

AIO3320A/21A

Analog Input and Multi-Function Digital I/O Card

Software Manual (V1.0)

健昇科技股份有限公司

JS AUTOMATION CORP.

新北市汐止區中興路 100 號 6 樓

6F., No.100, Zhongxing Rd.,

Xizhi Dist., New Taipei City, Taiwan

電話(TEL) : +886-2-2647-6936

傳真(FAX) : +886-2-2647-6940

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

<http://www.automation-js.com/>

E-mail : control.cards@automation.com.tw

Correction record

Version	Record
1.0	First publish

Contents

1. Compatibility about the AIO3320 and AIO3320A.....	4
2. How to install the software of AIO3320A.....	5
2.1 Install the PCI driver	5
3. Where to find the file you need	6
4. About the AIO3320A software	7
4.1 What you need to get started	7
4.2 Software programming choices.....	7
5. AIO3320A Language support.....	8
5.1 Building applications with the AIO3320A software library.....	8
5.2 AIO3320A Windows libraries	8
6. Basic concepts of analog I/O	9
6.1 Analog input (analog to digital converter)	9
6.2 Analog output (digital to analog converter)	9
6.3 Quantization error	10
6.4 Sample and hold	10
7. Function format and language difference	11
7.1 Function format	11
7.2 Variable data types	12
7.3 Programming language considerations	13
8. Flow chart of application implementation.....	15
8.1 AIO3320A Flow chart of application implementation (Basic AD operation)	15
8.2 AIO3320A Flow chart of application implementation (Embedded integral AD operation)	16
8.3 AIO3320A Flow chart of Timer / Counter / PWM application.....	17
8.4 AIO3320A Flow chart of interrupt.....	19
9. Software overview and dll function	20
9.1 Initialization and close	20
AIO3320A_initial	20
AIO3320A_initial_calibration	20
AIO3320A_close	20
AIO3320A_info	21
9.2 Analog input.....	22
AIO3320A_AD_config_set.....	22
AIO3320A_AD_config_read.....	23
AIO3320A_AD_value_read	23
AIO3320A_AD_value_read_no_calibration	24
AIO3320A_AD_data_read_no_calibration	24
AIO3320A_AD_integral_start.....	25
AIO3320A_AD_integral_all_read.....	25
AIO3320A_AD_integral_stop.....	26

9.3	Analog output	27
	AIO3320A_DA_set	27
	AIO3320A_DA_read	27
9.4	I/O Port R/W	28
	AIO3320A_port_set	28
	AIO3320A_port_read	28
	AIO3320A_point_set	29
	AIO3320A_point_read	29
	AIO3320A_dedicate_IO_set	30
	AIO3320A_dedicate_IO_read	30
9.5	Counter / Timer function	31
	AIO3320A_timer_set	32
	AIO3320A_counter_set	33
	AIO3320A_pwm_set	34
	AIO3320A_clock_frequency_set	34
	AIO3320A_GPT_set	35
	AIO3320A_GPT_enable	35
	AIO3320A_GPT_read	36
	AIO3320A_one_shot_command	36
	AIO3320A_parameter_read	37
9.6	Interrupt function	38
	AIO3320A_IRQ_link_process	39
	AIO3320A_IRQ_enable	39
	AIO3320A_IRQ_disable	39
	AIO3320A_IRQ_status_read	40
9.7	Error conditions	41
10.	Dll list	42
11.	AIO3320A Error codes summary	43
11.1	AIO3320A Error codes table	43

1. Compatibility about the AIO3320 and AIO3320A

The life cycle of industrial control cards normally remain several years but sometimes the chips phase out during its life cycle.

JS Automation tries to keep all the availability of its products before the market phase out. AIO3320A/1A is fully compatible with AIO3320/1, you can use it without any change of you old design but we strong recommend to use new driver and coding convention with new cards if you do not use it as just maintenance replacement of old version card.

card model	compatibility	
	hardware	software
AIO3320/1	----	use AIO3320.dll (new function do not support old version card)
AIO3320A/1A	direct compatible to AIO3320	AIO3320.dll
	direct compatible to AIO3320	use AIO3320A.dll with new coding convention that provides full functions

*1: old coding convention: the function call provides in AIO3320 software manual.

*2: new coding convention: the function call provides in AIO3320A software manual.

**We recommend to use new coding convention for AIO3320A for easier to upgrade if need.

2. How to install the software of AIO3320A

2.1 Install the PCI driver

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

In WinXP/7 and up system you should: (take Win XP 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. Do not response to the wizard, just Install the file
(..\AIO3320A\software\WinXP_7\ or if you download from website please execute the file AIO3320A_Install.exe to get the file)
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.

3. **Where to find the file you need**

WinXP/7 and up

The directory will be located at

.. \ **JS Automation** \AIO3320A\API\ (header files and lib files for VB,VC,BCB,C#)

.. \ **JS Automation** \AIO3320A\Driver\ (backup copy of AIO3320A drivers)

.. \ **JS Automation** \AIO3320A\exe\ (demo program and source code)

The system driver is located at ..\system32\Drivers and the DLL is located at ..\system.

For your easy startup, the demo program with source code demonstrates the card functions and help file.

4. **About the AIO3320A software**

AIO3320A 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 AIO3320A 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 AIO3320A functions within Windows' operation system environment.

4.1 What you need to get started

To set up and use your AIO3320A software, you need the following:

- AIO3320A software
- AIO3320A hardware
 - Main board
 - Wiring board (Option)

4.2 Software programming choices

You have several options to choose from when you are programming AIO3320A 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 AIO3320A software.

5. **AIO3320A Language support**

The AIO3320A software library is a DLL used with WinXP/7 and up. You can use these DLL with any Windows integrating development environment that can call Windows DLLs.

5.1 Building applications with the AIO3320A software library

The AIO3320A function reference topic contains general information about building AIO3320A applications, describes the nature of the AIO3320A files used in building AIO3320A applications, and explains the basics of making applications using the following tools:

Applications tools

- Microsoft Visual C/C++
- Borland C/C++
- Microsoft Visual C#
- Microsoft Visual Basic
- Microsoft VB.net

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

5.2 AIO3320A Windows libraries

The AIO3320A for Windows function library is a DLL called **AIO3320A.dll**. Since a DLL is used, AIO3320A functions are not linked into the executable files of applications. Only the information about the AIO3320A functions in the AIO3320A 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 AIO3320A functions in AIO3320A.dll.

Header Files and Import Libraries for Different Development Environments		
Language	Header File	Import Library
Microsoft Visual C/C++	AIO3320A.h	AIO3320AVC.lib
Borland C/C++	AIO3320A.h	AIO3320ABC.lib
Microsoft Visual C#	AIO3320A.cs	
Microsoft Visual Basic	AIO3320A.bas	
Microsoft VB.net	AIO3320A.vb	

Table 1

6. **Basic concepts of analog I/O**

6.1 Analog input (analog to digital converter)

Analog input (analog to digital conversion) is used for digital system to access external world's physical quantity. The most common specifications are: 5V, 10V for unipolar voltage input; +-5V, +-10V for bipolar voltage input. Also for current measuring: 0-20ma or 4-20ma is common for industrial applications.

In the conversion of analog signal to digital, the conversion time is defined as: the time required for an A/D converter to complete a single conversion to specified resolution and linearity for a full scale analog input change. Resolution is defined as: the smallest change that can be distinguished by an A/D converter. Resolution may be stated in percentage of full scale but is commonly expressed as the number of bit n where the converter has 2^n possible states.

Selecting a A/D, you must know how fast you need and how accuracy you required. As a thumb of rule, the conversion speed is recommended as 10 times of the under control process will get good result to re-construct the original waveform.

6.2 Analog output (digital to analog converter)

Analog output (digital to analog converter) is used for the digital system to simulate analog signal to external physical quantity. The most common specifications are: 5V, 10V for unipolar voltage output; +-5V, +-10V for bipolar voltage output. Also for current output: 0-20ma or 4-20ma is common for industrial applications (source or sink also possible).

In the conversion of digital to analog signal, the settling time is defined as: the time elapsed from the application of a full step input to a circuit to the time when the output has entered and remained within a specified error band around its final value. Resolution is defined as: the smallest change that can be distinguished by a D/A converter. Resolution may be stated in percentage of full scale but is commonly expressed as the number of bit n where the converter has 2^n possible states.

Selecting a D/A, you must know how fast the signal you will change and how accuracy you required. As a thumb of rule, the settling time decides the how fast your signal can be output. You cannot get accurate output shorter than the settling time.

6.3 Quantization error

On the fig. 6.1, if the vertical Y axis is the digital input you can see the output of D/A is step-wise. Same as if you take horizontal X axis as input, you can see the digital output that analog input quantized to digital output. The quantization error shown as fig 6.2

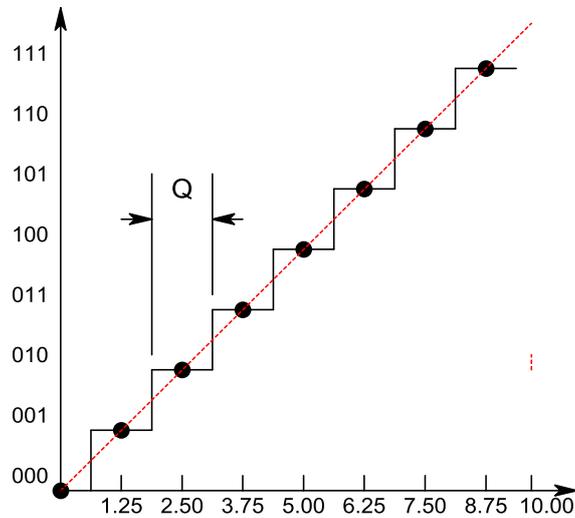


fig 6.1 Input-output relation for A/D or D/A

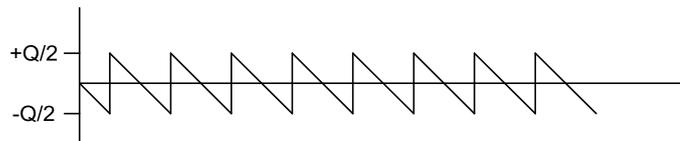


fig 6.2 Quantization error

6.4 Sample and hold

Sample and hold circuit often used to hold analog input to A/D, it will freeze the input and provide the A/D converter a near frozen state to achieve accurate conversion. (Be sure that if the input analog signal varies at the conversion period, you cannot get accurate conversion A/D data at the sample time slice). You can also use the circuit to hold the D/A signal which you can expand D/A channels with only one fast D/A converter to multiplex outputs.

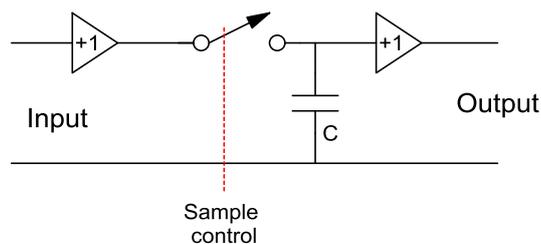


fig 6.3 Sample and hold

7. **Function format and language difference**

7.1 Function format

Every AIO3320A 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 AIO3320A function is the parameter **CardID** which is located the driver of AIO3320A 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 AIO3320A API. Read the following sections that apply to your programming language.

Note: Be sure to include the declaration functions of AIO3320A prototypes by including the appropriate AIO3320A header file in your source code. Refer to Building Applications with the AIO3320A 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 = AIO3320A_port_read(CardID, port, *data);
```

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

```
u8 CardID, port;  
u8 data,  
u32 Status;  
Status = AIO3320A_port_read (CardID, port, data);
```

7.3.2 Visual basic

The file AIO3320A.bas contains definitions for constants required for obtaining AIO Card information and declared functions and variable as global variables. You should use these constants symbols in the AIO3320A.bas, do not use the numerical values.

In Visual Basic, you can add the entire AIO3320A.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 AIO3320A.bas file for your project in Visual Basic 4.0, go to the **File** menu and select the **Add File...** option. Select AIO3320A.bas, which is browsed in the AIO3320A \ API directory. Then, select **Open** to add the file to the project.

To add the AIO3320A.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** AIO3320A.bas, which is in the AIO3320A \ 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 AIO3320ABC.lib AIO3320A.dll
```

Then add the **AIO3320ABC.lib** to your project and add

```
#include "AIO3320A.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 = AIO3320A_port_read(CardID, port, *data);
```

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

```
u8 CardID, port;
```

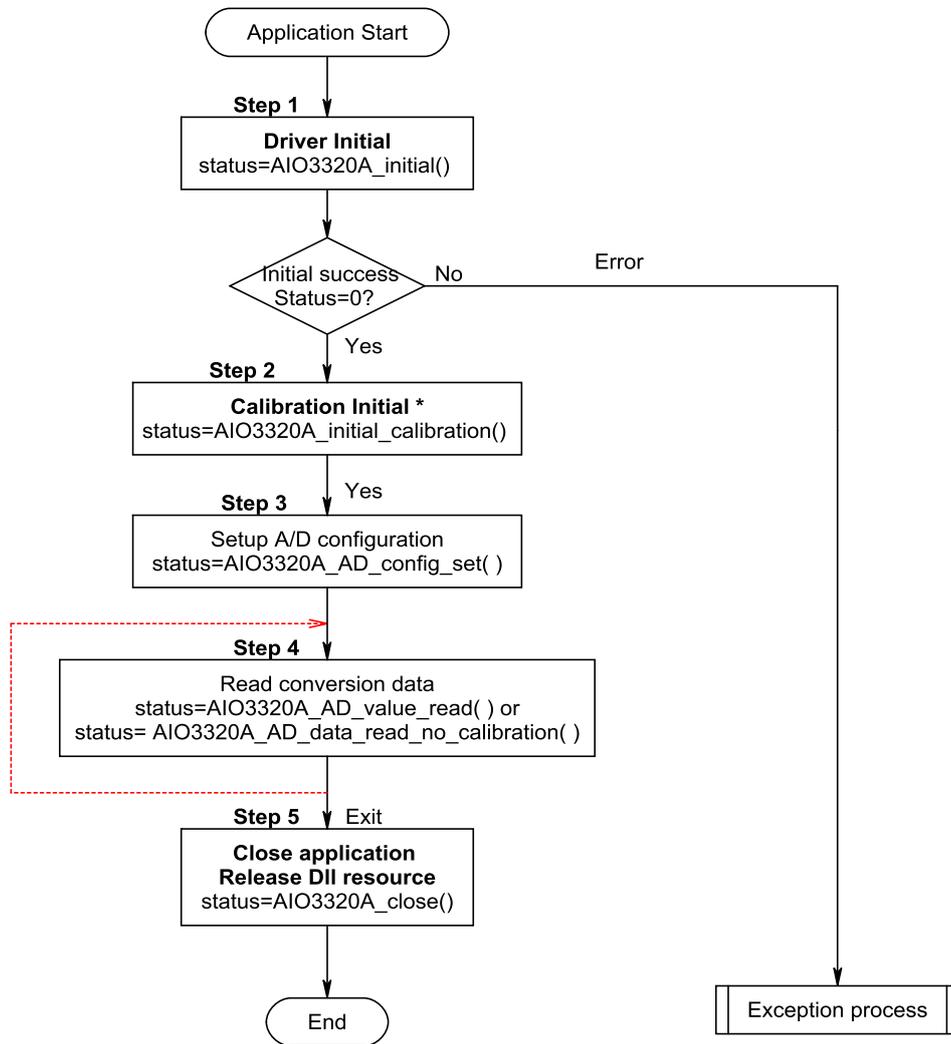
```
u8 data;
```

```
u32 Status;
```

```
Status =AIO3320A_port_read (CardID, port, data);
```

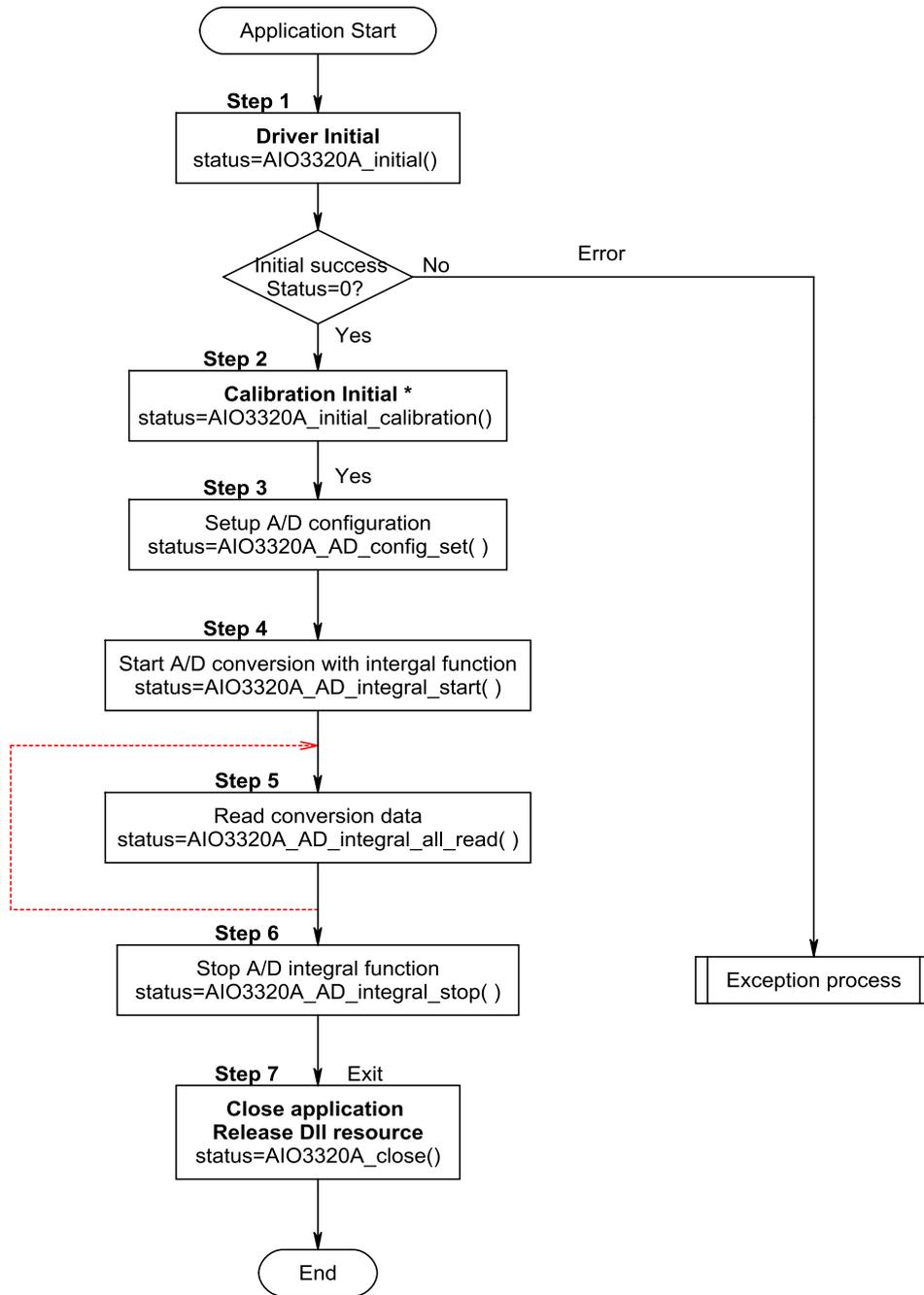
8. Flow chart of application implementation

8.1 AIO3320A Flow chart of application implementation (Basic AD operation)



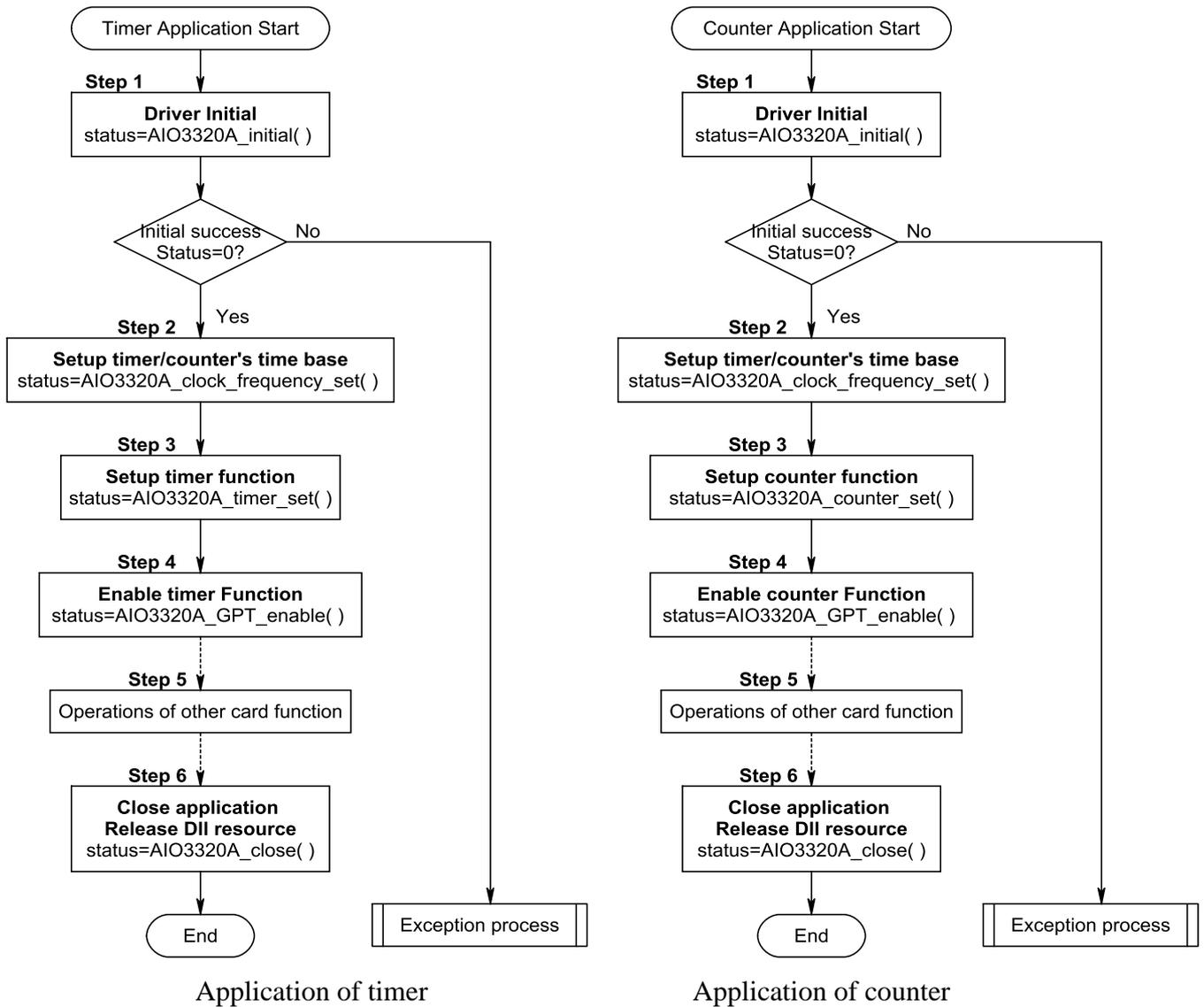
* If you need the data use factory pre-calibrated data to calibrate.

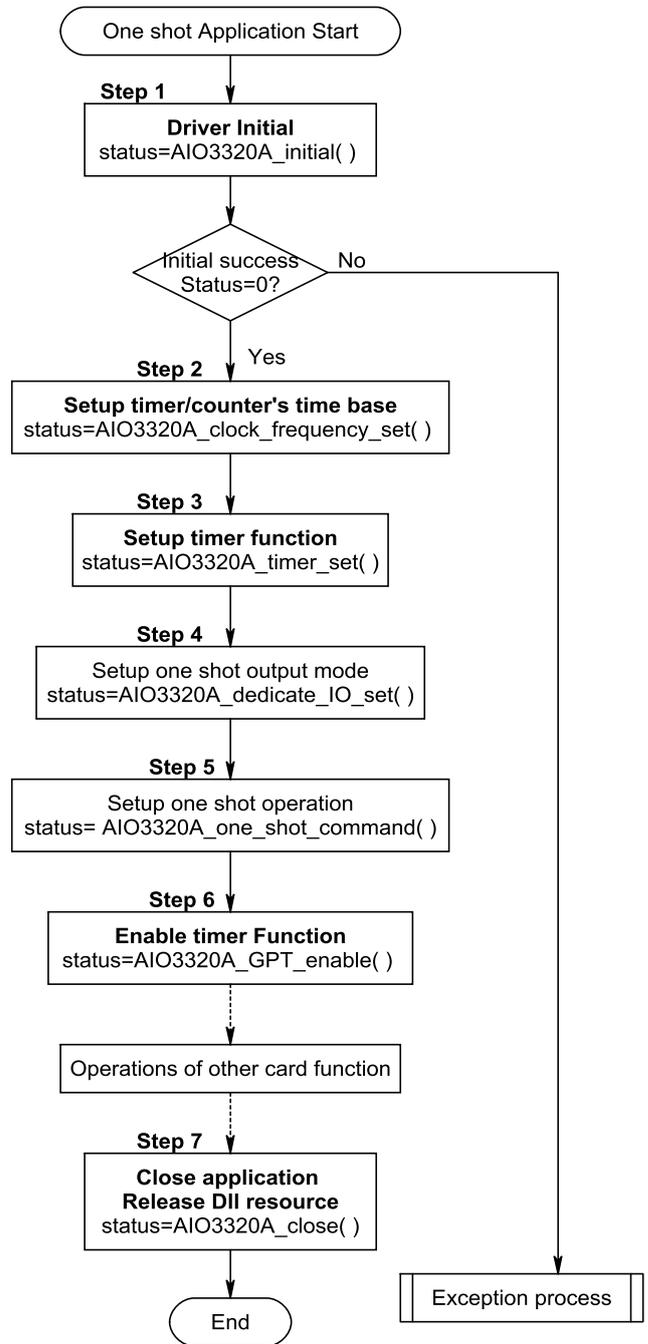
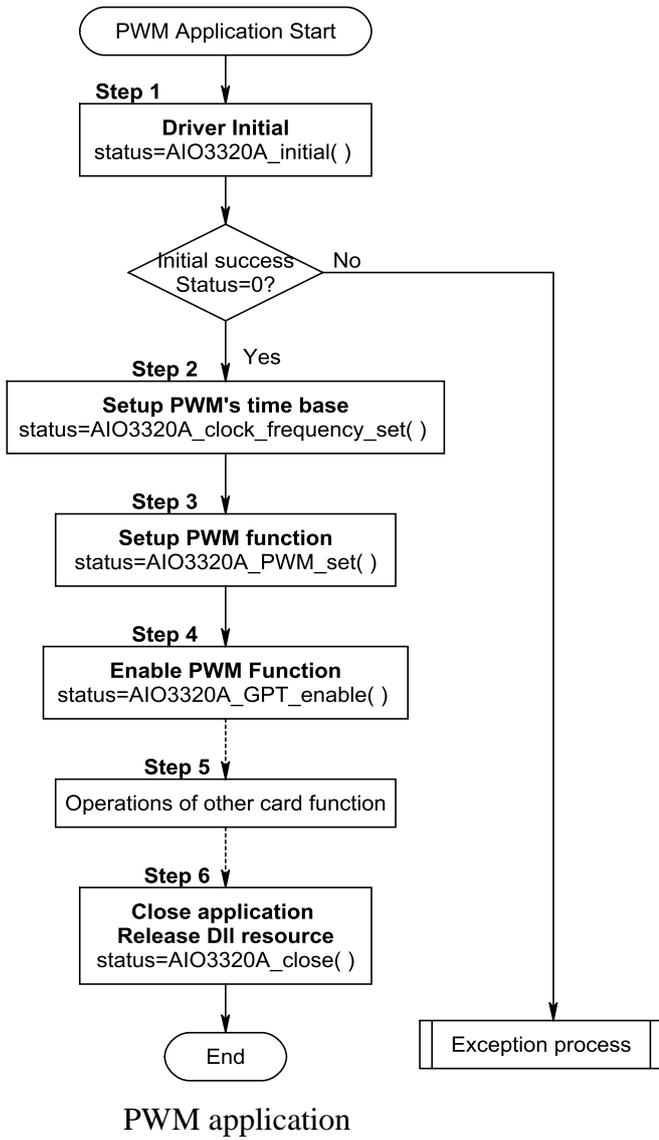
8.2 AIO3320A Flow chart of application implementation (Embedded integral AD operation)



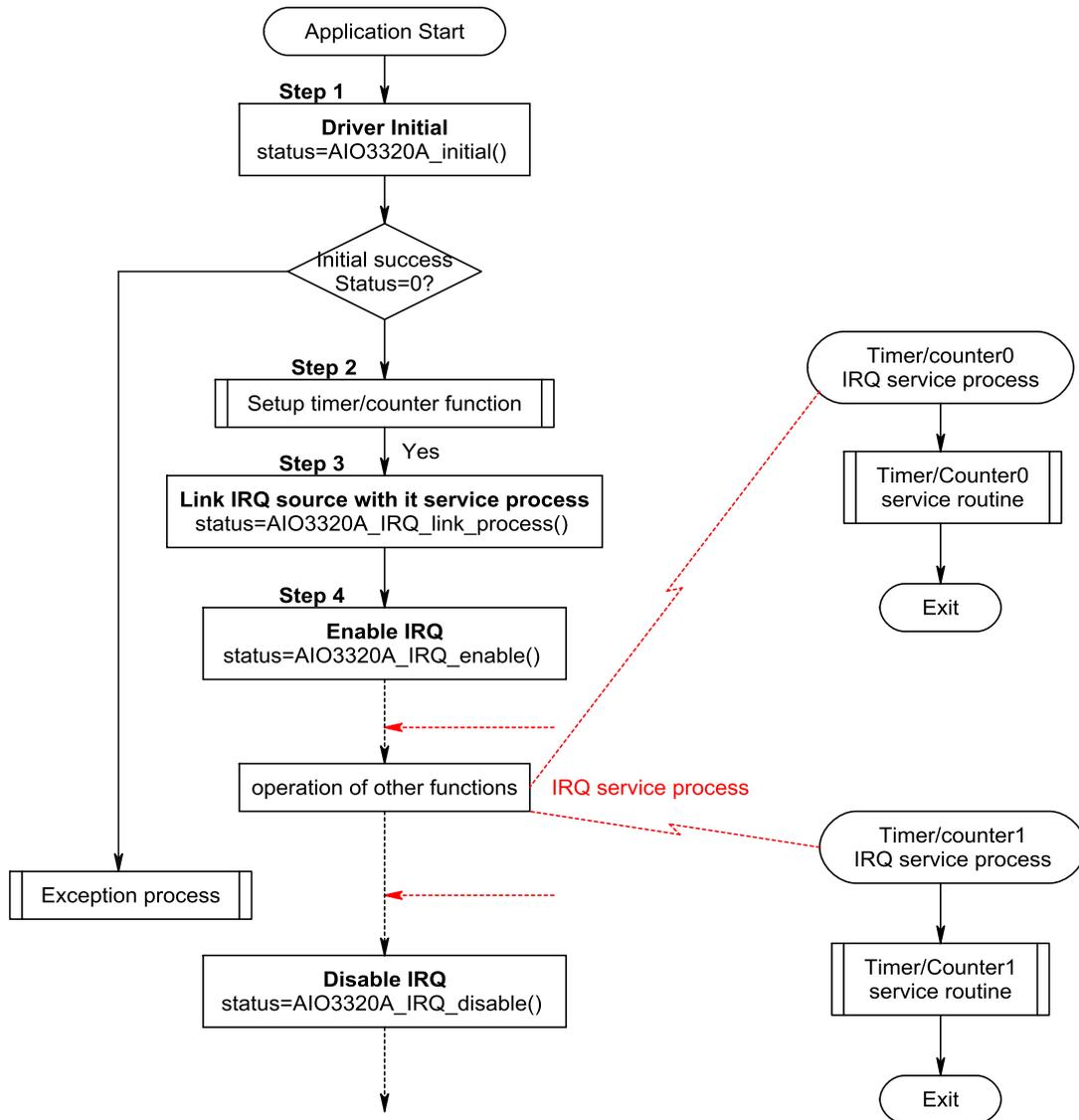
* If you need the data use factory pre-calibrated data to calibrate.

8.3 AIO3320A Flow chart of Timer / Counter / PWM application





8.4 AIO3320A Flow chart of interrupt



9. Software overview and dll function

9.1 Initialization and close

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

AIO3320A_initial() will do.

Before you want to A/D conversion, the factory pre-calibrated data should be initialized for A/D conversion.

AIO3320A_initial_calibration() will read factory calibrated data to working area.

Once you want to close your application, call

AIO3320A_close() to release all the resource.

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

AIO3320A_info() to get the address.

● **AIO3320A_initial**

Format : u32 status =AIO3320A_initial(void)

Purpose: Initial the AIO3320A resource when start the Windows applications.

● **AIO3320A_initial_calibration**

Format : u32 status =AIO3320A_initial_calibration(u8 CardID)

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW

● **AIO3320A_close**

Format : u32 status =AIO3320A_close(void);

Purpose: Release the AIO3320A resource when close the Windows applications.

● **AIO3320A info**

Format : u32 status =AIO3320A_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

9.2 Analog input

The AIO3320A now are 8 channels 16 bit AD cards. There are different channel D/A numbers for AIO3320A (2 channels) and AIO3320A (4 channels).

You must configure the input range of the specific channel by:

AIO3320A_AD_config_set() and read back the configuration for verification by:
AIO3320A_AD_config_read()

To read the input voltage value with the factory pre-calibrated data by:

AIO3320A_AD_value_read(), it can be also read without the calibration by
AIO3320A_AD_value_read_no_calibration()

If your application only need the raw AD data, you can read AD data by:

AIO3320A_AD_data_read_no_calibration()

If your environment is noisy or you need to get more accurate data, you can use the integral function built in dll by:

AIO3320A_AD_integral_start() to start the embedded integration of the system.
AIO3320A_AD_integral_all_read() to read all channels.
AIO3320A_AD_integral_stop() to stop integration function.

● **AIO3320A AD config set**

Format : `u32 status = AIO3320A_AD_config_set(u8 CardID,u8 channel,u8 mode)`

Purpose: Set A/D configuration.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
channel	u8	A/D channel number 0~7: 8 channels AD
mode	u8	scale range: 0: 0V ~ 5V 1: -5V ~ +5V 2: 0V ~ 10V 3: -10V ~ +10V 255 : AD stop operation.

Note: 0~5V, 0~20mA or 4~20mA are set by hardware but software must configure to meet. For the first time after power on or the first time after application opened, you should configure the AD mode as the hardware setting else **incorrect** reading will be.

● **AIO3320A AD config read**

Format : u32 status = AIO3320A_AD_config_read(u8 CardID,u8 channel,u8 *mode)

Purpose: Read A/D config.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
channel	u8	A/D channel number 0~7: 8 channels AD

Output:

Name	Type	Description
mode	u8	scale range: 0: 0V ~ 5V 1: -5V ~ +5V (Default) 2: 0V ~ 10V 3: -10V ~ +10V 255 : AD stop operation.

● **AIO3320A AD value read**

Format : u32 status = AIO3320A_AD_value_read(u8 CardID, u8 channel, f32 *value)

Purpose: Read voltage value with pre-calibration data.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
channel	u8	A/D channel number 0~7: 8 channels AD

Output:

Name	Type	Description
value	f32	Voltage value based on the AD converted and calibrated data. Say if the AD scale range is set at 0~5V then the voltage value returned will be in the 0~5 range.

Note: Use *AIO3320A_initial_calibration()* to load calibration data first then the further calibration will be effective.

● **AIO3320A AD value read no calibration**

Format : u32 status = AIO3320A_AD_value_read_no_calibration (u8 CardID, u8 channel, f32 *value)

Purpose: Read voltage value.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
channel	u8	A/D channel number 0~7: 8 channels AD

Output:

Name	Type	Description
value	f32	Voltage value based on the AD converted data only. Say if the AD scale range is set at 0~10V then the voltage value returned will be in the 0~10 range.

● **AIO3320A AD data read no calibration**

Format : u32 status = AIO3320A_AD_data_read_no_calibration (u8 CardID, u8 channel, u16 *data)

Purpose: Read A/D data.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
channel	u8	A/D channel number 0~7: 8 channels AD

Output:

Name	Type	Description
data	u16	AD 16 bit data, 0~65535: if unipolar AD mode, 0~5V or 0~10V range 0~32767: if bipolar AD mode, in 0~5V or 0~10V range 65535~32768: if bipolar AD mode, in 0~ -5V or 0 ~ -10V range

● **AIO3320A AD integral start**

Format : u32 status = AIO3320A_AD_integral_start(u8 CardID,u8 mode)

Purpose: start AD conversion with integral constant.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by jumper setting
mode	u8	0: immediately access, no integration 1: integration time 100ms 2: integration time 200ms 3: integration time 300ms 4: integration time 400ms 5: integration time 500ms 6: integration time 600ms 7: integration time 700ms 8: integration time 800ms 9: integration time 900ms 10: integration time 1s

● **AIO3320A AD integral all read**

Format : u32 status = AIO3320A_AD_integral_all_read(u8 CardID,u16 data[8])

Purpose: read integral result of AD conversion data.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by jumper setting

Output:

Name	Type	Description
data[8]	u16	Channel 0 ~7 AD data

Note:

To read all channels in integral

Start integral mode by **AIO3320A_AD_integral_start**. (AD interrupt is not allowed to use)

Read all channels by **AIO3320A_AD_integral_all_read**.

Stop AD integration function by **AIO3320A_AD_integral_stop**.

- **AIO3320A AD integral stop**

Format : u32 status = AIO3320A_AD_integral_stop(u8 CardID)

Purpose: stop AD integral conversion.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by jumper setting

9.3 Analog output

The AIO3320A provides 2 channels 17 bit DA and AIO3321A has 4 channels 17 bit DA. You can command the DA output in 16 bit magnitude with an extra sign bit to control the output polarity.

AIO3320A_DA_set() to output DA data.

AIO3320A_DA_read() to read back DA data you send.

● **AIO3320A DA set**

Format : u32 status = AIO3320A_DA_set(u8 CardID,u8 channel, u16 data,u8 sign)

Purpose: DA output

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by jumper setting
channel	u8	DA channel no. (0~7 for AIO3320A) (0~15 for AIO3321A)
data	u16	data to be input -10V ~+10V : 0~65535, sign= 0 or 1
sign	u8	0: output is positive voltage 1: output is negative voltage

● **AIO3320A DA read**

Format : u32 status = AIO3320A_DA_read(u8 CardID,u8 channel, u16 *data, u8 * sign)

Purpose: read back DA data

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by jumper setting
channel	u8	DA channel no. (0~7 for AIO3320A) (0~15 for AIO3321A)

Output:

Name	Type	Description
data	u16	data to be output -10V ~+10V : 0~65535, sign= 0 or 1
sign	u8	0: output is positive voltage 1: output is negative voltage

9.4 I/O Port R/W

There are one 8 bit isolated digital input port and 8 bit isolated digital output port associate with AIO3320A/1A card. The IN1,IN0 also serve as special function input and OUT1 and OUT0 serve as special output of timer/counter function.

To output the data to OUT7~OUT0 use:

AIO3320A_port_set() and read the input data by

AIO3320A_port_read()

If you want to operate on bit base, use

AIO3320A_point_set() will set/reset bit and

AIO3320A_point_read() to read back input bit data.

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

AIO3320A_dedicate_IO_set() can be used to change the IN1,IN0 as general or timer/counter dedicated input,OUT1,OUT0 as general or timer/counter dedicated output. To read back status of dedicated I/O use:

AIO3320A_dedicate_IO_read()

● **AIO3320A port set**

Format : u32 status = AIO3320A_port_set(u8 CardID,u8 data)

Purpose: To output data

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
data	u8	data to be output

● **AIO3320A port read**

Format : u32 status = AIO3320A_port_read (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: input port 1: output port

Output:

Name	Type	Description
data	u8	port data

● **AIO3320A point set**

Format : u32 status = AIO3320A_point_set(u8 CardID,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
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

● **AIO3320A point read**

Format : u32 status = AIO3320A_point_read (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: input port 1: output port
point	u8	bit designation 7~0: bit7~ bit0

Output:

Name	Type	Description
state	u8	bit data to be output

● **AIO3320A dedicate IO set**

Format : u32 status = AIO3320A_dedicate_IO_set(u8 CardID,u8 enable)

Purpose: To set IN0, IN1, O0, O1 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 IN1,OUT1 for timer/counter1 IN0,OUT0 for timer/counter0

● **AIO3320A dedicate IO read**

Format : u32 status = AIO3320A_dedicate_IO_read(u8 CardID,u8 *enable)

Purpose: To set IN0, IN1, O0, O1 as dedicate I/O or general I/O

Parameters:

Input:

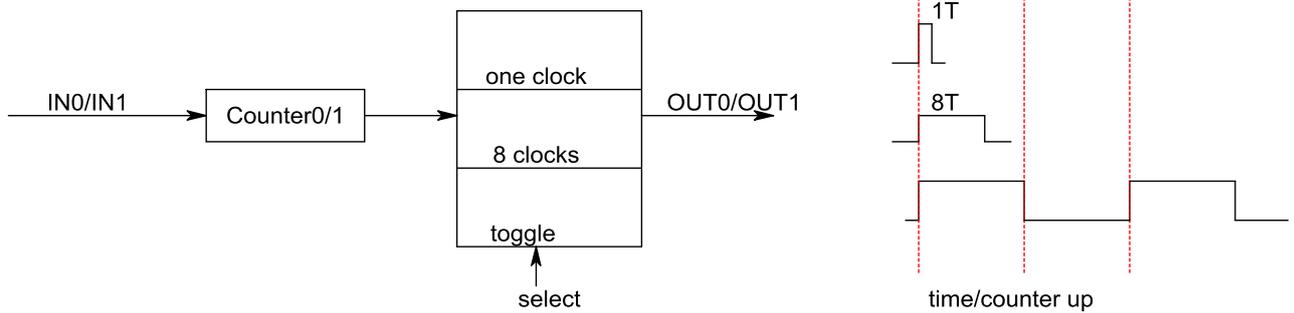
Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW

Output:

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

9.5 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

AIO3320A_timer_set() to configure as timer and its output mode

AIO3320A_counter_set() to configure as counter and its input and output mode

AIO3320A_pwm_set() to configure as PWM generator.

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

AIO3320A_clock_frequency_set() 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

AIO3320A_GPT_set() will do.

Once you have setup the data

AIO3320A_GPT_enable() will enable or disable the counter/timer function.

To check the current counter/timer value

AIO3320A_GPT_read() will do.

The special function of one shot mode can be applied by

AIO3320A_one_shot_command().

To readback the parameters you set

AIO3320A_parameter_read() will response the current working mode and the parameters you set for individual timer, counter.

● **AIO3320A timer set**

Format : u32 status = AIO3320A_timer_set(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

● **AIO3320A counter set**

Format : `u32 status = AIO3320A_counter_set(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 (4/33MHz) while Ti is low 1: Counter decrease with system clock (4/33MHz) 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 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 `AIO3320A_dedicate_IO_set()`.

● **AIO3320A_pwm_set**

Format : u32 status = AIO3320A_pwm_set(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 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 low (the high word is 0x5).
3. The PWM output is limited by the response time of output interface. The pulse width less than 100us may be distorted.

● **AIO3320A_clock_frequency_set**

Format : u32 status = AIO3320A_clock_frequency_set(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: 33MHz

● **AIO3320A GPT set**

Format : u32 status = AIO3320A_GPT_set(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,

● **AIO3320A GPT enable**

Format : u32 status = AIO3320A_GPT_enable (u8 CardID,u8 TimerID,u8 enable)

Purpose: To enable/disable timer/counter function that was set by AIO3320A_timer_set(), AIO3320A_counter_set(), AIO3320_pwm_set()

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

● **AIO3320A GPT read**

Format : status=AIO3320A_GPT_read(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

● **AIO3320A one shot command**

Format : status=AIO3320A_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 OUT0, Counter/Timer1 output is at OUT1, and before using this function, port should be configured as dedicated port by AIO3320A_dedicate_IO_set().

● **AIO3320A parameter read**

Format : status=AIO3320A_parameter_read(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>AIO3320A_counter_set</i> For To_mode of counter, refer to AIO3320A_counter_set For Ti_mode of timer, refer to AIO3320A_timer_set

9.6 Interrupt function

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

AIO3320A provide timer/counter interrupts, to use interrupt function use

AIO3320A_IRQ_link_process() to link your irq service routine,

AIO3320A_IRQ_enable() to enable it and

AIO3320A_IRQ_disable() 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 enable/disable status

AIO3320A_IRQ_status_read() will do.

● **AIO3320A IRQ link process**

Format : u32 status = AIO3320A_IRQ_link_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

● **AIO3320A IRQ enable**

Format : u32 status = AIO3320A_IRQ_enable (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

● **AIO3320A IRQ disable**

Format : u32 status = AIO3320A_IRQ_disable(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

● **AIO3320A IRQ status read**

Format : u32 status = AIO3320A_IRQ_status_read(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

9.7 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 AIO3320A functions.

10. Dll list

	Function Name	Description
1.	AIO3320A_initial()	Initial card
2.	AIO3320A_initial_calibration()	Calibration ad
3.	AIO3320A_close()	Close card
4.	AIO3320A_info()	Read the card information
5.	AIO3320A_AD_config_set()	Set AD configuration
6.	AIO3320A_AD_config_read()	Rad AD configuration
7.	AIO3320A_AD_value_read()	Read AD value
8.	AIO3320A_AD_value_read_no_calibration()	Read AD value no calibration
9.	AIO3320A_AD_data_read_no_calibration()	Read AD data no calibration
10.	AIO3320A_AD_integral_start()	Start AD integral mode
11.	AIO3320A_AD_integral_all_read()	Read all AD integral data
12.	AIO3320A_AD_integral_stop()	Stop AD integral mode
13.	AIO3320A_DA_set()	Set DA
14.	AIO3320A_DA_read()	Read DA
15.	AIO3320A_port_set()	Set Output port
16.	AIO3320A_port_read()	Read Output and Input pot
17.	AIO3320A_point_set()	Set Output point
18.	AIO3320A_point_read()	Read Output and Input point
19.	AIO3320A_dedicate_IO_set()	Set dedicate IO
20.	AIO3320A_dedicate_IO_read()	Read dedicate IO
21.	AIO3320A_timer_set()	Set timer mode
22.	AIO3320A_counter_set()	Set counter mode
23.	AIO3320A_pwm_set()	Set PWM mode
24.	AIO3320A_clock_frequency_set()	Set frequency
25.	AIO3320A_GPT_set()	Set GPT data
26.	AIO3320A_GPT_enable()	Control GPT mode
27.	AIO3320A_GPT_read()	Read GPT data
28.	AIO3320A_one_shot_command()	Set One Shot data
29.	AIO3320A_parameter_read()	Read GPT parameter
30.	AIO3320A_IRQ_link_process()	Link callback
31.	AIO3320A_IRQ_enable()	Enable IRQ
32.	AIO3320A_IRQ_disable()	Disable IRQ
33.	AIO3320A_IRQ_status_read()	Read IRQ status

11. AIO3320A Error codes summary

11.1 AIO3320A 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 AIO3320A card on the system.
102	JSDRV_DUPLICATE_ID	AIO3320A 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.
410	JSAIO_OPERATION_ERROR	Operation error.
500	JSAIO_EVENT_ERROR	Event error. Maybe no event created.