EMA8308/ EMA8308A
EMA8308D/ EMA8308DA
Ethernet Analog 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	EMA8308.dll v1.0

Contents

1. How to install the software of EMA8308

Please register as user's club member to download the
"Step by step installation of **EMA8308**" document from http://automation.com.tw

1.1 Install the EMA 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**EMA8308_**Install.exe to install the driver, Api and demo program automatically.

For a more detail descriptions, please refer "Step by step installation of **EMA8308**".

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 EMA83xx_Install.exe

If you use the default setting, a new directory .. \JS Automation**EMA8308** will generate to put the associate files.

.. / JS Automation /EMA8308/API (header files and VB,VC lib files)

.. / JS Automation /EMA8308/Driver (copy of driver code)

.. / JS Automation /EMA8308/exe (demo program and source code)

The dll is located at ..\system.

3. About the EMA8308 software

EMA8308 software includes a set of dynamic link library (DLL) based on socket that you can utilize to control the interface functions.

Your **EMA8308** 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 **EMA8308** functions within Windows' operation system environment.

3.1 What you need to get started

To set up and use your **EMA8308** software, you need the following:

- **EMA8308** software
- **EMA8308** hardware

3.2 Software programming choices

You have several options to choose from when you are programming **EMA8308** 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 **EMA8308** software.

4. EMA8308 Language support

The **EMA8308** 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 EMA8308 software library

The EMA8308 function reference section contains general information about building EMA8308 applications, describes the nature of the EMA8308 functions used in building EMA8308 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.

EMA8308 Windows Libraries

The EMA8308 for Windows function library is a DLL called EMA8308.dll. Since a DLL is used, EMA8308 functions are not linked into the executable files of applications. Only the information about the EMA8308 functions in the EMA8308 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 EMA8308 functions in EMA8308 .dll.

Header Files and Import Libraries for Different Development Environments		
Development Environment	Header File	Import Library
Microsoft C/C++	**EMA8308**.h	**EMA8308**VC.lib
Borland C/C++	**EMA8308**.h	**EMA8308**BC.lib
Microsoft Visual Basic	**EMA8308**.bas	

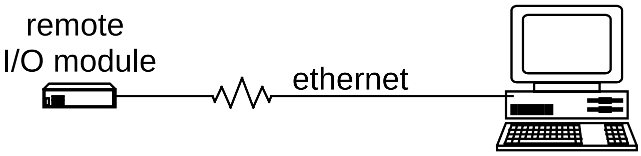
Table 1

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

I/O communicate via ethernet

The remote analog I/O is the function extension of the card type analog 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 analog I/O will do.

JS automation keeps the remote analog I/O function as close to the card type analog 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 Function format

Every EMA8308 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 **EMA8308** API. Read the following sections that apply to your programming language.

Note: Be sure to include the declaration functions of **EMA8308** prototypes by including the appropriate **EMA8308** header file in your source code. Refer to Chapter 4. **EMA8308** 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 = **EMA8308_**read_port (**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 = **EMA8308_**read_port (CardID, port, &data);

6.3.2 Visual basic

The file **EMA8308**.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 **EMA8308**.bas, do not use the numerical values.

In Visual Basic, you can add the entire **EMA8308**.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 **EMA8308**.bas file for your project in Visual Basic 4.0, go to the **File** menu and select the **Add File..**. **option**. Select **EMA8308**.bas, which is browsed in the **EMA8308** \ api directory. Then, select **Open** to add the file to the project.

To add the **EMA8308**.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 EMA8308**.bas, which is in the **EMA8308** \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 **EMA8308**bc.lib **EMA8308**.dll

Then add the **EMA8308bc.lib** to your project and add
#include “**EMA8308**.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 = **EMA8308_**read_port (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 = **EMA8308_**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.

7. Software overview and dll function

7.1 Initialization and close

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

EMA8308_initial( ) will do.

Once you want to close your application, call

EMA8308_close( ) to release all the resource.

To check the firmware version,

EMA8308_firmware_version_read( ) will do.

- **EMA8308_initial**

Format : **u32 status =EMA8308_initial (u32 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 **EMA8308** to a specified CardID number**.**

Parameters:

Input:

Name	Type	Description
CardID	u32	Assign CardID to the **EMA8308** 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:15120
Remote_port	u16	Assign a communicate port of **EMA8308** Default:6936
TimeOut	u16	Assign the max delay time of **EMA8308** response message,1000~10000 ms.

Output:

Name	Type	Description
CardType	u8	Get the Card Type of **EMA8308** 1: EMA-8308DA 2: EMA-8308D 3: EMA-8308A 4: EMA-8308

- **EMA8308_close**

Format : **u32 status =EMA8308_close (u32 CardID)**

Purpose: Release the **EMA8308** resource when closing the Windows applications.

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMA8308_initial function

- **EMA8308_firmware_version_read**

Format : **u32 status = EMA8308_firmware_version_read(u32 CardID, u8 Version[2])**

Purpose: Read the firmware version.

Parameters:

Input:

Name	Type	Description
CardID	u32	0~1999 CardID assigned by EMA8308_initial

Output:

Name	Type	Description

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

7.2 Analog Input/Output function

At the initial of A/D conversion, you can choose the input filter to filter off the noise, EMA module has build-in the filter and choose the bandwidth by:

EMA8308_AD_filter_set( )

EMA8308_AD_filter_read( )

To configure as single end or differential mode input by

EMA8308_AD_config_set( ) and read back the configuration data by

EMA8308_AD_config_read( ).

To setup the analog input range by

EMA8308_AD_range_set( ) and read back the range setting data by

EMA8308_AD_range_read( ).

After all the A/D working parameters set up, you can read the A/D value(already converted to voltage) by"

EMA8308_AD_channel_value_read( ) or raw conversion data by

EMA8308_AD_channel_data_read( )

Also the port value or data by:

EMA8308_AD_port_value_read( )

EMA8308_AD_port_data_read( )

To output analog voltage, use

EMA8308_DA_set( ) and read back the output register by

EMA8308_DA_read( )

● EMA8308_AD_filter_set

Format : u32 status = EMA8308_AD_filter_set (u32 CardID, u8 Mode)

Purpose: Configure AD Conversion rate.

Parameters:

Input:

Name	Type	Description		
CardID	u32	CardID assigned by EMA8308_initial function		
Mode	u8		Conversion rate	RMS noise
		0	7.03KHZ	23uV
		1	3.52KHZ	3.6uV
		2	1.76KHZ	2.1uV
		3	897HZ	1.5uV

EMA8308_AD_filter_read

Format : u32 status = EMA8308_AD_filter_read (u32 CardID, u8 *Mode)

Purpose: **R**ead back AD Conversion rate.

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMA8308_initial function

Output:

Name	Type	Description		
Mode	u8		Conversion rate	RMS noise
		0	7.03KHZ	23uV
		1	3.52KHZ	3.6uV
		2	1.76KHZ	2.1uV
		3	897HZ	1.5uV

EMA8308_AD_config_set

Format : u32 status = EMA8308_AD_config_set (u32 CardID,u8 Port, u8 State)

Purpose: Configure each port as differential or single end.

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMA8308_initial function
Port	u8	0: unused
State	u8	0: All channel is single end 1: Port0 is paired differential and Port1 is single end 2: Port0 is single end and Port1 is paired differential 3: All channel is paired differential

Note:

AD input differential connection please refer to chap 5 Basic concepts of analog I/O control.

- **EMA8308_AD_config_read**

Format : **u32 status = EMA8308_AD_config_read (u32 CardID,u8 Port ,u8 State)**

Purpose: read back each channel as differential or single end.

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMA8308_initial function
Port	u8	0: select A/D port0 and port1

Output:

Name	Type	Description
State	u8	0: All channel is single end 1: Port0 is paired differential and Port1 is single end 2: Port0 is single end and Port1 is paired differential 3: All channel is paired differential

EMA8308_AD_range_set

Format : **u32 status = EMA8308_AD_range_set(u32 CardID,u8 Port,**
_AD_Range *AD_range)

Purpose: set up each group conversion range

Parameters:

Input:

<table>
<tr><th>Name</th><th>Type</th><th colspan="2">Description</th></tr>
<tr><td>CardID</td><td>u32</td><td colspan="2">CardID assigned by EMA8308_initial function</td></tr>
<tr><td>Port</td><td>u8</td><td colspan="2">0: port0
1: port1</td></tr>
<tr><td rowspan="3">AD_range</td><td rowspan="3">_AD_range</td><td colspan="2">struct _AD_Range{
u8 ch0_ range,
u8 ch1_ range,
u8 ch2_ range,
u8 ch3_ range
u8 ch4_ range,
u8 ch5_ range,
u8 ch6_ range,
u8 ch7_ range
}</td></tr>
<tr><td>EMD8308D,
EMD8308DA</td><td>EMD8308,
EMD8308A</td></tr>
<tr><td>0: -5V ~5V
1: -10V ~10V
2: 0 ~20mA
3: 4 ~20mA</td><td>0: 0V ~ 5V
1: 0V ~10V
2: 0 ~20mA
3: 4 ~20mA</td></tr>
</table>

Note:

If the even channel is configured as differential input, the next odd number channel member is invalid.

For example ch0 is configured as differential input by EMA8308_AD_config_set, then the AD_Range.ch1_range is of no use.

EMA8308_AD_range_read

Format : **u32 status = EMA8308_AD_range_read(u32 CardID, u8 Port,**
_AD_Range *AD_range)

Purpose: read back each group conversion range setting

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMA8308_initial function
Port	u8	0: port0 1: port1

Output:

Name	Type	Description	

AD_range	_AD_range	struct _AD_Range{
		u8 ch0_ range,
		u8 ch1_ range,
		u8 ch2_ range,
		u8 ch3_ range
		u8 ch4_ range,
		u8 ch5_ range,
		u8 ch6_ range,
		u8 ch7_ range
		}	

		EMD8308D, EMD8308DA	EMD8308, EMD8308A
		0: -5V ~5V 1: -10V ~10V 2: 0 ~20mA 3: 4 ~20mA	0: 0V ~ 5V 1: 0V ~10V 2: 0 ~20mA 3: 4 ~20mA

- **EMA8308_AD_channel_value_read**

Format : **u32 status = EMA8308_AD_channel_value_read(u32 CardID,u8 Port, u8 Channel,f32 value)**

Purpose: Read back A/D conversion result by channel.

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMA8308_initial function
Port	u8	0:port0 1:port1
Channel	u8	0:channel 0 … 7:channel 7

Output:

Name	Type	Description
value	f32	Value, the AD value converted according

- **EMA8308_AD_channel_data_read**

Format : **u32 status = EMA8308_AD_channel_data_read(u32 CardID,u8 Port, u8 Channel , i32 data)**

Purpose: Read back A/D raw data by channel.

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMA8308_initial function
Port	u8	0:port0 1:port1
Channel	u8	0:channel 0 … 7:channel 7

Output:

Name	Type	Description
data	i32	data the AD raw data

- **EMA8308_AD_port_value_read**

Format : **u32 status = EMA8308_AD_port_value_read(u32 CardID, u8 Port, f32 Value[8])**

Purpose: Read back A/D conversion result by port.

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMA8308_initial function
Port	u8	0:Port0 1:Port1

Output:

Name	Type	Description		
Value[8]	f32	Mode	Differential Mode	single-end Mode
		Value[0]	differential	single-end
		Value[1]	unused	single-end
		Value[2]	differential	single-end
		Value[3]	unused	single-end
		Value[4]	differential	single-end
		Value[5]	unused	single-end
		Value[6]	differential	single-end
		Value[7]	unused	single-end

- **EMA8308_AD_port_data_read**

Format : **u32 status = EMA8308_AD_port_data_read(u32 CardID, u8 Port, i32 data[8])**

Purpose: Read back A/D raw data by port.

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMA8308_initial function
Port	u8	0:Port0 1:Port1

Output:

Name	Type	Description		
Data[8]	i32	Mode	Differential Mode	single-end Mode
		data[0]	differential	single-end
		data[1]	unused	single-end
		data[2]	differential	single-end
		data[3]	unused	single-end
		data[4]	differential	single-end
		data[5]	unused	single-end
		data[6]	differential	single-end
		data[7]	unused	single-end

- **EMA8308_DA_set**

Format : **u32 status = EMA8308_DA_set(u32 CardID ,u8 Port, u8 Channel , i16 Data)**

Purpose: Set the value of the D/A data.

Parameters:

Input:

<table>
<tr><th>Name</th><th>Type</th><th colspan="3">Description</th></tr>
<tr><td>CardID</td><td>u32</td><td colspan="3">CardID assigned by EMA8308_initial function</td></tr>
<tr><td>Port</td><td>u8</td><td colspan="3">0: unused</td></tr>
<tr><td>Channel</td><td>u8</td><td colspan="3">channel number:
0: DA0
1: DA1</td></tr>
<tr><td rowspan="8">Data</td><td rowspan="8">i16</td><td>model</td><td>EMA8308,
EMA8308D</td><td>EMA8308A,
EMA8308DA</td></tr>
<tr><td>resolution</td><td>12bit DA</td><td>16bit DA</td></tr>
<tr><td>data range</td><td>-2048 ~ 2047</td><td>-32768 ~ 32767</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td>output</td><td>data</td><td>data</td></tr>
<tr><td>0V</td><td>0</td><td>0</td></tr>
<tr><td>+10V</td><td>2047</td><td>32767</td></tr>
<tr><td>-10V</td><td>-2048</td><td>-32768</td></tr>
</table>

- **EMA8308_DA_read**

Format : **u32 status = EMA8308_DA_read(u32 CardID ,u8 Port, u8 Channel , i16 Data[2])**

Purpose: Read back the setting of the D/A data.

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMA8308_initial function
Port	u8	0: unused
Channel	u8	channel number: 0: DA0 1: DA1

Output:

Name	Type	Description
Data	i16	**EMA8308，EMA8308D** 12bit DA → -2048 ~ 2047 0: 0V 2047: +10V -2048: -10V **EMA8308A，EMA8308DA** 16bit DA → -32768 ~ 32767; 0: 0V 32767: +10V -32768: -10V

7.3 Configuration function

To change the socket port by

EMA8308_socket_port_change( ) and change IP by

EMA8308_IP_change( )

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

EMA8308_reboot( )

- **EMA8308_socket_port_change**

Format : **u32 status = EMA8308_socket_port_change (u32 CardID,u16 Remote_port);**

Purpose: To change the communicate port number of **EMA8308**.

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

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMA8308_initial function
Remote_port	u16	The new port number to be set Default port is: 6936

- **EMA8308_IP_change**

Format : **u32 status = EMA8308_IP_change (u32 CardID,u8 IP[4]);**

Purpose: To change the communicate IP of **EMA8308**.

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

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMA8308_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

- **EMA8308_reboot**

Format : **u32 status = EMA8308_reboot(u32 CardID);**

Purpose: To reboot **EMA8308**(about 10s).

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by **EMA8308_**initial function

7.4 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

EMA8308_security_unlock( ) and read back the status of security by

EMA8308_security_status_read( )

If you want to change password, use

EMA8308_password_change( ) will do.

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

EMA8308_password_set_default( ) *[*1]* will do.

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

- **EMA8308_security_unlock**

Format : **u32 status = EMA8308_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 EMA8308_initial function
password[8]	u8	The password previous set Default :password[8] = {'1','2','3','4','5','6','7','8'};

- **EMA8308_security_status_read**

Format : **u32 status = EMA8308_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 EMA8308_initial function

Output:

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

- **EMA8308_password_change**

Format : **u32 status = EMA8308_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 EMA8308_initial function
Oldpassword [8]	u8	The previous password
password[8]	u8	The new password to be set

- **EMA8308_password_set_default**

Format : **u32 status = EMA8308_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 EMA8308_initial function default :password[8] = {'1','2','3','4','5','6','7','8'};

7.5 Error codes and address

Every **EMA8308** 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 **EMA8308** function is the parameter **CardID** which is set by ***EMA8308_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.

8. DLL list

	Function Name	Description
1.	EMA8308_initial ( )	Map IP and get model parameter
2.	EMA8308_close( )	EMA8308 close
3.	EMA8308_firmware_version_read( )	Read the firmware version
4.	EMA8308_AD_filter_set ( )	Configure AD Conversion rate.
5.	EMA8308_AD_filter_read ( )	Read back AD Conversion rate.
6.	EMA8308_AD_config_set ( )	Configure each port as differential or single end
7.	EMA8308_AD_config_read ( )	Read back each port as differential or single end
8.	EMA8308_AD_range_set ( )	set up each group conversion range
9.	EMA8308_AD_range_read ( )	read back each group conversion range setting
10.	EMA8308_AD_channel_value_read( )	Read back A/D conversion result by channel.
11.	EMA8308_AD_channel_data_read( )	Read back A/D raw data by channel.
12.	EMA8308_AD_port_value_read( )	Read back A/D conversion result by port.
13.	EMA8308_AD_port_data_read( )	Read back A/D raw data by port.
14.	EMA8308_DA_set ( )	Set the value of the D/A data
15.	EMA8308_DA_read ( )	Read back the setting of the D/A data
16.	EMA8308_socket_port_change ( )	To change the communicate port number of EMA8308
17.	EMA8308_IP_change ( )	To change the communicate IP of EMA8308
18.	EMA8308_reboot ( )	To reboot EMA8308
19.	EMA8308_security_unlock ( )	Unlock security
20.	EMA8308_security_status_read ( )	Read lock status
21.	EMA8308_password_change ( )	Change password
22.	EMA8308_password_set_default ( )	Rest to factory default password

9. EMA8308 Error codes summary

9.1 EMA8308 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	CHANNEL_ERROR	Point parameter unacceptable or 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	CHANGEIP_ERROR	Can not change IP
313	AD_READ_ERROR	Can not read AD data
314	DA_SET_ERROR	Can not setup DA voltage / data
315	DA_READ_ERROR	Can not read DA voltage / data