTARY switch  |
| TimerID | u8   | timer/counter designation<br>0: timer/counter 0<br>1: timer/counter 1          |
| value   | u32  | data to be load  |
| preset  | u8   | 0: value is load to current register<br>1: value is preset to reload register, |

**Note:**

Current register is the register used as counting, the reload register is used as a preset data holding place, if the current register has decrease to zero, the new counter preset value will come from the reload register.

- **aio3322 GPT enable**

**Format :** u32 status = aio3322\_GPT\_enable(u8 CardID, u8 TimerID, u8 enable)

**Purpose:** To enable/disable timer/counter function that was set by aio3322\_set\_timer( ), aio3322\_set\_counter( ), aio3322\_set\_pwm( )

**Parameters:**

**Input:**

| Name    | Type | Description   |
|---------|------|---|
| CardID  | u8   | assigned by DIP/ROTARY switch   |
| TimerID | u8   | port designation<br>0: timer/counter0<br>1: timer/counter1                      |
| enable  | u8   | 0: disable timer/counter function<br>1: enable or resume timer/counter function |

- **aio3322 read GPT**

**Format :** status=aio3322\_read\_GPT (u8 CardID, u8 TimerID, u32 \*value, u8 preset)

**Purpose:** To read reload register or current register of timer/counter

**Parameters:**

**Input:**

| Name    | Type | Description  |
|---------|------|--|
| CardID  | u8   | assigned by DIP/ROTARY switch  |
| TimerID | u8   | timer/counter designation<br>0: timer/counter 0<br>1: timer/counter 1            |
| preset  | u8   | 0: value is read from current register<br>1: value is read from reload register, |

**Output:**

| Name  | Type | Description |
|-------|------|-------------|
| value | u32  | data load   |

- **aio3322 one shot command**

**Format :** status=aio3322\_one\_shot\_command (u8 CardID,u8 TimerID,u32 duration\_time)

**Purpose:** To generate a one shot output from timer0/1 at programmed duration

**Parameters:**

**Input:**

| Name          | Type | Description  |
|---------------|------|--|
| CardID        | u8   | assigned by DIP/ROTARY switch  |
| TimerID       | u8   | timer/counter designation<br>0: timer/counter 0 , output at DIO10<br>1: timer/counter 1 , output at DIO11                                  |
| duration_time | u32  | the duration time constant of one shot.<br>Duration time = time constant *T<br>for 4MHz frequency T=0.25us<br>for 33MHz frequency T=0.03us |

**Notes:**

Counter/Timer0 output is at OUT0, Counter/Timer1 output is at OUT1, and before using this function, port should be configured as dedicated port by aio3322\_set\_dedicate\_IO( ).

● **aio3322\_read\_parameter**

**Format :** status=aio3322\_read\_parameter(u8 CardID,u8 parameter,u32 \*data)

**Purpose:** To read parameter data from counter/timer

**Parameters:**

**Input:**

| Name      | Type | Description  |
|-----------|------|--|
| CardID    | u8   | assigned by DIP/ROTARY switch  |
| parameter | u8   | 0: read current working mode of timer/counter0<br>1: readback To_mode of Timer0<br>2: readback To_mode of Counter0<br>3: readback Ti_mode of Counter0<br>4: read current working mode of timer/counter1<br>5: readback To_mode of Timer1<br>6: readback To_mode of Counter1<br>7: readback Ti_mode of Counter1 |

**Output:**

| Name | Type | Description  |
|------|------|--|
| data | u32  | For reading current working mode of timer/counter<br>0: disable<br>1: Timer<br>2: Counter<br>3: PWM<br>For Ti_mode of counter, refer to <i>aio3322_set_counter</i><br>For To_mode of counter, refer to <i>aio3322_set_counter</i><br>For Ti_mode of timer, refer to <i>aio3322_set_timer</i> |

## Interrupt function

### ● aio3322 enable IRQ

**Format :** u32 status = aio3322\_enable\_IRQ(u8 CardID,u8 IRQ\_Source,  
HANDLE \*phEvent)

**Purpose:** Enable interrupt of timer/counter

**Parameters:**

**Input:**

| Name       | Type | Description                            |
|------------|------|--|
| CardID     | u8   | assigned by DIP/ROTARY switch          |
| IRQ_Source | u8   | 0: timer/counter0<br>1: timer/counter1 |

**Output:**

| Name    | Type   | Description                  |
|---------|--------|------------------------------|
| phEvent | HANDLE | The returned handle of event |

### ● aio3322 link IRQ process

**Format :** u32 status = aio3322\_link\_IRQ\_process(u8 CardID,u8 IRQ\_Source,void  
(\_\_stdcall \*callbackAddr(u8 CardID)))

**Purpose:** To link the interrupt source with the callback function.

**Parameters:**

**Input:**

| Name         | Type | Description                            |
|--------------|------|--|
| CardID       | u8   | assigned by DIP/ROTARY switch          |
| IRQ_Soure    | u8   | 0: timer/counter0<br>1: timer/counter1 |
| callbackAddr | void | the address of your callback function  |

### ● aio3322 disable IRQ

**Format :** u32 status = aio3322\_disable\_IRQ(u8 CardID,u8 IRQ\_Source)

**Purpose:** To disable interrupt from timer/counter

**Parameters:**

**Input:**

| Name       | Type | Description                            |
|------------|------|--|
| CardID     | u8   | assigned by DIP/ROTARY switch          |
| IRQ_Source | u8   | 0: timer/counter0<br>1: timer/counter1 |

**Note:**

in spite of interrupt is disabled the timer/counter is still working

● **aio3322 read IRQ status**

**Format :** u32 status = aio3322\_IRQ\_status(u8 CardID,u8 IRQ\_Source,u8 \* status)

**Purpose:** To read IRQ status

**Parameters:**

**Input:**

| Name       | Type | Description                            |
|------------|------|--|
| CardID     | u8   | assigned by DIP/ROTARY switch          |
| IRQ_Source | u8   | 0: timer/counter0<br>1: timer/counter1 |

**Output:**

| Name   | Type | Description             |
|--------|------|-------------------------|
| status | u8   | 0: disable<br>1: enable |

## Security function

### ● aio3322\_set\_password

**Format :** u32 status = aio3322\_set\_password(u8 CardID,u16 password[5])

**Purpose:** To set password and if the password is not all “0”, security function will be enabled.

**Note:** If the password is all “0”, the security function is disabled.

**Parameters:**

**Input:**

| Name        | Type | Description                   |
|-------------|------|-------------------------------|
| CardID      | u8   | assigned by DIP/ROTARY switch |
| password[5] | u16  | Password, 5 words             |

### ● aio3322\_change\_password

**Format :** u32 status = aio3322\_change\_password(u8 CardID,u16 Oldpassword[5],  
u16 password[5])

**Purpose:** To replace old password with new password.

**Parameters:**

**Input:**

| Name               | Type | Description                   |
|--------------------|------|-------------------------------|
| CardID             | u8   | assigned by DIP/ROTARY switch |
| Oldpassword<br>[5] | u16  | The previous password         |
| password[5]        | u16  | The new password to be set    |

### ● aio3322\_clear\_password

**Format :** u32 status = aio3322\_clear\_password(u8 CardID,u16 password[5])

**Purpose:** To clear password, to set password to all “0”, i.e. disable security function.

**Parameters:**

**Input:**

| Name        | Type | Description                   |
|-------------|------|-------------------------------|
| CardID      | u8   | assigned by DIP/ROTARY switch |
| password[5] | u16  | The password previous set     |

- **aio3322\_unlock\_security**

**Format :** u32 status = aio3322\_unlock\_security(u8 CardID,u16 password[5])

**Purpose:** To unlock security function and enable the further operation of this card

**Parameters:**

**Input:**

| Name        | Type | Description                   |
|-------------|------|-------------------------------|
| CardID      | u8   | assigned by DIP/ROTARY switch |
| password[5] | u16  | The password previous set     |

- **aio3322\_read\_security\_status**

**Format :** u32 status = aio3322\_read\_security\_status(u8 CardID,u8 \*lock\_status,  
u8 \*security\_enable )

**Purpose:** To read security status for checking if the card security function is unlocked.

**Parameters:**

**Input:**

| Name   | Type | Description                   |
|--------|------|-------------------------------|
| CardID | u8   | assigned by DIP/ROTARY switch |

**Output:**

| Name            | Type | Description   |
|-----------------|------|---|
| lock_status     | u8   | 0: security unlocked<br>1: locked<br>2: dead lock (must return to original maker to unlock) |
| security_enable | u8   | 0: security function disabled<br>1: security function enabled                               |

## 7.5 Dll list

|    | <b>Function Name</b>                            | <b>Descriptive Name</b>                           |
|----|---|---|
| 1  | aio3322_initial ( )                             | AIO3322 Initial                                   |
| 2  | aio3322_close ( )                               | AIO3322 Close                                     |
| 3  | aio3322_info ( )                                | Read the I/O address of specific card             |
| 4  | aio3322_get_DeviceHandle ( )                    | Read device handle                                |
| 5  | aio3322_smart_AtoD ( )                          | Read A/D in smart mode (convert to scale)         |
| 6  | aio3322_set_AD_command ( )                      | write command to AD chip                          |
| 7  | aio3322_start_AD_conversion ( )                 | start A/D conversion                              |
| 8  | aio3322_read_AD_status ( )                      | Read AD status                                    |
| 9  | aio3322_read_AD_data ( )                        | read AD conversion data                           |
| 10 | aio3322_set_DAQ_sampling_rate ( )               | setup auto data acquisition sampling rate         |
| 11 | aio3322_set_Digital_trigger_DAQ ( )             | setup digital trigger mode                        |
| 12 | aio3322_set_Analog_trigger_DAQ ( )              | setup analog trigger mode                         |
| 13 | aio3322_set_Software_trigger_DAQ ( )            | setup software trigger mode                       |
| 14 | aio3322_read_DAQ_mode ( )                       | Retrieve the set mode                             |
| 15 | aio3322_config_Analog_Trigger_conditi<br>on ( ) | setup analog trigger condition                    |
| 16 | aio3322_set_Trigger_IN_polarity ( )             | setup digital trigger input polarity              |
| 17 | aio3322_read_Trigger_IN_status ( )              | read digital trigger input status                 |
| 18 | aio3322_config_Trigger_OUT_mode ( )             | configure the trigger output mode                 |
| 19 | aio3322_set_Trigger_OUT_polarity ( )            | configure trigger output polarity                 |
| 20 | aio3322_load_Trigger_OUT_pulse_durati<br>on ( ) | configure the trigger output duration             |
| 21 | aio3322_write_Trigger_OUT ( )                   | set/reset trigger output                          |
| 22 | aio3322_readback_Trigger_OUT_status<br>( )      | read back trigger output status                   |
| 23 | aio3322_set_Trigger_DAQ_retaining_co<br>unt ( ) | configure the post-trigger buffer size            |
| 24 | aio3322_enable_Digital_trigger_DAQ ( )          | start waiting digital trigger                     |
| 25 | aio3322_enable_Analog_trigger_DAQ<br>( )        | start waiting for analog trigger condition occurs |
| 26 | aio3322_start_Software_trigger_DAQ ( )          | start DAQ immediately                             |
| 27 | aio3322_release_trigger_DAQ ( )                 | reset DAQ mode to normal mode                     |
| 28 | aio3322_read_DAQ_finish_flag ( )                | read the DAQ finished flag                        |
| 29 | aio3322_read_DAQ_value ( )                      | read the stored value from buffer                 |
| 30 | aio3322_read_DAQ_RAM_data ( )                   | read the stored raw data from buffer              |
| 31 | aio3322_read_DAQ_cycling_flag ( )               | read the DAQ in-action flag                       |

|    |   |   |
|----|---|---|
| 32 | aio3322_out_DA_data ( )                 | output D/A data   |
| 33 | aio3322_read_port ( )                   | Read I/O port data  |
| 34 | aio3322_read_point ( )                  | Read a specific point data of I/O port                    |
| 35 | aio3322_set_port ( )                    | Write data to output port                                 |
| 36 | aio3322_set_point ( )                   | Write bit of data to output port                          |
| 37 | aio3322_set_dedicate_IO ( )             | Set IN0, IN1, OIT0, OUT1 as dedicate I/O or general I/O   |
| 38 | aio3322_readback_dedicate_IO_status ( ) | Readback the mode previously set                          |
| 39 | aio3322_set_timer ( )                   | To set timer / counter at timer mode                      |
| 40 | aio3322_set_counter ( )                 | To set timer / counter at counter mode                    |
| 41 | aio3322_set_pwm ( )                     | Set timer/counter at PWM mode                             |
| 42 | aio3322_set_clock_frequency ( )         | Set the time base clock frequency                         |
| 43 | aio3322_load_GPT ( )                    | Write data to GPT(general purpose timer/counter register) |
| 44 | aio3322_GPT_enable ( )                  | Enable/disable timer/counter function                     |
| 45 | aio3322_read_GPT ( )                    | Read GPT(general purpose timer/counter register) data     |
| 46 | aio3322_one_shot_command ( )            | Generate an one shot                                      |
| 47 | aio3322_read_parameter ( )              | Read parameter  |
| 48 | aio3322_enable_IRQ ( )                  | Enable interrupt of timer/counter                         |
| 49 | aio3322_link_IRQ_process ( )            | Link IRQ process  |
| 50 | aio3322_disable_IRQ ( )                 | Disable interrupt of timer/counter                        |
| 51 | aio3322_read_IRQ_status ( )             | Read back IRQ status                                      |
| 52 | aio3322_set_password ( )                | Set password  |
| 53 | aio3322_change_password ( )             | Change password   |
| 54 | aio3322_clear_password ( )              | Clear password  |
| 55 | aio3322_unlock_security ( )             | Unlock security function                                  |
| 56 | aio3322_read_security_status ( )        | Read security status                                      |

## 8. AIO-3322/3323 Error codes summary

### 8.1 AIO3322/3323 Error codes table

| Error Code | Symbolic Name            | Description  |
|------------|--------------------------|--|
| 0          | JSDRV_NO_ERROR           | Success, No error.   |
| 1          | JSDRV_READ_DATA_ERROR    | Driver can not read correct data   |
| 2          | JSDRV_INIT_ERROR         | Driver initial error   |
| 3          | JSDRV_UNLOCK_ERROR       | Security unlock failure  |
| 4          | JSDRV_LOCK_COUNTER_ERROR | Dead lock, unlock failure more than 10 times   |
| 5          | JSDRV_SET_SECURITY_ERROR | Password overwrite error   |
| 100        | DEVICE_RW_ERROR          | Device Read/Write error  |
| 101        | JSDRV_NO_CARD            | No AIO3322/3323 card on the system.  |
| 102        | JSDRV_DUPLICATE_ID       | AIO3322/3323 CardID duplicate error.   |
| 104        | JSDRV_PAR_ERROR          | Bad parameter or illegal parameter   |
| 300        | JSDIO_ID_ERROR           | Function input parameter error. CardID setting error, CardID doesn't match the DIP/ROTARY switch setting |
| 301        | JSAIO_MODE_ERROR         | Mode parameter error.<br>Parameter out of range.   |
| 302        | JSAIO_CHANNEL_ERROR      | Channel parameter error.<br>Parameter out of range.  |
| 305        | JSAIO_CONVERSION_ERROR   | Conversion time over. Maybe no hardware or bad hardware.   |
| 306        | JSAIO_CONVERSION_BUSY    | A/D is busy in conversion  |
| 400        | JSAIO_PORT_ERROR         | Port parameter error.<br>Parameter out of range.   |
| 401        | JSAIO_STATE_ERROR        | State parameter error.<br>Parameter out of range.  |
| 402        | JSAIO_POINT_ERROR        | Point parameter error.<br>Parameter out of range.  |
| 403        | JSAIO_EEPROM_RW_ERROR    | Eeprom R/W error   |
| 405        | JSAIO_CALIBRATION_ERROR  | Calibration error.<br>Maybe out of range.  |
| 406        | JSAIO_TIMERID_ERROR      | TimerID parameter error.<br>Parameter out of range.  |
| 407        | JSAIO_TO_MODE_ERROR      | To_mode parameter error.<br>Parameter out of range.  |
| 408        | JSAIO_TI_MODE_ERROR      | Ti_mode parameter error.<br>Parameter out of range.  |
| 409        | JSAIO_PARAMETER_ERROR    | Parameter error.<br>Parameter out of range.  |
| 500        | JSAIO_EVENT_ERROR        | Event error.<br>Maybe no event created.  |