EMA8308/ EMA8308D
Ethernet Analog I/O module

Software Manual(V1.1)

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
1.1	based on V1.0
	correct parameters of AD function

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 **Ethernet IO module**".

2. Where to find the file you need

WinXP/7 and up

In WinXP/7 and later 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 WinXP/7 and later. 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

- Microsoft Visual C/C++
- Borland C/C++
- Microsoft Visual C#
- Microsoft Visual Basic
- Microsoft VB.net

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		
Language	**Header File**	**Import Library**
Microsoft Visual C/C++	EMA8308.h	EMA8308VC.lib
Borland C/C++	EMA8308.h	EMA8308BC.lib
Microsoft Visual C#	EMA8308.cs	
Microsoft Visual Basic	EMA8308.bas	
Microsoft VB.net	EMA8308.vb	

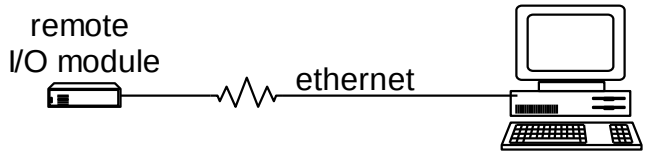
Table 1

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 **EMA8308BC**.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 0: assign by OS
Remote_port	u16	Assign a communicate port of **EMA8308** Default:6936
TimeOut_ms	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-8308D 3: 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 built-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_mode_set( ) and read back the configuration data by

EMA8308_AD_mode_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_channel_set( ) or

EMA8308_DA_port_set( ) and read back the output register by

EMA8308_DA_channel_read( )

EMA8308_DA_port_read( )

- **EMA8308_AD_filter_set**

Format : u32 status = EMA8308_AD_filter_set(u32 CardID,u8 port, u8 filter_mode)

Purpose: Configure AD Conversion rate.

Parameters:

Input:

Name	Type	Description		
CardID	u32	CardID assigned by EMA8308_initial function		
port	u8	unused		
filter_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 port, u8 * filter_mode)

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

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMA8308_initial function
port	u8	unused

Output:

Name	Type	Description		
filter_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_mode_set**

Format : u32 status = EMA8308_AD_mode_set(u32 CardID,u8 port, u8 ad_config)

Purpose: Configure each port as differential or single end.

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMA8308_initial function
port	u8	unused
ad_config	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_mode_read**

Format : u32 status = EMA8308_AD_mode_read(u32 CardID,u8 port ,u8 *ad_config)

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	unused

Output:

Name	Type	Description
ad_config	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,
u8 AD_range[8])

Purpose: set up each group conversion range

Parameters:

Input:

Name	Type	Description	
CardID	u32	CardID assigned by EMA8308_initial function	
port	u8	EMA8308D	EMA8308
		not used	0: single end port0 1: single end port1 2: differential
AD_range[8]	u8	EMA8308D	EMA8308
		0: -5V ~5V 1: -10V ~10V 2: 0 ~20mA 3: 4 ~20mA	0: 0V ~ 5V 1: 0V ~10V 2: 0 ~20mA 3: 4 ~20mA

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_mode_set, then the ad_range[1] is of no use.

- **EMA8308_AD_range_read**

Format : **u32 status = EMA8308_AD_range_read(u32 CardID, u8 port, u8 AD_range[8])**

Purpose: read back each group conversion range setting

Input:

Name	Type	Description	
CardID	u32	CardID assigned by EMA8308_initial function	
port	u8	EMA8308D	EMA8308
		not used	0: single end port0 1: single end port1 2: differential

Output:

Name	Type	Description	
AD_range[8]	u8	EMA8308D	EMA8308
		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	EMA8308D	EMA8308
		not used	0: single end port0 1: single end port1 2: differential
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	EMA8308D	EMA8308
		not used	0: single end port0 1: single end port1 2: differential
channel	u8	0:channel 0 … 7:channel 7	

Output:

Name	Type	Description
data	i32	data the AD raw data (25 bit) -16777216 ~ +16777216

- **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	EMA8308D	EMA8308
		not used	0: single end port0 1: single end port1 2: differential

Output:

Name	Type	Description
Value[8]	f32	Value, the AD value converted according

- **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	EMA8308D	EMA8308
		not used	0: single end port0 1: single end port1 2: differential

Output:

Name	Type	Description
data[8]	i32	data, the AD raw data

- **EMA8308_DA_channel_set**

Format : **u32 status = EMA8308_DA_channel_set(u32 CardID ,u8 port, u8 channel , i16 data)**

Purpose: Set the value 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
data	i16	**EMA8308，EMA8308D** 16bit DA → -32768 ~ 32767; 0: 0V 32767: +10V -32768: -10V

- **EMA8308_DA_channel_read**

Format : **u32 status = EMA8308_DA_channel_read(u32 CardID ,u8 port, u8 channel , i16 *data)**

Purpose: read the value 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** 16bit DA → -32768 ~ 32767; 0: 0V 32767: +10V -32768: -10V

- **EMA8308_DA_port_set**

Format : **u32 status = EMA8308_DA_port_set(u32 CardID ,u8 port, i16 data[2])**

Purpose: Set 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
data[2]	i16	**EMA8308，EMA8308D** 16bit DA → -32768 ~ 32767; 0: 0V 32767: +10V -32768: -10V

- **EMA8308_DA_port_read**

Format : **u32 status = EMA8308_DA_port_read(u32 CardID ,u8 port, 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

Output:

Name	Type	Description
data[2]	i16	**EMA8308，EMA8308D** 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 (ASCII) 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 (ASCII)
password[8]	u8	The new password to be set(ASCII)

- **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 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 run in abnormal state the system will fail to reset WDT then WDT will latch the system to prevent further harm. EMA8308 also provide the WDT function, which will detect the Ethernet connection state, once the connection is fail for a predefined period, the module will output the predefined status to the ports. You can enable or disable as your application required.

Use

EMA8308_WDT_set( ) to set up the WDT timer and the output state if the Ethernet connection fail to communicate.

EMA8308_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:

EMA8308_WDT_enable( ) and disable by:

EMA8308_WDT_disable( ).

- **EMA8308_WDT_set**

Format : u32 status = EMA8308_WDT_set(u32 CardID,u16 time,i16 DA_data[2])

Purpose: Set WDT(watch dog timer) configuration.

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMA8308_initial
time	u16	Set the WDT wait time.(10~10000) based on 0.1 sec time base. default: 10 (1s)
DA_data[2]	i16	Set the output default state, the state will keep while the connection failure. DA_data[0]: DA0 value DA_data[1]: DA1 value

Note: The predefined outputs will be complied with the of polarity it is configured.

- **EMA8308_WDT_read**

Format : u32 status = EMA8308_WDT_read (u32 CardID, u16 *time, i16 DA_data[2], u8 *enable)

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

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMA8308_initial

Output:

Name	Type	Description
time	u16	read the WDT wait time.
DA_data[2]	i16	DA_data[0]: DA0 value DA_data[1]: DA1 value
enable	u8	0: disable 1: enable

- **EMA8308_WDT_enable**

Format : u32 status = EMA8308_WDT_enable(u32 CardID)

Purpose: enableWDT(watch dog timer) .

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMA8308_initial

- **EMA8308_WDT_disable**

Format : u32 status = EMA8308_WDT_disable(u32 CardID)

Purpose: disable WDT (watch dog timer) .

Parameters:

Input:

Name	Type	Description
CardID	u32	CardID assigned by EMA8308_initial

7.6 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. Communication protocol

Although the dll have provide various function for the user, which enables the users to take the EMA module as if it is a non-ethernet I/O interface. But some users may want to coding their own software from the Ethernet basic functions, this chapter provides the detail of the communication protocol.

8.1 Host to module command format

IP header	UDP header	UDP data			
		Card_name	Password	Command	Data
20bytes	8bytes	7bytes	8bytes	1byte	32bytes

As shown above, the command format from PC to module is an UDP format, the first 20 bytes is the IP header, the next 8 byes is the UDP header then follows 48 bytes UDP data. The UDP data is defined as follows:

```
typedef struct _EMA8308_UDP_Tdata
    {
        u8   card_name[7];          // card_name={'E', 'M', 'A', '8', '3', '0', '8' }
        u8   password[8];           // password,8 words
        u8   command;               // command（EMA8308 UDP COMMAND LIST）
        Data      data_in;          // maximum 32 bytes
    }EMA8308_data;

typedef union _Data
    {
        u8   data_b[32];            //Data for byte
        u16  data_w[16];            //Data for word
        u32  data_l[8];             //Data for long
        u8   IP[4];                 //Data（IP Address）
        u16  socket_port;           //Data（Socket Port）
        u8   New_password[8];       //Data（New password,8 words）
        u8   MAC[6];                //Data（MAC Address）
        Analog_Data  Data;          //Data(Analog data, maximum 25 byte)
        WDT_DATA  WDT;              //Data(WDT data, maximum 6 byte)
        }Data;
```

8.2 Module to host command format

IP header	UDP header	UDP data		
		Data	Flag	Command
20bytes	8bytes	32bytes	1byte	1byte

As shown above, the command format from module to host is an UDP format, the first 20 bytes is the IP header, the next 8 byes is the UDP header then follows 34 bytes UDP data. The UDP data is defined as follows:

```
typedef struct _EMA8308_Rdata
{
    Receive_Data  Data       // Receive Data maximum 32byte
    u8   success_flag;      // Flag （0:Send command Failed   0x63:Send command successfully）
    u8   command            // command （EMA8308/A/D/DA UDP COMMAND LIST）
}EMA8308_receive;
```

The Receive_Data is defined as:

```
typedef   union _Receive_Data
{
        u8   data_b[32];          //Data for byte
        u16  data_w[16];          //Data for word
        u32  data_l[8];           //Data for long
        u8   Card_Type;           //Data（Card Type）
        Analog_Data   Data;       //Data(Analog data, maximum 25 byte)
        WDT_DATA   WDT;           //Data(WDT data, maximum 6 byte)
        u16 Version               //Data ( firmware version )
} Receive_Data

typedef struct _Analog_Data
    {
        u8   port[2];                 //Data(D/A port and A/D port)
        u8   channel[2];              //Data(D/A channels and A/D channels)
        u16  da_data[2];              //Data(D/A data,dir)
        u32  ad_data[4];              //Data(A/D data)
        u8   analog_config;           //Data(filter and A/D mode)
    }Analog_Data;

typedef struct _WDT_DATA
{
    u16  Timer_value;           // WDT timer value
    i16  DA_Data[2];            // WDT timer out DA output Voltage
    u8 state;
```

```
} WDT_DATA;
```

8.3 Definition of IP header

The IP header is defined as follows:

```
struct ipheader
{
    unsigned char ip_hl:4, ip_v:4; /* this means that each member is 4 bits */
    unsigned char ip_tos;    // type of service
    unsigned short int ip_len; //IP header total length
    unsigned short int ip_id; // identification
    unsigned short int ip_off; //fragment offset
    unsigned char ip_ttl; // time to live
    unsigned char ip_p; //protocol
    unsigned short int ip_sum; //header checksum
    unsigned int ip_src; //source ip address
    unsigned int ip_dst; //destination ip address
}; /* total ip header length: 20 bytes (=160 bits) */
```

8.4 Definition of UDP header

The UDP header is defined as follows:

```
struct udpheader
{
    unsigned short int uh_sport; // source port number
    unsigned short int uh_dport; //destination port number
    unsigned short int uh_len; // UDP package length
    unsigned short int uh_check; //UDP checksum
}; /* total udp header length: 8 bytes (=64 bits) */
```

8.5 EMA-8308 communication commands

- **GET_CARD_TYPE**

Function: ask the EMA-8308 module type

Host to Ethernet module:

IP header	UDP header	UDP data			
		Card_name	Password	0x1	Data
20bytes	8bytes	7bytes	8bytes	1byte	32bytes

Card_name: 'E', 'M', 'A', '8', '3', '0', '8'

Password: un-used

Command: 0x1

Data: un-used

Ethernet module to Host:

IP header	UDP header	UDP data		
		Data	Flag	0x1
20bytes	8bytes	32bytes	1byte	1byte

Command: 0x1

Flag: = 0x0 //command fail (this is not EMA-8308)

= 0x63 //command successful (this is EMA-8308)

Data: Data. Card_Type = 0x1 //Card Type：EMA-8308D

Data. Card_Type = 0x3 // Card Type：EMA-8308

- **REBOOT**

Function: reboot EMA-8308 module

Host to Ethernet module:

IP header	UDP header	UDP data			
		Card_name	Password	0x2	Data
20bytes	8bytes	7bytes	8bytes	1byte	32bytes

Card_name: ‘E’, ‘M’, ‘A’, ‘8’, ‘3’, ‘0’, ‘8’

Password: your password

Command: 0x2

Data: un-used

Ethernet module to Host:

IP header	UDP header	UDP data		
		Data	Flag	0x2
20bytes	8bytes	32bytes	1byte	1byte

Command: 0x2

Flag: = 0x0 //command fail

= 0x63 //command successful

Data: un-used

- **SOCKETPORT_CHANGE**

Function: change socket port of EMA-8308 module

Host to Ethernet module:

IP header	UDP header	UDP data			
		Card_name	Password	0x3	Data
20bytes	8bytes	7bytes	8bytes	1byte	32bytes

Card_name: 'E', 'M', 'A', '8', '3', '0', '8'

Password: your password

Command: 0x3

Data: socket_port //your new socket port number

other structure members are un-used.

Parameter	Type	Description
socket_port	u16	socket port number

Ethernet module to Host:

IP header	UDP header	UDP data		
		Data	Flag	0x3
20bytes	8bytes	32bytes	1byte	1byte

Command: 0x3

Flag: = 0x0 //command fail

= 0x63 //command successful

Data: un-used

- **PASSWORD_CHANGE**

Function: change password of EMA-8308 module

Host to Ethernet module:

IP header	UDP header	UDP data			
		Card_name	Password	0x4	Data
20bytes	8bytes	7bytes	8bytes	1byte	32bytes

Card_name: 'E', 'M', 'A', '8', '3', '0', '8'

Password: your password

Command: 0x4

Data: password[8] //your new password

other structure members are un-used.

Parameter	Type	Description
password[8]	u8	new password to be set

Ethernet module to Host:

IP header	UDP header	UDP data		
		Data	Flag	0x4
20bytes	8bytes	32bytes	1byte	1byte

Command: 0x4

Flag: = 0x0 //command fail

= 0x63 //command successful

Data: un-used

- **PASSWORD_RESTORE**

Function: restore password of EMA-8308 module

Host to Ethernet module:

IP header	UDP header	UDP data			
		Card_name	Password	0x5	Data
20bytes	8bytes	7bytes	8bytes	1byte	32bytes

Card_name: 'E', 'M', 'A', '8', '3', '0', '8'

Password: your password

Command: 0x5

Data: un-used.

Ethernet module to Host:

IP header	UDP header	UDP data		
		Data	Flag	0x5
20bytes	8bytes	32bytes	1byte	1byte

Command: 0x5

Flag: = 0x0 //command fail

= 0x63 //command successful

Data: un-used

- **IP_CHANGE**

Function: change IP of EMA-8308 module

Host to Ethernet module:

IP header	UDP header	UDP data			
		Card_name	Password	0x6	Data
20bytes	8bytes	7bytes	8bytes	1byte	32bytes

Card_name: ‘E’, ‘M’, ‘A’, ‘8’, ‘3’, ‘0’, ‘8’

Password: your password

Command: 0x6

Data: IP[4] //new IP address

other structure members are un-used.

Parameter	Type	Description
IP[4]	u8	new IP address to be set

Ethernet module to Host:

IP header	UDP header	UDP data		
		Data	Flag	0x6
20bytes	8bytes	32bytes	1byte	1byte

Command: 0x6

Flag: = 0x0 //command fail

= 0x63 //command successful

Data: un-used

- **GET_FIRMWARE_VERSION**

Function: read the firmware version of EMA-8308 module

Host to Ethernet module:

IP header	UDP header	UDP data			
		Card_name	Password	0x7	Data
20bytes	8bytes	7bytes	8bytes	1byte	32bytes

Card_name: ‘E’, ‘M’, ‘A’, ‘8’, ‘3’, ‘0’, ‘8’

Password: your password

Command: 0x7

Data: un-used

Ethernet module to Host:

IP header	UDP header	UDP data		
		Data	Flag	0x7
20bytes	8bytes	32bytes	1byte	1byte

Command: 0x7

Flag: = 0x0 //command fail

= 0x63 //command successful

Data: Version[0 ~ 1] // firmware version

Parameter	Type	Description
Version[1] Version[0]	u16	Version x.y x: Version[1] y: Version[0]

- **MAC_SET**

Function: the MAC address of EMA-8308 module

Host to Ethernet module:

IP header	UDP header	UDP data			
		Card_name	Password	0xfa	Data
20bytes	8bytes	7bytes	8bytes	1byte	32bytes

Card_name: 'E', 'M', 'A', '8', '3', '0', '8'

Password: your password

Command: 0xfa

Data: MAC[6] //new MAC address

other structure members are un-used.

Parameter	Type	Description
MAC[6]	u8	MAC address to be set

Ethernet module to Host:

IP header	UDP header	UDP data		
		Data	Flag	0xfa
20bytes	8bytes	32bytes	1byte	1byte

Command: 0xfa

Flag: = 0x0 //command fail

= 0x63 //command successful

Data: un-used

- **DA_PORT_SET**

Function: set the DA port

Host to Ethernet module:

IP header	UDP header	UDP data			
		Card_name	Password	0x40	Data
20bytes	8bytes	7bytes	8bytes	1byte	32bytes

Card_name: ‘E’, ‘M’, ‘A’, ‘8’, ‘3’, ‘0’, ‘8’

Password: your password

Command: 0x40

Data: Data. DA_Data [0 ~ 1]

other structure members are un-used.

Parameter	Type	Description
Data. DA_Data[0]	i16	D/A channel0 data
Data. DA_Data[1]	i16	D/A channel1 data

Ethernet module to Host:

IP header	UDP header	UDP data		
		Data	Flag	0x40
20bytes	8bytes	32bytes	1byte	1byte

Command: 0x40

Flag: = 0x0 //command fail

= 0x63 //command successful

Data: un-used

- **DA_PORT_READ**

Function: read the DA port

Host to Ethernet module:

IP header	UDP header	UDP data			
		Card_name	Password	0x41	Data
20bytes	8bytes	7bytes	8bytes	1byte	32bytes

Card_name: ‘E’, ‘M’, ‘A’, ‘8’, ‘3’, ‘0’, ‘8’

Password: your password

Command: 0x41

Data: un-used

Ethernet module to Host:

IP header	UDP header	UDP data		
		Data	Flag	0x41
20bytes	8bytes	32bytes	1byte	1byte

Command: 0x41

Flag: = 0x0 //command fail (this is not EMA-8308)

= 0x63 //command successful (this is EMA-8308)

Data: Data. DA_Data [0 ~ 1] // D/A channel(0 ~ 1) data

Name	Type	Description	
DA_Data	i16	model	EMA8308, EMA8308D
		resolution	16bit DA
		data range	-32768 ~ 32767
		output	data
		0V	0
		+10V	32767
		-10V	-32768

- **DA_CHANNEL_SET**

Function: to set the DA channel

Host to Ethernet module:

IP header	UDP header	UDP data			
		Card_name	Password	0x42	Data
20bytes	8bytes	7bytes	8bytes	1byte	32bytes

Card_name: 'E', 'M', 'A', '8', '3', '0', '8'

Password: your password

Command: 0x42

Data: Data. Channel [1], Data.da_data [0]

other structure members are un-used.

Parameter	Type	Description
Data. Channel[1]	u8	DA channel select 0: channel 0 1: channel 1
Data.da_data [0]	i16	DA data to be set

Ethernet module to Host:

IP header	UDP header	UDP data		
		Data	Flag	0x42
20bytes	8bytes	32bytes	1byte	1byte

Command: 0x42

Flag: = 0x0 //command fail (this is not EMA-8308)

= 0x63 //command successful (this is EMA-8308)

- **DA_CHANNEL_READ**

Function: to read the DA channel

Host to Ethernet module:

IP header	UDP header	UDP data			
		Card_name	Password	0x43	Data
20bytes	8bytes	7bytes	8bytes	1byte	32bytes

Card_name: ‘E’, ‘M’, ‘A’, ‘8’, ‘3’, ‘0’, ‘8’

Password: your password

Command: 0x43

Data: Data. Channel [1]

other structure members are un-used.

Parameter	Type	Description
Data. Channel[1]	u8	DA channel select 0: channel 0 1: channel 1

Ethernet module to Host:

IP header	UDP header	UDP data		
		Data	Flag	0x43
20bytes	8bytes	32bytes	1byte	1byte

Command: 0x43

Flag: = 0x0 //command fail (this is not EMA-8308)

= 0x63 //command successful (this is EMA-8308)

Data: Data. Channel [1],Data.da_data[0]

Parameter	Type	Description
Data. Channel[1]	u8	DA channel select 0: channel 0 1: channel 1
Data.da_data[0]	i16	DA data

- **AD_PORT_READ**

Function: to read the AD port

Host to Ethernet module:

IP header	UDP header	UDP data			
		Card_name	Password	0x50	Data
20bytes	8bytes	7bytes	8bytes	1byte	32bytes

Card_name: 'E', 'M', 'A', '8', '3', '0', '8'

Password: your password

Command: 0x50

Data: Data. Port [0 ~ 1]

other structure members are un-used.

Parameter	Type	Description
Data. Port [0]	u8	Choose A/ D port 0: channel 00 ~ 07 1:channel 10 ~ 17
Data. Port [1]	u8	0 : Choose A/D channel0 ~ channel3 1 : Choose A/D channel4 ~ channel7

Ethernet module to Host:

IP header	UDP header	UDP data		
		Data	Flag	0x50
20bytes	8bytes	32bytes	1byte	1byte

Command: 0x50

Flag: = 0x0 //command fail (this is not EMA-8308)

= 0x63 //command successful (this is EMA-8308)

Data: Data.ad_data [0 ~ 3]

Parameter	Type	Description
Data.ad_data [0 ~3]	i16	A/D data(channel ~ channel+3)

AD_Data data format：

MSB				LSB
Bit31	Bit30	Bit29	Bit28 ~ Bit5	Bit4 ~ Bit0
/EOC	DMY	SIG	AD_Data	unused

Input Voltage range	/EOC	DMY	SIG	Bit28	AD_DATA(Bit4 ~ Bit0 unused)
V_{in} > +10V	0	0	1	1	0x0
+10V > V_{in} > 0V	0	0	1	0	0xFFF_FFFF ~ 0x0
0V > V_{in} > -10V	0	0	0	1	0xFFF_FFFF ~ 0x0
V_{in} > -10V	0	0	0	0	0xFFF_FFFF

- **AD_CHANNEL_READ**

Function: to read the AD channel

Host to Ethernet module:

IP header	UDP header	UDP data			
		Card_name	Password	0x51	Data
20bytes	8bytes	7bytes	8bytes	1byte	32bytes

Card_name: 'E', 'M', 'A', '8', '3', '0', '8'

Password: your password

Command: 0x51

Data: Data. Channel [0] = 0 ~ 1; //Choose A/D port

Data. Channel [1] = 0 ~ 7; //Choose A/D channel

other structure members are un-used.

Parameter	Type	Description
Data. Channel [0]	u8	Choose A/ D port
Data. Channel [1]	u8	Choose A/D channel

Ethernet module to Host:

IP header	UDP header	UDP data		
		Data	Flag	0x51
20bytes	8bytes	32bytes	1byte	1byte

Command: 0x51

Flag: = 0x0 //command fail (this is not EMA-8308)

= 0x63 //command successful (this is EMA-8308)

Data: Data.ad_data [0]

Parameter	Type	Description
Data.ad_data [0]	u16	A/D data

AD_Data data format：

MSB				LSB
Bit31	Bit30	Bit29	Bit28 ~ Bit5	Bit4 ~ Bit0
/EOC	DMY	SIG	AD_Data	unused

Input Voltage range	/EOC	DMY	SIG	Bit28	AD_DATA(Bit4 ~ Bit0 unused)
V_{in} > +10V	0	0	1	1	0x0
+10V > V_{in} > 0V	0	0	1	0	0xFFF_FFFF ~ 0x0
0V > V_{in} > -10V	0	0	0	1	0xFFF_FFFF ~ 0x0
V_{in} > -10V	0	0	0	0	0xFFF_FFFF

- **AD_MODE_SET**

Function: to set the AD mode

Host to Ethernet module:

IP header	UDP header	UDP data			
		Card_name	Password	0x52	Data
20bytes	8bytes	7bytes	8bytes	1byte	32bytes

Card_name: 'E', 'M', 'A', '8', '3', '0', '8'

Password: your password

Command: 0x52

Data: Data. analog_config = 0 ~ 3; //A/D mode

other structure members are un-used.

Name	Type	Description
analog_config	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

Ethernet module to Host:

IP header	UDP header	UDP data		
		Data	Flag	0x52
20bytes	8bytes	32bytes	1byte	1byte

Command: 0x52

Flag: = 0x0 //command fail (this is not EMA-8308)

= 0x63 //command successful (this is EMA-8308)

Data: un-used

- **AD_MODE_READ**

Function: to read the AD mode

Host to Ethernet module:

IP header	UDP header	UDP data			
		Card_name	Password	0x53	Data
20bytes	8bytes	7bytes	8bytes	1byte	32bytes

Card_name: ‘E’, ‘M’, ‘A’, ‘8’, ‘3’, ‘0’, ‘8’

Password: your password

Command: 0x53

Data: un-used

Ethernet module to Host:

IP header	UDP header	UDP data		
		Data	Flag	0x53
20bytes	8bytes	32bytes	1byte	1byte

Command: 0x53

Flag: = 0x0 //command fail (this is not EMA-8308)

= 0x63 //command successful (this is EMA-8308)

Data: Data. analog_config = 0 ~ 3 //A/D mode

Name	Type	Description
analog_config	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

- **AD_FILTER_SET**

Function: to set the AD filter

Host to Ethernet module:

IP header	UDP header	UDP data			
		Card_name	Password	0x54	Data
20bytes	8bytes	7bytes	8bytes	1byte	32bytes

Card_name: ‘E’, ‘M’, ‘A’, ‘8’, ‘3’, ‘0’, ‘8’

Password: your password

Command: 0x54

Data: Data. analog_config = 0 ~ 3; // filter mode

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

Ethernet module to Host:

IP header	UDP header	UDP data		
		Data	Flag	0x54
20bytes	8bytes	32bytes	1byte	1byte

Command: 0x54

Flag: = 0x0 //command fail (this is not EMA-8308)

= 0x63 //command successful (this is EMA-8308)

Data: un-used

- **AD_FILTER_READ**

Function: to read the AD filter

Host to Ethernet module:

IP header	UDP header	UDP data			
		Card_name	Password	0x55	Data
20bytes	8bytes	7bytes	8bytes	1byte	32bytes

Card_name: ‘E’, ‘M’, ‘A’, ‘8’, ‘3’, ‘0’, ‘8’

Password: your password

Command: 0x55

Data: un-used

Ethernet module to Host:

IP header	UDP header	UDP data		
		Data	Flag	0x55
20bytes	8bytes	32bytes	1byte	1byte

Command: 0x55

Flag: = 0x0 //command fail (this is not EMA-8308)

= 0x63 //command successful (this is EMA-8308)

Data: Data. analog_config = 0 ~ 3 // filter mode

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

- **WDT_ENABLE**

Function: to enable WDT (watch dog timer)

Host to Ethernet module:

IP header	UDP header	UDP data			
		Card_name	Password	0x60	Data
20bytes	8bytes	7bytes	8bytes	1byte	32bytes

Card_name: ‘E’, ‘M’, ‘A’, ‘8’, ‘3’, ‘0’, ‘8’

Password: your password

Command: 0x60

Data: un-used

Ethernet module to Host:

IP header	UDP header	UDP data		
		Data	Flag	0x60
20bytes	8bytes	32bytes	1byte	1byte

Command: 0x60

Flag: = 0x0 //command fail (this is not EMA-8308)

= 0x63 //command successful (this is EMA-8308)

Data: un-used

- **WDT_DISABLE**

Function: to disable WDT (watch dog timer)

Host to Ethernet module:

IP header	UDP header	UDP data			
		Card_name	Password	0x61	Data
20bytes	8bytes	7bytes	8bytes	1byte	32bytes

Card_name: ‘E’, ‘M’, ‘A’, ‘8’, ‘3’, ‘0’, ‘8’

Password: your password

Command: 0x61

Data: un-used

Ethernet module to Host:

IP header	UDP header	UDP data		
		Data	Flag	0x61
20bytes	8bytes	32bytes	1byte	1byte

Command: 0x61

Flag: = 0x0 //command fail (this is not EMA-8308)

= 0x63 //command successful (this is EMA-8308)

Data: un-used

- **WDT_SET**

Function: to set WDT (watch dog timer) data

Host to Ethernet module:

IP header	UDP header	UDP data			
		Card_name	Password	0x62	Data
20bytes	8bytes	7bytes	8bytes	1byte	32bytes

Card_name: 'E', 'M', 'A', '8', '3', '0', '8'

Password: your password

Command: 0x62

Data: Data.WDT .DA_data[0 ~ 1],Data.WDT .Timer_value

Name	Type	Description	
WDT .Timer_value	u16	Set the WDT wait time.(10~10000 minisecond) based on 0.1 sec time base. default: 10 (1s)	
WDT.DA_Data[0~1]	i16	Set the output default state, the state will keep while the connection failure. DA_Data [0]: DA0 value DA_Data [1]: DA1 value	
DA_Data	i16	model	EMA8308, EMA8308D
		resolution	16bit DA
		data range	-32768 ~ 32767
		output	data
		0V	0
		+10V	32767
		-10V	-32768

Ethernet module to Host:

IP header	UDP header	UDP data		
		Data	Flag	0x62
20bytes	8bytes	32bytes	1byte	1byte

Command: 0x62

Flag: = 0x0 //command fail (this is not EMA-8308)

= 0x63 //command successful (this is EMA-8308)

Data: un-used

- **WDT_READ**

Function: to read WDT (watch dog timer) data

Host to Ethernet module:

IP header	UDP header	UDP data			
		Card_name	Password	0x63	Data
20bytes	8bytes	7bytes	8bytes	1byte	32bytes

Card_name: 'E', 'M', 'A', '8', '3', '0', '8'

Password: your password

Command: 0x63

Data: un-used

Ethernet module to Host:

IP header	UDP header	UDP data		
		Data	Flag	0x63
20bytes	8bytes	32bytes	1byte	1byte

Command: 0x63

Flag: = 0x0 //command fail (this is not EMA-8308)

= 0x63 //command successful (this is EMA-8308)

Data: u8 WDT .state ,WDT .DA_Data[0 ~ 1], WDT .Timer_value

Name	Type	Description	
WDT .Timer_value	u16	Set the WDT wait time.(10~10000 minisecond) based on 0.1 sec time base. default: 10 (1s)	
WDT.DA_Data[0~1]	u8i16	Set the output default state, the state will keep while the connection failure. DA_Data [0]: DA0 value DA_Data [1]: DA1 value	
WDT.state	u8	0: WDT disable 1: WDT enable	
DA_Data	i16	model	EMA8308, EMA8308D
		resolution	16bit DA
		data range	-32768 ~ 32767
		output	data
		0V	0
		+10V	32767
		-10V	-32768

9. 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_mode_set( )	Configure each port as differential or single end
7.	EMA8308_AD_mode_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_channel_set( )	Set the value of the D/A data
15.	EMA8308_DA_channel_read( )	Read back the value of the D/A data
16.	EMA8308_DA_port_set( )	Set the setting of the D/A data
17.	EMA8308_DA_port_read( )	Read back the setting of the D/A data
18.	EMA8308_socket_port_change( )	To change the communicate port number of EMA8308
19.	EMA8308_IP_change( )	To change the communicate IP of EMA8308
20.	EMA8308_reboot( )	To reboot EMA8308
21.	EMA8308_security_unlock( )	Unlock security
22.	EMA8308_security_status_read( )	Read lock status
23.	EMA8308_password_change( )	Change password
24.	EMA8308_password_set_default( )	Rest to factory default password
25.	EMA8308_WDT_set( )	Set WDT(watch dog timer) configuration.
26.	EMA8308_WDT_read( )	Read back WDT(watch dog timer) configuration
27.	EMA8308_WDT_enable( )	enable WDT(watch dog timer)
28.	EMA8308_WDT_disable( )	disable WDT(watch dog timer)

10. EMA8308 Error codes summary

10.1 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

11. EMA8308 UDP command list

Command	R/W	Descriptions	Mnemonics
0x1	R	Read Card Type	CARD TYPE READ
0x2	W	Soft Reboot	REBOOT
0x3	W	Change Socket Port	SOCKET PORT CHANGE_
0x4	W	Change Password	PASSWORD CHANGE
0x5	W	Restore Password	PASSWORD SET DEFAULT
0x6	W	Change IP Address	IP CHANGE
0x7	R	Read firmware version	FIRMWARE VERSION READ
0xFA	W	Set MAC Address	MAC SET
0xFB	W	Set offset and gain	CALIBRATION SET
0xFC	R	Read offset and gain	CALIBRATION READ
0x40	W	Set D/A port data	DA PORT SET
0x41	R	Readback D/A port data	DA PORT READ
0x42	W	Set D/A channel data	DA CHANNEL SET
0x43	R	Readback D/A channel data	DA CHANNEL READ
0x50	R	Readback A/D Port data	AD PORT READ
0x51	R	Readback A/D Channel data	AD CHANNEL READ
0x52	W	Set A/D Port Mode	AD MODE SET
0x53	R	Readback A/D Port Mode	AD MODE READ
0x54	W	Set A/D Filter	AD FILTER SET
0x55	R	Read A/D Filter	AD FILTER READ
0x60	W	Enable WDT	ENABLE WDT
0x61	W	Disable WDT	DISABLE WDT
0x62	W	Set WDT	SET WDT
0x63	R	Read WDT	READ WDT

12. Error codes table for UDP success_flag

Error code	Symbolic Name	Description
99	SUCCESS	No Error
100	COMMAND_ERROR	Command Error
101	PASSWORD_ERROR	Password Error
102	CHANGE_IP_ERROR	Set IP value error out of range Range：1 ~ 254
103	CHANGE_SOCKET_ERROR	Set socket port error out of range Range：value > 1000
104	CHANGE_MAC_ERROR	Set mac address error mac != 0xFFFF_FFFF_FFFF
120	PORT_ERROR	Choose Port error out of range Range：port < port_max
121	CHANNEL_ERROR	Choose channel error out of range Range：channel < channel_max
122	STATE_ERROR	Set state error
123	TIMER_VALUE_ERROR	Set Timer value error out of range Range：value_min < value < value_max
124	MODE_ERROR	Choose mode error out of range Range：mode > mode_max