

# **DIO-3216B**

## **Digital I/O Card**

### **Software Manual (V2.0)**

**健昇科技股份有限公司**

**JS AUTOMATION CORP.**

台北縣汐止市中興路 100 號 6 樓

6F, No. 100, Chungshin Rd.

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

TEL : 886-2-2647-6936

FAX : 886-2-2647-6940

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

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

## Correction record

Manual Version	Record
1.0	wdm3216B.sys V1.2
	wdm3216B.dll V1.2
2.0	wdm3216B.sys V2.0
	wdm3216B.dll V2.0
	DIO3216B.dll V2.0

**Note:** V2.0 only for DIO3216B

# Contents

1.	Special Notes on using DIO3216B as replacement of DIO3216A.....	4
2.	How to install the software of DIO3216B .....	5
2.1	Install the PCI driver.....	5
3.	Where to find the file you need .....	6
4.	About the DIO3216B software .....	7
4.1	What you need to get started.....	7
4.2	Software programming choices .....	7
5.	DIO3216B Language support.....	8
5.1	Building applications with the DIO3216B software library .....	8
5.2	DIO3216B Windows libraries .....	8
6.	Basic concepts of digital I/O control .....	9
7.	Software overview .....	12
7.1	Initialization and close.....	12
7.2	I/O Port R/W.....	12
7.3	Interrupt function.....	13
7.4	Software key function.....	14
7.5	Error conditions .....	14
8.	Flow chart of implement an application .....	15
8.1	DIO3216B Flow chart of implementation.....	15
9.	Function reference .....	16
9.1	Function format.....	16
9.2	Variable data types.....	17
9.3	Programming language considerations.....	18
9.4	DIO3216B Functions.....	20
Initialization and close .....	20	
DIO3216B_initial .....	20	
DIO3216B_close .....	20	
DIO3216B_info .....	20	
I/O Port R/W.....	21	
DIO3216B_read_port .....	21	
DIO3216B_set_port.....	21	
DIO3216B_set_out_point.....	21	
DIO3216B_read_in_point .....	22	
DIO3216B_read_out_point .....	22	
DIO3216B_set_debounce_time.....	23	
DIO3216B_read_debounce_time .....	23	
Interrupt function .....	24	
DIO3216B_enable_IRQ .....	24	
DIO3216B_disable_IRQ .....	24	

DIO3216B_link_IRQ_process.....	24
DIO3216B_set_IRQ_mask.....	25
DIO3216B_read_IRQ_mask .....	25
DIO3216B_read_IRQ_status.....	26
Software key function.....	27
DIO3216B_set_password .....	27
DIO3216B_change_password .....	27
DIO3216B_clear_password.....	27
DIO3216B_unlock_security .....	28
DIO3216B_read_security_status .....	28
9.5 Dll list .....	29
10. DIO-3216B Error codes table .....	30

## **1. Special Notes on using DIO3216B as replacement of DIO3216A**

DIO3216B is not fully software compatible with the previous model DIO3216A, but the major digital I/O functions is the same. If you use DIO3216A in your application before and now for the repair purpose, you use DIO3216B, you must confirm and take the following procedures to ensure the success of replacement.

1. Make sure that your old applications only use the digital I/O functions, if not, you can not replace the old card with DIO3216B
2. You must install the DIO3216B driver to replace the old driver. The new driver has build in digital I/O functions under the same syntax of old card. Please refer the DIO3216B.h file for more detail information. Under this condition, you do not need to re-compile your application program.
3. If you can re-compile your application program, we suggest you to use new function syntax with the DIO3216B card, refer the functions described at this manual.

## **2. How to install the software of DIO3216B**

### 2.1 Install the PCI driver

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

In Windows 2000/XP/Vista 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  
    \DIO3216B\Software\Win2K\_up\DIO3216B\_Install.exe
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**

#### **Windows 2000,XP and up**

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

`\DIO3216B\Software\Win2K_up\Install\DIO3216B_Install.exe`

The directory will be located at

**.. \JS Automation\DIO3216B\API** (header files and VB,VC lib files)

**.. \JS Automation\DIO3216B\Driver** (backup copy of DIO3216B drivers)

**.. \JS Automation\ DIO3216B\exe** (demo program and source code)

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

## 4. **About the DIO3216B software**

DIO3216B 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 DIO3216B 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 DIO3216B functions within Windows' operation system environment.

### 4.1 What you need to get started

To set up and use your DIO3216B software, you need the following:

- DIO3216B software
- DIO3216B hardware
  - Main board
  - Wiring board (Option)

### 4.2 Software programming choices

You have several options to choose from when you are programming DIO3216B 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 DIO3216B software.

## 5. **DIO3216B Language support**

The DIO3216B software library is a DLL used with Windows 2000/XP/Vista. You can use these DLL with any Windows integrating development environment that can call Windows DLLs.

### 5.1 Building applications with the DIO3216B software library

The DIO3216B function reference topic contains general information about building DIO3216B applications, describes the nature of the DIO3216B files used in building DIO3216B applications, and explains the basics of making applications using the following tools:

#### **Applications tools**

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

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

### 5.2 DIO3216B Windows libraries

The DIO3216B for Windows function library is a DLL called **dio3216b.dll**. Since a DLL is used, DIO3216B functions are not linked into the executable files of applications. Only the information about the DIO3216B functions in the DIO3216B 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 DIO3216B functions in dio3216b.dll.

Header Files and Import Libraries for Different Development Environments		
<b>Development Environment</b>	<b>Header File</b>	<b>Import Library</b>
<b>Microsoft C/C++</b>	DIO3216B.h	DIO3216BVC.lib
<b>Borland C/C++</b>	DIO3216B.h	DIO3216BBC.lib
<b>Microsoft Visual Basic</b>	DIO3216B.bas	

**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.**

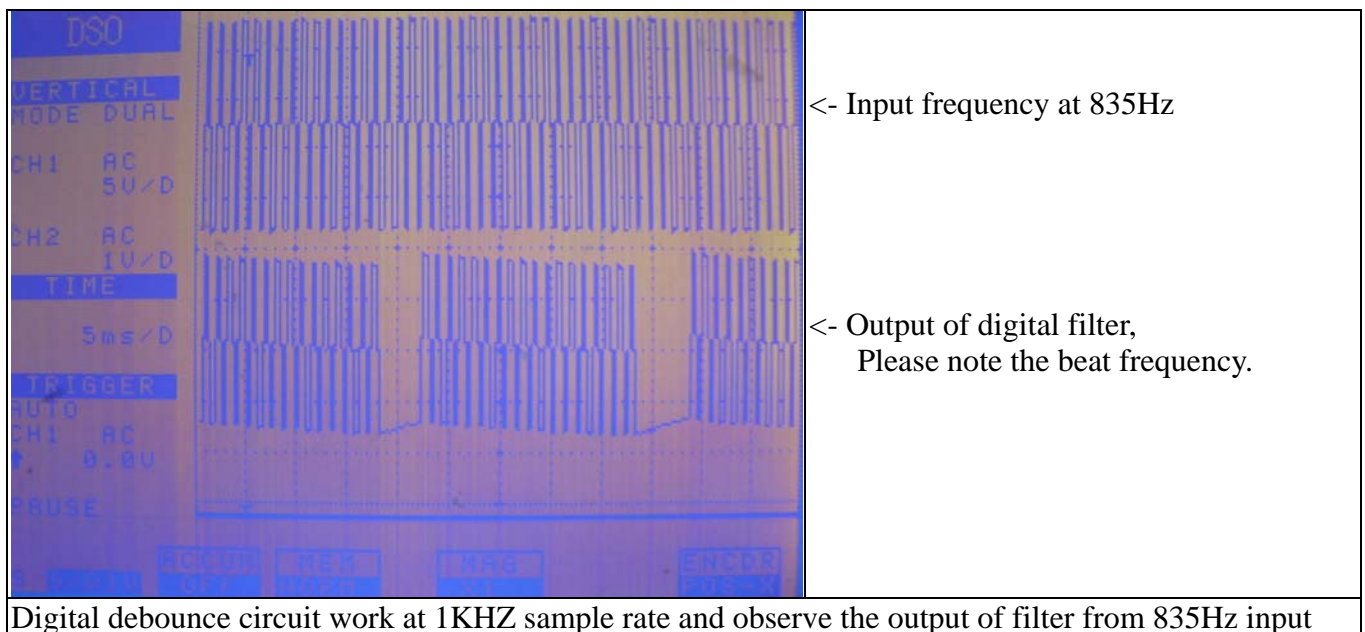
SCR (or triac) is seldom direct connect to digital output, but his relative SSR is the most often selection. In fact, SSR is a compact package of trigger circuit and triac. You can choose zero cross trigger (output command only turn on the output at power phase near zero to eliminate surge) or direct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



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. Software overview

These topics describe the features and functionality of the DIO3216B boards and briefly describes its functions.

### 7.1 Initialization and close

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

[\*DIO3216B initial\(\)\*](#) will do.

Once you want to close your application, call

[\*DIO3216B close\(\)\*](#) to release all the resource.

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

[\*DIO3216B info\(\)\*](#) to get the address .

### 7.2 I/O Port R/W

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

[\*DIO3216B read\\_port\(\)\*](#) to read a byte data from I/O port,

[\*DIO3216B set\\_port\(\)\*](#) to output byte data to output port,

[\*DIO3216B set\\_out\\_point\(\)\*](#) to set output bit,

[\*DIO3216B read\\_in\\_point\(\)\*](#) to read I/O bit,

[\*DIO3216B read\\_out\\_point\(\)\*](#) to read back output bit.

Mechanical contact or noisy environment always induced unstable state at digital input, the DIO3216B provides software selectable debounce function (the former digital IO cards use hardware debounce and fixed at one frequency). You can filter out the pulse width at 10ms (100Hz), 5ms (200Hz), 1ms (1KHz) or no filter as you need.

Use [\*DIO3216B set\\_debounce\\_time\(\)\*](#) to select the debounce frequency and read back the setting by [\*DIO3216B read\\_debounce\\_time\(\)\*](#).

### 7.3 Interrupt function

Sometimes you want your application to take care of the I/O while special event occurs, interrupt function is the right choice. DIO3216B provide IN00 and IN01 as external event trigger input. To use it, first of all you must enable IRQ function by:

[\*\*\*DIO3216B enable IRQ\(\)\*\*\*](#).

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

[\*\*\*DIO3216B disable IRQ\(\)\*\*\*](#) to release the resource.

Next tell the driver your interrupt service routine by

[\*\*\*DIO3216B link IRQ process\(\)\*\*\*](#)

Finally you should enable / disable the hardware of the interrupt source,

[\*\*\*DIO3216B set IRQ mask\(\)\*\*\*](#) will do and your program is waiting an interrupt to service.

[\*\*\*DIO3216B read IRQ mask\(\)\*\*\*](#) to read back the setting.

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

[\*\*\*DIO3216B read IRQ status\(\)\*\*\*](#) to identify the source of interrupt.

For the application program do not use interrupt, you can also use [\*\*\*DIO3216B read IRQ status\(\)\*\*\*](#) as fast response input latch. You just [\*\*\*DIO3216B set IRQ mask\(\)\*\*\*](#) to setup the points you want to monitor and polling by [\*\*\*DIO3216B read IRQ status\(\)\*\*\*](#) to check the input. After reading, the status on card will automatically cleared.

#### 7.4 Software key function

Since DIO3216B 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

[\*DIO3216B set password\(\)\*](#) to set password and start the security function.

[\*DIO3216B change password\(\)\*](#) to change it.

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

[\*DIO3216B 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

[\*DIO3216B unlock security\(\)\*](#) unlock the security.

You can also use

[\*DIO3216B read security status\(\)\*](#) to check the current status of security.

#### 7.5 Error conditions

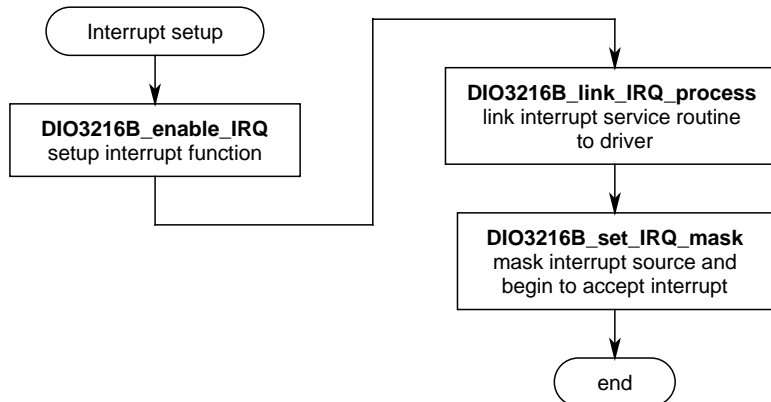
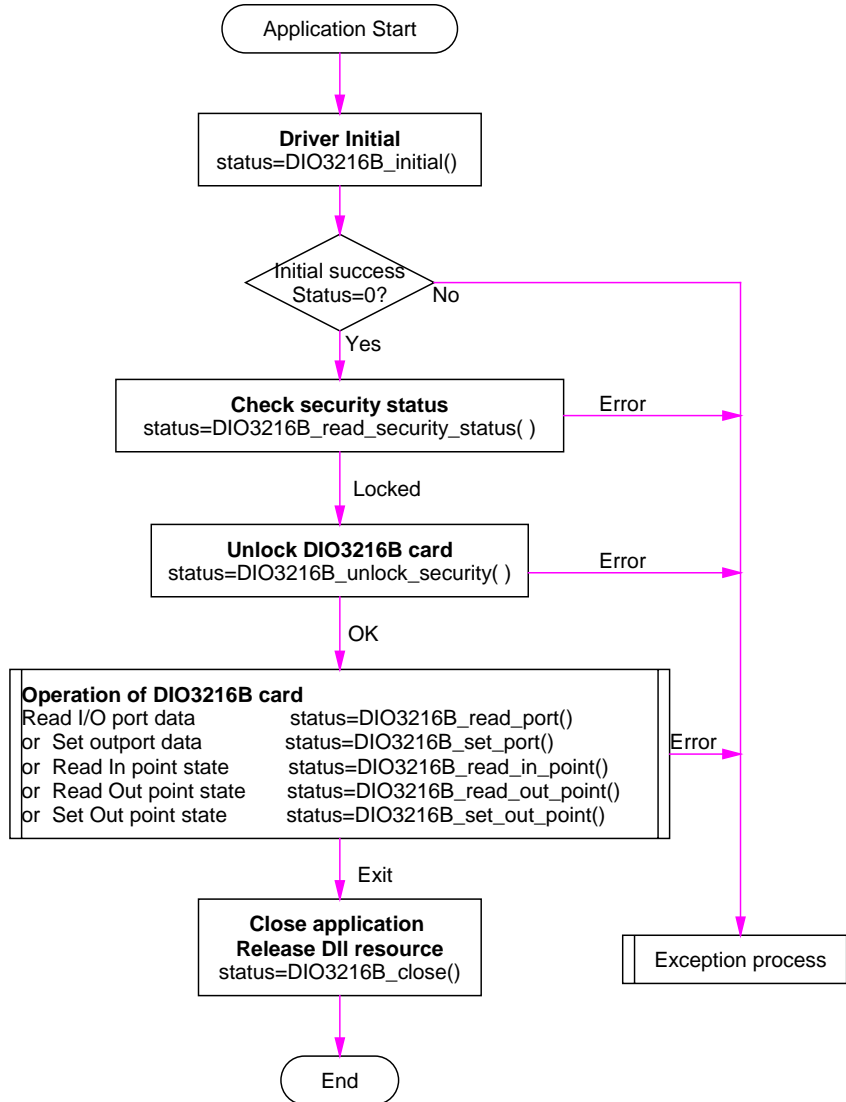
DIO3216 cards minimize error conditions. There are three possible fatal failure modes:

- ◆ System Fail Status Bit Valid
- ◆ Communication Loss
- ◆ Hardware not ready

These error types may indicate an internal hardware problem on the board. Error Codes contains a detailed listing of the error status returned by DIO3216B functions.

## 8. Flow chart of implement an application

### 8.1 DIO3216B Flow chart of implementation



## 9. **Function reference**

### 9.1 Function format

Every DIO3216B 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 DIO3216B function is the parameter **CardID** which is located the driver of DIO3216B 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**)

These topics contain detailed descriptions of each DIO3216B function. The functions are arranged alphabetically by function name. Refer to DIO3216B Function Reference for additional information.

## 9.2 Variable data types

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

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

**Table 2**

### 9.3 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 DIO3216B API. Read the following sections that apply to your programming language.

**Note:** Be sure to include the declaration functions of DIO3216B prototypes by including the appropriate DIO3216B header file in your source code. Refer to Building Applications with the DIO3216B Software Library for the header file appropriate to your compiler.

#### 9.3.1 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 = DIO3216B_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, port;
```

```
u8 data,
```

```
u32 Status;
```

```
Status = DIO3216B_read_port (CardID, port, &data);
```

#### 9.3.2 Visual basic

The file DIO3216B.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 DIO3216B.bas, do not use the numerical values.

In Visual Basic, you can add the entire DIO3216B.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 DIO3216B.bas file for your project in Visual Basic 4.0, go to the **File** menu and select the **Add File.. option**. Select DIO3216B.bas, which is browsed in the DIO3216B \ api directory. Then, select **Open** to add the file to the project.

To add the DIO3216B.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** DIO3216B.bas, which is in the DIO3216B \ api directory. Then, select **Open** to add the file to the project.

### 9.3.3 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 DIO3216BBC.lib DIO3216B.dll**

Then add the **DIO3216BBC.lib** to your project and add

**#include "DIO3216B.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 = DIO3216B\_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, port;*

*u8 data;*

*u32 Status;*

*Status = DIO3216B\_read\_port (CardID, port, &data);*

**Initialization and close**

● **DIO3216B\_initial**

**Format :** u32 status =DIO3216B\_initial (void)

**Purpose:** Initial the DIO3216B resource when start the Windows applications.

● **DIO3216B\_close**

**Format :** u32 status =DIO3216B\_close (void);

**Purpose:** Release the DIO3216B resource when close the Windows applications.

● **DIO3216B\_info**

**Format :** u32 status =DIO3216B\_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

## I/O Port R/W

### ● DIO3216B read port

**Format :** u32 status = DIO3216B\_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: input port IN10-IN17 2: output port OUT00-OUT07 3: output port OUT10-OUT17

**Output:**

Name	Type	Description
data	u8	I/O data

### ● DIO3216B set port

**Format :** u32 status = DIO3216B\_set\_port (u8 CardID,u8 port, u8 data)

**Purpose:** Sets the output data.

**Parameters:**

**Input:**

Name	Type	Description
CardID	u8	assigned by Rotary SW
port	u8	2: output port for OUT00-OUT07 3: output port for OUT10-OUT17
data	u8	bitmap of output values

### ● DIO3216B set out point

**Format :** u32 status =DIO3216B\_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 8~15 for OUT10-OUT17
state	u8	state of output point

- **DIO3216B read in point**

**Format :** u32 status =DIO3216B\_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 8~15 for IN10-IN17

**Output:**

Name	Type	Description
state	u8	state of point of input

- **DIO3216B read out point**

**Format :** u32 status =DIO3216B\_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 0~7 for OUT00-OUT07 8~15 for OUT10-OUT17

**Output:**

Name	Type	Description
state	u8	state of output point

● **DIO3216B set debounce time**

**Format :** u32 status = DIO3216B\_set\_debounce\_time (u8 CardID , u8 port , u8 data)

**Purpose:** Set the input port debounce time

**Parameters:**

**Input:**

Name	Type	Description
CardID	u8	assigned by Rotary SW
port	u8	port number 0: input port0 for IN00-IN07 1: input port1 for IN10-IN17
data	u8	Debounce time selection: 0: no debounce 1: filter out over 100Hz (default) 2: filter out over 200Hz 3: filter out over 1KHz

● **DIO3216B read debounce time**

**Format :** u32 status = DIO3216B\_read\_debounce\_time (u8 CardID , u8 port , u8 \*data)

**Purpose:** Read the input port debounce time setting

**Parameters:**

**Input:**

Name	Type	Description
CardID	u8	assigned by Rotary SW
port	u8	port number 0: input port0 for IN00-IN07 1: input port1 for IN10-IN17

**Output:**

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

## Interrupt function

### ● DIO3216B enable IRQ

**Format :** u32 status = DIO3216B\_enable\_IRQ (u8 CardID, HANDLE \*phEvent)

**Purpose:** Enable interrupt from selected source

**Parameters:**

**Input:**

Name	Type	Description
CardID	u8	assigned by Rotary SW

**Output:**

Name	Type	Description
phEvent	HANDLE	event handle

### ● DIO3216B disable IRQ

**Format :** u32 status = DIO3216B\_disable\_IRQ (u8 CardID)

**Purpose:** Disable interrupt from selected source

**Parameters:**

**Input:**

Name	Type	Description
CardID	u8	assigned by Rotary SW

### ● DIO3216B link IRQ process

**Format :** u32 status = DIO3216B\_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

- **DIO3216B set IRQ mask**

**Format :** u32 status = DIO3216B\_set\_IRQ\_mask (u8 CardID, u8 Data)

**Purpose:** Mask interrupt from IN00~ IN07

**Parameters:**

**Input:**

Name	Type	Description
CardID	u8	assigned by Rotary SW
Data	U8	Bit0 =0 disable irq from IN00 =1 enable irq from IN00 Bit1 =0 disable irq from IN01 =1 enable irq from IN01 .... Bit7 =0 disable irq from IN07 =1 enable irq from IN07

- **DIO3216B read IRQ mask**

**Format :** u32 status = DIO3216B\_read\_IRQ\_mask (u8 CardID, u8 \*mask)

**Purpose:** Read back the mask of interrupt

**Parameters:**

**Input:**

Name	Type	Description
CardID	u8	assigned by Rotary SW

**Output:**

Name	Type	Description
mask	u8	Bit0 =0 disable irq from IN00 =1 enable irq from IN00 Bit1 =0 disable irq from IN01 =1 enable irq from IN01 .... Bit7 =0 disable irq from IN07 =1 enable irq from IN07

● **DIO3216B read IRQ status**

**Format :** u32 status = DIO3216B\_read\_IRQ\_status (u8 CardID, u8 \*Event\_Status)

**Purpose:** To read back the interrupt source to identify IN00~ IN07

**Parameters:**

**Input:**

Name	Type	Description
CardID	u8	assigned by Rotary SW

**Output:**

Name	Type	Description
Event_Status	u8	Bit0 =1 irq from IN00 Bit1 =1 irq from IN01 .... Bit7 =1 irq from IN07

**Note:**

1. This command can be used in non-interrupt application for scanning the input change, but once you read the status, the on chip register will also be cleared for next transition input.
2. For interrupt application, you can read the status for identification of interrupt sources to determine the response you want to take.

## Software key function

### ● DIO3216B set password

**Format :** u32 status = DIO3216B\_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.

### ● DIO3216B change password

**Format :** u32 status = DIO3216B\_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

### ● DIO3216B clear password

**Format :** u32 status = DIO3216B\_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

● **DIO3216B unlock security**

**Format :** u32 status = DIO3216B\_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

● **DIO3216B read security status**

**Format :** u32 status = DIO3216B\_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.

## 9.5 Dll list

	<b>Function Name</b>	<b>Descriptive Name</b>
1	DIO3216B_initial( )	DIO3216B Initial
2	DIO3216B_close( )	DIO3216B Close
3	DIO3216B_info( )	get OS. Assigned address
4	DIO3216B_read_port( )	Read Port Data (word)
5	DIO3216B_set_port( )	Set Output port(word)
6	DIO3216B_set_out_point( )	Set Output Point State(bit)
7	DIO3216B_read_in_point( )	Read Input Point State(bit)
8	DIO3216B_read_out_point( )	Read Output Point State(bit)
9	DIO3216B_set_debounce_time( )	Set input port debounce time
10	DIO3216B_read_debounce_time( )	Read input port debounce time
11	DIO3216B_enable_IRQ( )	Enable interrupt function
12	DIO3216B_disable_IRQ( )	Disable interrupt function
13	DIO3216B_link_IRQ_process( )	Link interrupt service routine to driver
14	DIO3216B_set_IRQ_mask( )	Set interrupt mask
15	DIO3216B_read_IRQ_mask( )	
16	DIO3216B_read_IRQ_status( )	Read back irq status
17	DIO3216B_set_password( )	Set software key
18	DIO3216B_change_password( )	Change software key
19	DIO3216B_clear_password( )	Clear software key
20	DIO3216B_unlock_security( )	Unlock software key
21	DIO3216B_read_security_status( )	Read software key status

## 10. DIO-3216B Error codes table

<b>Error Code</b>	<b>Symbolic Name</b>	<b>Description</b>
<b>0</b>	JSDRV_NO_ERROR	No error.
<b>2</b>	JSDRV_INIT_ERROR	Driver initial error
<b>3</b>	JSDRV_UNLOCK_ERROR	Software key unlock error
<b>4</b>	JSDRV_LOCK_COUNTER_ERROR	Software key unlock error count over
<b>5</b>	JSDRV_SET_SECURITY_ERROR	Software key setting error
<b>100</b>	DEVICE_RW_ERROR	Device Read/Write error
<b>101</b>	JSDRV_NO_CARD	No DIO3216B card on the system.
<b>102</b>	JSDRV_DUPLICATE_ID	DIO3216B CardID duplicate error.
<b>300</b>	JSDIO_ID_ERROR	Function input parameter error. CardID setting error, CardID doesn't match the DIP SW setting
<b>301</b>	JSDIO_PORT_ERROR	Function input parameter error. Parameter out of range.
<b>302</b>	JSDIO_IN_POINT_ERROR	Function input parameter error. Parameter out of range.
<b>303</b>	JSDIO_OUT_POINT_ERROR	Function input parameter error. Parameter out of range.
<b>304</b>	JSDIO_VERSION_ERROR	Hardware version can not match with software version
<b>306</b>	JSDIO_DEBOUNCE_MODE_ERROR	Bad debounce time parameter