EX-94317

Digital I/O Card

Software Manual (V1.0)

TOPSCCC TECHNOLOGY CO.,LTD.

http://www.topsccc.com.tw

http://www.topsccc.com

E-mail：info@topsccc.com

Correction record

Manual Version	Record
1.0	wdm94317.sys V1.0
	wdm94317.dll V1.0
	EX94317.dll V1.0

Contents

1. How to install the software of EX94317 4
   1.1 Install the PCI driver 4
2. Where to find the file you need 5
3. About the EX94317 software 6
   3.1 What you need to get started 6
   3.2 Software programming choices 6
4. EX94317 Language support 7
   4.1 Building applications with the EX94317 software library 7
   4.2 EX94317 Windows libraries 7
5. Basic concepts of digital I/O control 8
6. Software overview 11
   6.1 Initialization and close 11
   6.2 I/O Port R/W 11
   6.3 Timer function 11
   6.4 Interrupt function 12
   6.5 Error conditions 13
7. Flow chart of application implementation 14
   7.1 EX94317 Flow chart of application implementation 14
8. Function reference 15
   8.1 Function format 15
   8.2 Variable data types 16
   8.3 Programming language considerations 17
   8.4 EX94317 Functions 19
      Initialization and close 19
         EX94317_initial 19
         EX94317_close 19
         EX94317_info 19
      I/O Port R/W 20
         EX94317_config_set 20
         EX94317_config_read 20
         EX94317_port_set 21
         EX94317_port_read 21
         EX94317_point_set 22
         EX94317_point_read 22
         EX94317_debounce_time_set 23
         EX94317_debounce_time_read 23
      Timer function 24
         EX94317_timer_set 24
         EX94317_timer_start 24

EX94317_timer_stop 24
EX94317_TC_set 25
EX94317_TC_read 25
Interrupt function 26
EX94317_IRQ_polarity_set 26
EX94317_IRQ_polarity_read 26
EX94317_IRQ_mask_set 27
EX94317_IRQ_mask_read 27
EX94317_IRQ_process_link 28
EX94317_IRQ_enable 28
EX94317_IRQ_disable 28
EX94317_IRQ_status_read 29
8.5 Dll list 30
9. EX94317 Error codes table 31

1. How to install the software of EX94317

1.1 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
   \EX94317\Software\Win2K_up\EX94317_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://topsccc.com.tw/

to download the complementary documents.

2. Where to find the file you need

Windows2000,XP,Vista

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

\EX94317\software\Win2K_up\install\EX94317_Install.exe

The directory will be located at

.. \TOPSCCC\EX94317\API (header files and VB,VC lib files)

.. \TOPSCCC\EX94317\Driver (backup copy of EX94317 drivers)

.. \TOPSCCC\EX94317\exe (demo program and source code)

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

3. About the EX94317 software

EX94317 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 EX94317 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 EX94317 functions within Windows' operation system environment.

3.1 What you need to get started

To set up and use your EX94317 software, you need the following:

- EX94317 software
- EX94317 hardware
  - Main board
  - Wiring board (Option)

3.2 Software programming choices

You have several options to choose from when you are programming EX94317 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 EX94317 software.

4. EX94317 Language support

The EX94317 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.

4.1 Building applications with the EX94317 software library

The EX94317 function reference topic contains general information about building EX94317 applications, describes the nature of the EX94317 files used in building EX94317 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.

4.2 EX94317 Windows libraries

The EX94317 for Windows function library is a DLL called **EX94317.dll**. Since a DLL is used, EX94317 functions are not linked into the executable files of applications. Only the information about the EX94317 functions in the EX94317 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 EX94317 functions in EX94317.dll.

Header Files and Import Libraries for Different Development Environments		
Development Environment	**Header File**	**Import Library**
Microsoft C/C++	EX94317.h	EX94317VC.lib
Borland C/C++	EX94317.h	EX94317BC.lib
Microsoft Visual Basic	EX94317.bas	

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 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 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.**

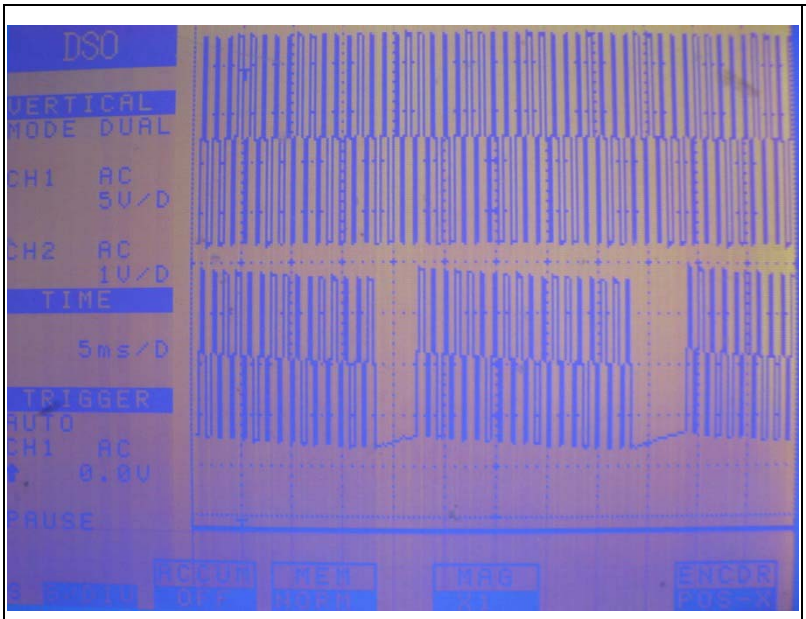
SCR (or triac) is seldom direct connect to digital output, but his relative SSR is the most often selection. In fact, SSR is a compact package of trigger circuit and triac. You can choose zero cross trigger (output command only turn on the output at power phase near zero to eliminate surge) or direct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



<- 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 EX94317 boards and briefly describes its functions.

6.1 Initialization and close

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

EX94317_initial( ) will do.

Once you want to close your application, call

EX94317_close( ) to release all the resource.

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

EX94317_info( ) to get the address .

6.2 I/O Port R/W

Before using a IO port, you must configure the port direction (as input or as output) first by

EX94317_config_set( ) and any time you can read back configuration by

EX94317_config_read( )

Then you can use the following functions for I/O port output value reading and control:

EX94317_port_set( ) to output byte data to output port,

EX94317_port_read( ) to read a byte data from I/O port,

EX94317_point_set ( ) to set output bit,

EX94317_point_read( ) to read I/O bit,

Mechanical contact or noisy environment always induced unstable state at digital input, the EX94317 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

EX94317_debounce_time_set( ) to select the debounce frequency and read back the setting by

EX94317_debounce_tme_read( ).

6.3 Timer function

There is a build in 32 bit timer run on 1us time base, you can set the timer constant by

EX94317_timer_set( ) and

EX94317_timer_start( ) to star its operation,

EX94317_timer_stop( ) to stop operation.

For the timer related registers use:

EX94317_TC_set( ) to set registers,

EX94317_TC_read( ) to read back 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. EX94317 provide IN00 ~ IN07 as external event trigger input. You may configure the trigger polarity by:

EX94317_IRQ_polarity_set( ) and read back by

EX94317_IRQ_polarity__read( )

For timer ans digital IO interrupts, you can mask off the source you don' want by

EX94317_IRQ_mask_set( ) and read back by

EX94317_IRQ_mask_read( ).

After all the above is prepared, you must first link your service routine to the driver by

EX94317_IRQ_process_link( )

Now all is ready, you can enable the interrupt by

EX94317_IRQ_enable( ) or disable by

EX94317_IRQ_disable( ).

To read back the interrupt status (at interrupt service routine or polling routine) use

EX94317_IRQ_status_read( ).

After reading the status register on card will be cleared.

6.5 Error conditions

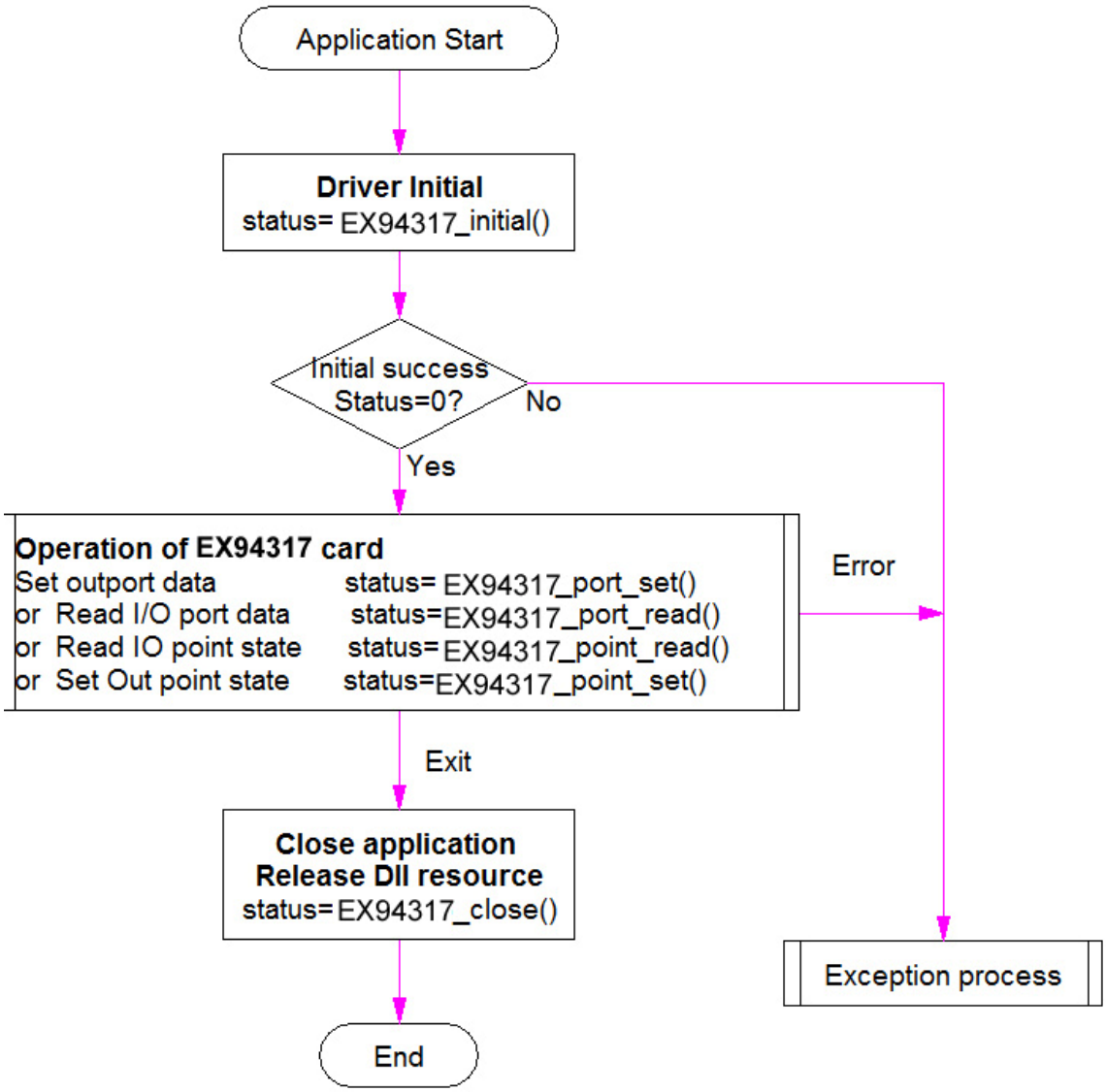
EX94317 cards minimize error conditions. There are three possible fatal failure modes:

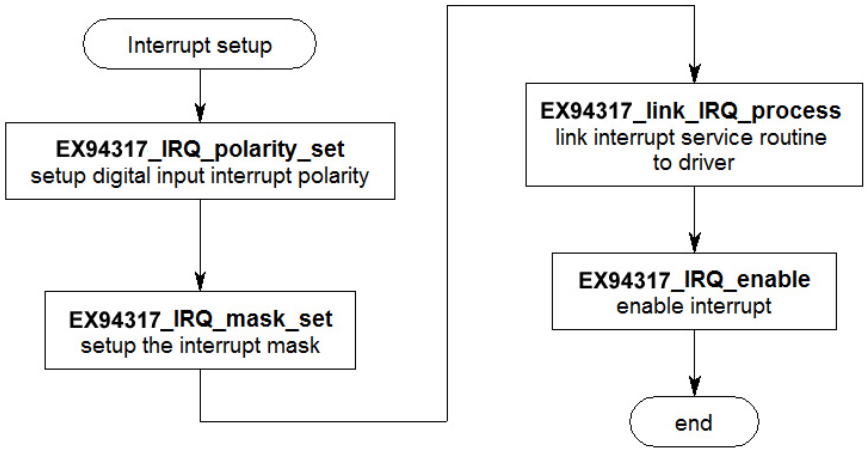
- System Fail Status Bit Valid
- Communication Loss
- Hardware not ready

These error types may indicate an internal hardware problem on the board. Error Codes contains a detailed listing of the error status returned by EX94317 functions.

7. Flow chart of application implementation

7.1 EX94317 Flow chart of application implementation





8. Function reference

8.1 Function format

Every EX94317 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 EX94317 function is the parameter **CardID** which is located the driver of EX94317 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 EX94317 function. The functions are arranged alphabetically by function name. Refer to EX94317 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	-1.797683134862315E+308 to 1.797683134862315E+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 EX94317 API. Read the following sections that apply to your programming language.

Note: Be sure to include the declaration functions of EX94317 prototypes by including the appropriate EX94317 header file in your source code. Refer to Building Applications with the EX94317 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 = EX94317_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, port;
u8 data,
u32 Status;
Status = EX94317_port_read (CardID, port, &data);
```

8.3.2 Visual basic

The file EX94317.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 *EX94317*.bas, do not use the numerical values.

In Visual Basic, you can add the entire EX94317.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 EX94317.bas file for your project in Visual Basic 4.0, go to the **File** menu and select the **Add File... option**. Select EX94317.bas, which is browsed in the EX94317 \ api directory. Then, select **Open** to add the file to the project.

To add the EX94317.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** EX94317.bas, which is in the EX94317 \ 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 EX94317BC.lib EX94317.dll

Then add the **EX94317BC.lib** to your project and add

#include "EX94317.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 = EX94317_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, port;
u8 data;
u32 Status;
Status = EX94317_port_read (CardID, port, &data);
```

8.4 EX94317 Functions

Initialization and close

- **EX94317_initial**

Format : **u32 status =EX94317_initial (void)**

Purpose: Initial the EX94317 resource when start the Windows applications**.**

- **EX94317_close**

Format : **u32 status = EX94317_close (void);**

Purpose: Release the EX94317 resource when close the Windows applications.

- **EX94317_info**

Format : **u32 status = EX94317_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	physical I/O address assigned to DIO block by OS
TC_address	u16	physical I/O address assigned to timer block by OS

I/O Port R/W

- **EX94317_config_set**

Format : u32 status =EX94317_config_set (u8 CardID, u8 configuration)

Purpose: Sets port configuration.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by Rotary SW
configuration	u8	B0: 0: port0 as input port (default) 1: port0 as output port B1: 0: port1 as input port (default) 1: port1 as output port … B5: 0: port5 as input port (default) 1: port5 as output port

- **EX94317_config_read**

Format : u32 status =EX94317_config_read (u8 CardID,u8 *configuration)

Purpose: read port configuration.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by Rotary SW

Output:

Name	Type	Description
configuration	u8	B0: 0: port0 as input port (default) 1: port0 as output port B1: 0: port1 as input port (default) 1: port1 as output port … B5: 0: port5 as input port (default) 1: port5 as output port

- **EX94317_port_set**

Format : **u32 status = EX94317_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	port number 0: port0 1: port1 … 5: port5
data	u8	bitmap of output values If port is configured as input, the data is registered and do not output. If port is configured as output, the data is registered and output.

- **EX94317_port_read**

Format : **u32 status = EX94317_port_read (u8 CardID , u8 port , u8 *data)**

Purpose: Read the register or input values of the I/O port.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by Rotary SW
port	u8	port number 0: port0 1: port1 … 5: port5

Output:

Name	Type	Description
data	u8	I/O data If port is configured as input, the data is external input data. If port is configured as output, the data is the output register data.

- **EX94317_point_set**

Format : u32 status =EX94317_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	port number 0: port0 1: port1 … 5: port5
point	u8	point number 0~7 for bit0~bit7
state	u8	state of output point If port is configured as input, the data is registered and do not output. If port is configured as output, the data is registered and output.

- **EX94317_point_read**

Format : u32 status =EX94317_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	port number 0: port0 1: port1 … 5: port5
point	u8	point number of input 0~7 for bit0~bit7

Output:

Name	Type	Description
state	u8	state of point of input If port is configured as input, the data is external input data. If port is configured as output, the data is the output register data.

- **EX94317_debounce_time_set**

Format : u32 status = EX94317_debounce_time_set (u8 CardID,u8 port ,u8 debounce_time)

Purpose: debounce the input port signal

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by Rotary SW
port	u8	port number 0: port0 1: port1 … 5: port5
debounce_time	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

Note: only valid for port configured as input

- **EX94317_debounce_time_read**

Format : u32 status = EX94317_debounce_time_read (u8 CardID,u8 port , u8 *debounce_time)

Purpose: To read back configuration of debounce mode

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by Rotary SW
port	u8	port number 0: port0 1: port1 … 5: port5

Output:

Name	Type	Description
debounce_time	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

Timer function

- **EX94317_timer_set**

Format : **u32 status = EX94317_timer_set (u8 CardID,u32 time_constant)**

Purpose: set time constant.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
time_constant	u32	time_constant based on 1us time base

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.

- **EX94317_timer_start**

Format : **u32 status = EX94317_timer_start (u8 CardID)**

Purpose: start timer function.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW

- **EX94317_timer_stop**

Format : **u32 status = EX94317_timer_stop (u8 CardID)**

Purpose: stop timer function.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW

- **EX94317_TC_set**

Format : **u32 status= EX94317_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 index = TC_CONTROL 0: stop timer operation 1: timer run For index = PRELOAD or TIMER Data is the constant to be load

Note:

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

- **EX94317_TC_read**

Format : **u32 status= EX94317_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

Interrupt function

- **EX94317_IRQ_polarity_set**

Format : u32 status = EX94317_IRQ_polarity_set (u8 CardID, u8 polarity)

Purpose: Sets the IRQ polarity of port0 (IO00~IO07)

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by Rotary SW
polarity	u8	polarity values: b7~b0 bit data =0, normal polarity (default) bit data =1, invert polarity

- **EX94317_IRQ_polarity_read**

Format : u32 status = EX94317_IRQ_polarity_read (u8 CardID , u8 *polarity)

Purpose: Read the IRQ polarity of the port0.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by Rotary SW

Output:

Name	Type	Description
polarity	u8	polarity data: b7~b0 bit data =0, normal polarity bit data =1, invert polarity

- **EX94317_IRQ_mask_set**

Format : **u32 status = EX94317_IRQ_mask_set (u8 CardID,u8 source, u8 mask)**

Purpose: Mask interrupt from port0 b7~b0 or timer

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by Rotary SW
source	u8	0: digital io block 1: timer block
mask	u8	Digital block: Any bit set to 1 of b7~b0 means port0 b7~b0 can generate interrupt Timer block: B0=1 means timer count up can generate interrupt B0=0 timer will not generate interrupt while time up

- **EX94317_IRQ_mask_read**

Format : **u32 status = EX94317_IRQ_mask_read (u8 CardID,u8 source,u8 *mask)**

Purpose: read back interrupt Mask of port0 b7~b0 or timer

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by Rotary SW
source	u8	0: digital io block 1: timer block

Output:

Name	Type	Description
mask	u8	Digital block: Any bit set to 1 of b7~b0 means port0 b7~b0 can generate interrupt Timer block: b0=1 means timer count up can generate interrupt b0=0 timer will not generate interrupt while time up

- **EX94317_IRQ_process_link**

Format : **u32 status = EX94317_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

- **EX94317_IRQ_enable**

Format : **u32 status = EX94317_IRQ_enable (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

- **EX94317_IRQ_disable**

Format : **u32 status = EX94317_IRQ_disable (u8 CardID)**

Purpose: Disable interrupt from selected source

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by Rotary SW

- **EX94317_IRQ_status_read**

Format : u32 status = EX94317_IRQ_status_read (u8 CardID,u8 source, u8 *Event_Status)

Purpose: To read back the interrupt status to identify the source

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by Rotary SW
source	U8	0: digital io block 1: timer block

Output:

Name	Type	Description
Event_Status	U8	Digital block: Any bit set to 1 of b7~b0 means port0 b7~b0 generate interrupt Timer block: B0=1 means timer count up occurred. B0=0 means timer not count up.

Note:

1. Status read back will also clear the on board status register.
2. The status will reflect the on board digital input or timer count up status are irrelevant to the IRQ_MASK

8.5 Dll list

	Function Name	Description
1	EX94317_initial( )	EX94317 Initial
2	EX94317_close( )	EX94317 Close
3	EX94317_info( )	get OS. Assigned address
4	EX94317_config_set( )	Port direction configuration
5	EX94317_config_read( )	Read back configuration
6	EX94317_port_set( )	Read Port Data (word)
7	EX94317_port_read( )	Set Output port(word)
8	EX94317_point_set ( )	Set Output Point State(bit)
9	EX94317_point_read( )	Read Input Point State(bit)
10	EX94317_debounce_time_set( )	Set input port debounce time
11	EX94317_debounce_ time_read( )	Read back input port debounce time
12	EX94317_timer_set( )	Set timer constant
13	EX94317_timer_start( )	Start timer operation
14	EX94317_timer_stop( )	Stop timer operation
15	EX94317_TC_set( )	load data to timer related registers
16	EX94317_TC_read( )	Read back the setting of timer related registers
17	EX94317_IRQ_polarity_set( )	Sets the IRQ polarity of port0
18	EX94317_IRQ_polarity__read( )	Read back the setting of IRQ polarity
19	EX94317_IRQ_mask_set( )	Mask off the IRQ
20	EX94317_IRQ_mask_read( )	Read back the mask
21	EX94317_IRQ_process_link( )	Link irq service routine
22	EX94317_IRQ_enable( )	Enable interrupt function
23	EX94317_IRQ_disable( )	Disable interrupt function
24	EX94317_IRQ_status_read( )	Read back the IRQ status

9. EX94317 Error codes table

Error Code	Symbolic Name	Description
0	DRV_NO_ERROR	No error.
1	DRV_READ_DATA_ERROR	Driver read data error
2	DRV_INIT_ERROR	Driver initial error
100	DEVICE_RW_ERROR	Device Read/Write error
101	DRV_NO_CARD	No EX94317 card on the system.
102	DRV_DUPLICATE_ID	EX94317 CardID duplicate error.
103	DRV_NOT_INSTALL	Driver not installed or bad installation
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.
305	SOURCE_ERROR	Bad source parameter
306	DEBOUNCE_MODE_ERROR	Bad debounce time parameter
406	INDEX_ERROR	Bad index parameter