EMD7201
Ethernet Digital I/O module

Software Manual (V1.2)

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	EMD7201.dll v1.0
1.1	Add counter function, change function name
1.2	change explanation of channel parameter

Contents

1. How to install the software of EMD7201

Please register as user’s club member to download the
“Step by step installation of EMD7201” 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\EMD7201_Install.exe to install the driver, Api and demo program automatically.
For a more detail descriptions, please refer “Step by step installation of EMD7201”.

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

If you use the default setting, a new directory .. \JS Automation\EMD7201 will generate to put the associate files.

.. / JS Automation /EMD7201/API (header files and VB,VC lib files)

.. / JS Automation /EMD7201/Driver (copy of driver code)

.. / JS Automation /EMD7201/exe (demo program and source code)

The dll is located at ..\system.

3. About the EMD7201 software

EMD7201 software includes a set of dynamic link library (DLL) based on socket that you can utilize to control the interface functions.

Your EMD7201 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 EMD7201 functions within Windows' operation system environment.

3.1 What you need to get started

To set up and use your EMD7201 software, you need the following:

- EMD7201 software
- EMD7201 hardware

3.2 Software programming choices

You have several options to choose from when you are programming EMD7201 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 EMD7201 software.

4. EMD7201 Language support

The EMD7201 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 EMD7201 software library

The EMD7201 function reference section contains general information about building EMD7201 applications, describes the nature of the EMD7201 functions used in building EMD7201 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.

EMD7201 Windows Libraries

The EMD7201 for Windows function library is a DLL called **EMD7201.dll**. Since a DLL is used, EMD7201 functions are not linked into the executable files of applications. Only the information about the EMD7201 functions in the EMD7201 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 EMD7201 functions in EMD7201 .dll.

Header Files and Import Libraries for Different Development Environments		
Development Environment	Header File	Import Library
Microsoft C/C++	EMD7201.h	EMD7201VC.lib
Borland C/C++	EMD7201.h	EMD7201BC.lib
Microsoft Visual Basic	EMD7201.bas	

Table 1

5. Software overview

5.1 Initialization and close

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

EMD7201_IP_mapping( ) will do.

Once you want to close your application, call

EMD7201_close( ) to release all the resource.

5.2 Input/Output function

To read input or output register status use

EMD7201_port_read( )

To control the output, use

EMD7201_port_set( )

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

EMD7201_point_read ( )

To control a point of output, use

EMD7201_point_set ( )

5.3 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:

EMD7201_counter_mask_set( ) then enable the function by

EMD7201_counter_enable( ) and any time to stop by

EMD7201_counter_disable( ).

To read the counter value by

EMD7201_counter_read( ) and use

EMD7201_counter_clear( ) to clear counter.

5.4 Miscellaneous function

To change the communication port as you need by:

EMD7201_change_socket_port( )^{*1}

To change IP,

EMD7201_change_IP( )^{*1}

To reboot EMD7201 module for module alarm or to validate the system configuration change by:

EMD7201_reboot( )^{*1}

5.5 Software key function

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

To operate the EMD7201, you must unlock the module first by

EMD7201_security_unlock( )

To verify the lock status by

EMD7201_security_status_read( )

You can change password for your convienence by

EMD7201_password_change( )

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

EMD7201_password_set_default( ) *[*1]*

5.6 Error codes and address

Every EMD7201 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 EMD7201 function is the parameter **CardID** which is set by ***EMD7201_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 proceed the next communication.

5.7 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

5.8 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 EMD7201 API. Read the following sections that apply to your programming language.

Note: Be sure to include the declaration functions of EMD7201 prototypes by including the appropriate EMD7201 header file in your source code. Refer to Chapter 4. EMD7201 Language Support for the header file appropriate to your compiler.

5.8.1 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 = EMD7201_port_read (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 = EMD7201_port_read ( CardID, port, &data);
```

5.8.2 Visual basic

The file EMD7201.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 EMD7201.bas, do not use the numerical values.
In Visual Basic, you can add the entire EMD7201.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 EMD7201.bas file for your project in Visual Basic 4.0, go to the **File** menu and select the **Add File..**. **option**. Select EMD7201.bas, which is browsed in the EMD7201 \ api directory. Then, select **Open** to add the file to the project.

To add the EMD7201.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** EMD7201.bas, which is in the EMD7201 \api directory. Then, select **Open** to add the file to the project.

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

5.8.3 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 EMD7201bc.lib EMD7201.dll
```

Then add the **EMD7201bc.lib** to your project and add
#include “EMD7201.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 = EMD7201_port_read ( 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 = EMD7201_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.

5.9 EMD7201 Functions

Initialization and close

- **EMD7201_IP_mapping**

Format : **u32 status =EMD7201_IP_mapping (u8 CardID,u8 IP_Address[4] , u16 Host_Port,u16 Remote_port,u16 TimeOut, u8 *CardType)**

Purpose: To map IP and PORT of an existing EMD7201 to a specified CardID number**.**

Parameters:

Input:

Name	Type	Description
CardID	u8	0~255 Assign CardID to the EMD7201 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 EMD7201 Default:6936
TimeOut	u16	Assign the max delay time of EMD7201 response message,1000~10000 ms.

Output:

Name	Type	Description
CardType	u8	Get the Card Type of EMD7201 1: 4 in / 4 out (EMD7201-4) 2: 8 in / 8 out (EMD7201-8) 3: TTL 8 in / 8 out (EMD7201-T)

- **EMD7201_close**

Format : **u32 status =EMD7201_close (u8 CardID)**

Purpose: Release the EMD7201 resource when closing the Windows applications.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMD7201_IP_mapping

Input/output function

- **EMD7201_port_read**

Format : **u32 status = EMD7201_port_read(u8 CardID , u8 port , u8 *data)**

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

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMD7201_IP_mapping
port	u8	port number 0: INPORT, input port bit7~bit0 1: OUTPORT, output port bit7~bit0

Output:

Name	Type	Description
data	u8	b7 ~ b0, I/O port data b7: IN07 for input, OUT07 for output … b0: IN00 for input, OUT00 for output

- **EMD7201_port_set**

Format : **u32 status = EMD7201_port_set (u8 CardID, u8 data)**

Purpose: Set the output port data.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMD7201_IP_mapping
data	u8	b7 ~ b0, bitmap of output port values b7: OUT07 for output … b0: OUT00 for output

- **EMD7201_point_read**

Format : **u32 status =EMD7201_point_read(u8 CardID, u8 port, u8 point, u8 *state)**

Purpose: Read bit state of input port.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMD7201_IP_mapping
port	u8	port number 0: INPORT 1: OUTPORT
point	u8	point number of inport 0~7 for IN00~IN07

Output:

Name	Type	Description
state	u8	state of designated point

- **EMD7201_point_set**

Format : **u32 status =EMD7201_point_set(u8 CardID, u8 point, u8 state)**

Purpose: Set bit status of output port

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMD7201_IP_mapping
point	u8	Output point number 0~7 for OUT00~OUT07
state	u8	state of outport

Counter function

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

- ### EMD7201_counter_mask_set

Format : **u32 status = EMD7201_counter_mask_set(u8 CardID,u8 channel);**

Purpose: To set the counter channel mask.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMD7201_IP_mapping
Channel	u8	b7 ~ b0, b7: 0: IN07 counter disable 1: IN07 counter enable … b0: 0: IN00 counter disable 1: IN00 counter enable

- ### EMD7201_counter_enable

Format : **u32 status = EMD7201_counter_enable(u8 CardID);**

Purpose: To enable the counter.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMD7201_IP_mapping

- ### EMD7201_counter_disable

Format : **u32 status = EMD7201_counter_disable(u8 CardID);**

Purpose: To disable the counter.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMD7201_IP_mapping

- **EMD7201_counter_read**

Format : **u32 status = EMD7201_counter_read(u8 CardID,u32 counter[8]);**

Purpose: To read all the counter value.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMD7201_IP_mapping

Output:

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

- **EMD7201_counter_clear**

Format : **u32 status = EMD7201_counter_clear (u8 CardID, u8 channel);**

Purpose: To reset the counter value.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMD7201_IP_mapping
Channel	u8	b7~b0, b7: 0: no function 1: clear IN07 counter … b0: 0: no function 1: clear IN00 counter

Miscellaneous function

- **EMD7201_change_socket_port**

Format : **u32 status = EMD7201_change_socket_port(u8 CardID,u16 Remote_port);**

Purpose: To change the communicate port number of EMD7201.

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

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMD7201_IP_mapping
Remote_port	u16	The new port number to be set

- **EMD7201_change_IP**

Format : **u32 status = EMD7201_change_IP(u8 CardID, u8 IP[4]);**

Purpose: To change the communicate IP of EMD7201.

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

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMD7201_IP_mapping
IP[4]	U8	The new IP to be set

- **EMD7201_reboot**

Format : **u32 status = EMD7201_reboot(u8 CardID);**

Purpose: To reboot EMD7201 (about 10s).

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMD7201_IP_mapping

Software key function

- **EMD7201_security_unlock**

Format : **u32 status = EMD7201_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 EMD7201_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'};

- **EMD7201_security_status_read**

Format : **u32 status = EMD7201_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 EMD7201_IP_mapping

Output:

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

- **EMD7201_password_change**

Format : **u32 status = EMD7201_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 EMD7201_IP_mapping
Oldpassword [8]	u8	The previous password
password[8]	u8	The new password to be set

- **EMD7201_password_set_default**

Format : **u32 status = EMD7201_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 EMD7201_IP_mapping default : password[8] = {'1','2','3','4','5','6','7','8'};

5.10 DLL list

	Function Name	Description
1.	EMD7201_IP_mapping( )	Assign IP and get model parameter
2.	EMD7201_close( )	EMD7201 close
3.	EMD7201_port_read( )	Read EMD7201's input or output register status
4.	EMD7201_port_set ( )	Set EMD7201's output
5.	EMD7201_point_read( )	Read EMD7201's input or output register point status
6.	EMD7201_point_set( )	Set output point
7.	EMD7201_counter_mask_set( )	Set counter channel mask
8.	EMD7201_counter_enable( )	Enable counter function
9.	EMD7201_counter_disable( )	Disable counter function
10.	EMD7201_counter_read( )	Read counter value
11.	EMD7201_counter_clear( )	Clear designated counter
12.	EMD7201_change_socket_port( )	change the communication port
13.	EMD7201_change_IP( )	change the IP of EMD7201
14.	EMD7201_reboot( )	reboot EMD7201 module
15.	EMD7201_security_unlock( )	Unlock security
16.	EMD7201_security_status_read( )	Read lock status
17.	EMD7201_password_change( )	Change password
18.	EMD7201_password_set_default( )	Rest to factory default password

6. EMD7201 Error codes summary

6.1 EMD7201 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