DIO3208B

Digital I/O Card

Software Manual (V3.0)

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

TEL：+886-2-2647-6936

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	wdm3208B.sys V1.0
	drv3208B.dll V1.0
	DIO3208B.dll V1.0
2.0	wdm3208B.sys V2.0
	drv3208B.dll V2.0
	DIO3208B.dll V2.0
2.1	revised to new manual style
3.0	disable the software key function with return value always true

Note: V2.0 only for DIO3208B

Contents

1. Using the DIO3208B to replace DIO3208/A

Owing to phase out of the DIO3208/A, the new card DIO3208B is designed as the replacement. In fact, the DIO3208B can simulate the DIO3208/A only on the digital I/O function and provides new powerful timer/counter functions.

If your application does not use any of the timer/counter functions, you can just replace the card and re-install the driver software, everything is no change. But **if your application uses the timer/counter function, please use the new function and re-coding your program.**

2. How to install the software of DIO3208B

2.1 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 Win2K/XP/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 (..\DIO3208B\Software\Win2K_up\ or if you download from website please execute the file DIO3208B_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

Win2K/XP/7 and up

The directory will be located at

.. \ JS Automation \DIO3208B\API (header files and lib files for VB,VC,BCB,C#)

.. \ JS Automation \DIO3208B\Driver (backup copy of DIO3208B drivers)

.. \ JS Automation \DIO3208B\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 DIO3208B software

DIO3208B 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 DIO3208B 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 DIO3208B functions within Windows' operation system environment.

4.1 What you need to get started

To set up and use your DIO3208B software, you need the following:

- DIO3208B software
- DIO3208B hardware

  Main board

  Wiring board (Option)

4.2 Software programming choices

You have several options to choose from when you are programming DIO3208B 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 DIO3208B software.

5. DIO3208B Language support

The DIO3208B software library is a DLL used with Win2K/XP/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 DIO3208B software library

The DIO3208B function reference topic contains general information about building DIO3208B applications, describes the nature of the DIO3208B files used in building DIO3208B 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 DIO3208B Windows libraries

The DIO3208B for Windows function library is a DLL called **DIO3208B.dll**. Since a DLL is used, DIO3208B functions are not linked into the executable files of applications. Only the information about the DIO3208B functions in the DIO3208B 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 DIO3208B functions in DIO3208B.dll.

Header Files and Import Libraries for Different Development Environments		
Language	**Header File**	**Import Library**
Microsoft Visual C/C++	DIO3208B.h	DIO3208BVC.lib
Borland C/C++	DIO3208B.h	DIO3208BBC.lib
Microsoft Visual C#	DIO3208B.cs	
Microsoft Visual Basic	DIO3208B.bas	
Microsoft VB.net	DIO3208B.vb	

Table 1

6. Basic concepts of digital I/O control

The digital I/O control is the most common type of PC based application. For example, on the main board, printer port is the TTL level digital I/O.

Types of I/O calssified by isolation

If the system and I/O are not electrically connected, we call it is isolated. There are many kinds of isolation: by transformer, by photo-coupler, by magnetic coupler,… Any kind of device, they can brake the electrical connection without braking the signal is suitable for the purpose.

Currently, photo-coupler isolation is the most popular selection, isolation voltage up to 2000V or over is common. But the photo-coupler is limited by the response time, the high frequency type cost a lot. The new selection is magnetic coupler, it is design to focus on high speed application.

The merit of isolation is to avoid the noise from outside world to enter the PC system, if the noise comes into PC system without elimination, the system maybe get "crazy" by the noise disturbance. Of course the isolation also limits the versatile of programming as input or output at the same pin as the TTL does. The inter-connection of add-on card and wiring board maybe extend to several meters without any problem.

The non-isolated type is generally the TTL level input/output. The ground and power source of the input/output port come from the system. Generally you can program as input or output at the same pin as you wish. **The connection of wiring board and the add-on board is limited to 50cm or shorter** (depends on the environmental noise condition).

Types of Output calssified by driver device

There are several devices used as output driver, the relay, transistor or MOS FET, SCR and SSR. Relay is electric- mechanical device, it life time is about 1,000,000 times of switching. But on the other hand it has many selections such as high voltage or high current. It can also be used to switch DC load or AC load.

Transistor and MOS FET are basically semi-permanent devices. If you have selected the right ratings, it can work without switching life limit. But the transistor or MOS FET can only work in DC load condition.

The transistor or MOS FET also have another option is source or sink. For PMOS or PNP transistor is source type device, the load is one terminal connects to output and another connects to common ground, but NPN or NMOS is one terminal connects to output and the other connects to VCC+. **If you are concerned about hazard from high DC voltage while the load is floating, please choose the source type driver device.**

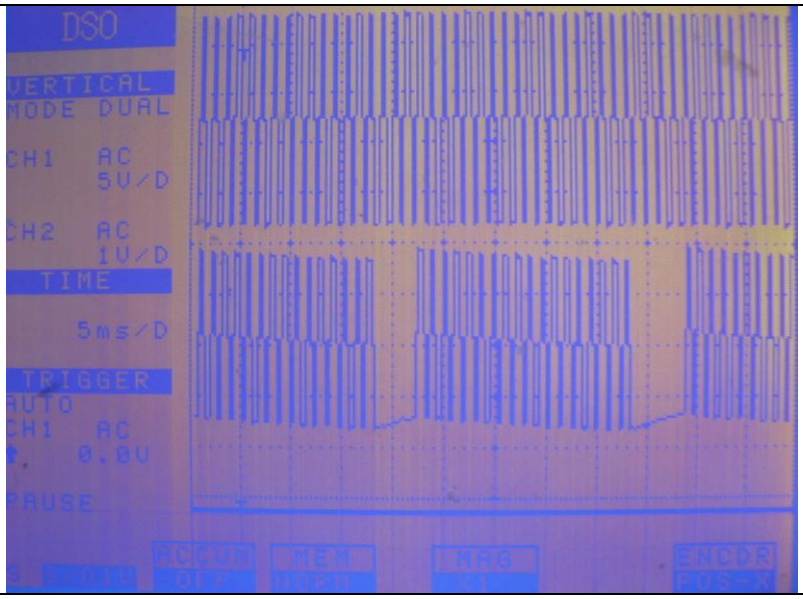
SCR (or triac) is seldom direct connect to digital output, but his relative SSR is the most often selection. In fact, SSR is a compact package of trigger circuit and triac. You can choose zero cross trigger (output command only turn on the output at power phase near zero to eliminate surge) or direct turn on type. SSR is working in AC load condition.

Input debounce

Debounce is the function to filter the input jitters. From the microscope view of a switch input, you will see the contact does not come to close or release to open clearly. In most cases, it will contact-release-contact-release… for many times then go to steady state (ON or OFF). If you do not have the debounce function, you will read the input at high state and then next read will get low state, this maybe an error data for your decision of contact input.

Debounce can be implemented by hardware or software. Analog hardware debounce circuit will have fixed time constant to filter out the significant input signal, if you want to change the response time, the only way is to change the circuit device.

If digital debounce is implemented, maybe several filter frequency you can choose. To choose the filter frequency, please keep the Nyquist–Shannon sampling theorem in mind: filter sample frequency must at least twice of the input frequency. The following sample is a bad selection of debounce filter, the input frequency is not as low as less than half of the sample frequency, the output will generate a beat frequency. The DIO3208B has built-in digital debounce function by hardware, you can choose the debounce frequency from 100Hz, 200Hz up to 1KHz and if you need faster input frequency, you can program it no debounce (only limit by the photo-isolator response time).



<- Input frequency at 835Hz

<- Output of digital filter,
Please note the beat frequency.

Digital debounce circuit work at 1KHZ sample rate and observe the output of filter from 835Hz input

Software debounce will consumes the CPU time a lot, we do not recommend to use except for you really know you want.

Input interrupt

You can scan the input by polling, but the CPU will spend a lot of time to do null task. Another way is use a timer to sample the input at adequate time (remind the Nyquist–Shannon sampling theorem, at least double of the input frequency). The third one is directly allows the input to generate interrupt to CPU. To use direct interrupt from input, the noise coupled from input must take special care not to mal-trigger the interrupt.

Read back of Output status

Some applications need to read back the output status, if the card do not provide output status read back, you can use a variable to store the status of output before you really command it output. Some cards provide the read back function but please note that **the read back status is come from the output register, not from the real physical output.**

7. Basic concepts of timer / counter function

The DIO3208B provides powerful timer/counter functions. There are 3 working modes to select: Timer mode, counter mode and PWM mode. The simplified function block diagram shown as follows:

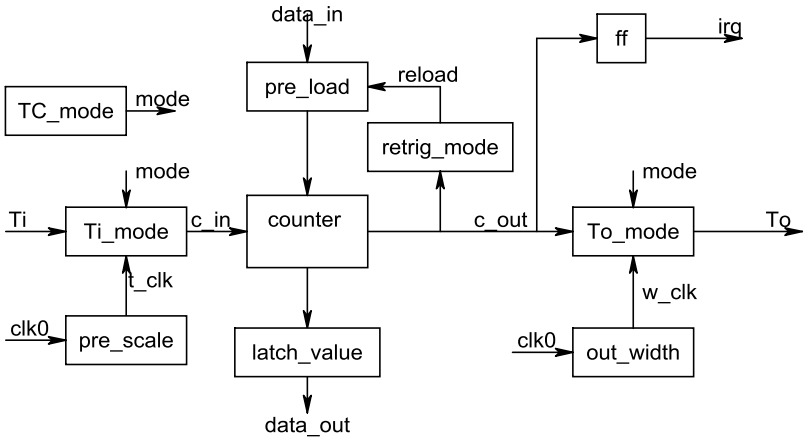


fig 7.1 timer/counter function block diagram

Timer function

The timer function based on 1us time base. There is a timer input (IN00) and a timer output (OUT00). The digital input IN00 can be used as the gate control of timer or general purpose digital input. The digital output OUT00 can be timer/counter output or general purpose digital output. They depend on the working mode of timer counter you choose.

The timer mode can be simplified as the following block. The kernel function is the counter.

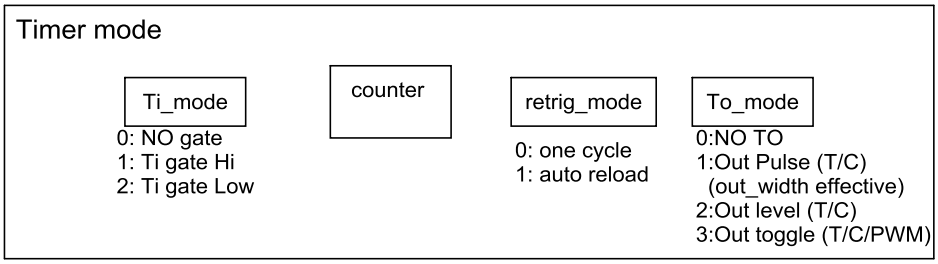


fig 7.2 function block of timer mode

Retrigger mode:

one cycle : The timer can work one time (timer up (timer down count to zero) will also stop itself).

auto reload : The timer will auto reload as timer up, it is so called continuous mode.

Ti_mode: there are 3 working modes

NO_GATE: IN00 as general purpose input, no gate function

GATED_HIGH: IN00 is gate input, the timer will be counting while the gate input is high and halt timer counting while it is low after the timer function starts.

GATED_LOW: IN00 is gate input, the timer will be counting while the gate input is low and halt timer counting while it is high after the timer function starts.

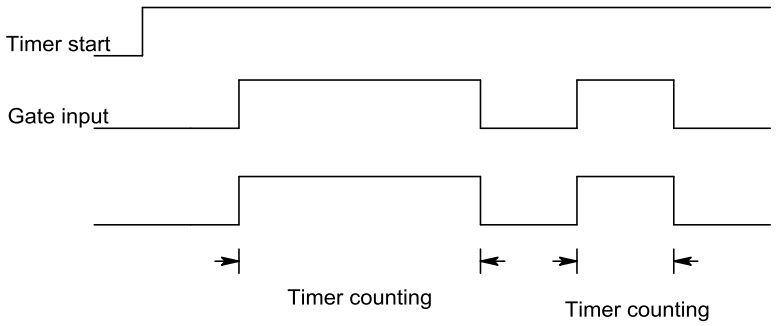


fig 7.3 timer counting with Ti gate function (GATED_HIGH for example)

To_mode: There are 4 working modes of timer / counter output

NO_TOUT : OUT00 as general digital output

OUT_PULSE : OUT00 as timer cross zero pulse output. (output width can be set)

OUT_LEVEL : OUT00 on timer cross zero output will make.

OUT_TOGGLE: OUT00 on timer cross zero toggles.

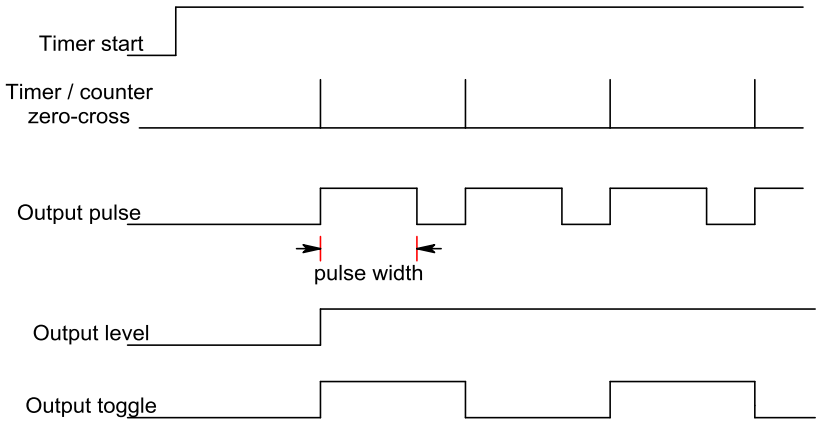


fig 7.4 Different output mode of timer / counter zero crossing

Counter function

You can think counter function as timer function except timer counts the 1us time base and counter counts the pulse from external signal input IN00.

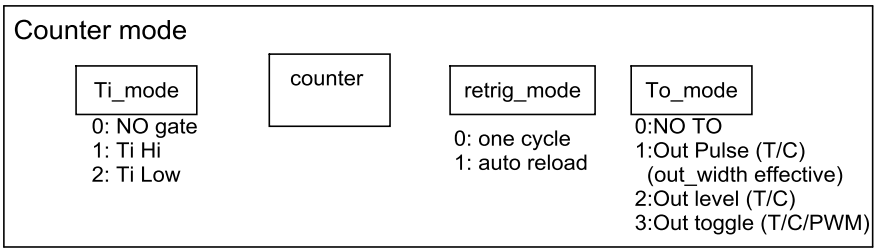


fig 7.5 function block of counter mode

Retrigger mode:

one cycle : The counter can work one time (counter down count to zero will also stop itself).

auto reload : The counter will auto reload counter constant as count down to zero, it is so called continuous mode.

Ti_mode: there are 3 working modes

NO_GATE: IN00 as general purpose input, no gate function

Ti_HIGH: IN00 is counter trigger input, the counter will be counting while the trigger input is low to high transition after the timer function starts.

Ti_LOW: IN00 is counter trigger input, the counter will be counting while the trigger input is high to low transition after the timer function starts.

To_mode: There are 4 working modes of timer / counter output

NO_TOUT : OUT00 as general digital output

OUT_PULSE : OUT00 as counter cross zero pulse output. (output width can be set)

OUT_LEVEL : OUT00 on counter cross zero output will make.

OUT_TOGGLE: OUT00 on counter cross zero toggles.

(refer fig 7.4 Different output mode of timer / counter zero crossing)

PWM function

PWM function is a special mode of timer. generally it has 2 time constant, one is the period (or frequency) of the waveform and the other is the duty. DIO3208B divide the time constant to 2 16bit parts. You must set the period longer than duty to have it function correctly.

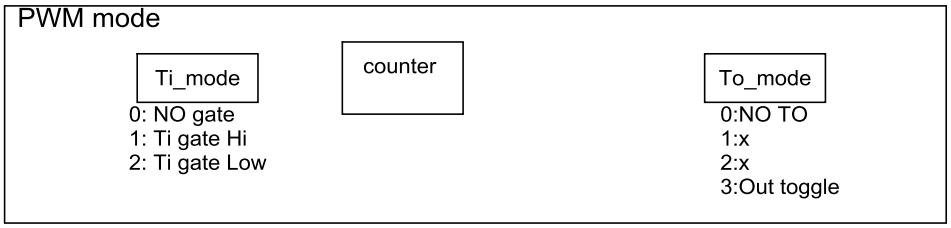


fig 7.6 function block of PWM mode

Ti_mode: there are 3 working modes

NO_GATE: IN00 as general purpose input, no gate function

GATED_HIGH: IN00 is gate input, while the gate input is high the PWM generates output and while the input low stops the PWM output after PWM function starts.

GATED_LOW: IN00 is gate input, while the gate input is low the PWM generates output and while the input high stops the PWM output after PWM function starts..

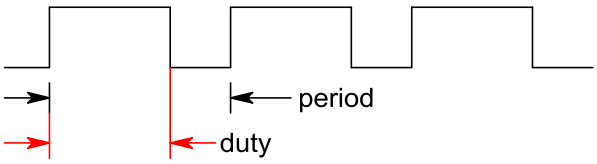
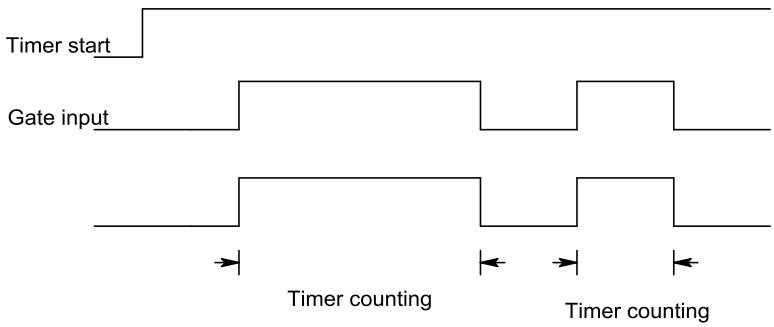


fig 7.7 PWM waveform and definition



8. Function format and language difference

8.1 Function format

Every DIO3208B 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. (refer chapter 12 DIO3208 Error codes summary)
Note: **Status** is a 32-bit unsigned integer.

The first parameter to almost every DIO3208B function is the parameter **CardID** which is located the driver of DIO3208B 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**)

8.2 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	-1.797685123862315E+308 to 1.797685123862315E+308	double	Double (for example: voltage Number)	Double

Table 2

8.3 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 DIO3208B API. Read the following sections that apply to your programming language.

Note: Be sure to include the declaration functions of DIO3208B prototypes by including the appropriate DIO3208B header file in your source code. Refer to Building Applications with the DIO3208B Software Library for the header file appropriate to your compiler.

8.3.1 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 = DIO3208B_read_port(u8 CardID, u8 port, u8*data);

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

```
u8 CardID=0, port=0;      //assume CardId=0 and port=0
u8 data,
u32 Status;
Status = DIO3208B_read_port (CardID, port, &data);
```

8.3.2 Visual basic

The file DIO3208B.bas contains definitions for constants required for obtaining DIO Card information and declared functions and variable as global variables. You should use these constants symbols in the DIO3208B.bas, do not use the numerical values.

In Visual Basic, you can add the entire DIO3208B.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 DIO3208B.bas file for your project in Visual Basic 4.0, go to the **File** menu and select the **Add File... option**. Select Dio3208.bas, which is browsed in the DIO3208B \ API directory. Then, select **Open** to add the file to the project.

To add the DIO3208B.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** DIO3208B.bas, which is in the DIO3208B \ API directory. Then, select **Open** to add the file to the project.

8.3.3 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 DIO3208BBC.lib DIO3208B.dll

Then add the **DIO3208BBC.lib** to your project and add

#include "DIO3208B.h" to main program.

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

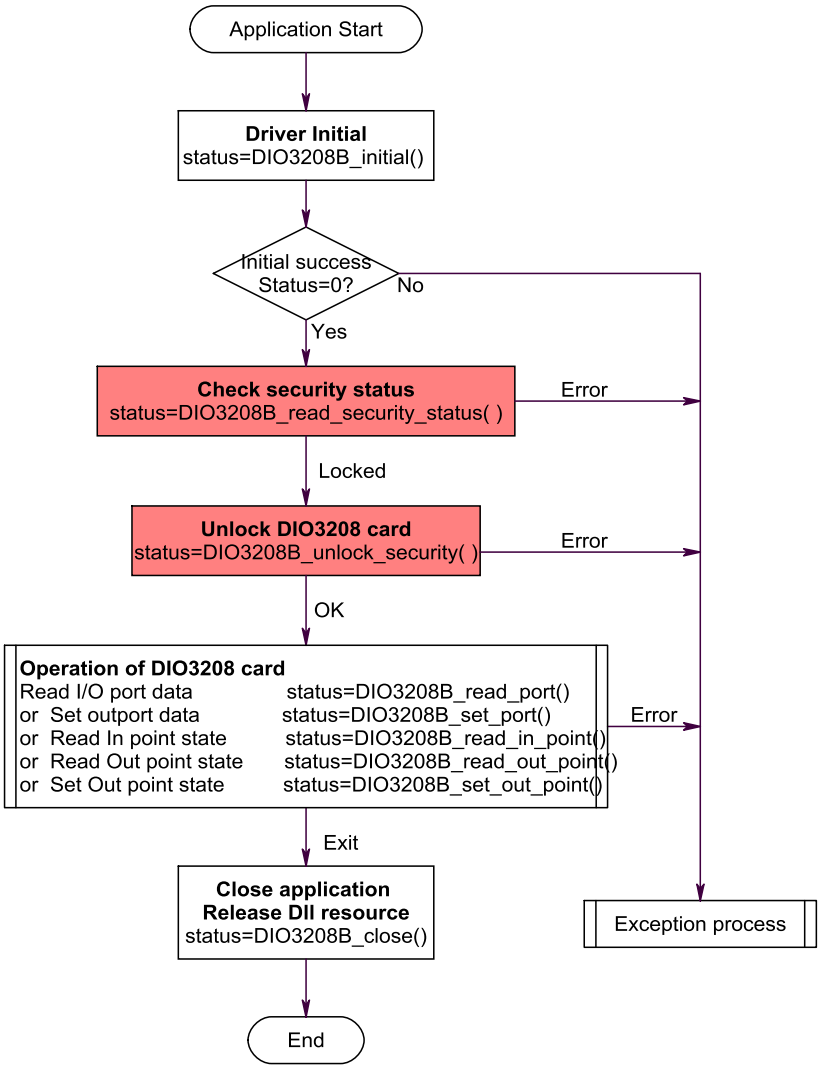
Status = DIO3208B_read_port(u8 CardID, u8 port, u8*data);

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

```
u8 CardID=0, port=0;      //assume CardId=0 and port=0
u8 data;
u32 Status;
Status = DIO3208B_read_port (CardID, port, &data);
```

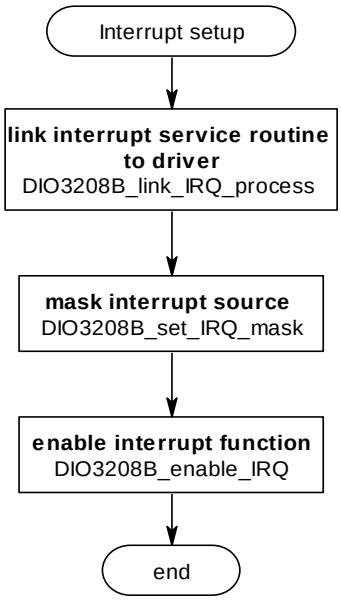
9. Flow chart of application implementation

9.1 DIO3208B Flow chart of application implementation

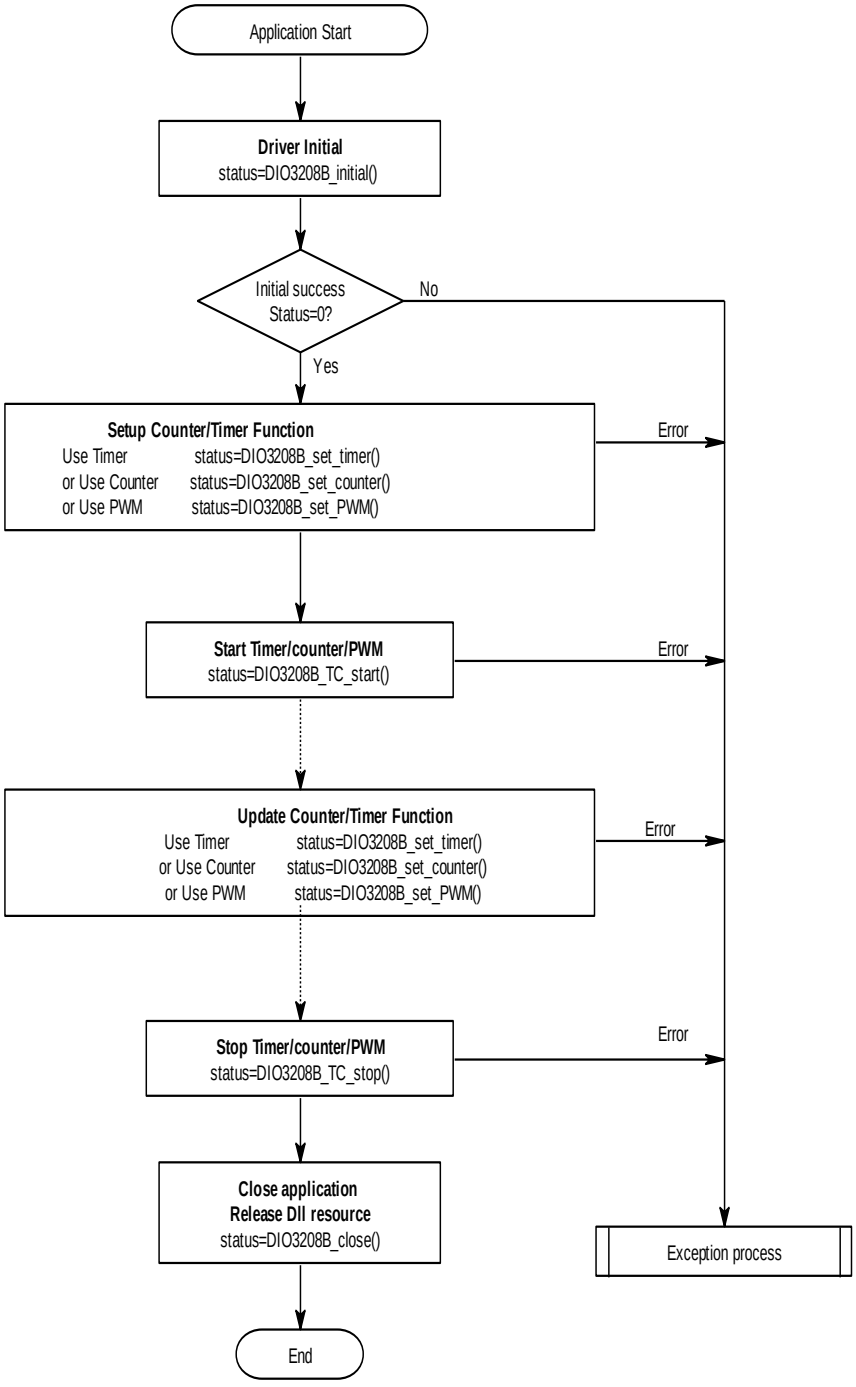


* If you do not use the password function, no need to unlock

9.2 Flow of Interrupt setup



9.3 DIO3208B Flow chart of Timer / Counter / PWM application



10. Software overview and dll function

10.1 Initialization and close

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

DIO3208B_initial( ) will do.

Once you want to close your application, call

DIO3208B_close( ) to release all the resource.

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

DIO3208B_info( ) to get the address .

- **DIO3208B_initial**

Format : **u32 status =DIO3208B_ initial (void)**

Purpose: Initial the DIO3208B resource when start the Windows applications.

- **DIO3208B_close**

Format : **u32 status =DIO3208B_close (void);**

Purpose: Release the DIO3208B resource when close the Windows applications.

- **DIO3208B_info**

Format : **u32 status =DIO3208B_info(u8 CardID, u16 *DIO_address,u16 *TC_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
DIO_address	u16	DIO physical I/O address assigned by OS
TC_address	u16	TC physical I/O address assigned by OS

10.2 I/O Port R/W

Before using a input port, if you already know the maximum response time of the input signal you can setup the debounce time to filter out the undesired noise signal and get a noise-free signal. If you do not know the exact response, please use the conservative setting i.e. 100Hz (sample rate 200Hz) is a common choice.

Use ***DIO3208B_set_debounce_time( )*** to configure the debounce time.

DIO3208B_read_debounce_time( ) to read back the configuration data.

Use the following functions for I/O port output value reading and control:

DIO3208B_read_port( ) to read a byte data from I/O port,

DIO3208B_set_port( ) to output byte data to output port,

DIO3208B_set_out_point( ) to set output bit,

DIO3208B_read_in_point( ) to read I/O bit,

DIO3208B_read_out_point( ) to read back output bit.

- **DIO3208B_set_debounce_time**

Format : **u32 status = DIO3208B_set_debounce_time (u8 CardID , u8 data)**

Purpose: Set the input port debounce time

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by Rotary SW
data	u8	Debounce time selection: 0: no debounce 1: filter out over 100Hz (sampled at 200Hz,default) 2: filter out over 200Hz (sampled at 400Hz) 3: filter out over 1KHz (sampled at 2KHz)

- **DIO3208B_read_debounce_time**

Format : **u32 status = DIO3208B_read_debounce_time (u8 CardID , u8 *data)**

Purpose: Read back the input port debounce time configuration

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by Rotary SW

Output:

Name	Type	Description
data	u8	Debounce time selection: 0: no debounce 1: filter out over 100Hz (sampled at 200Hz,default) 2: filter out over 200Hz (sampled at 400Hz) 3: filter out over 1KHz (sampled at 2KHz)

- **DIO3208B_read_port**

Format : **u32 status = DIO3208B_read_port (u8 CardID , u8 port , u8 *data)**

Purpose: Read the output values of the I/O port.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by Rotary SW
port	u8	port number 0: input port IN00-IN07 1: output port OUT00-OUT07

Output:

Name	Type	Description
data	u8	I/O data depends on port selection b0: IN00 or OUT00 ... b7: IN07 or OUT07

- **DIO3208B_set_port**

Format : **status = DIO3208B_set_port (u8 CardID, u8 data)**

Purpose: Sets the output data.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by Rotary SW
data	u8	I/O data to output b0: OUT00 ... b7: OUT07

- **DIO3208B_set_out_point**

Format : **status =DIO3208B_set_out_point(u8 CardID, u8 point, u8 state)**

Purpose: Sets the bit data of output port.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by Rotary SW
point	u8	point number 0~7 for OUT00~OUT07
state	u8	state of output point

- **DIO3208B_read_in_point**

Format : **u32 Status =DIO3208B_read_in_point(u8 CardID, u8 point, u8 *state)**

Purpose: Read the input state of the input points.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by Rotary SW
point	u8	point number of input 0~7 for IN00~IN07

Output:

Name	Type	Description
state	u8	state of point of input

- **DIO3208B_read_out_point**

Format : **u32 status =DIO3208B_read_out_point(u8 CardID, u8 point, u8 *state)**

Purpose: Read the output state of the output points.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by Rotary SW
point	u8	point number of output 0~7 for OUT00~OUT07

Output:

Name	Type	Description
state	u8	state of output point

10.3 Timer / Counter function

There are 3 major functions of counter/timer function—to work as timer, as counter or as PWM generator.

If your timer/counter function need external input or output, you must switch IN00 (Tin / Gate) and OUT00 (Tout) as dedicated I/O (disable general purpose I/O) while configuring the operation as timer/counter/PWM.

To configure the working mode use

DIO3208B_set_timer( ) to configure as timer and its output mode

DIO3208B_set_counter( ) to configure as counter and its input and output mode

DIO3208B_set_PWM( ) to configure as PWM generator.

To start/stop the operation by:

DIO3208B_start_TC( )

DIO3208B_stop_TC( )

To read or load dedicated timer/counter registers for advanced application, use

DIO3208B_set_TC( ) set TC dedicated registers

DIO3208B_read_TC( ) read TC dedicated registers

Note1: For timer / counter function, Ti/Gate means IN00.

Note2: For timer / counter function, Tout means OUT00 for timer / counter.

- **DIO3208B_set_timer**

Format : u32 status = DIO3208B_set_timer(u8 CardID, Timer_struct * Timer_struct)

Purpose: To setup timer operation mode or update timer

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch

Timer_struct	struct	Timer_struct
		{
		u16 TI_GATE_MODE,
		// **0: NO_GATE**
		//Always count without gate function,
		//IN00 is digital input.
		// **1:GATED_HIGH**
		//IN00 is gate input, after command start_TC,
		//if internal logic active high timer will start
		//counting and logic low will halt counting.
		// **2: GATED_LOW**
		//IN00 is gate input, after command start_TC,
		//if internal logic active low will start timer
		//counting and logic high will halt counting.
		
		u32 time_constant,
		// Timer constant based on 1us clock
		
		u16 TO_MODE,
		// **0: NO_TOUT** ,
		//OUT00 use as general digital output
		// **1: OUT_PULSE**
		//OUT00:timer cross zero output pulse.
		//(out_width effective)
		// **2: OUT_LEVEL**
		//OUT00: timer cross zero output will make.
		// **4: OUT_TOGGLE**
		//OUT00: timer cross zero toggles output
		
		u16 TOUT_WIDTH,
		// Output pulse width based on 1us clock, only
		//valid in Tout_mode is OUT_PULSE
		
		u16 cont_single
		// **0: SINGLE_CYCLE**
		//single cycle mode, timer will stop operation
		//when time constant count down to zero.
		// **1: ALWAYS_RUN**
		//continuous operation mode, timer will reload
		//time constant and continue operation when
		//time constant count down to zero.
		}

- **DIO3208B_set_counter**

Format : u32 status = DIO3208B_set_counter(u8 CardID, Counter_struct *Counter_struct)

Purpose: To setup timer operation mode or update counter

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch

Counter_struct	struct	Counter_struct
		{
		u16 TI_GATE_MODE,
		// **0: invalid**
		// **1:Ti_HIGH**
		//IN00 is counter pulse input, after command
		//start_TC, counter will count on internal logic
		//active high transition
		// **2:Ti_LOW**
		//IN00 is counter pulse input, after command
		//start_TC, counter will count on internal logic
		//active low transition
		
		u32 counter_constant,
		// Counter constant
		
		u16 TO_MODE,
		// **0: NO_TOUT**
		// OUT00 use as general digital output
		// **1: OUT_PULSE**
		//OUT00: timer cross zero output pulse.
		//(out_width effective)
		// **2: OUT_LEVEL**
		//OUT00: timer cross zero output will make.
		// **4:OUT_TOGGLE**
		//OUT00: timer cross zero toggles output
		
		u16 TOUT_WIDTH,
		// Output pulse width based on 1us clock, only
		//valid in Tout_mode is OUT_PULSE
		
		u16 cont_single
		// **0: SINGLE_CYCLE**
		//single cycle mode, counter will stop operation
		//when time constant count down to zero.
		
		// **1: ALWAYS_RUN**
		// continuous operation mode, counter will reload
		//time constant and continue operation when time
		//constant count down to zero.
		}

- **DIO3208B_set_PWM**

Format : u32 status = DIO3208B_set_PWM(u8 CardID, PWM_struct *PWM_struct)

Purpose: To setup PWM operation mode or update PWM.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch

PWM_struct	struct	PWM_struct
		{
		u16 TI_GATE_MODE,
		// **0: NO_GATE**
		//Always count without gate function,
		//IN00 is digital input.
		// **1:GATED_HIGH**
		//IN00 is gate input, after command start_TC,
		//if internal logic active high PWM will start
		//counting and logic low will freeze PWM.
		// **2: GATED_LOW**
		//IN00 is gate input, after command start_TC,
		//if internal logic active low will start PWM
		//counting and logic high will freeze PWM.
		
		u16 PWM_freq;
		// PWM frequency clock count based on 33MHz
		// clock
		u16 PWM_duty;
		//PWM duty clock count based on 33MHz clock
		
		u16 TO_MODE,
		// **0: NO_TOUT**
		// OUT00 use as general digital output
		// **4:OUT_TOGGLE**
		//OUT00: timer cross zero toggles output
		}

Note:

1. PWM base clock is based on33MHz, say if you want your PWM frequency is 20KHz, please put the PWM_freq = (33MHz/20KHz) = 1650
2. PWM duty must less than PWM_freq for proper operation, from the example above, the PWM_duty value can be 1 ~ 1649. For 50% duty, the PWM_duty will be 1650/2 = 825

- **DIO3208B_start_TC**

Format : **u32 status = DIO3208B_start_TC(u8 CardID)**

Purpose: To start timer/counter/PWM operation mode

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch

- **DIO3208B_stop_TC**

Format : **u32 status = DIO3208B_stop_TC(u8 CardID)**

Purpose: To stop timer/counter/PWM operation mode

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch

- **DIO3208B_set_TC**

Format : **u32 status=DIO3208B_set_TC(u8 CardID,u8 index,u32 data)**

Purpose: To set data to counter/timer register

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
index	u8	0: TC_CONTROL 1: TC_MODE 2: TI_GATE_MODE 3: TO_MODE 4: RETRIGGER_MODE 5: PRELOAD 6: COUNTER 7:OUT_WIDTH
data	u32	register data to be set

Note: please refer the next segment "Note:Meaning of setting or return value of different index"

- **DIO3208B_read_TC**

Format : u32 status=DIO3208B_read_TC(u8 CardID,u8 index,u32 *data)

Purpose: To read data from counter/timer

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
index	u8	0: TC_CONTROL 1: TC_MODE 2: TI_GATE_MODE 3: TO_MODE 4: RETRIGGER_MODE 5: PRELOAD 6: COUNTER 7:OUT_WIDTH

Output:

Name	Type	Description
data	u32	Data read back

Note: Meaning of setting or return value of different index

index	register	value	meaning
0	TC_CONTROL	0	STOP, stop operation of TC
		1	START, start operation of TC
1	TC_MODE	0	TIMER_MODE
		1	COUNTER_MODE
		3	SW_DEC (a write will software decrease counter by 1 and return to
		4	PWM_MODE
2	TI_GATE_MODE	0	NO_GATE
		1	GATED_HIGH
		2	GATED_LOW
3	TO_MODE	0	NO_TOUT
		1	OUT_PULSE
		2	OUT_LEVEL
		4	OUT_TOGGLE
4	RETRIGGER_MODE	0	SINGLE_CYCLE
		1	ALWAYS_RUN
5	PRELOAD	1~0xffffffff	Counter or timer or PWM preload value
6	COUNTER	1~0xffffffff	Set (write): will write preload and counter Read : will read counter on the fly
7	OUT_WIDTH	1~0xffff	Output pulse width

Note: For example, you want to watch the counting on the fly, use

DIO3208B_read_TC(u8 CardID,u8 index,u32 *data) //CardID as you assign, index=6

To read back the counter value.

10.4 Interrupt function

Sometimes you want your application to take care of the I/O while special event occurs, interrupt function is the right choice.

First, tell the driver your interrupt service routine by

DIO3208B_link_IRQ_process( )

Next, you should enable / disable the hardware of the interrupt source by,

DIO3208B_set_IRQ_mask( )

DIO3208B_read_IRQ_mask( ) to read back the IRQ mask status.

To enable the IRQ function,

DIO3208B_enable_IRQ( ) to start waiting the interrupt.

If you do not use interrupt any more and you will close your application program, be sure to use

DIO3208B_disable_IRQ( ) to release the resource.

In interrupt service routine, if you want to know the interrupt status, use

DIO3208B_read_IRQ_status( ) to identify the source of interrupt.

- **DIO3208B_link_IRQ_process**

Format : u32 status = DIO3208B_link_IRQ_process (u8 CardID,

void (__stdcall *callbackAddr)(u8 CardID));

Purpose: Link irq service routine to driver

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by Rotary SW
callbackAddr	void	callback address of service routine

- **DIO3208B_set_IRQ_mask**

Format : u32 status = DIO3208B_set_IRQ_mask (u8 CardID, u8 source, u8 mask)

Purpose: Select interrupt source

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by Rotary SW
source	u8	0:DIO 1:TC
mask	u8	For DIO b7: 0, disable IN07 as interrupt source 1, enable IN07 as interrupt source b6: 0, disable IN06 as interrupt source 1, enable IN06 as interrupt source b0: 0, disable IN00 as interrupt source 1, enable IN00 as interrupt source for TC b0: 0, disable TC counter counts to zero as interrupt source 1, enable TC counter counts to zero as interrupt source

- **DIO3208B_read_IRQ_mask**

Format : **u32 status = DIO3208B_read_IRQ_mask (u8 CardID, u8 source, u8 *mask)**

Purpose: Select interrupt source

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by Rotary SW
source	u8	0:DIO 1:TC

Output:

Name	Type	Description
mask	u8	Return the setting value of mask register

- **DIO3208B_enable_IRQ**

Format : **u32 status = DIO3208B_enable_IRQ (u8 CardID, HANDLE *phEvent)**

Purpose: Enable interrupt from unmasked source

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by Rotary SW

Output:

Name	Type	Description
phEvent	HANDLE	event handle

- **DIO3208B_disable_IRQ**

Format : **u32 status = DIO3208B_disable_IRQ (u8 CardID)**

Purpose: Disable interrupt from unmasked source

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by Rotary SW

- **DIO3208B_read_IRQ_status**

Format : u32 status = DIO3208B_read_IRQ_status (u8 CardID, u8 source, u8 *Event_Status)

Purpose: To read back the interrupt status and clears the on board status register

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by Rotary SW
source	u8	0:DIO 1:TC

Output:

Name	Type	Description
Event_Status	u8	For DIO b7: 1, IN07 interrupted b6: 1, IN06 interrupted … b0: 1, IN00 interrupted for TC b0: 1, TC counter counts to zero interrupted

Note:

1.DIO3208B_IRQ_status( ) function can be used in the user's interrupt service routine to identify the irq source if multiple source is allowed.

2.If you do not use interrupt function, you can still use DIO3208B_IRQ_status( ) to catch the fast changing input status.

3.Before return from DIO3208B_IRQ_status( ), the status hardware will be cleared.

10.5 Software key function

From the dll version 4.0 and later, we remove the software key function owing to some customers complained about the card locked on some unknown occasion. We only remain the functions to comply with the existing programs but the returned value always true.

Since DIO3208B 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

DIO3208B_set_password( ) to set password and start the security function.

DIO3208B_change_password( ) to change it.

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

DIO3208B_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

DIO3208B_unlock_security( ) unlock the security.

You can also use

DIO3208B_read_security_status( ) to check the current status of security.

- **DIO3208B_set_password**

Format : u32 status = DIO3208B_set_password(u8 CardID,u16 password[5]);

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

Parameters:

Input:

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

Note on password:

If the password is all "0", the security function is disabled.

- **DIO3208B_change_password**

Format : u32 status = DIO3208B_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

- **DIO3208B_clear_password**

Format : u32 status = DIO3208B_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

- **DIO3208B_unlock_security**

Format : u32 status = DIO3208B_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

- **DIO3208B_read_security_status**

Format : u32 status = DIO3208B_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

Note on security status:

The security should be unlocked before using any other function of the card, and any attempt to unlock with the wrong passwords more than 10 times will cause the card at dead lock status. Any further operation even with the correct password will not unlock the card. The only way is to send back to the card distributor or the original maker to unlock to virgin state.

11. DLL list

	Function Name	Description
1	DIO3208B_initial( )	DIO3208B Initial
2	DIO3208B_close( )	DIO3208B Close
3	DIO3208B_info( )	get OS. assigned address
4	DIO3208B_set_debounce_time( )	Set input port digital debounce time
5	DIO3208B_read_debounce_time( )	Read back input port digital debounce time
6	DIO3208B_read_port( )	Read Port Data
7	DIO3208B_set_port( )	Set Output port
8	DIO3208B_set_out_point( )	Set Output Point State(bit)
9	DIO3208B_read_in_point( )	Read Input Point State(bit)
10	DIO3208B_read_out_point( )	Read Output Point State(bit)
11	DIO3208B_set_timer( )	Setup or update timer
12	DIO3208B_set_counter( )	Setup or update counter
13	DIO3208B_set_PWM( )	Setup or update PWM
14	DIO3208B_start_TC( )	Start timer/counter/PWM operation
15	DIO3208B_stop_TC( )	Stop timer/counter/PWM operation
16	DIO3208B_set_TC( )	Set TC register
17	DIO3208B_read_TC( )	Read TC register
18	DIO3208B_link_IRQ_process( )	Link interrupt service routine to driver
19	DIO3208B_set_IRQ_mask( )	Set interrupt source mask
20	DIO3208B_read_IRQ_mask( )	Read interrupt source mask
21	DIO3208B_enable_IRQ( )	Enable interrupt function
22	DIO3208B_disable_IRQ( )	Disable interrupt function
23	DIO3208B_IRQ_status( )	Read back irq status
24	DIO3208B_set_password( )	Set software key
25	DIO3208B_change_password( )	Change software key
26	DIO3208B_clear_password( )	Clear software key
27	DIO3208B_unlock_security( )	Unlock software key
28	DIO3208B_read_security_status( )	Read software key status

12. DIO3208B Error codes summary

Error Code	Symbolic Name	Description
0	JSDRV_NO_ERROR	No error.
1	JSDRV_READ_DATA_ERROR	Read data error
2	JSDRV_INIT_ERROR	Driver initial error
3	JSDRV_UNLOCK_ERROR	Software key unlock error
4	JSDRV_LOCK_COUNTER_ERROR	Software key unlock error count over
5	JSDRV_SET_SECURITY_ERROR	Software key setting error
100	DEVICE_RW_ERROR	Device Read/Write error
101	JSDRV_NO_CARD	No DIO3208B card on the system.
102	JSDRV_DUPLICATE_ID	DIO3208B CardID duplicate error.
300	JSDIO_ID_ERROR	Function input parameter error. CardID setting error, CardID doesn't match the DIP SW setting
301	JSDIO_PORT_ERROR	Function input parameter error. Parameter out of range.
302	JSDIO_IN_POINT_ERROR	Function input parameter error. Parameter out of range.
303	JSDIO_OUT_POINT_ERROR	Function input parameter error. Parameter out of range.
304	JSDIO_VERSION_ERROR	Hardware version cannot match with software version
305	JSDIO_SOURCE_ERROR	TC source select error
306	JSDIO_DEBOUNCE_MODE_ERROR	Digital input debounce mode error
406	JSDIO_INDEX_ERROR	TC register index error
407	JSDIO_TO_mode_error	Timer output mode error
408	JSDIO_TI_mode_error	Timer input mode error
500	JSDIO_CARDTYPE_ERROR	Card type error