DIO6208/6216

PCI-104
Digital I/O Card

Software Manual (V1.0)

健昇科技股份有限公司

JS AUTOMATION CORP.

新北市汐止區中興路 100 號 6 樓
6F., No.100, Zhongxing Rd.,
Xizhi Dist., New Taipei City, Taiwan
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
2.0	wdm6216.sys V2.0
	drv6216.dll V2.0
	DIO6216.dll V2.0

Contents

1. How to install the software of DIO6208/6216

Note: DIO6208 use the same software driver with the DIO6216, if the descriptions do not specially point out "not for 6208" will be applied to both type.

1.1 Install the PCI-104 driver

The PCI 104 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 (..\DIO6208_16\Software\Win2K_up\ or if you download from website please execute the file DIO6216_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.

2. Where to find the file you need

Win2K/XP/7 and up

The directory will be located at

.. \ JS Automation \DIO6216\API\   (header files and lib files for VB,VC,BCB,C#)

.. \ JS Automation \DIO6216\Driver\   (backup copy of DIO6216 drivers)

.. \ JS Automation \DIO6216\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.

3. About the DIO6208/6216 software

DIO6216 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 DIO6216 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 DIO6216 functions within Windows' operation system environment.

3.1 What you need to get started

To set up and use your DIO6216 software, you need the following:

- DIO6216 software
- DIO6216 hardware
  - Main board
  - Wiring board (Option)

3.2 Software programming choices

You have several options to choose from when you are programming DIO6216 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 DIO6216 software.

4. DIO6216 Language support

The DIO6216 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.

4.1 Building applications with the DIO6216 software library

The DIO6216 function reference topic contains general information about building DIO6216 applications, describes the nature of the DIO6216 files used in building DIO6216 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.

4.2 DIO6216 Windows libraries

The DIO6216 for Windows function library is a DLL called **DIO6216.dll**. Since a DLL is used, DIO6216 functions are not linked into the executable files of applications. Only the information about the DIO6216 functions in the DIO6216 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 DIO6216 functions in DIO6216.dll.

Header Files and Import Libraries for Different Development Environments		
Language	**Header File**	**Import Library**
Microsoft Visual C/C++	DIO6216.h	DIO6216VC.lib
Borland C/C++	DIO6216.h	DIO6216BC.lib
Microsoft Visual C#	DIO6216.cs	
Microsoft Visual Basic	DIO6216.bas	
Microsoft VB.net	DIO6216.vb	

Table 1

5. 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

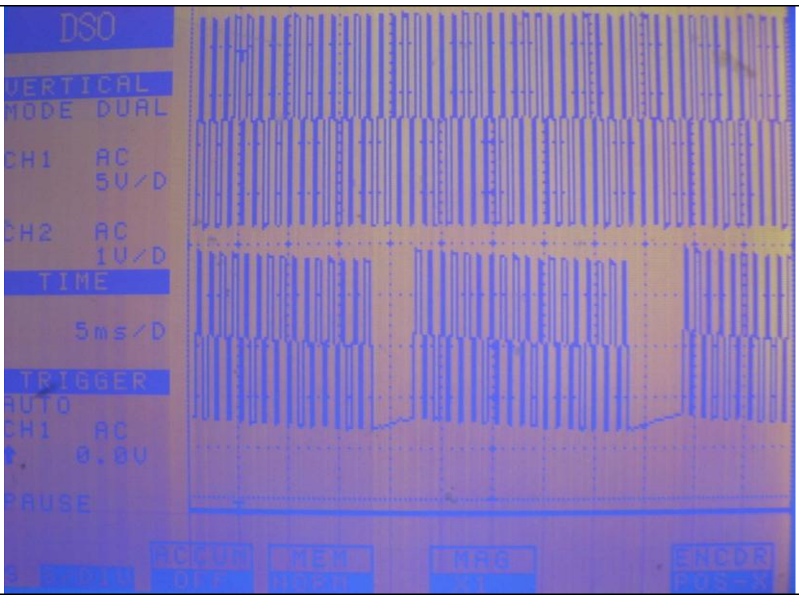
There are several devices used as output driver, the relay, transistor or MOS FET, SCR and SSR.
Relay is electric- mechanical device, it life time is about 1,000,000 times of switching. But on the other hand it has many selections such as high voltage or high current. It can also be used to switch DC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

 DSO VERTICAL MODE DUAL CH1 AC 5V/D CH2 AC 1V/D TIME 5ms/D TRIGGER AUTO CH1 AC 0.0V PAUSE ACCUM OFF · MEM · MAG x1 · ENCDR POS-X	<- 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.**

6. Software overview

These topics describe the features and functionality of the DIO6216 boards and briefly describes the DIO6216 functions.

Owing to the PCI-104 use stack method to connect the bus, the lower layer stack will be closest to the PCI signal. To compensate the distance, the DIO6216 requires the CardID setting and clock selection according to the physical stacking layer, i.e. the closest one CardID=0 and CLK also select 0, the next to the bottom layer will be set jumper to CardID=1 and CLK also to 1….

6.1 Initialization and close

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

DIO6216_initial( ) will do.

Once you want to close your application, call

DIO6216_close( ) to release all the resource.

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

DIO6216_info( ) to get the address .

6.2 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 ***DIO6216_debounce_time_set ( )*** to configure the debounce time.

DIO6216_debounce_time_read ( ) to read back the configuration data.

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

DIO6216_port_read ( ) to read a byte data from I/O port,

DIO6216_port_set ( ) to output byte data to output port,

DIO6216_point_set ( ) to set output bit,

DIO6216_point_read ( ) to read I/O bit,

The input and output port can also set the polarity to meet the logic convention, use

DIO6216_port_polarity_set ( ) to configure and

DIO6216_port_polarity_read ( ) to read back the setting.

6.3 Timer / Counter function

The on card timer is a 32 bits counter based on 1MHZ time base. To configure the working mode use ***DIO6216_timer_set ( )*** to configure as timer and its output mode

To start/stop the operation by:

DIO6216_timer_start ( )

DIO6216_timer_stop ( )

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

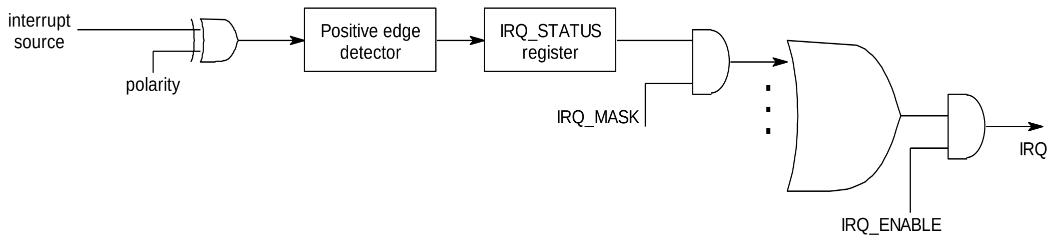
DIO6216_TC_set ( ) set TC dedicated registers

DIO6216_TC_read ( ) read TC dedicated registers

6.4 Interrupt function

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

The DIO6216 card interrupt model is as follows:



For digital input, the interrupt source is IN00~IN07, the physical input is changed polarity by POLARITY_SET, then the on board hardware detect the positive edge transition to trigger the IRQ_STATUS register, it is irrelevant to the IRQ_MASK. If the IRQ_MASK is set to 1 and IRQ_ENABLE are also set, the interrupt will generate. By this model, you can see that you can check the fast changing input by IRQ_STATUS without using interrupt.

To configure the IN00~IN07 interrupt polarity (also the input polarity), use

DIO6216_port_polarity_set( ) and read back the setting data by

DIO6216_port_polarity_read( )**.**

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

DIO6216_IRQ_mask_set ( )

DIO6216_IRQ_mask_read ( ) to read back the IRQ mask status.

After all is prepared, tell the driver your interrupt service routine by

DIO6216_IRQ_process_link ( )

To enable the IRQ function,

DIO6216_IRQ_enable ( ) to start waiting the interrupt.

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

DIO6216_IRQ_disable ( ) to release the resource.

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

DIO6216_IRQ_status_read ( ) to identify the source of interrupt. If you do not use the interrupt function (IRQ disabled) and want to scan the interrupt generated only, this command can also use to verify the external trigger status.

6.5 Software key function

Since DIO6216 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

DIO6216_password_set ( ) to set password and start the security function.

DIO6216_password_change ( ) to change it.

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

DIO6216_password_clear ( ) 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

DIO6216_security_unlock ( ) unlock the security.

You can also use

DIO6216_security_status_read ( ) to check the current status of security.

6.6 Error conditions

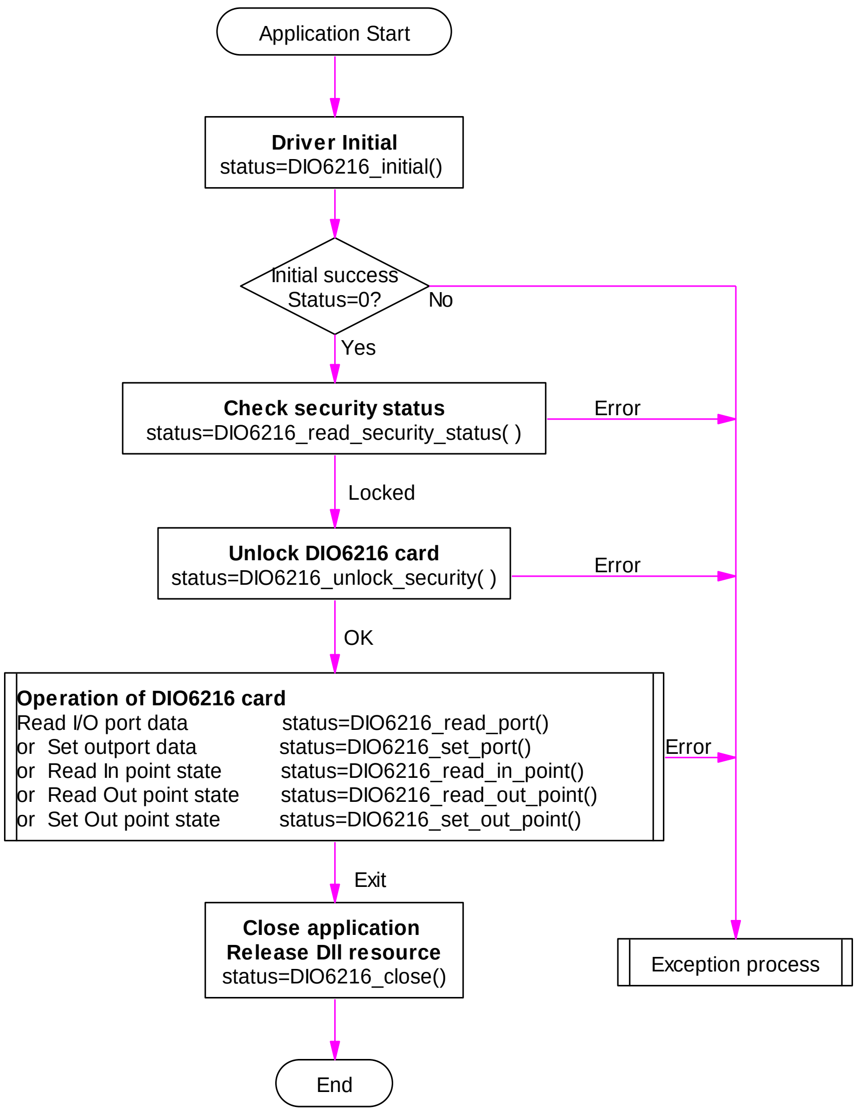
DIO6216 cards minimize error conditions. There are three possible fatal failure modes:

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

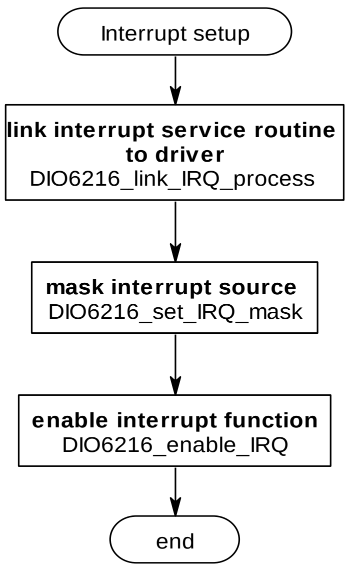
These error types may indicate an internal hardware problem on the board. Error Codes contains a detailed listing of the error status returned by DIO6216 functions.

7. Flow chart of application implementation

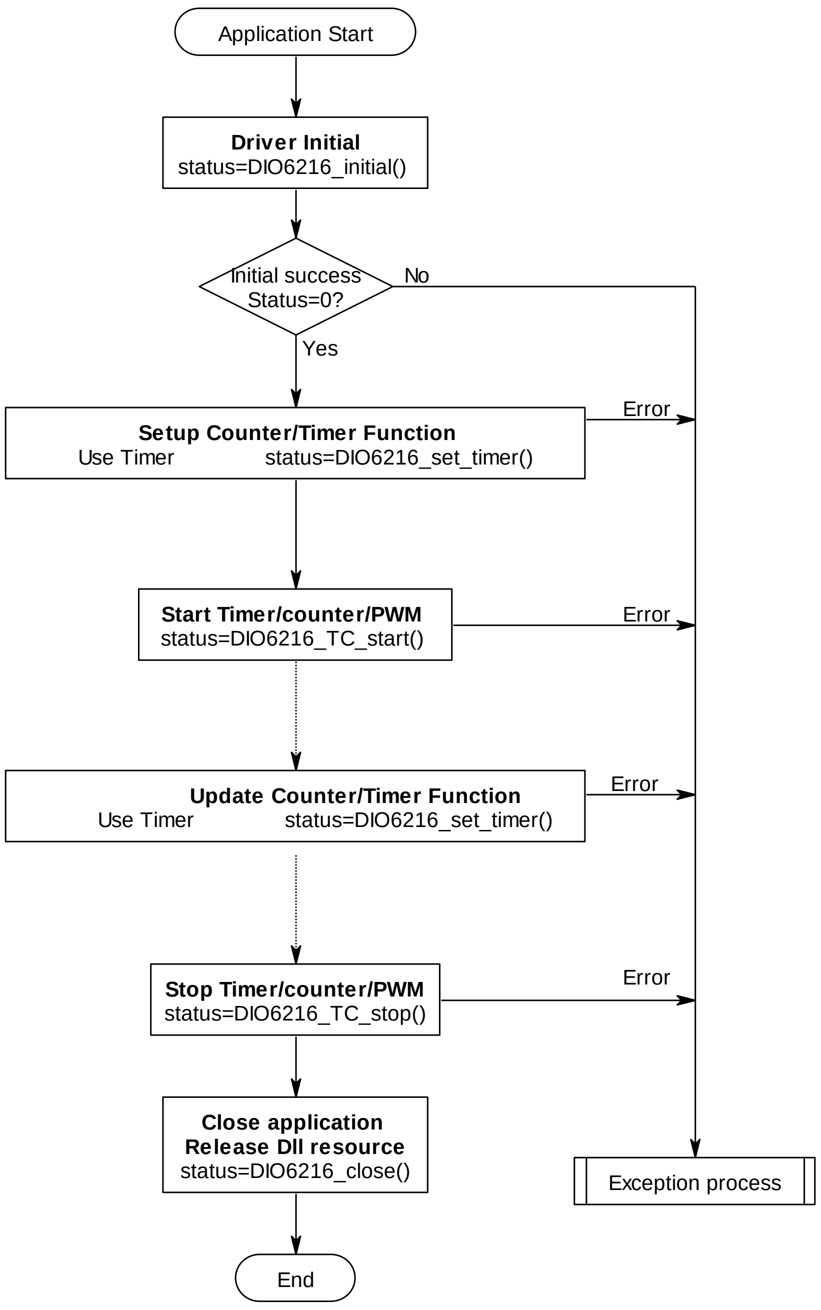
7.1 DIO6216 Flow chart of application implementation



Application Start
Driver Initial
status=DIO6216_initial()
Initial success
Status=0?
No
Yes
Check security status
status=DIO6216_read_security_status( )
Error
Locked
Unlock DIO6216 card
status=DIO6216_unlock_security( )
Error
OK
Operation of DIO6216 card
Read I/O port data status=DIO6216_read_port()
or Set outport data status=DIO6216_set_port()
or Read In point state status=DIO6216_read_in_point()
or Read Out point state status=DIO6216_read_out_point()
or Set Out point state status=DIO6216_set_out_point()
Error
Exit
Close application
Release Dll resource
status=DIO6216_close()
Exception process
End



7.2 DIO6216 Flow chart of Timer / Counter / PWM application



8. Function reference

8.1 Function format

Every DIO6216 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 DIO6216 function is the parameter **CardID** which is located the driver of DIO6216 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 (**0x0-0x3**)

These topics contain detailed descriptions of each DIO6216 function. The functions are arranged alphabetically by function name. Refer to DIO6216 Function Reference for additional information.

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 DIO6216 API. Read the following sections that apply to your programming language.

Note: Be sure to include the declaration functions of DIO6216 prototypes by including the appropriate DIO6216 header file in your source code. Refer to Building Applications with the DIO6216 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 = DIO6216_ port _read (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 = DIO6216_port_read (CardID, port, &data);
```

8.3.2 Visual basic

The file DIO6216.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 DIO6216.bas, do not use the numerical values.

In Visual Basic, you can add the entire DIO6216.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 DIO6216.bas file for your project in Visual Basic 4.0, go to the **File** menu and select the **Add File... option**. Select Dio6216.bas, which is browsed in the DIO6216 \ API directory. Then, select **Open** to add the file to the project.

To add the DIO6216.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** DIO6216.bas, which is in the DIO6216 \ 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 DIO6216bc.lib DIO6216.dll

Then add the **DIO6216bc.lib** to your project and add

#include “DIO6216.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 = DIO6216_port_read (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 = DIO6216_port_read (CardID, port, &data);
```

8.4 DIO6216 Functions

Note:

Owing to the PCI-104 use stack method to connect the bus, the lower layer stack will be closest to the PCI signal. To compensate the distance, the DIO6216 requires clock selection according to the physical stacking layer, i.e. the closest one CLK also select 0, the next to the bottom layer will be set CLK jumper to 1….

Initialization and close

- **DIO6216_initial**

Format : **u32 status =DIO6216_ initial (void)**

Purpose: Initial the DIO6216 resource when start the Windows applications.

- **DIO6216_close**

Format : **u32 status =DIO6216_close (void);**

Purpose: Release the DIO6216 resource when close the Windows applications.

- **DIO6216_info**

Format : **u32 status =DIO6216_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	timer physical I/O address assigned by OS

I/O Port R/W

- **DIO6216_debounce_time_set**

Format : **u32 status = DIO6216_debounce_time_set (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	0: input port 0 1: input port 1 **(not for 6208)**
data	u8	Debounce time selection: 0: no debounce 1: filter out duration less than 10ms (default) 2: filter out duration less than 5ms 3: filter out duration less than 1ms

- **DIO6216_debounce_time_read**

Format : **u32 status = DIO6216_debounce_time_read (u8 CardID , u8 port, u8 *data)**

Purpose: Read back the input port debounce time configuration

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by Rotary SW
port	u8	0: input port 0 1: input port 1**(not for 6208)**

Output:

Name	Type	Description
data	u8	Debounce time selection: 0: no debounce 1: filter out duration less than 10ms (default) 2: filter out duration less than 5ms 3: filter out duration less than 1ms

- **DIO6216_port_read**

Format : **u32 status = DIO6216_port_read (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 for IN00-IN07 1: input port for IN10-IN17 **(not for 6208)** 2: output port OUT00-OUT07 3: output port OUT10-OUT17 **(not for 6208)**

Output:

Name	Type	Description
data	u8	I/O data

Note:

If the port to be read is output, the data read back will be the output register data not physical output data.

- **DIO6216_port_set**

Format : **u32 status = DIO6216_port_set (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 **(not for 6208)**
data	u8	bitmap of output values

- **DIO6216_point_set**

Format : **u32 status =DIO6216_point_set (u8 CardID,u8 port, u8 point, u8 state)**

Purpose: Sets the bit data of output port.

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 **(not for 6208)**
point	u8	point number 0~7 for OUTx0~OUTx7
state	u8	state of output point

- **DIO6216_point_read**

Format : **u32 status =DIO6216_point_read (u8 CardID, u8 port, u8 point, u8 *state)**

Purpose: Read the input state of the input points.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by Rotary SW
port	u8	0: input port for IN00-IN07 1: input port for IN10-IN17 **(not for 6208)** 2: output port for OUT00-OUT07 3: output port for OUT10-OUT17 **(not for 6208)**
point	u8	point number of input 0~7 for INx0~INx7 or 0~7 for OUTx0~OUTx7

Output:

Name	Type	Description
state	u8	state of point of input

Note:

If the point to be read is output, the data read back will be the output register data not physical output data.

- **DIO6216_port_polarity_set**

Format : **u32 status =DIO6216_port_polarity_set (u8 CardID,u8 port, u8 polarity)**

Purpose: Sets the polarity of port.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by Rotary SW
port	u8	0: input port for IN00~IN07 1: input port for IN10~IN17 **(not for 6208)** 2: output port for OUT00-OUT07 3: output port for OUT10-OUT17 **(not for 6208)**
polarity	u8	b7~b0 for INx7~INx0 or OUTx7~OUTx0 any bit =0, polarity is normal any bit=1, polarity is inverse

- **DIO6216_port_polarity_read**

Format : **u32 status =DIO6216_port_polarity_read (u8 CardID, u8 port, u8 * polarity)**

Purpose: Read the setting of polarity.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by Rotary SW
port	u8	0: input port for IN00-IN07 1: input port for IN10-IN17 **(not for 6208)** 2: output port for OUT00-OUT07 3: output port for OUT10-OUT17 **(not for 6208)**

Output:

Name	Type	Description
polarity	u8	b7~b0 for INx7~INx0 or OUTx7~OUTx0 any bit =0, polarity is normal any bit=1, polarity is inverse

Timer / Counter function

- **DIO6216_timer_set**

Format : **u32 status = DIO6216_timer_set (u8 CardID, u32 time_constant)**

Purpose: To setup timer operation mode or update timer

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
time_constant	u32	Timer constant based on 1us clock

Note:

1. Time constant is based on 1us clock, period T= (time_constant +1) * 1us

2. If you also enable the timer interrupt, the period T must at least larger than the system interrupt response time else the system will be hanged by excess interrupts.

- **DIO6216_timer_start**

Format : **u32 status = DIO6216_timer_start (u8 CardID)**

Purpose: To start timer operation mode

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch

- **DIO6216_timer_stop**

Format : **u32 status = DIO6216_timer_stop (u8 CardID)**

Purpose: To stop timer operation mode

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch

- **DIO6216_TC_set**

Format : **u32 status=DIO6216_TC_set (u8 CardID,u8 index,u32 data)**

Purpose: To load data to timer related registers

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
index	u8	0: TC_CONTROL 1: PRELOAD 2: TIMER
data	u32	For TC_CONTROL 0: stop timer operation 1: timer run For PRELOAD or TIMER Data is the constant to be load

Note:

PRELOAD is the register for timer to re-load, the value will be valid while timer count to zero and reload the data.

- **DIO6216_TC_read**

Format : **u32 status=DIO6216_TC_read (u8 CardID,u8 index,u32 *data)**

Purpose: To read data from timer related registers

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
index	u8	0: TC_CONTROL 1: PRELOAD 2: TIMER

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	0~1	0:timer stops operation 1: timer runs
1	PRELOAD	1~0xffffffff	timer preload value
2	TIMER	1~0xffffffff	Timer value on the fly

Note:

1. For example, you want to watch the counting on the fly, use
   DIO6216_TC_read (CardID, TIMER,*data)   //CardID as you assign, TIMER =2
   To read back the timer value.

Interrupt function

- **DIO6216_IRQ_mask_set**

Format : **u32 status = DIO6216_IRQ_mask_set (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 timer counts to zero as interrupt source 1, enable timer counts to zero as interrupt source

- **DIO6216_IRQ_mask_read**

Format : **u32 status = DIO6216_IRQ_mask_read (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

- **DIO6216_IRQ_process_link**

Format : **u32 status = DIO6216_IRQ_process_link (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

Note:

Before using interrupt function, please refer section 6.4 Interrupt function for the hardware model.

- **DIO6216_IRQ_enable**

Format : **u32 status = DIO6216_IRQ_enable (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

- **DIO6216_IRQ_disable**

Format : **u32 status = DIO6216_IRQ_disable (u8 CardID)**

Purpose: Disable interrupt from unmasked source

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by Rotary SW

- **DIO6216_IRQ_status_read**

Format : **u32 status = DIO6216_IRQ_status_read (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, timer counts to zero interrupted

Note:

1. DIO6216_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 DIO6216_IRQ_status( ) to catch the fast changing input status.
3. After calling DIO6216_IRQ_status( ), the status hardware will be cleared.

Software key function

- **DIO6216_password_set**

Format : **u32 status = DIO6216_password_set (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.

- **DIO6216_password_change**

Format : **u32 status = DIO6216_password_change (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

- **DIO6216_password_clear**

Format : **u32 status = DIO6216_password_clear (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

- **DIO6216_security_unlock**

Format : **u32 status = DIO6216_security_unlock (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

- **DIO6216_security_status_read**

Format : **u32 status = DIO6216_security_status_read (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.

8.5 Dll list

	Function Name	**Description**
1	DIO6216_initial( )	DIO6216 Initial
2	DIO6216_close( )	DIO6216 Close
3	DIO6216_info( )	get OS. assigned address
4	DIO6216_debounce_time_set( )	Set input port digital debounce time
5	DIO6216_debounce_time_read( )	Read back input port digital debounce time
6	DIO6216_port_read( )	Read Port Data
7	DIO6216_port_set( )	Set Output port
8	DIO6216_point_set( )	Set Output Point State(bit)
9	DIO6216_point_read( )	Read Input Point State(bit)
10	DIO6216_port_polarity_set( )	Sets the polarity of port.
11	DIO6216_port_polarity_read( )	Read the setting of polarity.
12	DIO6216_timer_set( )	Setup or up date timer
13	DIO6216_timer_start( )	Start timer operation
14	DIO6216_timer_stop( )	Stop timer operation
15	DIO6216_TC_set( )	Set TC registers
16	DIO6216_TC_read( )	Read TC registers
17	DIO6216_IRQ_mask_set( )	Set interrupt source mask
18	DIO6216_IRQ_mask_read( )	Read interrupt source mask
19	DIO6216_IRQ_process_link( )	Link interrupt service routine to driver
20	DIO6216_IRQ_enable( )	Enable interrupt function
21	DIO6216_IRQ_disable( )	Disable interrupt function
22	DIO6216_IRQ_status_read( )	Read back irq status
23	DIO6216_password_set( )	Set software key
24	DIO6216_password_change( )	Change software key
25	DIO6216_password_clear( )	Clear software key
26	DIO6216_security_unlock( )	Unlock software key
27	DIO6216_security_status_read( )	Read software key status

9. DIO6216 Error codes summary

Error Code	Symbolic Name	Description
0	DRV_NO_ERROR	No error.
1	DRV_READ_DATA_ERROR	Read data error
2	DRV_INIT_ERROR	Driver initial error
3	DRV_UNLOCK_ERROR	Software key unlock error
4	DRV_LOCK_COUNTER_ERROR	Software key unlock error count over
5	DRV_SET_SECURITY_ERROR	Software key setting error
100	DEVICE_RW_ERROR	Device Read/Write error
101	DRV_NO_CARD	No DIO6216 card on the system.
102	DRV_DUPLICATE_ID	DIO6216 CardID duplicate error.
300	ID_ERROR	Function input parameter error. CardID setting error, CardID doesn’t match the DIP SW setting
301	PORT_ERROR	Function input parameter error. Parameter out of range.
302	IN_POINT_ERROR	Function input parameter error. Parameter out of range.
303	OUT_POINT_ERROR	Function input parameter error. Parameter out of range.
304	VERSION_ERROR	Hardware version can not match with software version
305	SOURCE_ERROR	TC source select error
306	DEBOUNCE_MODE_ERROR	Digital input debounce mode error
406	INDEX_ERROR	TC register index error
407	TO_MODE_ERROR	Timer output mode error
408	TI_MODE_ERROR	Timer input mode error