EMD7216
Ethernet Digital I/O module

Software Manual (V1.1)

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	EMD7216.dll v1.0
1.1	enhance explanation of channel parameter

Contents

1. How to install the software of EMD7216

Please register as user’s club member to download the
“Step by step installation of EMD7216” 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\EMD7216_Install.exe to install the driver, Api and demo program automatically.

For a more detail descriptions, please refer “Step by step installation of EMD7216”.

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\EMD7216 will generate to put the associate files.

.. / JS Automation /EMD7216/API (header files and VB,VC lib files)

.. / JS Automation /EMD7216/Driver (copy of driver code)

.. / JS Automation /EMD7216/exe (demo program and source code)

The dll is located at ..\system.

3. About the EMD7216 software

EMD7216 software includes a set of dynamic link library (DLL) based on socket that you can utilize to control the interface functions.

Your EMD7216 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 EMD7216 functions within Windows' operation system environment.

3.1 What you need to get started

To set up and use your EMD7216 software, you need the following:

- EMD7216 software
- EMD7216 hardware

3.2 Software programming choices

You have several options to choose from when you are programming EMD7216 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 EMD7216 software.

4. EMD7216 Language support

The EMD7216 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 EMD7216 software library

The EMD7216 function reference section contains general information about building EMD7216 applications, describes the nature of the EMD7216 functions used in building EMD7216 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.

EMD7216 Windows Libraries

The EMD7216 for Windows function library is a DLL called **EMD7216.dll**. Since a DLL is used, EMD7216 functions are not linked into the executable files of applications. Only the information about the EMD7216 functions in the EMD7216 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 EMD7216 functions in EMD7216 .dll.

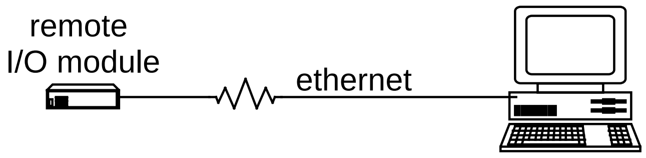
Header Files and Import Libraries for Different Development Environments		
Development Environment	Header File	Import Library
Microsoft C/C++	EMD7216.h	EMD7216VC.lib
Borland C/C++	EMD7216.h	EMD7216BC.lib
Microsoft Visual Basic	EMD7216.bas	

Table 1

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

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 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 Function format

Every EMD7216 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.

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 EMD7216 API. Read the following sections that apply to your programming language.

Note: Be sure to include the declaration functions of EMD7216 prototypes by including the appropriate EMD7216 header file in your source code. Refer to Chapter 4. EMD7216 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 = EMD7216_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 = EMD7216_port_read ( CardID, port, &data);
```

6.3.2 Visual basic

The file EMD7216.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 EMD7216.bas, do not use the numerical values.
In Visual Basic, you can add the entire EMD7216.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 EMD7216.bas file for your project in Visual Basic 4.0, go to the **File** menu and select the **Add File..**. **option**. Select EMD7216.bas, which is browsed in the EMD7216 \ api directory. Then, select **Open** to add the file to the project.

To add the EMD7216.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** EMD7216.bas, which is in the EMD7216 \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 EMD7216bc.lib EMD7216.dll
```

Then add the **EMD7216bc.lib** to your project and add
#include "EMD7216.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 = EMD7216_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 = EMD7216_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 EMD7216 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,

EMD7216_initial( ) will do.

Once you want to close your application, call

EMD7216_close( ) to release all the resource.

- **EMD7216_initial**

Format : **u32 status =EMD7216_initial(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 EMD7216 to a specified CardID number**.**

Parameters:

Input:

Name	Type	Description
CardID	u8	0~255 Assign CardID to the EMD7216 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 EMD7216 Default: 6936
TimeOut	u16	Assign the max delay time of EMD7216 response message,1000~10000 ms.

Output:

Name	Type	Description
CardType	u8	not defined

- **EMD7216_close**

Format : **u32 status =EMD7216_close (u8 CardID)**

Purpose: Release the EMD7216 resource when closing the Windows applications.

Parameters:

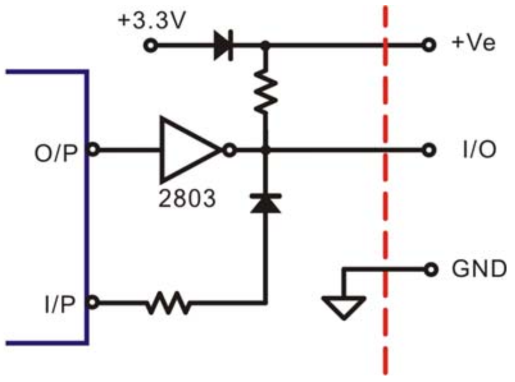
Input:

Name	Type	Description
CardID	u8	CardID assigned by EMD7216_initial

7.2 Digital I/O

Each digital I/O point can be configured as input or output. If it is configured as input, the input level can work up to the external terminal voltage Ve (not exceed the output buffer rating 45Vdc).

If it is configured as output, the output buffer can drive up to 450ma peak current and steady stae up to 45ma, the working voltage up to 45Vdc. You can see the circuit diagram as follows.



First of all, each point must configure as input or output

EMD7216_port_config_set( ) and read back to verify by

EMD7216_port_config_read( )

To control the output, use

EMD7216_port_set( )

To read input or output register status use

EMD7216_port_read( )

To control a point of output, use

EMD7216_point_set( )

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

EMD7216_point_read( )

- **EMD7216_port_config_set**

Format : **u32 status = EMD7216_port_config_set(u8 CardID ,u8 port, u8 config)**

Purpose: Set the data of the I/O port.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMD7216_initial
Port	u8	port number 0: IO_00~07 1: IO_10~17
config	u8	Configure the IO as input or output bit0: 0: IO_n0 as input 1: IO_n0 as output … bit7: 0: IO_n7 as input 1: IO_n7 as output

- **EMD7216_port_config_read**

Format : **u32 status = EMD7216_port_config_read(u8 CardID ,u8 port, u8 *config)**

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

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMD7216_initial
port	u8	port number 0: IO_00~07 1: IO_10~17

Output:

Name	Type	Description
config	u8	Configure the IO as input or output bit0: 0: IO_n0 as input 1: IO_n0 as output … bit7: 0: IO_n7 as input 1: IO_n7 as output

- **EMD7216_port_set**

Format : **u32 status = EMD7216_port_set (u8 CardID, u8 port,u8 data)**

Purpose: Set the output port data.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMD7216_initial
port	u8	port number 0: IO_00~07 1: IO_10~17
data	u8	b7 ~ b0, bitmap of output port values b7: IO_07 or IO_17 depend on port assignment … b0: IO_00 or IO_10 depend on port assignment

- **EMD7216_port_read**

Format : **u32 status = EMD7216_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 EMD7216_initial
port	u8	port number 0: IO_00~07 1: IO_10~17

Output:

Name	Type	Description
data	u8	b7 ~ b0, bitmap of I/O port data b7: IO_07 or IO_17 depend on port assignment … b0: IO_00 or IO_10 depend on port assignment

● EMD7216_point_set

Format : **u32 status =EMD7216_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 EMD7216_initial
point	u8	point number of inport 0x00~0x07 for IO_00~07 0x10~0x17 for IO_10~17
state	u8	state of out port

● EMD7216_point_read

Format : **u32 status =EMD7216_point_read(u8 CardID,u8 point, u8 *state)**

Purpose: Read bit state of input port.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMD7216_initial
point	u8	point number of in port 0x00~0x07 for IO_00~07 0x10~0x17 for IO_10~17

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:

EMD7216_counter_mask_set( ) then enable the function by

EMD7216_counter_enable( ) and any time to stop by

EMD7216_counter_disable( ).

To read the counter value by

EMD7216_counter_read( ) and use

EMD7216_counter_clear( ) to clear counter.

Each point configured as input can work as low frequency counter (max 200Hz). The remote I/O module will count the input signal for you without any attention to the signal transition.

- **EMD7216_counter_mask_set**

Format : **u32 status = EMD7216_counter_mask_set(u8 CardID,u8 port,u8 channel);**

Purpose: To set the counter channel mask.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMD7216_initial
port	u8	port number 0: IO_00~07 1: IO_10~17
Channel	u8	b7 ~ b0, b7: 0: IO_n7 counter disable 1: IO_n7 counter enable … b0: 0: IO_n0 counter disable 1: IO_n0 counter enable

- **EMD7216_counter_enable**

Format : **u32 status = EMD7216_counter_enable(u8 CardID);**

Purpose: To enable the counter.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMD7216_initial

- **EMD7216_counter_disable**

Format : **u32 status = EMD7216_counter_disable(u8 CardID);**

Purpose: To disable the counter.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMD7216_initial

- **EMD7216_counter_read**

Format : **u32 status = EMD7216_counter_read(u8 CardID, u8 port, u32 counter[8]);**

Purpose: To read all the counter value.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMD7216_initial
port	u8	port number 0: IO_00~07 1: IO_10~17

Output:

Name	Type	Description
counter	u32	counter value counter[0] for IO_n0 … counter[7] for IO_n7

- **EMD7216_counter_clear**

Format : **u32 status = EMD7216_counter_clear (u8 CardID,u8 port,u8 channel);**

Purpose: To reset the counter value.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMD7216_initial
port	u8	port number 0: IO_00~07 1: IO_10~17
Channel	u8	b7: 0: no function 1: clear IO_n7 counter … b0: 0: no function 1: clear IO_n0 counter

7.4 Miscellaneous function

The module IP and communication port must be confirmed with the gateway and software to ensure the correct Ethernet communication.

To change the communication port as you need by:

EMD7216_change_socket_port( )[*1]

To change IP,

EMD7216_change_IP( )[*1]

To reboot EMD7216 module for module alarm or to validate the system configuration change by:

EMD7216_reboot( )[*1]

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

- **EMD7216_change_socket_port**

Format : **u32 status = EMD7216_change_socket_port(u8 CardID,u16 Remote_port);**

Purpose: To change the communicate port number of EMD7216.

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

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMD7216_initial
Remote_port	u16	The new port number to be set

- **EMD7216_change_IP**

Format : **u32 status = EMD7216_change_IP(u8 CardID,u8 IP[4]);**

Purpose: To change the communicate IP of EMD7216.

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

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMD7216_initial
IP[4]	U8	The new IP to be set

- **EMD7216_reboot**

Format : **u32 status = EMD7216_reboot(u8 CardID);**

Purpose: To reboot EMD7216(about 10s).

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMD7216_initial

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 EMD7216, you must unlock the module first by

EMD7216_security_unlock( )

To verify the lock status by

EMD7216_security_status_read( )

You can change password for your convienence by

EMD7216_password_change( )

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

EMD7216_password_set_default( ) [*2]

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

- **EMD7216_security_unlock**

Format : **u32 status = EMD7216_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 EMD7216_initial
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'};

- **EMD7216_security_status_read**

Format : **u32 status = EMD7216_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 EMD7216_initial

Output:

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

- **EMD7216_password_change**

Format : **u32 status = EMD7216_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 EMD7216_initial
Oldpassword [8]	u8	The previous password
password[8]	u8	The new password to be set

- **EMD7216_password_set_default**

Format : **u32 status = EMD7216_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 EMD7216_initial default : password[8] = {'1','2','3','4','5','6','7','8'};

7.6 WDT (watch dog timer)

In the industrial environment, we want the controller work as stable as possible but we are not God, we can not always put the controller by guess. To ensure the controller will not harm to the system or human, people always put a WDT to monitor the controller, if the controller fail to reset it, WDT will latch the system to prevent further harm. EDM7216 also provide the hardware WDT function, you can enable or disable as your application requirement.

Use

EMD7216_WDT_set( ) to set up the WDT timer and the output state if the system fail to communicate.

EMD7216_WDT_read( ) to read back the configuration.

To enable the WDT function to monitor the communication (you must periodically communicate with the remote I/O module to keep it alive) by:

EMD7216_WDT_enable( ) and disable by:

EMD7216_WDT_disable( ).

- **EMD7216_WDT_set**

Format : **u32 status = EMD7216_WDT_set(u8 CardID,u16 time,u8 state[2])**

Purpose: Set WDT(watch dog timer) configuration.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMD7216_initial
time	u16	Set the WDT wait time.(10~10000) based on 0.1 sec time base. default: 10 (1s)
state[2]	u8	Set the output default state, the state will keep while the system failure.

- **EMD7216_WDT_read**

Format : **u32 status = EMD7216_WDT_read (u8 CardID, u16 *time, u8 state[2], u8 *enable)**

Purpose: Read back WDT(watch dog timer) configuration.

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMD7216_initial

Output:

Name	Type	Description
time	u16	read the WDT wait time.
state[2]	u8	read the output default state
enable	u8	0: disable 1: enable

- **EMD7216_WDT_enable**

Format : **u32 status = EMD7216_WDT_enable(u8 CardID)**

Purpose: enableWDT(watch dog timer) .

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMD7216_initial

- **EMD7216_WDT_disable**

Format : **u32 status = EMD7216_WDT_disable(u8 CardID)**

Purpose: disable WDT(watch dog timer) .

Parameters:

Input:

Name	Type	Description
CardID	u8	CardID assigned by EMD7216_initial

7.7 Error codes and address

Every EMD7216 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 EMD7216 function is the parameter **CardID** which is set by ***EMD7216_initial***. You can utilize multiple devices with different card ID within one application; to do so, simply pass the appropriate **CardID** to each function.

8. DLL list

	Function Name	Description
1.	EMD7216_initial( )	Assign IP and get model parameter
2.	EMD7216_close( )	EMD7216 close
3.	EMD7216_port_config_set( )	Configure the point as input or output
4.	EMD7216_port_config_read( )	Read configure the point
5.	EMD7216_port_set( )	Set EMD7216's output
6.	EMD7216_port_read( )	Read EMD7216's input or output register status
7.	EMD7216_point_set( )	Set output point
8.	EMD7216_point_read( )	Read EMD7216's input or output register point status
9.	EMD7216_counter_mask_set( )	Set counter channel mask
10.	EMD7216_counter_enable( )	Enable counter function
11.	EMD7216_counter_disable( )	Disable counter function
12.	EMD7216_counter_read( )	Read counter value
13.	EMD7216_counter_clear( )	Clear designated counter
14.	EMD7216_change_socket_port( )	change the communication port
15.	EMD7216_change_IP( )	change the IP of EMD7216
16.	EMD7216_reboot( )	reboot EMD7216 module
17.	EMD7216_security_unlock( )	Unlock security
18.	EMD7216_security_status_read( )	Read lock status
19.	EMD7216_password_change( )	Change password
20.	EMD7216_password_set_default( )	Rest to factory default password
21.	EMD7216_WDT_set( )	Set up WDT timer and output states
22.	EMD7216_WDT_read( )	Read WDT timer and output state setting
23.	EMD7216_WDT_enable( )	Enable WDT function
24.	EMD7216_WDT_disable( )	Disable WDT function

9. EMD7216 Error codes summary

9.1 EMD7216 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
312	CHANGE_IP_ERROR	Change IP error
313	MASK_ERROR	Set mask error
314	COUNTER_ENABLE_ERROR	Can not enable counter
315	COUNTER_DISABLE_ERROR	Can not disable counter
316	COUNTER_READ_ERROR	Fail to read counter
317	COUNTER_CLEAR_ERROR	Fail to clear counter
318	TIME_ERROR	Set the time error