DIO3232/3232A

Digital I/O Card

Software Manual (V2.2)

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
1.1->1.2	For driver version 2.0 up
	1 Add software key function
	DIO3232_set_password( )
	DIO3232_change_password( )
	DIO3232_clear_password( )
	DIO3232_unlock_security( )
	DIO3232_read_security_status( )
1.2->1.3	For driver version 4.0 up
	1 Revise how to install
1.3->1.4	Modify all function name (lowercase → uppercase)
2.0	1 revised to new manual style
	2 disable the software key function with return value always true
2.1	add Chapt.1 Software compatibility of DIO3232A
2.2	add cross reference table of I/O port naming

Contents

1. Software compatibility of DIO3232A

JS Automation has revised the DIO3232 with new chip and design and naming the new version as DIO3232A. We try to fully compatible with the old version DIO3232 on software, if you do not want to recompile or modify the existing software.

Owing to DIO3232A is hardware down compatible with DIO3232 and up compatible with DIO3232B; for the new design, we suggest installing the DIO3232B driver which has full basic functions for DIO3232A and DIO3232B and new advanced functions for DIO3232B. With new naming convention, you can update DIO3232A to DIO3232B easily if you need more function

2. How to install the software of DIO3232

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 WinXP/7 and up system you should:

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 (..\DIO3232\Software\WinXP_7\ or if you download from website please execute the file DIO3232_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 \DIO3232\API (header files and lib files for VB,VC,BCB,C#)

.. \ JS Automation \DIO3232\Driver (backup copy of DIO3232 drivers)

.. \ JS Automation \DIO3232\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 DIO3232 software

DIO3232 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 DIO3232 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 DIO3232 functions within Windows' operation system environment.

4.1 What you need to get started

To set up and use your DIO3232 software, you need the following:

- DIO3232 software
- DIO3232 hardware

  Main board

  Wiring board (Option)

4.2 Software programming choices

You have several options to choose from when you are programming DIO3232 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 DIO3232 software.

5. DIO3232 Language support

The DIO3232 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 DIO3232 software library

The DIO3232 function reference topic contains general information about building DIO3232 applications, describes the nature of the DIO3232 files used in building DIO3232 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

5.2 DIO3232 Windows libraries

The DIO3232 for Windows function library is a DLL called **DIO3232.dll**. Since a DLL is used, DIO3232 functions are not linked into the executable files of applications. Only the information about the DIO3232 functions in the DIO3232 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 DIO3232 functions in DIO3232.dll.

Header Files and Import Libraries for Different Development Environments		
Language	**Header File**	**Import Library**
Microsoft Visual C/C++	DIO3232.h	DIO3232VC.lib
Borland C/C++	DIO3232.h	DIO3232BC.lib
Microsoft Visual C#	DIO3232.cs	
Microsoft Visual Basic	DIO3232.bas	
Microsoft VB.net	DIO3232.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 classified 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 break the electrical connection without breaking 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 classified 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.

*DIO3232 has built-in hardware debounce circuit which limits the input frequency up to 2.2KHz. With this function, the DIO3232 differs from some other company's product; they do not have built-in debounce circuit, sometimes you will get double or triple signal input while fast scanning.

***DIO3232A use build-in digital debounce fixed at 2KHz**

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. DIO3232 provides IN0 and IN1 as interrupt input.

Read back of Output status

Some applications need to read back the output status, if the card does 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. Function format and language difference

7.1 Function format

Every DIO3232 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 DIO3232 function is the parameter **CardID** which is located the driver of DIO3232 board you want to use those given operation. The **CardID** is assigned by DIP 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 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 DIO3232 API. Read the following sections that apply to your programming language.

Note: Be sure to include the declaration functions of DIO3232 prototypes by including the appropriate DIO3232 header file in your source code. Refer to Building Applications with the DIO3232 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 = DIO3232_read_port(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 = read_port (CardID, port, &data);
```

7.3.2 Visual basic

The file DIO3232.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 DIO3232.bas, do not use the numerical values.

In Visual Basic, you can add the entire DIO3232.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 DIO3232.bas file for your project in Visual Basic 4.0, go to the **File** menu and select the **Add File... option**. Select DIO3232.bas, which is browsed in the DIO3232 \ API directory. Then, select **Open** to add the file to the project.

To add the DIO3232.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** DIO3232.bas, which is in the DIO3232 \ 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 DIO3232bc.lib DIO3232.dll

Then add the **DIO3232bc.lib** to your project and add

#include "DIO3232.h" to main program.

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

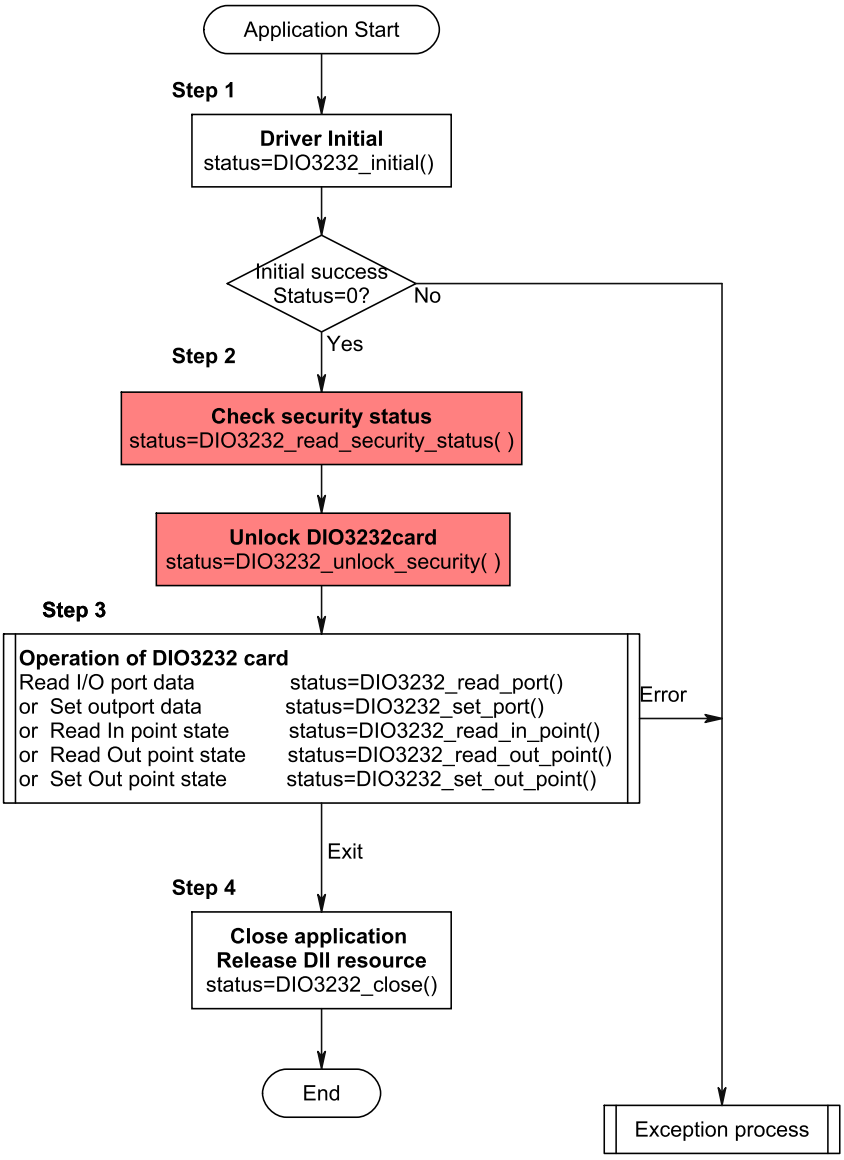
Status = DIO3232_read_port(CardID, port, data);

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

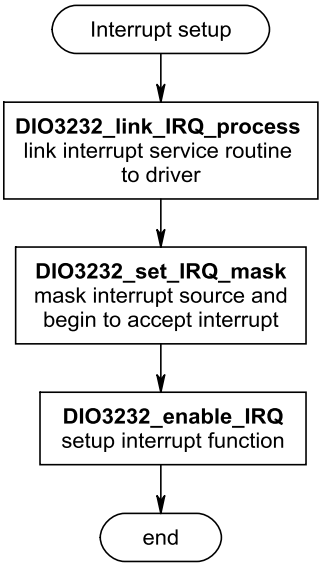
u16 CardID, port;
u8 data;
u32 Status;
Status = read_port (CardID, port, &data);

8. Flow chart of application implementation

8.1 DIO3232 Flow chart of application implementation



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


Interrupt setup
DIO3232_link_IRQ_process
link interrupt service routine
to driver
DIO3232_set_IRQ_mask
mask interrupt source and
begin to accept interrupt
DIO3232_enable_IRQ
setup interrupt function
end

9. Software overview and dll function

9.1 Initialization

You need to initialize each time you run your application.

DIO3232_initial( ) to initial the resources of the driver.

DIO3232_close( ) to close the resources of the driver.

DIO3232_info( ) get the information of address assigned by the OS.

DIO3232_get_device_handle( ) get the driver handle.

- **DIO3232_initial**

  Format : **u32 status =DIO3232_initial (void)**

  Purpose: Initial the DIO3232 resource when start the Windows applications.

- **DIO3232_close**

  Format : **u32 status =DIO3232_close (void);**

  Purpose: Release the DIO3232 resource when close the Windows applications.

- **DIO3232_info**

  Format : **u32 status =DIO3232_info(u8 CardID,u16 *Ven_ID,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(**0x0-0xF**)

  Output:

Name	Type	Description
Ven_ID	u16	return sub system ID(0x3232)
address	u16	physical I/O address assigned by OS

- **DIO3232_get_device_handle**

  Format : **u32 status =DIO3232_get_device_handle(u8 CardID,HANDLE *DeviceHandle)**

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

  Parameters:

  Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY witch(**0x0-0xF**)

  Output:

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

9.2 I/O Port R/W

There are 8 ports on DIO3232 card, the 0~3 port designed as input and 4~7 as output. The input port 0~3 mapped to IN00~IN37 and output ports 4~7 mapped to OUT00~OUT37 with the first digit as the designation of number of port, the second is the bit number. You can only output data to output ports but as for input reading, the input ports and output ports (output register read back) are available .

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

Set I/O port output

DIO3232_set_port( )

Set I/O point output

DIO3232_set_out_point( )

Read I/O port

DIO3232_read_port( )

Read I/O point

DIO3232_read_in_point( )

DIO3232_read_out_point( )

Note: The new wiring board has different point designation, the difference list as follows.

Wiring board I/O point designation cross reference

port	wiring board (ADP3232DIN)	wiring board (ADP3232ADIN)
0	0 ~ 7	00 ~ 07
1	8 ~ 15	10 ~ 17
2	16 ~ 23	20 ~ 27
3	23~ 31	30 ~ 37

- **DIO3232_set_port**

Format : **u32 status = DIO3232_set_port (u8 CardID, u8 port , u8 data)**

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch(**0x0-0xF**)
port	u8	port number 4~7 (output port)
data	u8	bitmap of output values port5 as example: b0: OUT8 b1: OUT 9 ... b7: OUT 15 For ADP3232ADIN wiring board b0: OUT10 b1: OUT11 ... b7: OUT17

- **DIO3232_set_out_point**

Format : **u32 status =DIO3232_set_out_point(u8 CardID, u8 point, u8 state)**

Purpose: Sets the output state of the I/O point.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch(**0x0-0xF**)
point	u8	output point number (0~31) refer the Wiring board I/O point designation cross reference
state	u8	point of output state 0: inactive 1: active

- **DIO3232_read_port**

Format : **u32 status = DIO3232_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 DIP/ROTARY switch(**0x0-0xF**)
port	u8	port number (0~7)

Output:

Name	Type	Description
data	u8	bitmap of port values port7 as example: b0: OUT24 (OUT b1: OUT25 ... b7: OUT31 For ADP3232A (v1.3) wiring board b0: OUT30 b1: OUT31 ... b7: OUT37

- **DIO3232_read_in_point**

Format : **u32 status =DIO3232_read_in_point(u8 CardID, u8 point, u8 *state)**

Purpose: Read the input state of the I/O points.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch(**0x0-0xF**)
point	u8	point number (0~31) refer the Wiring board I/O point designation cross reference

Output:

Name	Type	Description
state	u8	point of output state 0: inactive 1: active

- **DIO3232_read_out_point**

Format : **u32 status =DIO3232_read_out_point(u8 CardID, u8 point, u8 *state)**

Purpose: Read the output state of the I/O points.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch(**0x0-0xF**)
point	u8	point number (0~31) refer the Wiring board I/O point designation cross reference

Output:

Name	Type	Description
state	u8	point of output state 0: inactive 1: active

9.3 Interrupt function

The DIO3232 card provides 2 input points as interrupt source: IN00 and IN01. To use the external interrupt function you must link the service routine (a function that will be called back during interrupt occurs) to the interrupt handle by

DIO3232_link_IRQ_process( )

Next, you should setup the IRQ mask for the interrupt by:

DIO3232_set_IRQ_mask( )

Last, enable it by:

DIO3232_enable_IRQ( ) and also you can disable IRQ by:

DIO3232_disable_IRQ( )

On the service routine, you can check the interrupt source (if multiple interrupt source) by:

DIO3232_IRQ_status( )

- **DIO3232_link_IRQ_process**

Format : **u32 status = DIO3232_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 DIP/ROTARY switch(**0x0-0xF**)
callbackAddr	void	callback address of service routine

- **DIO3232_set_IRQ_mask**

Format : **u32 status = DIO3232_set_IRQ_mask (u8 CardID, u16 Data)**

Purpose: Mask interrupt from IN0, IN1

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch(**0x0-0xF**)
Data	u16	bit0: 0, disable irq from IN00 1, enable irq from IN00 bit1: 0, disable irq from IN01 1, enable irq from IN01

- **DIO3232_enable_IRQ**

Format : **u32 status = DIO3232_enable_IRQ (u8 CardID, HANDLE *phEvent)**

Purpose: Enable interrupt from IN0, IN1

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch(**0x0-0xF**)

Output:

Name	Type	Description
phEvent	HANDLE	event handle

- **DIO3232_disable_IRQ**

Format : **u32 status = DIO3232_disable_IRQ (u8 CardID)**

Purpose: Disable interrupt from IN0, IN1

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch(**0x0-0xF**)

- **DIO3232_IRQ_status**

Format : **u32 status = DIO3232_IRQ_status (u8 CardID, u32 *Event_Status)**

Purpose: To read back the interrupt source to identify IN0 or IN1

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch(**0x0-0xF**)

Output:

Name	Type	Description
Event_Status	u32	bit0: 1, irq source from IN00 bit1: 1, irq source from IN01

9.4 Software key function

From the dll version 5.0 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 DIO3232 is a general purpose card, anyone who can buy from the market. Your program is the fruit of your intelligence, un-authorized copy maybe prevent by the security function enabled.

You can use

DIO3232_set_password( ) to set password and start the security function. Use

DIO3232_change_password( ) to change it.

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

DIO3232_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

DIO3232_unlock_security( ) unlock the security.

You can also use

DIO3232_read_security_status( ) to check the current status of security.

Note:

Any attempt to unlock the software security function with wrong passwords more than 10 times will "dead lock" the card. We also suggest locking the card under the demo program (comes with the card) and unlocking when your application program starts. Lock and unlock cycles is limited by the semiconductor's life read/write cycles.

- **DIO3232_set_password**

Format : **u32 status = DIO3232_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 switch(**0x0-0xF**)
password[5]	u16	Password, 5 words

Note on password:

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

- **DIO3232_change_password**

Format : **u32 status = DIO3232_change_password(u8 CardID,u16 Oldpassword[5], u16 password[5]);**

Purpose: To replace old password with new password.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch(**0x0-0xF**)
Oldpassword [5]	u16	The previous password
password[5]	u16	The new password to be set

- **DIO3232_clear_password**

Format : **u32 status = DIO3232_clear_password(u8 CardID,u16 password[5])**

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch(**0x0-0xF**)
password[5]	u16	The password previous set

- **DIO3232_unlock_security**

Format : **u32 status = DIO3232_unlock_security(u8 CardID,u16 password[5])**

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch(**0x0-0xF**)
password[5]	u16	The password previous set

- **DIO3232_read_security_status**

Format : **u32 status = DIO3232_read_security_status(u8 CardID,u8 *lock_status, u8 *security_enable);**

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch(**0x0-0xF**)

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.

10. Dll list

	Function Name	Description
1	DIO3232_initial( )	DIO3232 Initial
2	DIO3232_close( )	DIO3232 Close
3	DIO3232_info( )	get OS. assigned address
4	DIO3232_get_device_handle( )	Read device handle
5	DIO3232_set_port( )	Set Output port(word)
6	DIO3232_set_out_point( )	Set Output Point State(bit)
7	DIO3232_read_port( )	Read Port Data (word)
8	DIO3232_read_in_point( )	Read Input Point State(bit)
9	DIO3232_read_out_point( )	Read Output Point State(bit)
10	DIO3232_link_IRQ_process( )	Link interrupt service routine to driver
11	DIO3232_set_IRQ_mask( )	Set interrupt mask
12	DIO3232_enable_IRQ( )	Enable interrupt function
13	DIO3232_disable_IRQ( )	Disable interrupt function
14	DIO3232_IRQ_status( )	Read back irq status
15	DIO3232_set_password( )	Set software key
16	DIO3232_change_password( )	Change software key
17	DIO3232_clear_password( )	Clear software key
18	DIO3232_unlock_security( )	Unlock software key
19	DIO3232_read_security_status( )	Read software key status

11. Port-point reference table

11.1 DIO3232 I/O Port-Point table

DIO3232 I/O Port table (for ADP3232DIN wiring board)								
Bit / Port	***Bit 7***	***Bit 6***	***Bit 5***	***Bit 4***	***Bit 3***	***Bit 2***	***Bit 1***	***Bit 0***
Port 0	IN 7	IN 6	IN 5	IN 4	IN 3	IN 2	IN 1	IN 0
Port 1	IN 15	IN 14	IN 13	IN 12	IN 11	IN 10	IN 9	IN 8
Port 2	IN 23	IN 22	IN 21	IN 20	IN 19	IN 18	IN 17	IN 16
Port 3	IN 31	IN 30	IN 29	IN 28	IN 27	IN 26	IN 25	IN 24
Port 4	OUT 7	OUT 6	OUT 5	OUT 4	OUT 3	OUT 2	OUT 1	OUT 0
Port 5	OUT 15	OUT 14	OUT 13	OUT 12	OUT 11	OUT 10	OUT 9	OUT 8
Port 6	OUT 23	OUT 22	OUT 21	OUT 20	OUT 19	OUT 18	OUT 17	OUT 16
Port 7	OUT 31	OUT 30	OUT 29	OUT 28	OUT 27	OUT 26	OUT 25	OUT 24

DIO3232 I/O Port table(for ADP3232ADIN wiring board)								
Bit / Port	***Bit 7***	***Bit 6***	***Bit 5***	***Bit 4***	***Bit 3***	***Bit 2***	***Bit 1***	***Bit 0***
Port 0	IN07	IN06	IN05	IN04	IN03	IN02	IN01	IN00
Port 1	IN17	IN16	IN15	IN14	IN13	IN12	IN11	IN10
Port 2	IN27	IN26	IN25	IN24	IN23	IN22	IN21	IN20
Port 3	IN37	IN36	IN35	IN34	IN33	IN32	IN31	IN30
Port 4	OUT07	OUT06	OUT05	OUT04	OUT03	OUT02	OUT01	OUT00
Port 5	OUT17	OUT16	OUT15	OUT14	OUT13	OUT12	OUT11	OUT10
Port 6	OUT27	OUT26	OUT25	OUT24	OUT23	OUT22	OUT21	OUT20
Port 7	OUT37	OUT36	OUT35	OUT34	OUT33	OUT32	OUT31	OUT30

12. DIO3232 Error codes summary

12.1 DIO3232 Error codes table

Error Code	Symbolic Name	Description
0	JSDRV_NO_ERROR	No error.
2	JSDRV_INIT_ERROR	Driver initial error
3	JSDRV_UNLOCK_ERROR	Security unclock failure
4	JSDRV_LOCK_COUNTER_ERROR	Dead lock, unclock 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 DIO3232 card on the system.
102	JSDRV_DUPLICATE_ID	DIO3232 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. (In port ≠ 0~3 ; Out port ≠ 4~7)
302	JSDIO_IN_POINT_ERROR	Function input parameter error. Parameter out of range. (point > 31)
303	JSDIO_OUT_POINT_ERROR	Function input parameter error. Parameter out of range. (point > 31)