EMC7432/EMC7485
Ethernet to serial converter module

Software Manual (V1.0)

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

Correction record

Version	Record
1.0	EMC74xx.dll v1.0

Contents

1. How to install the software of EMC74xx

1.1 Install the EMC driver

The ether net module can not found by OS as PCI cards. You can just install the driver without the module installed. Execute the file ..\install\EMC74xx_Install.exe to install the driver, Api and demo program automatically.

For a more detail descriptions, please refer “Step by step installation of EMC74xx”.

2. Where to find the file you need

Windows2000 and up

In Windows 2000 and up, the demo program can be setup by

If you use the default setting, a new directory .. \JS Automation\EMC74xx will generate to put the associate files.

.. / JS Automation /EMC74xx/API (header files and VB,VC lib files)

.. / JS Automation /EMC74xx /Driver (copy of driver code)

.. / JS Automation /EMC74xx /exe (demo program and source code)

The dll is located at ..\system.

3. About the EMC74xx software

EMC74xx software includes a set of dynamic link library (DLL) based on socket that you can utilize to control the interface functions.

Your EMC74xx software package includes setup driver, 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 EMC74xx functions within Windows' operation system environment.

If you only want to use as a general COM port, the virtual COM driver only will do.

3.1 What you need to get started

To set up and use your EMC74xx software, you need the following:

- EMC74xx software
- EMC74xx hardware

3.2 Software programming choices

You have several options to choose from when you are programming EMC74xx 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 EMC74xx software.

4. Language support

The EMC74xx software library is a DLL used with Windows 2000and up. You can use these DLL with any Windows integrating development environment that can call Windows DLLs.

4.1 Building applications with the EMC74xx software library

The EMC74xx function reference section contains general information about building EMC74xx applications, describes the nature of the EMC74xx functions used in building EMC74xx 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.

EMC74xx Windows Libraries

The EMC74xx for Windows function library is a DLL called **EMC74xx.dll**. Since a DLL is used, EMC74xx functions are not linked into the executable files of applications. Only the information about the EMC74xx functions in the EMC74xx 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 EMC74xx functions in EMC74xx.dll.

Header Files and Import Libraries for Different Development Environments		
Development Environment	Header File	Import Library
Microsoft C/C++	EMC74xx.h	EMC74xxVC.lib
Borland C/C++	EMC74xx.h	EMC74xxBC.lib
Microsoft Visual Basic	EMC74xx.bas	

Table 1

5. Function format and language difference

5.1 Function format

Every EMC74xx 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 EMC74xx function is the parameter **CardID** which is set by ***EMC74xx_IP_mapping*** . You can utilize multiple devices with different card ID within one application; to do so, simply pass the appropriate **CardID** to each function.

5.2 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

5.3 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 EMC74xx API. Read the following sections that apply to your programming language.

Note: Be sure to include the declaration functions of EMC74xx prototypes by including the appropriate EMC74xx header file in your source code. Refer to Chapter 4. EMC74xx Language Support for the header file appropriate to your compiler.

5.3.1 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 = EMC74xx_read_port (u8 CardID, u8 port, u8 *data);
```

where **CardID** and **port** are input parameters, and **data** is an output parameter.
To use the function in C language, consider the following example:

```
u8 CardID=0, port=0 ;   //assume CardID is 0 and port also 0
u8 data,
u32 Status;
Status = EMC74xx_read_port ( CardID, port, &data);
```

5.3.2 Visual basic

The file EMC74xx.bas contains definitions for constants required for obtaining LSI Card information and declared functions and variable as global variables. You should use these constants symbols in the EMC74xx.bas, do not use the numerical values.
In Visual Basic, you can add the entire EMC74xx.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 EMC74xx.bas file for your project in Visual Basic 4.0, go to the **File** menu and select the **Add File..**. **option**. Select EMC74xx.bas, which is browsed in the EMC74xx \ api directory. Then, select **Open** to add the file to the project.

To add the EMC74xx.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** EMC74xx.bas, which is in the EMC74xx \api directory. Then, select **Open** to add the file to the project.

If you want to use under .NET environment, please download "

5.3.3 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 EMC74xxbc.lib EMC74xx.dll
```

Then add the **EMC74xxBC.lib** to your project and add
#include “EMC74xx.h” to main program.
Now you may use the dll functions in your program. For example, the Read Input function has the following format:

```
Status = EMC74xx_read_port ( CardID, port, &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 is 0 and port also 0
u8 data,
u32 Status;
Status = EMC74xx_read_port ( CardID, port, &data);
```

* If you are using Delphi, please refer to **http://www.drbob42.com/headconv/index.htm** for more detail about the difference of C++ and Delphi.

6. Software overview and dll function

6.1 Initialization and close

You need to initialize system resource and port and IP each time you run your application,

EMC74xx_Initial( ) will do.

Once you want to close your application, call

EMC74xx_close( ) to release all the resource.

EMC74xx_Initial

Format : **u32 Status =EMC74xx_initial (u8 CardID,u8 IP_Address[4],u16 Host_Port,u16 Remote_port,u16 TimeOut_ms,u8 *CardType)**

Purpose: To map IP and PORT of an existing EMC74xx to a specified CardID number**.**

Parameters:

Input:

Name	Type	Description
CardID	u8	Assign CardID to the EMC74xx of a corresponding IP address.

IP_Address[4]	u8	4 words of IP address
		For example:
		if IP address is “192.168.0.100” then
		IP_Address[0]=192
		IP_Address[1]=168
		IP_Address[2]=0
		IP_Address[3]=100
		Default:192.168.0.100

Host_Port	u16	Assign a communicate port of host PC Default: 25122
Remote_port	u16	Assign a communicate port of EMC74xx Default: 6936
TimeOut	u16	Assign the max delay time of EMC74xx response message,1000~10000 ms.

Output:

Name	Type	Description
CardType	u8	Get the Card Type of EMC74xx 1: EMC7485 2: EMC7432

- **EMC74xx_close**

Format : **u32 Status =EMC74xx_close (u8 CardID)**

Purpose: Release the EMC74xx resource when closing the Windows applications.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMC74xx_initial function

6.2 Configuration function

To change the socket port by

EMC74xx_socket_port_change( ) and change IP by

EMC74xx_IP_change( )

Sometimes you need to reset the system (hot reset), you can commend by

EMC74xx_reboot( )

- **EMC74xx_socket_port_change**

Format : **u32 status = EMC74xx_socket_port_change (u8 CardID,u16 Remote_port);**

Purpose: To change the communicate port number of EMC74xx.

After using this function, please wait for reboot (about 10s) to validate the change.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMC74xx_initial function
Remote_port	u16	The new port number to be set Default port is: 6936

- **EMC74xx_IP_change**

Format : **u32 status = EMC74xx_IP_change (u8 CardID,u8 IP[4]);**

Purpose: To change the communicate IP of EMC74xx.

After using this function, please wait for reboot (about 10s) to validate the change.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMC74xx_initial function

IP[4]	u8	The new IP to be set
		Default IP is: 192.168.0.100
		IP_Address[0]=192
		IP_Address[1]=168
		IP_Address[2]=0
		IP_Address[3]=100

- **EMC74xx_reboot**

Format : **u32 status = EMC74xx_reboot(u8 CardID);**

Purpose: To reboot EMC74xx (about 10s).

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMC74xx_initial function

6.3 Software key function

To prevent un-authorized person to change the settings and outputs, software key is an essential protection. If you want to commend to change settings or output, you must unlock first by

EMC74xx_security_unlock( ) and read back the status of security by

EMC74xx_security_status_read( )

If you want to change password, use

EMC74xx_password_change( ) will do.

If you forget the password and you want to reset password to factory default value remotely,

EMC74xx_password_set_default( ) *1 will do.

1 Command concerning the system rebooting, please wait for about 10s to proceed the next communication.

- **EMC74xx_security_unlock**

Format : **u32 status = EMC74xx_security_unlock (u8 CardID,u8 password[8])**

Purpose: To unlock security function and enable the further operation.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMC74xx_initial function
password[8]	u8	The password previous set Default: password[8] = {'1','2','3','4','5','6','7','8'};

- **EMC74xx_security_status_read**

Format : **u32 status = EMC74xx_security_status_read(u8 CardID, u8 *lock_status);**

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

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMC74xx_initial function

Output:

Name	Type	Description
lock_status	u8	0: security unlocked 1: locked

- **EMC74xx_password_change**

Format : **u32 status = EMC74xx_password_change(u8 CardID, u8 Oldpassword[8], u8 password[8])**

Purpose: To replace old password with new password.

After using this function, please wait for reboot (about 10s) to validate the change.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMC74xx_initial function
Oldpassword [8]	u8	The previous password
password[8]	u8	The new password to be set

- **EMC74xx_password_set_default**

Format : **u32 status = EMC74xx_password_set_default (u8 CardID)**

Purpose: Set password to default.

After using this function, please wait for reboot (about 10s) to validate the change.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMC74xx_initial function default :password[8] = {'1','2','3','4','5','6','7','8'};

6.4 TTL I/O

The Ethernet to serial converter module provides extra 8-bit TTL I/O for compact integration of various applications need to control on/off or detect external signals.

First of all, you must setup each TTL pin as input or output.

EMC74xx_port_config_set( ) will do.

EMC74xx_port_config_read( ) to read back for verification.

EMC74xx_port_set( ) to set the output data;

EMC74xx_port_read( ) to read the input status.

- **EMC74xx_port_config_set**

Format : **u32 status = EMC74xx_port_config_set(u8 CardID, u8 TTL_config);**

Purpose: To setup the TTL configuration.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMC74xx_initial function
TTL_config	u8	Configure the TTL's I/O as input or output bit0: 0: TTL0 as output 1: TTL0 as input … bit7: 0: TTL7 as output 1: TTL7 as input

- **EMC74xx_port_config_read**

Format : **u32 status = EMC74xx_port_config_read(u8 CardID, u8 &TTL_config);**

Purpose: To read the TTL configuration.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMC74xx_initial function

Output:

Name	Type	Description
TTL_config	u8	bit0: 0: TTL0 as output 1: TTL0 as input … bit7: 0: TTL7 as output 1: TTL7 as input

- **EMC74xx_port_set**

Format : **u32 status = EMC74xx_port_set(u8 CardID,u8 port);**

Purpose: To set the TTL output value.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMC74xx_initial function
port	u8	Set the TTL output value. bit0: TTL0 value … bit7: TTL7 value

- **EMC74xx_port_read**

Format : **u32 status = EMC74xx_port_read(u8 CardID,u8 &port);**

Purpose: To read all the TTL port value.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMC74xx_initial function

Output:

Name	Type	Description
port	u8	Read the TTL port value. bit0: TTL0 value … bit7: TTL7 value

6.5 Counter function

TTL inputs can be used as low frequency counter (less than 200 pulses per second), you can mask off the counter function on unwanted inputs by:

EMC74xx_counter_mask_set( ), then enable or disable the counter function:

EMC74xx_counter_enable( ) to enable counter function;

EMC74xx_counter_disable( ) to disable counter function.

The counter can be write or clear by using:

EMC74xx_counter_read( ) to read counter on the fly;

EMC74xx_counter_clear( ) to clear counter data.

- **EMC74xx_counter_mask_set**

Format : **u32 status = EMC74xx_counter_mask_set(u8 CardID,u8 channel);**

Purpose: To set the counter channel mask.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMC74xx_initial function
Channel	u8	b7: 0: TTL7 counter disable 1: TTL7 counter enable … b0: 0: TTL0 counter disable 1: TTL0 counter enable

- **EMC74xx_counter_enable**

Format : **u32 status = EMC74xx_counter_enable(u8 CardID);**

Purpose: To enable the counter function.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMC74xx_initial function

- **EMC74xx_counter_disable**

Format : **u32 status = EMC74xx_counter_disable(u8 CardID);**

Purpose: To disable the counter function.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMC74xx_initial function

- **EMC74xx_counter_read**

Format : **u32 status = EMC74xx_counter_read(u8 CardID,u32 counter[8]);**

Purpose: To read all the counter value.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMC74xx_initial function

Output:

Name	Type	Description
counter	u32	counter value counter[0] for TTL0 … counter[7] for TTL7

- **EMC74xx_counter_clear**

Format : **u32 status = EMC74xx_counter_clear(u8 CardID,u8 channel);**

Purpose: To reset the counter value.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMC74xx_initial function
Channel	u8	b7: 0: no function 1: clear TTL7 counter … b0: 0: no function 1: clear TTL0 counter

6.6 RS232/422/485 setup

As a serial to Ethernet converter, we must setup the serial port parameters of the module to meet the communication protocol.

EMC74xx_serial_port_set( ) is used for parameters setting and

EMC74xx_serial_port_read( ) is used to read back for verification.

- **EMC74xx_serial_port_set**

Format : **u32 status = EMC74xx_serial_port_set(u8 CardID, u8 baud_rate, u8 data_bit, u8 parity, u8 stop_bits, u8 flow_control, u8 mode);**

Purpose: To set the serial port configuration.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMC74xx_initial function
baud_rate	u8	Set the baud rate. 0: 1200 1: 2400 2: 4800 3: 9600(default) 4: 19200 5: 38400 6: 57600 7: 115200 8: 921600
data_bit	u8	Communication data bit setting 0: 5 bits 1: 6 bits 2: 7 bits 3: 8 bits (default)
parity	u8	Communication parity setting 0:Odd 1:Even 2:None(default)
stop_bits	u8	Communication stop bit setting 0: 1 bit (default) 1: 1.5 bit
flow_control	u8	Flow control setting 0:Xon/Xoff 1: Hardware(default) 2:None
mode	u8	Mode setting for EMC7485 set as 1: RS422 (default) 2: RS485 for EMC7432 this parameter is of no use.

- **EMC74xx_serial_port_read**

Format : **u32 status = EMC74xx_serial_port_read(u8 CardID, u8 &baud_rate, u8 &data_bit, u8 &parity, u8 &stop_bits, u8 &flow_control, u8 &mode);**

Purpose: To read the serial port configuration.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMC74xx_initial function

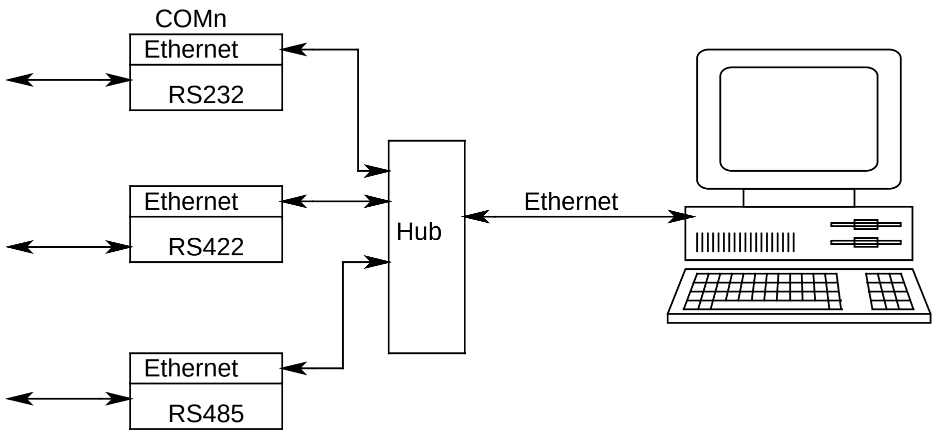
Output:

Name	Type	Description
baud_rate	u8	Set the baud rate. 0: 1200 1: 2400 2: 4800 3: 9600(default) 6: 57600 5: 38400 4: 19200 7: 115200 8: 921600
data_bit	u8	Communication data bit setting 0: 5 bits 1: 6 bits 2: 7 bits 3: 8 bits
parity	u8	Communication data bit setting 0:Odd 1:Even 2:None(default)
stop_bits	u8	Communication stop bit setting 0: 1 bit (default) 1: 1.5 bit
flow_control	u8	Flow control setting 0: Xon/Xoff 1: Hardware(default) 2: None
mode	u8	Mode setting for EMC7485 set as 1: RS422 (default) 2: RS485 for EMC7432 this parameter is of no use.

6.7 Virtual COM port

The serial to Ethernet module is in fact use Ethernet to connect to computer but from the user side, we would rather take it as a COM port for the existing program or the traditional COM port programmers.

As the followings shown, the virtual COM port will take the RS232 (RS422 or RS485) as the computer inside COM port but it really connect to the serial port via Ethernet.



Using

EMC74xx_VSPM_install( ) to add a virtual COM port to the system,

EMC74xx_VSPM_remove( ) to remove the virtual COM port and release resource.

EMC74xx_VSPM_set( ) to setup the Ethernet IP of the converter module to the virtual COM port.

EMC74xx_VSPM_connect( ) connect the virtual COM port (logic device) to the converter module (physical device).

EMC74xx_VSPM_info( ) to read the Ethernet IP of the converter module from the virtual COM port.

EMC74xx_VSPM_close( ) to close the connection.

- **EMC74xx_VSPM_install**

Format : **u32 Status = EMC74xx_VSPM_install(u8 *vID);**

Purpose: To add a virtual com port module.

Parameters:

Output:

Name	Type	Description
vID	u8	Return the number vID virtual com port module.

- **EMC74xx_VSPM_remove**

Format : **u32 Status =EMC74xx_VSPM_remove(u8 vID);**

Purpose: To remove a virtual com port module.

Parameters:

Input:

Name	Type	Description
vID	u8	vID assigned by EMC74xx_VSPM_install function.

- **EMC74xx_VSPM_set**

Format : **u32 Status = EMC74xx_VSPM_set(u8 vID,u8 ip[4]);**

Purpose: To set the virtual COM port IP.

Parameters:

Input:

Name	Type	Description
vID	u8	vID assigned by EMC74xx_VSPM_install function.

ip[4]	u8	IP of virtual COM device (the converter module)
		Default IP is: 192.168.0.100
		IP[0]=192
		IP[1]=168
		IP[2]=0
		IP [3]=100

- **EMC74xx_VSPM_connect**

Format : **u32 Status = EMC74xx_VSPM_connect(u8 vID);**

Purpose: The virtual COM (logic device) connect to the remote IP (converter module)

Parameters:

Input:

Name	Type	Description
vID	u8	vID assigned by EMC74xx_VSPM_install function.

Note: use EMC74xx_VSPM_set to set the IP first.

- **EMC74xx_VSPM_info**

Format : **u32 Status = EMC74xx_VSPM_info(u8 vID, u8 *status, u8 remote_IP[4]);**

Purpose: Get the virtual COM information.

Parameters:

Input:

Name	Type	Description
vID	u8	vID assigned by EMC74xx_VSPM_install function.

Output:

Name	Type	Description
status	u8	Return the virtual device status 0:idle 1:connect

remote_IP[4]	u8	IP of virtual COM port (converter module)
		Default IP is: 192.168.0.100
		IP[0]=192
		IP[1]=168
		IP[2]=0
		IP [3]=100

- **EMC74xx_VSPM_close**

Format : **u32 Status = EMC74xx_VSPM_close(u8 vID);**

Purpose: To close the virtual COM (logic device) connection.

Parameters:

Input:

Name	Type	Description
vID	u8	vID assigned by EMC74xx_VSPM_install function.

7. DLL list

	Function Name	Description
1.	EMC74xx_Initial ( )	Map IP and get model parameter
2.	EMC74xx_close( )	EMC74xx close
3.	EMC74xx_socket_port_change ( )	To change the communicate port number of EMC74xx
4.	EMC74xx_IP_change ( )	To change the communicate IP of EMC74xx
5.	EMC74xx_reboot ( )	To reboot EMC74xx
6.	EMC74xx_security_unlock ( )	Unlock security
7.	EMC74xx_security_status_read ( )	Read lock status
8.	EMC74xx_password_change ( )	Change password
9.	EMC74xx_password_set_default ( )	Rest to factory default password
10.	EMC74xx_port_config_set( )	To setup the TTL configuration
11.	EMC74xx_port_config_read( )	To read the TTL configuration
12.	EMC74xx_port_set( )	To set the TTL outport value
13.	EMC74xx_port_read( )	To read all the TTL port value
14.	EMC74xx_counter_mask_set( )	To set the counter channel mask
15.	EMC74xx_counter_enable( )	To enable the counter function
16.	EMC74xx_counter_disable( )	To disable the counter function
17.	EMC74xx_counter_read( )	To read all the counter value
18.	EMC74xx_counter_clear( )	To reset the counter value
19.	EMC74xx_serial_port_set( )	To set the serial port configuration
20.	EMC74xx_serial_port_read( )	To read the serial port configuration
21.	EMC74xx_VSPM_install( )	To add a virtual com port module
22.	EMC74xx_VSPM_remove( )	To remove a virtual com port module
23.	EMC74xx_VSPM_set( )	To set the virtual COM port IP
24.	EMC74xx_VSPM_connect( )	The virtual COM connect to the remote IP
25.	EMC74xx_VSPM_info( )	Get the virtual COM information
26.	EMC74xx_VSPM_close( )	To close the virtual COM connection

8. EMC74xx Error code table

Error Code	Symbolic Name	Description
0	JSDRV_NO_ERROR	No error.
1	INITIAL_SOCKET_ERROR	Sock can not initialized, maybe Ethernet hardware problem
2	IP_ADDRESS_ERROR	IP address is not acceptable
3	UNLOCK_ERROR	Unlock fail
4	LOCK_COUNTER_ERROR	Unlock error too many times
5	SET_SECURITY_ERROR	Fail to set security
100	DEVICE_RW_ERROR	Can not reach module
101	NO_CARD	Can not reach module
102	DUPLICATE_ID	CardID is already used
300	ID_ERROR	CardID is not acceptable
301	PORT_ERROR	Port parameter unacceptable or unreachable
305	PARAMETERS_ERROR	Parameters error
306	CHANGE_SOCKET_ERROR	Can not change socket
307	UNLOCK_SECURITY_ERROR	Fail to unlock security
308	PASSWORD_ERROR	Password mismatched
309	REBOOT_ERROR	Can not reboot
310	TIME_OUT_ERROR	Too long to response
311	CREATE_SOCKET_ERROR	Socket can not create
312	CHANGEIP_ERROR	Error while change IP
313	COUNTER_MASK_SET_ERROR	Count mask error
314	COUNTER_ENABLE_ERROR	Can not enable counter function
315	COUNTER_DISABLE_ERROR	Can not disable counter function
316	COUNTER_READ_ERROR	Can not read counter data
317	COUNTER_CLEAR_ERROR	Can not clear counter data
318	PORT_CONFIG_SET_ERROR	Can not setup port configuration
319	PORT_CONFIG_READ_ERROR	Can not read port configuration
320	PORT_SET_ERROR	Can not set port data
321	PORT_READ_ERROR	Can not read port data
322	SERIAL_PORT_SET_ERROR	Can not set serial port configuration
323	SERIAL_PORT_READ_ERROR	Can not read serial port configuration
330	VSPM_REMOTE_ERROR	Can not talk with physical device
331	VSPM_CONNECT_ERROR	Can not connect virtual COM

332	VSPM_CLOSE_ERROR	
333	VSPM_INFO_ERROR	Can not get virtual COM info
334	VSPM_NOT_FOUND	Can not find virtual COM