EMD-8204/08
Ethernet Digital I/O module

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
1.0	EMD-820x.dll v1.0

Contents

1. How to install the software of EMD820x

Please register as user's club member to download the
"Step by step installation of EMD820x" document from http://automation.com.tw

1.1 Install the EMD 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\EMD820x_Install.exe to install the driver, Api and demo program automatically.
For a more detail descriptions, please refer "Step by step installation of EMD820x".

2. Where to find the file you need

Windows2000, XP and up

In Windows 2000,XP,Vista system, the demo program can be setup by EMD820x_Install.exe.

If you use the default setting, a new directory .. \JS Automation\EMD820x will generate to put the associate files.

.. / JS Automation /EMD820x/API (header files and VB,VC lib files)

.. / JS Automation /EMD820x/Driver (copy of driver code)

.. / JS Automation /EMD820x/exe (demo program and source code)

The dll is located at ..\system.

3. About the EMD820x software

EMD820x software includes a set of dynamic link library (DLL) based on socket that you can utilize to control the interface functions.

Your EMD820x 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 EMD820x functions within Windows' operation system environment.

3.1 What you need to get started

To set up and use your EMD820x software, you need the following:

- EMD820x software
- EMD820x hardware

3.2 Software programming choices

You have several options to choose from when you are programming EMD820x 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 EMD820x software.

4. EMD820x Language support

The EMD820x 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 EMD820x software library

The EMD820x function reference section contains general information about building EMD820x applications, describes the nature of the EMD820x functions used in building EMD820x 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.

EMD820x Windows Libraries

The EMD820x for Windows function library is a DLL called **EMD820x.dll**. Since a DLL is used, EMD820x functions are not linked into the executable files of applications. Only the information about the EMD820x functions in the EMD820x 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 EMD820x functions in EMD820x .dll.

Header Files and Import Libraries for Different Development Environments		
Development Environment	Header File	Import Library
Microsoft C/C++	EMD820x.h	EMD820xVC.lib
Borland C/C++	EMD820x.h	EMD820xBC.lib
Microsoft Visual Basic	EMD820x.bas	

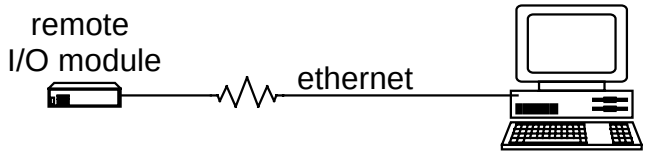
Table 1

5. Basic concept of the remote digital I/O module

I/O communicate via ethernet

The remote digital I/O is the function extension of the card type digital I/O. If the under control target is at a long distance away, the card type is limited by the wiring, it is very difficult to go far away but the ether net remote digital I/O will do.

JS automation keeps the remote digital I/O function as close to the card type digital I/O as possible. Users can port their application from card type to remote or from remote to card at the shortest working time.



The module response or be commanded by the controller through Ethernet by UDP protocol. As a member on the Ethernet, it must have a distinguished IP and a predefined port. At factory, we set the default IP at 192.168.0.100 and the port at 6936 for the remote module.

If you want to control the module through internet, you must configure your network to pass the message to the module, say, your gateway allows the message from outside to go to the module and also the message from the module can go out to the internet. Please check with your internet engineer to set up the environment.

6. Function format and language difference

6.1 Error codes and address

Every EMD820x 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 EMD820x function is the parameter **CardID** which is set by ***EMD820x_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.

[*1] Command concerning the system rebooting, please wait for about 10s to precede the next communication.

6.2 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

6.3 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 EMD820x API. Read the following sections that apply to your programming language.

Note: Be sure to include the declaration functions of EMD820x prototypes by including the appropriate EMD820x header file in your source code. Refer to Chapter 4. EMD820x Language Support for the header file appropriate to your compiler.

6.3.1 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 = EMD820x_port_read (u32 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:

```
u32 CardID=0, port=0 ;    //assume CardID is 0 and port also 0
u8 data,
u32 Status;
Status = EMD820x_port_read ( CardID, port, &data);
```

6.3.2 Visual basic

The file EMD820x.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 EMD820x.bas, do not use the numerical values.
In Visual Basic, you can add the entire EMD820x.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 EMD820x.bas file for your project in Visual Basic 4.0, go to the **File** menu and select the **Add File...** **option**. Select EMD820x.bas, which is browsed in the EMD820x \ api directory. Then, select **Open** to add the file to the project.

To add the EMD820x.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** EMD820x.bas, which is in the EMD820x \api directory. Then, select **Open** to add the file to the project.

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

6.3.3 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 EMD820xbc.lib EMD820x.dll

Then add the **EMD820xbc.lib** to your project and add

#include “EMD820x.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 = EMD820x_port_read ( CardID, port, &data);
```

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

```
u32 CardID=0, port=0 ;    //assume CardID is 0 and port also 0
u8 data,
u32 Status;
Status = EMD820x_port_read ( 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.

7. Software overview and dll function

These topics describe the features and functionality of the EMD820x module and the detail of the dll function.

7.1 Initialization and close

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

EMD820x_initial( ) will do.

Once you want to close your application, call

EMD820x_close( ) to release all the resource.

To check the firmware version,

EMD820x_firmware_version_read( ) will do.

- **EMD820x_initial**

Format : u32 status =EMD820x_initial (u32 CardID,u8 IP_Address[4] , u16 Host_Port,u16 Remote_port,u16 TimeOut, u8 *CardType)

Purpose: To map IP and PORT of an existing EMD820x to a specified CardID number**.**

Parameters:

Input:

Name	Type	Description
CardID	u32	0~1999 Assign CardID to the EMD820x of a corresponding IP address.
IP_Address[4]	u8	4 words of IP address, Default:192.168.0.100 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
Host_Port	u16	Assign a communicate port of host PC Default:15120
Remote_port	u16	Assign a communicate port of EMD820x Default:6936
TimeOut	u16	Assign the max delay time of EMD820x response message,1000~10000 ms.

Output:

Name	Type	Description
CardType	u8	Get the Card Type of EMD820x 0: 4 in / 4 out (EMD8204) 1: 8 in / 8 out (EMD8208)

- **EMD820x_close**

Format : **u32 status =EMD820x_close (u32 CardID)**

Purpose: Release the EMD820x resource when closing the Windows applications.

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMD820x_IP_mapping

- **EMD820x_firmware_version_read**

Format : **u32 status =EMD820x_firmware_version_read(u32 CardID, u8 Version[2])**

Purpose: Read the firmware version.

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMD820x_initial

Output:

Name	Type	Description
Version[2]	u8	the firmware version x.y x = Version[1] y = Version[0]

7.2 Input/Output function

Input and output polarity setting can give you the logic polarity as you need. Say, you use the positive logic in your application program and the input maybe short to ground as active, change the polarity to take the short to ground (active) input to be read as logic ‘1’.

To set the polarity setting by

EMD820x_port_polarity_set( )

To read back the input and output polarity setting by

EMD820x_port_polarity_read( )

To control the output, use

EMD820x_port_set( )

To read input or output register status use

EMD820x_port_read( )

To set the point setting by

EMD820x_point_polarity_set( )

To read back the point of input and output polarity setting by

EMD820x_point_polarity_read( )

To control a point of output, use

EMD820x_point_set( )

To read a point data of input or output register, use

EMD820x_point_read( )

- **EMD820x_port_polarity_set**

Format : **u32 status =EMD820x_port_polarity_set(u32 CardID, u8 port, u8 state)**

Purpose: Set polarity of input port or output port.

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMD820x_IP_mapping
port	u8	port number 1: INPORT 0: OUTPORT
state	u8	state of designated polarity

- **EMD820x_port_polarity_read**

Format : **u32 status =EMD820x_port_polarity_read(u32 CardID, u8 port, u8 *data)**

Purpose: Read polarity of input port or output port.

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMD820x_IP_mapping
port	u8	port number 1: INPORT 0: OUTPORT

Output:

Name	Type	Description
data	u8	state of designated polarity

- **EMD820x_port_set**

Format : **u32 status = EMD820x_port_set (u32 CardID , u8 port , u8 data)**

Purpose: Set the output port data.

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMD820x_IP_mapping
port	u8	port number 0: OUTPORT, output port bit7~bit0
data	u8	b7 ~ b0, bitmap of output port values b7: OUT7 for output … b0: OUT0 for output

- **EMD820x_port_read**

Format : **u32 status = EMD820x_port_read(u32 CardID , u8 port , u8 *data)**

Purpose: Read back the data of the I/O port.

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMD820x_IP_mapping
port	u8	port number 1: INPORT, input port bit7~bit0 0: OUTPORT, output port bit7~bit0

Output:

Name	Type	Description
data	u8	b7 ~ b0, I/O port data b7: IN7 for input, OUT7 for output … b0: IN0 for input, OUT0 for output

- **EMD820x_point_polarity_set**

Format : **u32 status =EMD820x_point_polarity_set(u32 CardID, u8 port, u8 point, u8 state)**

Purpose: Set polarity of input point or output point.

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMD820x_IP_mapping
port	u8	port number 1: INPORT 0: OUTPORT
point	u8	point number of inport or outport 0~7 for IN0 ~ IN7 or OUT0 ~ OUT7
state	u8	state of designated polarity

- **EMD820x_point_polarity_read**

Format : **u32 status =EMD820x_point_polarity_read(u32 CardID, u8 port, u8 point, u8 *data)**

Purpose: Read polarity of input point or output point.

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMD820x_IP_mapping
Port	u8	port number 1: INPORT 0: OUTPORT
point	u8	point number of inport or outport 0~7 for IN0 ~ IN7 or OUT0 ~ OUT7

Output:

Name	Type	Description
Data	u8	state of designated polarity

- **EMD820x_point_set**

Format : **u32 status =EMD820x_point_set(u32 CardID, u8 port, u8 point, u8 state)**

Purpose: Set bit status of output point

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMD820x_IP_mapping
port	u8	port number 0: OUTPORT
point	u8	Output point number 0~7 for OUT0~OUT7
state	u8	state of outport

- **EMD820x_point_read**

Format : **u32 status =EMD820x_point_read(u32 CardID, u8 port, u8 point, u8 *state)**

Purpose: Read bit state of input point .

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMD820x_IP_mapping
port	u8	port number 1: INPORT 0: OUTPORT
point	u8	point number of inport or outport 0~7 for IN0 ~ IN7 or OUT0 ~ OUT7

Output:

Name	Type	Description
state	u8	state of designated point

7.3 Counter function

You can use the digital input as a low speed counter (no more than 200pps). First you can set which input channel you will want to work as counter by:

EMD820x_counter_mask_set( ) then enable the function by

EMD820x_counter_enable( ) and any time to stop by

EMD820x_counter_disable( ).

To read the counter value by

EMD820x_counter_read( ) and use

EMD820x_counter_clear( ) to clear counter.

If the digital input speed is faster than 200 pps，the counter might be error.

- **EMD820x_counter_mask_set**

Format : u32 status = EMD820x_counter_mask_set(u32 CardID, u8 port,u8 channel);

Purpose: To set the counter channel mask.

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMD820x_IP_mapping
port	u8	unused
Channel	u8	b7 ~ b0, b7: 0: IN07 counter disable 1: IN07 counter enable … b0: 0: IN00 counter disable 1: IN00 counter enable

- **EMD820x_counter_enable**

Format : u32 status = EMD820x_counter_enable(u32 CardID);

Purpose: To enable the counter.

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMD820x_IP_mapping

- **EMD820x_counter_disable**

Format : **u32 status = EMD820x_counter_disable(u32 CardID);**

Purpose: To disable the counter.

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMD820x_IP_mapping

- **EMD820x_counter_read**

Format : **u32 status = EMD820x_counter_read(u32 CardID, u8 port,u32 counter[8]);**

Purpose: To read all the counter value.

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMD820x_IP_mapping
port	u8	unused

Output:

Name	Type	Description
counter	u32	counter value counter[0] for IN00 … counter[7] for IN07

- **EMD820x_counter_clear**

Format : **u32 status = EMD820x_counter_clear (u32 CardID, u8 port,u8 channel);**

Purpose: To reset the counter value.

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMD820x_IP_mapping
port	u8	unused
Channel	u8	b7~b0, b7: 0: no function 1: clear IN07 counter … b0: 0: no function 1: clear IN00 counter

7.4 Miscellaneous function

To change the communication port as you need by:

EMD820x_change_socket_port( ) [*1]

To change IP,

EMD820x_change_IP( ) [*1]

To reboot EMD820x module for module alarm or to validate the system configuration change by:

EMD820x_reboot( ) [*1]

- **EMD820x_change_socket_port**

Format : **u32 status = EMD820x_change_socket_port(u32 CardID,u16 Remote_port);**

Purpose: To change the communicate port number of EMD820x.

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

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMD820x_IP_mapping
Remote_port	u16	The new port number to be set

- **EMD820x_change_IP**

Format : **u32 status = EMD820x_change_IP(u32 CardID, u8 IP[4]);**

Purpose: To change the communicate IP of EMD820x.

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

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMD820x_IP_mapping
IP[4]	u8	The new IP to be set

- **EMD820x_reboot**

Format : **u32 status = EMD820x_reboot(u32 CardID);**

Purpose: To reboot EMD820x (about 10s).

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMD820x_IP_mapping

7.5 Software key function

Software key is used to protect the modification of IO state and system configuration by un-authorized person.

To operate the EMD820x, you must unlock the module first by

EMD820x_security_unlock( )

To verify the lock status by

EMD820x_security_status_read( )

You can change password for your convenience by

EMD820x_password_change( )

If you forget the password you set, you can recover the factory default password by:

EMD820x_password_set_default( ) *[*1]*

- **EMD820x_security_unlock**

Format : u32 status = EMD820x_security_unlock(u32 CardID,u8 password[8])

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

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMD820x_IP_mapping
password[8]	u8	The password previous set Use a-z, A-Z, 0-9 characters. For example: u8 password[8] = {'1','2','3','4','5','6','7','8'}; u8 password[8] = {'1','2','3','a', 'A',NULL,NULL,NULL}; default : password[8] = {'1','2','3','4','5','6','7','8'};

- **EMD820x_security_status_read**

Format : **u32 status = EMD820x_security_status_read(u32 CardID,u8 *lock_status);**

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

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMD820x_IP_mapping

Output:

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

- **EMD820x_password_change**

Format : **u32 status = EMD820x_password_change(u32 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	u32	CardID assigned by EMD820x_IP_mapping
Oldpassword [8]	u8	The previous password
password[8]	u8	The new password to be set

- **EMD820x_password_set_default**

Format : **u32 status = EMD820x_password_set_default(u32 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	u32	CardID assigned by EMD820x_IP_mapping default : password[8] = {'1','2','3','4','5','6','7','8'};

7.6 DLL list

	Function Name	Description
1.	EMD820x_Initial( )	Assign IP and get model parameter
2.	EMD820x_close( )	EMD820x close
3.	EMD820x_firmware_version_read( )	Read the firmware version
4.	EMD8216_port_polarity_set( )	Set polarity of input port or output port
5.	EMD8216_port_polarity_read( )	Read polarity of input port or output port.
6.	EMD820x_port_set ( )	Set the output port data.
7.	EMD820x_port_read( )	Read back the data of the I/O port.
8.	EMD820x_point_polarity_set( )	Set polarity of input port or output port.
9.	EMD820x_point_polarity_read( )	Read polarity of input port or output port
10.	EMD820x_point_set( )	Set bit status of output port
11.	EMD820x_point_read( )	Read bit state of input port.
12.	EMD820x_counter_mask_set( )	Set counter channel mask
13.	EMD820x_counter_enable( )	Enable counter function
14.	EMD820x_counter_disable( )	Disable counter function
15.	EMD820x_counter_read( )	Read counter value
16.	EMD820x_counter_clear( )	Clear designated counter
17.	EMD820x_change_socket_port( )	change the communication port
18.	EMD820x_change_IP( )	change the IP of EMD820x
19.	EMD820x_reboot( )	reboot EMD820x module
20.	EMD820x_security_unlock( )	Unlock security
21.	EMD820x_security_status_read( )	Read lock status
22.	EMD820x_password_change( )	Change password
23.	EMD820x_password_set_default( )	Rest to factory default password

8. EMD820x Error codes summary

8.1 EMD820x Error codes 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 already used
300	ID_ERROR	CardID is not acceptable
301	PORT_ERROR	Port parameter unacceptable or unreachable
302	IN_POINT_ERROR	Input point unreachable
303	OUT_POINT_ERROR	Output point unreachable
305	PARAMETERS_ERROR	Parameter 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	CREAT_SOCKET_ERROR	Socket can not create