

AIO-338x

Analog I/O and Digital I/O Card

Software Manual (V1.1)

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

Correction record

Version	Record
1.0	for driver V1.0 up
1.1	fix time base 4/16M error

Contents

1.	How to install the software of AIO338x	4
1.1	Install the PCI driver.....	4
2.	Where to find the file you need.....	5
3.	About the AIO-338x software.....	6
3.1	What you need to get started.....	6
3.2	Software programming choices	6
4.	AIO338x Language support.....	7
4.1	Building applications with the AIO338x software library.....	7
4.2	AIO338x Windows libraries	7
5.	Software overview	8
5.1	Initialization and close	8
5.2	Analog input	8
5.3	Analog output	9
5.4	Digital I/O.....	9
5.5	Counter / Timer function	9
5.6	Interrupt function	10
5.7	Security function.....	10
5.8	Error conditions	10
6.	Flow chart of implement an application.....	11
6.1	AIO3382/3384 Flow chart of implementation.....	11
6.2	AIO3382/3384 Flow chart of Timer / Counter / PWM application.....	12
6.3	AIO3382/3384 Flow chart of interrupt.....	14
7.	Function reference.....	15
7.1	Error codes and CardID	15
7.2	Variable data types	16
7.3	Programming language considerations	17
7.4	AIO3380 Functions	19
	Initialization and close	19
	aio3380_initial	19
	aio3380_close	19
	aio3380_info	19
	aio3380_get_DeviceHandle.....	19
	aio3380_initial_calibration	20
	Analog Input	21
	aio3380_smart_AtoD.....	21
	aio3380_set_AD_command	21
	aio3380_start_AD_conversion	22
	aio3380_read_AD_status.....	22
	aio3380_read_AD_data	22
	aio3380_AD_calibration.....	23
	aio3380_smart_AtoD_no_calibr.....	23

Analog output	24
aio3380_smart_DtoA.....	24
aio3380_set_DA_data.....	24
aio3380_Readback_DA_data	25
aio3380_start_DA_conversion	25
Digital I/O.....	26
aio3380_set_port_dir	26
aio3380_read_port_dir.....	26
aio3380_read_port	27
aio3380_read_point	27
aio3380_set_port.....	27
aio3380_set_point.....	28
aio3380_set_dedicate_IO.....	28
aio3380_readback_dedicate_IO_status	29
Counter / Timer function	30
aio3380_set_timer.....	30
aio3380_set_counter	31
aio3380_set_pwm	32
aio3380_set_clock_frequency	32
aio3380_load_GPT	33
aio3380_read_GPT	33
aio3380_GPT_enable.....	34
aio3380_one_shot_command	34
aio3380_set_gate_CNTR.....	35
aio3380_read_parameter.....	36
Interrupt function	37
aio3380_enable_IRQ	37
aio3380_link_IRQ_process	37
aio3380_disable_IRQ	37
aio3380_read_IRQ_status.....	38
Security function.....	39
aio3380_set_password.....	39
aio3380_change_password	39
aio3380_clear_password.....	39
aio3380_unlock_security	40
aio3380_read_security_status.....	40
7.5 Dll list	41
8. AIO-3382/3384 Error codes summary.....	43
8.1 AIO3382/3384 Error codes table.....	43

1. **How to install the software of AIO338x**

1.1 Install the PCI driver

The PCI card is a plug and play card, once you add a new card on 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 (..\aio338x\software\win98_2k_xp\driver or if you download from website please execute the self-unzip file aio338x_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

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

\aio338x\software\win98_2k_xp\install\Setup338xVBdemo.exe and header files setup by

\aio338x\software\win98_2k_xp\install\Setup338xAPI.exe

The directory will be located at

../ JS Automation /AIO338x/Api (header files and VB,VC lib files)

../ JS Automation /AIO338x/Example (easy source code of initial user)

../ JS Automation /AIO338x/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

\aio338x\software\nt\install\Setup338xVBdemo.exe

3. About the AIO-338x software

AIO338x software includes a set of dynamic link library (DLL) and system driver that you can utilize to control the I/O card's ports and points separately.

Your AIO338x 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 AIO338x functions within Windows' operation system environment.

3.1 What you need to get started

To set up and use your AIO338x software, you need the following:

- AIO338x software
- AIO338x hardware
 - Main board
 - Wiring board (Option)

3.2 Software programming choices

You have several options to choose from when you are programming AIO338x 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 AIO338x software.

4. **AIO338x Language support**

The AIO338x 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 AIO338x software library

The AIO338x function reference topic contains general information about building AIO338x applications, describes the nature of the AIO338x files used in building AIO338x 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 AIO338x Windows libraries

The AIO338x for Windows function library is a DLL called **AIO3380.dll**. Since a DLL is used, AIO338x functions are not linked into the executable files of applications. Only the information about the AIO338x functions in the AIO338x 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 AIO338x functions in aio3380.dll.

Header Files and Import Libraries for Different Development Environments		
Development Environment	Header File	Import Library
Microsoft C/C++	aio3380.h	aio3380vc.lib
Borland C/C++	aio3380.h	aio3380bc.lib
Microsoft Visual Basic	aio3380.bas	

Table 1

5. Software overview

These topics describe the features and functionality of the AIO338x boards and briefly describes the AIO338x functions. Be sure to check the model number (in hardware manual) for functions.

5.1 Initialization and close

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

[*ai03380 initial\(\)*](#) will do.

Once you want to close your application, call

[*ai03380 close\(\)*](#) to release all the resource.

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

[*ai03380 info\(\)*](#) to get the address.

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

[*ai03380 get DeviceHandle\(\)*](#)

If you want to use the factory calibration data to calibrate the input data, load the calibration data by

[*ai03380 initial calibration\(\)*](#)

5.2 Analog input

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

[*ai03380 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

[*ai03380 set AD command\(\)*](#) to select channel, mode.

[*ai03380 start AD conversion\(\)*](#) to start A/D conversion.

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

[*ai03380 read AD status\(\)*](#) will do.

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

[*ai03380 read AD data\(\)*](#).

If you want to get a more accurate to calibrate the data, use

[*ai03380 AD calibration\(\)*](#) to get the calibrated conversion data from conversion data.

You can also use

[*ai03380 smart AtoD no calibr\(\)*](#) to read the conversion data (without calibration) to simplify the data acquisition process.

5.3 Analog output

The analog output is as simple as digital output

[*aio3380 smart DtoA\(\)*](#) to select channel and output data.

If you want to manually control the data output, first you must select the channel and setup the data

[*aio3380 set DA data\(\)*](#) will do.

Of course anytime you can read back the setup data by

[*aio3380 Readback DA data\(\)*](#)

After data setup, you can output all the DA channel by using

[*aio3380 start DA conversion\(\)*](#)

5.4 Digital I/O

There are 2 byte programmable TTL i/o on card. To use the i/o port, you must configure as input or output as your need. Use

[*aio3380 set port dir\(\)*](#) will do.

If you want to confirm your configuration use

[*aio3380 read port dir\(\)*](#).

To read a port data, use

[*aio3380 read port\(\)*](#) and to read bit data use [*aio3380 read point\(\)*](#).

To write a port data, use

[*aio3380 set port\(\)*](#) and write bit data [*aio3380 set point\(\)*](#) will do.

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

[*aio3380 set dedicate IO\(\)*](#) can be used to change the I/O bit0,bit1 (of port0 and port1) as general or timer/counter dedicated I/O.

To read back status of dedicated I/O use:

[*aio3380 readback dedicate IO status\(\)*](#)

5.5 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

[*aio3380 set timer\(\)*](#) to configure as timer and its output mode

[*aio3380 set counter\(\)*](#) to configure as counter and its input and output mode

[*aio3380 set pwm\(\)*](#) to configure as PWM generator.

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

[*aio3380 set clock frequency\(\)*](#) can be used to choose the 4MHz or 16MHz clock source.

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

[*aio3380 load GPT\(\)*](#) will do.

To check the current counter/timer value

[*aio3380 read GPT\(\)*](#) will do.

Once you have setup the data

[*ai03380 GPT enable\(\)*](#) will enable or disable the counter/timer function.

The special function of one shot mode can be applied by

[*ai03380 one shot command\(\)*](#).

If you want to count the time elapsed during two triggers,

[*ai03380 set gate CNTR\(\)*](#) is the right selection.

To readback the parameters you set

[*ai03380 read parameter\(\)*](#) will response the current working mode and the parameters you set for individual timer, counter.

5.6 Interrupt function

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

AIO3380 provide timer/counter interrupts, to use interrupt function use

[*ai03380 enable IRO\(\)*](#) to enable it and

[*ai03380 link IRO process\(\)*](#) to link your irq service routine,

[*ai03380 disable IRO\(\)*](#) 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

[*ai03380 read IRO status\(\)*](#) will do.

5.7 Security function

Since AIO3380 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

[*ai03380 set password\(\)*](#) to set password and start the security function. Use

[*ai03380 change password\(\)*](#) to change it.

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

[*ai03380 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

[*ai03380 unlock security\(\)*](#) unlock the security.

You can also use

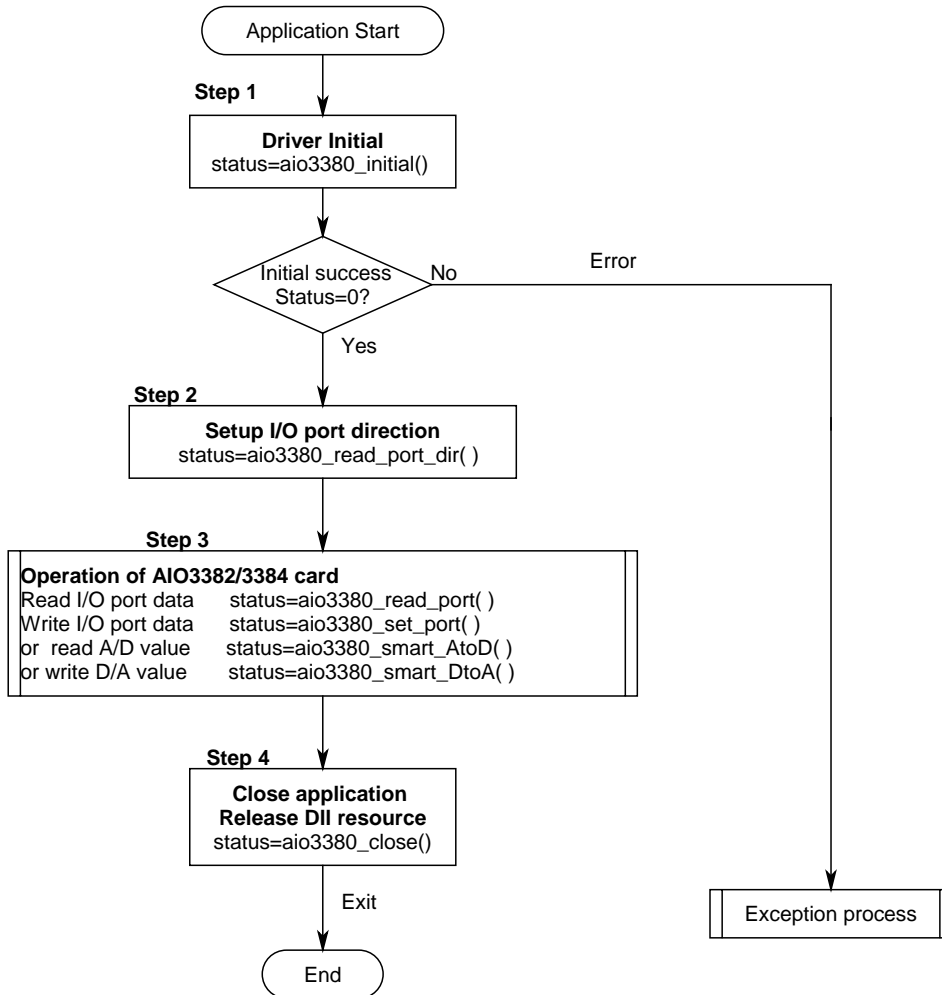
[*ai03380 read security status\(\)*](#) to check the current status of security.

5.8 Error conditions

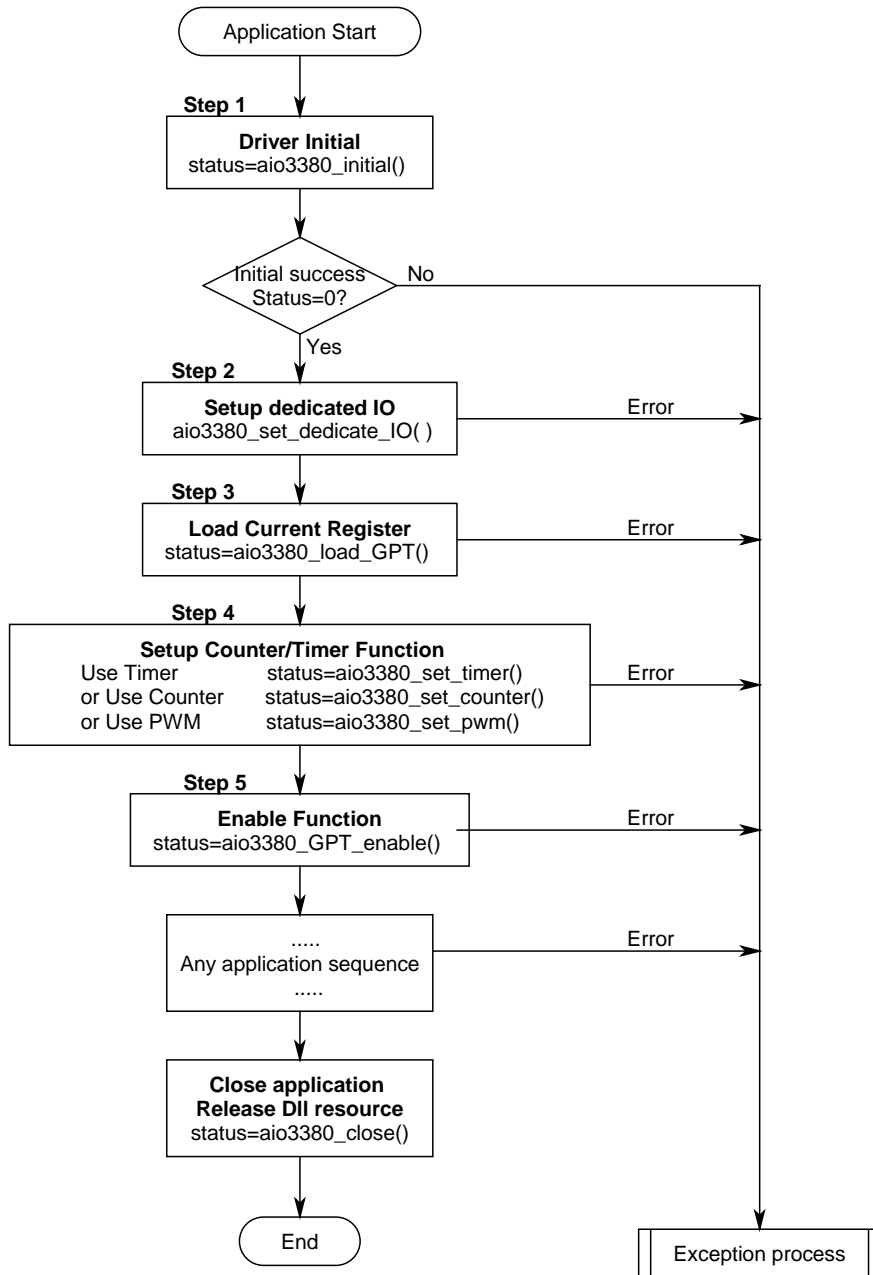
These error types may indicate an internal hardware problem on the board. Error Codes summary contains a detailed listing of the error status returned by AIO3380 functions.

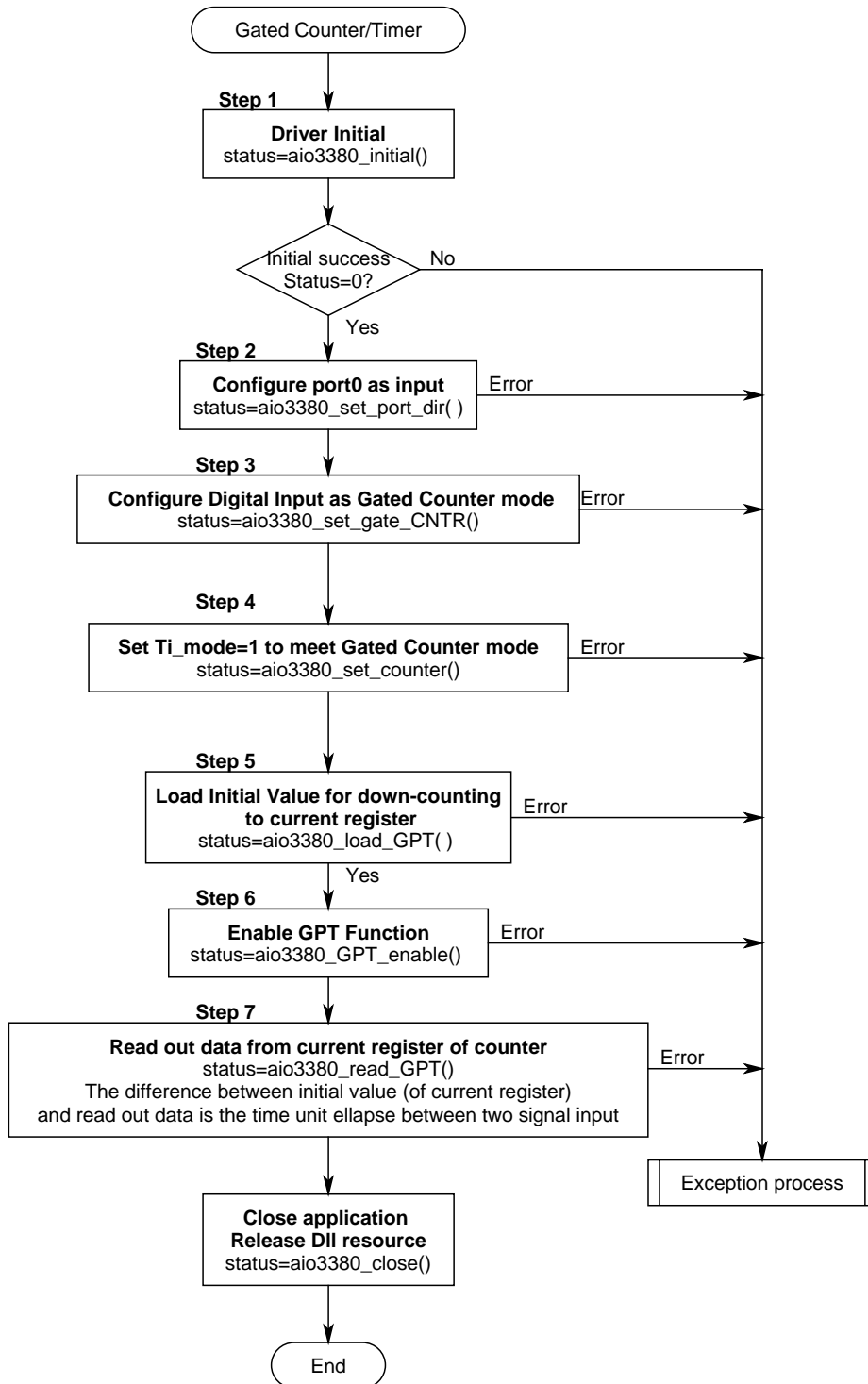
6. Flow chart of implement an application

6.1 AIO3382/3384 Flow chart of implementation

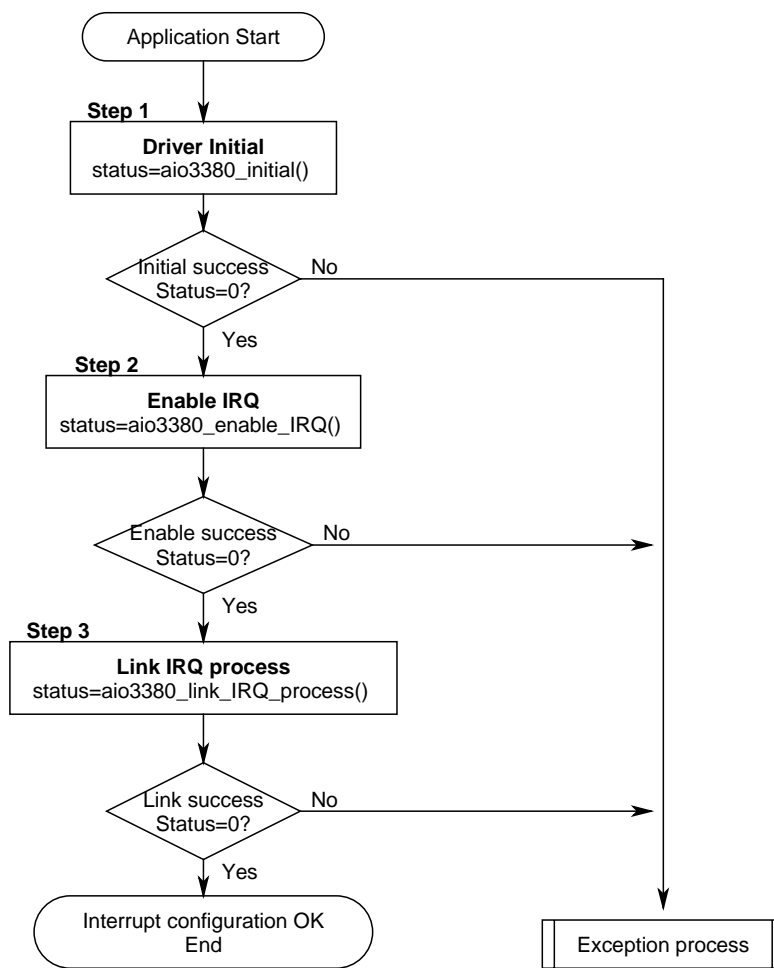


6.2 AIO3382/3384 Flow chart of Timer / Counter / PWM application





6.3 AIO3382/3384 Flow chart of interrupt



7. **Function reference**

7.1 Error codes and CardID

Every AIO3380 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 AIO3380 function is the parameter **CardID** which is located the driver of AIO3380 board you want to use those given operation. The **CardID** is assigned by DIP/ROTARY SW. 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 SW (**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)
u8	8-bit ASCII character	0 to 255	char	Not supported by BASIC. For functions that require character arrays, use string types instead.	Byte
I16	16-bit signed integer	-32,768 to 32,767	short	Integer (for example: deviceNum%)	SmallInt
U16	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
I32	32-bit signed integer	-2,147,483,648 to 2,147,483,647	long	Long (for example: count&)	LongInt
U32	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.
F32	32-bit single-precision floating-point value	-3.402823E+38 to 3.402823E+38	float	Single (for example: num!)	Single
F64	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 AIO3380 API. Read the following sections that apply to your programming language.

Note : Be sure to include the declaration functions of AIO3380 prototypes by including the appropriate AIO3380 header file in your source code. Refer to section 4.1 Building Applications with the AIO3380 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 = aio3380_read_port(CardID, *data);
```

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

```
u8 CardID;
```

```
u8 data;
```

```
u32 Status;
```

```
Status = aio3380_set_port (CardID, data);
```

7.3.2 Visual basic

The file aio3380.bas contains definitions for constants required for obtaining AIO3380 card information and declared functions and variable as global variables. You should use these constants symbols in the aio3380.bas, do not use the numerical values.

In Visual Basic, you can add the entire aio3380.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 aio3380.bas file for your project in Visual Basic 4.0, go to the **File** menu and select the **Add File... option**. Select aio3380.bas, which is browsed in the aio3380 \ api directory. Then, select **Open** to add the file to the project.

To add the aio3380.bas file to your project in Visual Basic 5.0 and 6.0, go to the **Project** menu and select **Add Module**. Click on the Existing tab page. **Select** aio3380.bas, which is in the aio3380 \ 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 aio3380bc.lib aio3380.dll
```

Then add the **aio3380bc.lib** to your project and add

```
#include "aio3380.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 = aio3380_read_port(CardID, *data);
```

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

```
u16 CardID;
```

```
u8 data;
```

```
u32 Status;
```

```
Status = aio3380_set_port (CardID, data);
```

7.4 AIO3380 Functions

These topics contain detailed descriptions of each AIO3380 function. The functions are arranged by dll function block.

Initialization and close

- **aio3380_initial**

Format : u32 Status =aio3380_initial (void)

Purpose: Initial the AIO3380 resource when start the Windows applications.

- **aio3380_close**

Format : u32 Status =aio3380_close (void);

Purpose: Release the AIO3380 resource when close the Windows applications.

- **aio3380_info**

Format : u32 status =aio3380_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 SW

Output:

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

- **aio3380_get_DeviceHandle**

Format : u32 status =aio3380_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 SW

Output:

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

- **aio3380 initial calibration**

Format : u32 Status =aio3380_initial_calibration(u8 CardID)

Purpose: Read the factory calibrated data for the future calibration.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW

Analog Input

● aio3380 smart AtoD

Format : u32 status = aio3380_smart_AtoD(u8 CardID,u8 channel,u8 mode,f32 *value)

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
channel	u8	A/D channel number 0~7
mode	u8	mode / scale range: 0: 0V ~ 5V 1: -5V ~ +5V 2: 0V ~ 10V 3: -10V ~ +10V 4: 0~20ma 5: 4~20ma

Output:

Name	Type	Description
value	f32	data (converted)

● aio3380 set AD command

Format : u32 status = aio3380_set_AD_command(u8 CardID,u8 channel,u8 mode)

Purpose: Set channel selection and mode to A/D .

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
channel	u8	select A/D channel number (0~7)
mode	u8	scale range: 0: 0V ~ 5V 1: -5V ~ +5V 2: 0V ~ 10V 3: -10V ~ +10V 4: 0~20ma 5: 4~20ma

- **aio3380 start AD conversion**

Format : u32 status = aio3380_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 SW

- **aio3380 read AD status**

Format : u32 status = aio3380_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 SW

Output:

Name	Type	Description
status	u8	0: busy, not ready 1: ready (end of conversion)

- **aio3380 read AD data**

Format : u32 status = aio3380_read_AD_data(u8 CardID,i16 *data)

Purpose: Read (on previous selected channel) A/D conversion result (data).

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW

Output:

Name	Type	Description
data	i16	0~4095 : for 0~5V, 0~10V mode for 0~20ma, 4~20ma mode -2048~2047: for -5V~5V, -10V ~10V mode

- **aio3380 AD calibration**

Format : u32 status = aio3380_AD_calibration(u8 CardID,u8 channel ,i16 data,
f32 *value)

Purpose: To calibrate the conversion data with factory pre-calibrated data.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
channel	u8	A/D channel number data read from
data	i16	data to be calibrated

Output:

Name	Type	Description
value	f32	calibrated data

- **aio3380 smart AtoD no calibr**

Format : u32 status = aio3380_smart_AtoD_no_calibr(u8 CardID,u8 channel,u8 mode,
f32 *value)

Purpose: Read A/D value and convert to scale.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
channel	u8	A/D channel number
mode	u8	scale range: 0: 0V ~ 5V 1: -5V ~ +5V 2: 0V ~ 10V 3: -10V ~ +10V

Output:

Name	Type	Description
value	f32	data (converted but not calibrated)

Analog output

● **aio3380 smart DtoA**

Format : u32 status = aio3380_smart_DtoA(u8 CardID,u8 channel,i16 data)

Purpose: Output DA data

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
channel	u8	DA channel no. (0~1 for AIO3382) (0~3 for AIO3384)
data	i16	data to be output -10V ~+10V : -32768 ~32767 0~20ma: 0~32767 4~20ma: 0~32767

● **aio3380 set DA data**

Format : u32 status = aio3380_set_DA_data(u8 CardID,u8 channel,i16 data)

Purpose: To setup DA data

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
channel	u8	DA channel no. (0~1 for AIO3382) (0~3 for AIO3384)
data	i16	data to be output -10V ~+10V : -32768~32767 0~20ma: 0~32767 4~20ma: 0~32767

- **aio3380 Readback DA data**

Format : u32 status = aio3380_Readback_DA_data(u8 CardID,u8 channel,i16 *data)

Purpose: To readback setup DA data

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
channel	u8	DA channel no. (0~1 for AIO3382) (0~3 for AIO3384)
data	i16	data to be output -10V ~+10V : -32768 ~32767 0~20ma: 0~32767 4~20ma: 0~32767

- **aio3380 start DA conversion**

Format : u32 status = aio3380_start_DA_conversion(u8 CardID)

Purpose: To output setup DA data

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW

Digital I/O

● aio3380 set port dir

Format : u32 status = aio3380_set_port_dir(u8 CardID,u8 port,u8 dir)

Purpose: To set the TTL port0, port1 as input or output.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
port	u8	port designation 0: port0 1: port1
dir	u8	direction 0: input 1: output

● aio3380 read port dir

Format : u32 status = aio3380_read_port_dir(u8 CardID,u8 port,u8 *dir)

Purpose: To readback the TTL port0, port1 direction setting.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
port	u8	port designation 0: port0 1: port1

Output:

Name	Type	Description
dir	u8	direction 0: input 1: output

- **aio3380_read_port**

Format : u32 status = aio3380_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 SW
port	u8	0: port 0 1: port 1

Output:

Name	Type	Description
data	u8	port data

- **aio3380_read_point**

Format : u32 status = aio3380_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 SW
port	u8	0: port 0 1: port 1
point	u8	bit designation 0: bit0 1: bit 1 2: bit 2 3: bit 3 4: bit 4 5: bit 5 6: bit 6 7: bit 7

Output:

Name	Type	Description
state	u8	bit data

- **aio3380_set_port**

Format : u32 status = aio3380_set_port(u8 CardID,u8 port,u8 data)

Purpose: To output data

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
port	u8	0: port 0 1: port 1
data	u8	data to be output

- **aio3380 set point**

Format : u32 status = aio3380_set_point(u8 CardID,u8 port,u8 point,u8 state)

Purpose: To set/reset specific bit output to port

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
port	u8	0: port 0 1: port 1
point	u8	bit designation 0: bit0 1: bit 1 2: bit 2 3: bit 3 4: bit 4 5: bit 5 6: bit 6 7: bit 7
state	u8	bit data to be output

- **aio3380 set dedicate IO**

Format : u32 status = aio3380_set_dedicate_IO(u8 CardID,u8 enable)

Purpose: To set IO00, IO01, IO10, IO11 as dedicate I/O or general I/O

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
enable	u8	0: general I/O 1: dedicate I/O for timer/counter for timer/counter0: IO00(In),IO10(Out) for timer/counter1: IO01(In),IO11(Out)

- **aio3380 readback dedicate IO status**

Format : u32 status = aio3380_readback_dedicate_IO_status(u8 CardID,u8 *enable)

Purpose: To readback IO00, IO01, IO10, IO11 as dedicate I/O or general I/O

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW

Output:

Name	Type	Description
enable	u8	0: general I/O 1: dedicate I/O for timer/counter for timer/counter0: IO00(In),IO10(Out) for timer/counter1: IO01(In),IO11(Out)

Counter / Timer function

- **aio3380 set timer**

Format : u32 status = aio3380_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 SW
TimerID	u8	timer/counter designation 0: timer/counter0 1: timer/counter1
To_mode	u8	designation of timer output mode: 0: To will pulse low for one clock cycle when terminal count is reached 1: To will pulse low for 8 clock cycle when terminal count is reached 2: To will toggle whenever terminal count is reached 3: no output function

● **aio3380 set counter**

Format : u32 status = aio3380_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 SW
TimerID	u8	timer/counter designation 0: timer/counter0 1: timer/counter1
To_mode	u8	counter output mode designation: 0: To will pulse low for one clock cycle when terminal count is reached 1: To will pulse low for 8 clock cycle when terminal count is reached 2: To will toggle whenever terminal count is reached 3: no output function
Ti_mode	u8	counter input mode designation: 0: Counter decrease with system clock (4/16MHz) while Ti is low 1: Counter decrease with system clock (4/16MHz) while Ti is high 2: Counter decrease 1 while Ti is high to low transition 3: Counter decrease 1 while Ti is low to high transition

Notes:

1. Counter/Timer0 input is at IO00, Counter/Timer1 input is at IO01.
2. Counter/Timer0 output is at IO10, Counter/Timer1 output is at IO11, and before using this function, port should be configured as dedicated port by aio3380_set_dedicate_IO().

● **aio3380 set_pwm**

Format : u32 status = aio3380_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 SW
TimerID	u8	timer/counter designation 0: timer/counter0 1: timer/counter1

Note:

1. You must configure digital i/o port1 as output to get PWM out from IO10(timer/counter0) ,IO11(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 IO10. 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).

● **aio3380 set_clock_frequency**

Format : u32 status = aio3380_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 SW
freq_mode	u8	timer/counter time base frequency 0: 4MHz 1: 16MHz

● **aio3380 load GPT**

Format : u32 status = aio3380_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 SW
TimerID	u8	timer/counter designation 0: timer/counter 0 1: timer/counter 1
value	u32	data to be load
preset	u8	0: value is load to current register 1: value is preset to reload register,

Note:

1. Current register is the working register of counting, load the current register will stop the on-going action of timer/counter.
2. Reload register is used at timer/counter re-loading (automatically), it can be used on the fly to change time constant or counter constant.

● **aio3380 read GPT**

Format : status=aio3380_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 SW
TimerID	u8	timer/counter designation 0: timer/counter 0 1: timer/counter 1
preset	u8	0: value is read from current register 1: value is read from reload register,

Output:

Name	Type	Description
value	u32	data load

- **aio3380 GPT enable**

Format : u32 status = aio3380_GPT_enable(u8 CardID,u8 TimerID,u8 enable)

Purpose: To enable/disable timer/counter function that was set by aio3380_set_timer(),
aio3380_set_counter(), aio3380_set_pwm()

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
TimerID	u8	port designation 0: timer/counter0 1: timer/counter1
enable	u8	0: disable timer/counter function 1: enable or resume timer/counter function

- **aio3380 one shot command**

Format : status=aio3380_one_shot_command (u8 CardID,u8 TimerID,u32 duration_time)

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

Parameters:

Input:

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

Notes:

Counter/Timer0 output is at IO10, Counter/Timer1 output is at IO11, and before using this function, port should be configured as dedicated port by aio3380_set_dedicate_IO().

● **aio3380 set gate CNTR**

Format : u32 status = aio3380_set_gate_CNTR(u8 CardID,u8 TimerID,u8 enable)

Purpose: To enable/disable gated timer/counter function and the gated input for timer0/timer1.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
TimerID	u8	port designation 0: timer/counter0 1: timer/counter1
enable	u8	0: disable gated timer/counter function 1: enable or resume timer/counter function for Counter/Timer0, the gate input : set: IO00 reset: IO02 for Counter/Timer1, the gate input : set: IO01 reset: IO03

Note: The gate input will set/reset the Counter/Timer on rising edge.

- **aio3380_read_parameter**

Format : status=aio3380_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 SW
parameter	u8	0: read current working mode of timer/counter0 1: readback To_mode of Timer0 2: readback To_mode of Counter0 3: readback Ti_mode of Counter0 4: read current working mode of timer/counter1 5: readback To_mode of Timer1 6: readback To_mode of Counter1 7: readback Ti_mode of Counter1

Output:

Name	Type	Description
data	u32	For reading current working mode of timer/counter 0: disable 1: Timer 2: Counter 3: PWM For Ti_mode of counter, refer to <i>aio3380_set_counter</i> For To_mode of counter, refer to <i>aio3380_set_counter</i> For Ti_mode of timer, refer to <i>aio3380_set_timer</i>

Interrupt function

● aio3380 enable IRQ

Format : u32 status = aio3380_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 SW
IRQ_Source	u8	0: timer/counter0 1: timer/counter1

Output:

Name	Type	Description
phEvent	HANDLE	The returned handle of event

● aio3380 link IRQ process

Format : u32 status = aio3380_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 SW
IRQ_Soure	u8	0: timer/counter0 1: timer/counter1
callbackAddr	void	the address of your callback function

● aio3380 disable IRQ

Format : u32 status = aio3380_disable_IRQ(u8 CardID,u8 IRQ_Source)

Purpose: To disable interrupt from timer/counter but please note that “timer/counter is still working”

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
IRQ_Source	u8	0: timer/counter0 1: timer/counter1

● **aio3380 read IRQ status**

Format : u32 status = aio3380_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 SW
IRQ_Source	u8	0: timer/counter0 1: timer/counter1

Output:

Name	Type	Description
status	u8	0: disable 1: enable

Security function

● aio3380 set password

Format : u32 status = aio3380_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 SW
password[5]	u16	Password, 5 words

● aio3380 change password

Format : u32 status = aio3380_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 SW
Oldpassword [5]	u16	The previous password
password[5]	u16	The new password to be set

● aio3380 clear password

Format : u32 status = aio3380_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 SW
password[5]	u16	The password previous set

- **aio3380 unlock security**

Format : u32 status = aio3380_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 SW
password[5]	u16	The password previous set

- **aio3380 read security status**

Format : u32 status = aio3380_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 SW

Output:

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

7.5 Dll list

	Function Name	Descriptive Name
1	aio3380_initial()	AIO3380 Initial
2	aio3380_close()	AIO3380 Close
3	aio3380_info()	Read the I/O address of specific card
4	aio3380_get_DeviceHandle()	Read device handle
5	aio3380_initial_calibration()	Read the factory calibrated data for the future calibration
6	aio3380_smart_AtoD()	Read A/D in smart mode (convert to scale)
7	aio3380_set_AD_command()	write command to AD chip
8	aio3380_start_AD_conversion()	start A/D conversion
9	aio3380_read_AD_status()	Read AD status
10	aio3380_read_AD_data()	read AD conversion data
11	aio3380_AD_calibration()	To calibrate the conversion data with factory pre-calibrated data
12	aio3380_smart_AtoD_no_calibr()	Read A/D value and convert to scale.
13	aio3380_smart_DtoA()	output D/A data
14	aio3380_set_DA_data()	setup DA data
15	aio3380_Readback_DA_data()	readback setup DA data
16	aio3380_start_DA_conversion()	output setup data
17	aio3380_set_port_dir()	setup port direction
18	aio3380_read_port_dir()	readback port direction setting
19	aio3380_read_port()	Read I/O port data
20	aio3380_read_point()	Read a specific point data of I/O port
21	aio3380_set_port()	Write data to output port
22	aio3380_set_point()	Write bit data to output port
23	aio3380_set_dedicate_IO()	Set IO00, IO01, IO10, IO11 as dedicate I/O or general I/O
24	aio3380_readback_dedicate_IO_status()	Readback the mode previously set for dedicated IO
25	aio3380_set_timer()	To set timer / counter at timer mode
26	aio3380_set_counter()	To set timer / counter at counter mode
27	aio3380_set_pwm()	Set timer/counter at PWM mode
28	aio3380_set_clock_frequency()	Set the time base clock frequency
29	aio3380_load_GPT()	Write data to GPT(general purpose timer/counter register)
30	aio3380_read_GPT()	Read GPT(general purpose timer/counter register) data
31	aio3380_GPT_enable()	Enable/disable timer/counter function
32	aio3380_one_shot_command()	Generate an one shot

33	aio3380_set_gate_CNTR()	Set gated counter function
34	aio3380_read_parameter()	Read parameter from timer/counter
35	aio3380_enable_IRQ()	Enable interrupt of timer/counter
36	aio3380_link_IRQ_process()	Link IRQ process
37	aio3380_disable_IRQ()	Disable interrupt of timer/counter
38	aio3380_read_IRQ_status()	Readback IRQ status
39	aio3380_set_password()	Set password
40	aio3380_change_password()	Change password
41	aio3380_clear_password()	Clear password
42	aio3380_unlock_security()	Unlock security function
43	aio3380_read_security_status()	Read security status

8. AIO-3382/3384 Error codes summary

8.1 AIO3382/3384 Error codes table

Error Code	Symbolic Name	Description
0	JSDRV_NO_ERROR	Success, No error.
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	SDRV_SET_SECURITY_ERROR	Password overwrite error
100	DEVICE_RW_ERROR	Device Read/Write error
101	JSDRV_NO_CARD	No AIO3382/3384 card on the system.
102	JSDRV_DUPLICATE_ID	AIO3382/3384 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 SW setting
301	JSAIO_MODE_ERROR	Mode parameter error. Parameter out of range.
302	JSAIO_CHANNEL_ERROR	Channel parameter error. 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. Parameter out of range.
401	JSAIO_STATE_ERROR	State parameter error. Parameter out of range.
402	JSAIO_POINT_ERROR	Point parameter error. Parameter out of range.
403	JSAIO_EEPROM_RW_ERROR	Eeprom R/W error
405	JSAIO_CALIBRATION_ERROR	Calibration error. Maybe out of range.
406	JSAIO_TIMERID_ERROR	TimerID parameter error. Parameter out of range.
407	JSAIO_TO_MODE_ERROR	To_mode parameter error. Parameter out of range.
408	JSAIO_TI_MODE_ERROR	Ti_mode parameter error. Parameter out of range.
409	JSAIO_PARAMETER_ERROR	Parameter error. Parameter out of range.
500	JSAIO_EVENT_ERROR	Event error. Maybe no event created.