Ezi-IO® Ethernet
Input/Output Module AD

User Manual

Communication Function

(**Rev.01**)

FASTECH

Table of Contents

1 . Communication Protocol

1 - 1 . Communication Function

Ezi-IO Ethernet AD products can control up to 254(1~254) modules through using Ethernet communication.

1 - 1 - 1 . Communication Specifications

Item	Specifications
Communication Speed	10/100base-T/TX
Communication Type(Protocol)	TCP (Port No. : **2001,2002**)
	UDP (Port No. : **3001,3002**)
Max. Cablin Length	100m
Min. Cabling Length between drives	More than 20cm
Number of Connected Axes	254 axes (No. 01~FE)

- Port No. 2001, 3001 : For GUI
- Port No. 2002, 3002 : For User Library
- Port No. 2001, 3001 cannot be used when using provided User Library file.

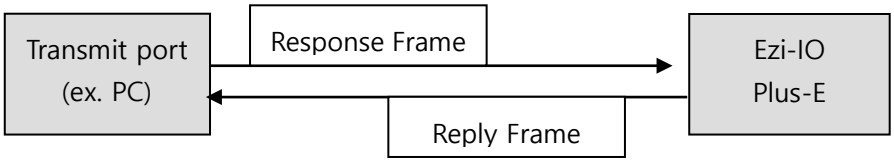
1 - 1 - 2 . Ethernet IP Address

1) Subnet Mask : 255.255.255.0
2) Gateway : 192.168.0.1
3) IP address : 192.168.0.xxx (xxx is set by external switch.)

- When connecting Ezi-IO Ethernet AD directly from PC or Ethernet device, be sure to set the Network setting according to the above IP address.
  If the setting is not set or different, connection is not available.
- If the switches are to set 255(FF), IP Address is automatically set.
  Because DHCP is used, IP Address is set automatically when using router.
  (Connect the Ethernet to Ethernet IN connector.)
- Be sure to set the IP Address by switches, when connecting directly to the controller(PC, /PLC).
- Set the IP Address automatically only when default IP Address is not used.
  When IP is set automatically, connect the user program(GUI) and save the IP Address.
  Then, turn off the power and set the last number of IP with the switch.
- When the switch is set to 0, the IP setting becomes the initial (default) value.

1 - 1 - 3 . Ethernet Protocol

1) Overview of communication FRAME



2) Basic structure of FRAME

UDP Header	*Frame Data*
8 bytes	5~257 bytes

UDP Header contains the following information :

① Transmit port number : 2 bytes
② Receiving port number : 2 bytes
③ Date length : 2 bytes, Total legth of UDP Header and Frame Data
④ Checksum : 2 bytes

1 - 1 - 4 . Receiving Frame Data Structure

The detailed configuration of the receiving ***Frame Data*** is as follows.

Header	Length	Sync No.	Reserved	Frame type	Data
1 byte	1 byte	1 byte	1 byte (0x00)	1 byte	0 ~ 252 bytes

① Header : 0xAA, Displaying the beginning of the Frame.
② Length : Length of Data after Length
(Sync No. + Reserved + Frame type + Data)
③ Reserved : 1 byte (Input as "0x00")
④ Sync No. : Checking whether the command is executed as the Sync number of the packet in the drive module.
The value must be changed every time a new command is sent.
⑤ Frame type : Specifying the command type of the Frame.
Refer to「Frame type and Data configuration」section for command types.
⑥ Data : The data structure and length of this clause are determined by the Frame type.
Refer to「Frame type and Data configuration」section for detailed structure.

1 - 1 - 5 . Response Frame Stucture and Communication Error

Basic structure of the Frame at the response side is same as receiving side. However, 'communication status' is added in the ***Frame Data*** clause.

<table>
<tr><th>Header</th><th>Length</th><th>Sync No.</th><th>Reserved</th><th>Frame type</th><th colspan="2">Data</th></tr>
<tr><td rowspan="2">1 byte</td><td rowspan="2">1 byte</td><td rowspan="2">1 byte</td><td rowspan="2">1 byte (0x00)</td><td rowspan="2">1 byte</td><td>1 byte</td><td>0 ~ 251 bytes</td></tr>
<tr><td>Communication Status</td><td>Response data</td></tr>
</table>

① Header : 0xAA, Displaying the beginning of the Frame.

② Length : Length of Data after Length
(Sync No. + Reserved + Frame type + Data)

③ Sync No. : Same as Response Frame
(Recognize as error status, if Sync No. does not match with the data at the time of reception.)

④ Reserved : 1 byte(0x00)

⑤ Frame type : Same as Response Frame
(Recognize as error status, if Frame type does not match with the data at the time of transmission.)

⑥ Data : In reply, 1 byte of data indicating communication status(error/normal) is included.
Simple execution command has communication stauts Data only.

The contents of byte indicating communication status are as follows.

Hexa code	Decimal code	Description
0x00	0	Communication is in normal state.
0x80	128	Frame type error : Received Frame type command cannot be recognized.
0x81	129	Data error, ROM data reading/writimg error : The received data is out of specified range.
0x82	130	Response Frame error: Received Frame is out of specification.

Caution	**1) If 'Header' and 'Length' value of response Frame is abnormal, there is no response from the drive.** **2) If the '130' communication status error is occurred, the size of response date is 0 btye.**

1 - 2 . Structure of Frame

1 - 2 - 1 . Frame type Data Configuration

(1) The following table displays the content and configuration of data by Frame type.

● 0xXX of Frame type is value of Hexa, the value in () is Decimal.

<table>
<tr><th>Frame type</th><th>Library Name</th><th>Description</th></tr>
<tr><td>0x01
(1)</td><td>FAS_
GetSlaveInfo</td><td>
Requiring type of connected slave and program version information.

Sending : 0 byte

Response : 1~248 bytes
<table>
<tr><td>1 byte</td><td>1 byte</td><td>0~246 bytes</td></tr>
<tr><td>Communication Status</td><td>Slave type</td><td>ACII string with NULL byte (strlen() + 1 byte)</td></tr>
</table>
◆ Slave type
170 : Ezi-IO-EN-AD08 series
</td></tr>
<tr><td>0x5A
(90)</td><td>FAS_SetADConfig</td><td>
Setting parameter values required to convert anlaog to digital for each CH.

Sending : 6 byte
<table>
<tr><td>1 byte</td><td>1 byte</td><td>4 bytes</td></tr>
<tr><td>Selecting CH</td><td>Parameter number</td><td>Parameter value</td></tr>
</table>
<table>
<tr><th>CH No.</th><th>CH select input value</th><th>CH No.</th><th>CH select Input value</th></tr>
<tr><td>1</td><td>0</td><td>5</td><td>4</td></tr>
<tr><td>2</td><td>1</td><td>6</td><td>5</td></tr>
<tr><td>3</td><td>2</td><td>7</td><td>6</td></tr>
<tr><td>4</td><td>3</td><td>8</td><td>7</td></tr>
</table>
<table>
<tr><th>Parameter No.</th><th>Function</th><th>Min. Value</th><th>Max. Value</th><th>Default Value</th></tr>
<tr><td>0</td><td>Analog input range (Conversion range)</td><td>0</td><td>4</td><td>0</td></tr>
<tr><td>1</td><td>Filter buffer length</td><td>0</td><td>1000</td><td>0</td></tr>
<tr><td>2</td><td>AD Conversion Offset</td><td>-1000</td><td>1000</td><td>0</td></tr>
</table>
◆ No. 0 AD input range is set as No.1 parameter value, when No.1 of input range switch(SW3) is OFF. When No.1 of SW3 is ON, it is not.
Even one CH is set as current input by current/voltage mode switch, OFF the No.1 of SW3.
0 : -10 ~ 10[V]
1 : -5 ~ 5[V]
</td></tr>
</table>

<table>
<tr><th>Frame type</th><th>Library Name</th><th>Description</th></tr>
<tr><td></td><td></td><td>
2 : -2.5~ 2.5[V]
3 : 0 ~ 10[V]
4 : 0 ~ 20[mA] (Select '4' for current input)

Response : 7 byte
<table>
<tr><td>1 byte</td><td>1 byte</td><td>1 byte</td><td>4 bytes</td></tr>
<tr><td>Communication Status</td><td>Selecting CH</td><td>Parameter No.</td><td>DATA value</td></tr>
</table>
Returning setting value
AD input value is returned currently set value accoding to No.1 of SW3.
</td></tr>
<tr><td>0x5B
(91)</td><td>FAS_GetADConfig</td><td>
Reading CH parameter setting value.

Sending : 4 bytes
<table>
<tr><td>1 byte</td><td>1 byte</td></tr>
<tr><td>Selecting CH</td><td>Parameter No.</td></tr>
</table>
<table>
<tr><td>CH No.</td><td>CH select input value</td><td>CH No.</td><td>CH select input value</td></tr>
<tr><td>1</td><td>0</td><td>5</td><td>4</td></tr>
<tr><td>2</td><td>1</td><td>6</td><td>5</td></tr>
<tr><td>3</td><td>2</td><td>7</td><td>6</td></tr>
<tr><td>4</td><td>3</td><td>8</td><td>7</td></tr>
</table>
<table>
<tr><td>Parmater No.</td><td>Function</td></tr>
<tr><td>0</td><td>Filter length</td></tr>
<tr><td>1</td><td>AD Offset</td></tr>
<tr><td>2</td><td>Analog input range
(Conversion range)</td></tr>
</table>
◆ No.2 analaog input range
Setting value reads 16~19 accroding to switch setting, when input range is set by input range setting switch(SW3).
<table>
<tr><td>Setting Value</td><td>Range</td><td>Setting Value</td><td>Range</td></tr>
<tr><td>0</td><td>-10 ~ 10[V]</td><td>16</td><td>-10 ~ 10[V]</td></tr>
<tr><td>1</td><td>-5 ~ 5[V]</td><td>17</td><td>-5 ~ 5[V]</td></tr>
<tr><td>2</td><td>-2.5 ~ 2.5[V]</td><td>18</td><td>-2.5 ~ 2.5[V]</td></tr>
<tr><td>3</td><td>0 ~ 10[V]</td><td>19</td><td>0 ~ 10[V]</td></tr>
<tr><td>4</td><td>0 ~ 20[mA]</td><td></td><td></td></tr>
</table>
Response : 1 byte
<table>
<tr><td>1 byte</td><td>4 bytes</td></tr>
<tr><td>Communication Status</td><td>Parameter value</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Frame type</th><th>Library Name</th><th>Description</th></tr>
<tr><td>0x5C
(92)</td><td>FAS_ReadADValue</td><td>
Loading AD conversion value of 1 CH

Sending : 1bytes
<table>
<tr><td>1 byte</td></tr>
<tr><td>Selecting CH</td></tr>
</table>
<table>
<tr><th>CH No.</th><th>CH select input value</th><th>CH No.</th><th>CH select input value</th></tr>
<tr><td>1</td><td>0</td><td>5</td><td>4</td></tr>
<tr><td>2</td><td>1</td><td>6</td><td>5</td></tr>
<tr><td>3</td><td>2</td><td>7</td><td>6</td></tr>
<tr><td>4</td><td>3</td><td>8</td><td>7</td></tr>
</table>
Response : 3 bytes
<table>
<tr><td>1 byte</td><td>2 bytes</td></tr>
<tr><td>Communication Status</td><td>AD conversion value</td></tr>
</table>
</td></tr>
<tr><td>0x5D
(93)</td><td>FAS_ReadADAll Value</td><td>
Loading AD conversion value of 8CH.

Sending : 1bytes
<table>
<tr><td>1 byte</td></tr>
<tr><td>offset</td></tr>
</table>
Send the Offset value as ‘0’.

Response : 17 bytes
<table>
<tr><td>1 byte</td><td>2 bytes</td><td>2 bytes</td><td>·○·○·○</td><td>2 bytes</td><td>2 bytes</td></tr>
<tr><td>Communication Status</td><td>AD 1CH</td><td>AD 2CH</td><td>·○·○·○</td><td>AD 7CH</td><td>AD 8CH</td></tr>
</table>
</td></tr>
<tr><td>0xCC
(204)</td><td>FAS_LoadADConfig</td><td>
Loading AD conversion setting value of ROM memory.
Changing RAM data of the board to loaded data.

Sending : 0 byte

Response : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>Communication Status</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Frame type</th><th>Library Name</th><th>Description</th></tr>
<tr><td>0xCD
(205)</td><td>FAS_SaveADConfig</td><td>Saving currently set AD conversion setting value to ROM momory. Ensuring that the values are saved even when the power is OFF.

Sending : 0 byte

Response : 1 byte

<table><tr><td>1 byte</td></tr><tr><td>Communication Status</td></tr></table></td></tr>
</table>

1 - 3 . Programming Method

There are 2 programming methods to use Ezi-IO Ethernet AD.

First method is using Visual C++ language under Window system of PC.
In this case, use the library(Refer to 「2. Library for PC program」 provided with the product.

Second method is sending the command character directly without using library. The user must create a low-level protocol program such as Protocol Test program. It is mainly used when a PLC is used as a host controller.

2 . Library for PC Programming

2 - 1 . Library Configuration

(1) For C++

C++ header file(*.h) and library file(*.lib or *.dll) are required. These files are included in "₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include₩₩". Following contects must be included in a source file for development.

#include "₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include₩₩FAS_EziMotionPlusE.h"
#include "₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include₩₩ReturnCodes_Define.h"
#include "₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include₩₩MOTION_DEFINE.h"
#include "₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include₩₩COMM_Define.h"

Library files are as follows:
1) For 32bit
"₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include₩₩EziMotionPlusE.lib"
"₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include₩₩EziMotionPlusE.dll"

2) For 64bit
"₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include_x64₩₩EziMotionPlusE.lib"
"₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include_x64₩₩EziMotionPlusE.dll"

A sample program source using library is included in
"₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩Examples₩₩C++₩₩".

(2) For C#

C# header file and library file are required. These files are included in "₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩". The following content mus be included in a source file for development.

#include "₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include₩₩MOTION_DEFINE_PlusE.cs"

Library files are as follows:
1) For 32bit
"₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include₩₩LIB_EziMOTIONPlusE.cs"
2) For 64bit
"₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include_x64₩₩LIB_EziMOTIONPlusE.cs"

A sample program source using library is included in
"₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩Examples₩₩C#₩₩".

(3) The following table describes returned values when using each library(DLL) function. The returned values can be check at the library(DLL) function. In case of programming method using protocol, this function is not provided.

Item	Name	Return value	Description
Normal	FMM_OK	0(0x00)	The function has normally performed the command.
Input Error	FMM_INVALID_SLAVE_NUM	3(0x03)	Wrong Slave number is input.
Connection Error	FMC_DISCONNECTED	5(0x05)	The board is disconnected.
	FMC_TIMEOUT_ERROR	6(0x06)	No response in 100msec.
	FMC_CRCFAILED_ERROR	7(0x07)	Checksum Error during communication.
	FMC_RECVPACKET_ERROR	8(0x08)	Protocol level error occurred in the packet received from drive.

(4) The following table shows returned values included in all libraries. The user can monitor the result(communication status, running status) judjed by the drive.
Provided to programming methods of both using library(DLL) and protocol.

Item	Name	Return Value	Description
Normal	FMP_OK	0(0x00)	Communication is normally performed.
Input Error	FMP_FRAMETYPEERROR	128 (0x80)	Command is not recognizable by the board.
	FMP_DATAERROR	129 (0x81)	Input data is out of the range.
Connection Error	FMP_PACKETERROR	130 (0x82)	Protocol level error occurs in the packet that the drive received.

2 - 2 . Board Link Function

Function Name	Description
FAS_Connect	The drive module tries to connect with UDP Protocol. : Returns TRUE if the connection is succeeded, and returns FALSE if the connection is failed.
FAS_ConnectTCP	The drive module tries to connect with TCP Protocol. : Returns TRUE if the connection is succeeded, and returns FALSE if the connection is failed.
FAS_Reconnect	Reconnect with existing IP, Protocol, iBdID.
FAS_Close	The drive tries to disconnect communication with the drive module.
FAS_GetSlaveInfo	The drive reads drive type and program version. : Returns drive type and version information.
FAS_IsSlaveExist	The drive checks whether the relvant drive exists. : Returns TRUE, if it exists. Otherwise, FALSE will returned.
FAS_IsBdIDExist	The drive checks whether BdID is used for the IP Address. : Returns TRUE, if it exists. Otherwise, FALSE will returned.
FAS_IsIPAddressExist	The drive checks whether IP Address is assigned to the BdID. : Returns TRUE, if it exists. Otherwise, FALSE will returned.
FAS_EnableLog	Controls output of the Log related to communication error.
FAS_SetLogPath	Sets the storage path of output Log : Returns TRUE if the path exists, and returns FALSE if the path does not exist or is not accessible.
FAS_SetLogLevel	Outputs Log according to set level. : Only the Log related to internal communication errors are displayed. (LOG_LEVEL_COMM)
FAS_PrintCustomLog	Outputs arbitrary Log.

FAS_Connect

FAS_Connect is a function to connect Ezi-IO Ethernet AD.

Syntax

```
BOOL FAS_Connect(
    BYTE sb1, BYTE sb2, BYTE sb3, BYTE sb4
    BYTE iBdID
);
```

Parameters

sb1~4

Enter the IP address of the drive to connect to.

ex) In case of 192.168.0.2,

sb1 = 192, sb2 = 168, sb3 = 0, sb4 = 2

iBdID

Unique ID of the board to connect to. The ID(value) is set by the user.

Same ID cannot be used like IP Address.

Return Value

When connection is succeeded, TRUE will return. Otherwise, FALSE will return.

Remarks

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcInit()
{
        BYTE sb1 = 192, sb2 = 168, sb3 = 0, sb4 = 2; // IP :192.168.0.2
        BYTE iBdID = 0; // unique board number of 192.168.0.2
        char lpBuff[256];
        int nBuffSize = 256;
        BYTE nType;
        int nRtn;

        // Try to connect
        if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
        {
                // Connection failed
                MessageBox(_T("Connection failed!"));
                return;
        }
```

```
        if (FAS_IsSlaveExist(iBdID) == FALSE)
        {
                // The board number does not exist.
                // Check the board number of Ezi-IO.
                return;
        }

        nRtn = FAS_GetSlaveInfo(iBdID, &nType, lpBuff, nBuffSize);
        if (nRtn != FMM_OK)
        {
                // Command has not been performed normally.
                // Refer to ReturnCodes_Define.h.
        }

        printf("₩tType : %d ₩n", nType);
        printf("₩tVersion : %d ₩n", lpBuff);

        // Disconnect.
        FAS_Close(iBdID);
}
```

See Also

FAS_Close

FAS_ConnectTCP

FAS_ConnectTCP is a function to connect Ezi-IO Ethernet AD to TCP Protocol.

Syntax

```
BOOL FAS_ConnectTCP(
    BYTE sb1, BYTE sb2, BYTE sb3, BYTE sb4
    int iBdID
);
```

Parameters

sb1~4

Enter the IP address of the drive to connect to.

ex) In case of 192.168.0.2,

sb1 = 192, sb2 = 168, sb3 = 0, sb4 = 2

iBdID

Unique ID of the board to connect to. The ID(value) is set by the user.

Same ID cannot be used like IP Address.

Return Value

When connection is succeeded, TRUE will return. Otherwise, FALSE will return.

Remarks

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcInit()
{
        BYTE sb1 = 192, sb2 = 168, sb3 = 0, sb4 = 2; // IP :192.168.0.2
        int iBdID = 0 // unique board number of 192.168.0.2
        char lpBuff[256];
        int nBuffSize = 256;
        BYTE nType;
        int nRtn;

        // Try to connect.
        if (FAS_ConnectTCP(sb1, sb2, sb3, sb4, iBdID) == FALSE)
        {
                // Disconnect.
                MessageBox(_T("Connection failed!"));
                return;
```

```
        }

        if (FAS_IsSlaveExist(iBdID) == FALSE)
        {
                // The board number does not exist.
                // Check the board number of Ezi-IO.
                return;
        }

        nRtn = FAS_GetSlaveInfo(iBdID, &nType, lpBuff, nBuffSize);
        if (nRtn != FMM_OK)
        {
                // Command has not been performed normally.
                // Refer to ReturnCodes_Define.h.
        }

        printf("Port : %d (board %d) ₩n", iBdID);
        printf("₩tType : %d ₩n", nType);
        printf("₩tVersion : %d ₩n", lpBuff);

        // Disconnect.
        FAS_Close(iBdID);
}
```

See Also

FAS_Close

FAS_Reconnect

This function reconnects to Ezi-IO Ethernet AD without using FAS_Connect().

Syntax

```
void FAS_Reconnect(int iBdID);
```

Parameters

iBdID

ID number of the drive to reconnect to.

Remarks

This command connects communication again without using FAS_Connect(), when the connection is terminated after being connected with FAS_Connect().

Example

Refer to FAS_Connect library.

See Also

FAS_Connect

FAS_Close

This function disconnects communication of the ID.

Syntax

```
void FAS_Close(
    BYTE iBdID
);
```

Parameters

iBdID

ID number to disconnect.

Remarks

Example

Refer to FAS_Connect library.

See Also

FAS_Connect

FAS_GetSlaveInfo

This function reads version information string of the board.

Syntax

```
int FAS_GetSlaveInfo(
    BYTE iBdID,
    BYTE pType,
    LPSTR lpBuff,
    int nBuffSize
);
```

Parameters

iBdID

The ID number of the board. iBdID set by FAS_Connect function.

pType

Type number of the board.

lpBuff

Buffer Pointer to receive version information string.

nBuffSize

Memory allocation size of lpBuff.

Return Value

FMM_OK : Command has been performed normally.
FMM_NOT_OPEN : The board has not been connected yet.
FMM_INVALID_SLAVE_NUM : The board of iBdID doest not exist.

Remarks

Example

Refer to FAS_Connect library.

See Also

FAS_IsSlaveExist

This function checks whether the drive is connected.

Syntax

```
BOOL FAS_IsSlaveExist(
    BYTE iBdID,
);
```

Parameters

iBdID

The ID number of the board. iBdID set by FAS_Connect function.

Return Value

TRUE : Connected Status

FALSE : Disconnected Status

Remarks

This function is provided to programming method using library only. It is not provided to Protocol progamming method.

Example

Refer to FAS_Connect library.

See Also

FAS_Connect

FAS_IsBdIDExist

This function check whether the drive is connected.

Syntax

```
BOOL FAS_IsBdIDExist(int iBdID, BYTE* sb1, BYTE* sb2, BYTE* sb3, BYTE* sb4 );
```

Parameters

iBdID

The ID number of the board. ibdID set by FAS_Connect function.

sb1, sb2, sb3, sb4

IP Address. (Ex,192.168.0.10 → sb1:192, sb2:168, sb3:0, sb4:10)

Return Value

TRUE : Use the corresponding BdID

FALSE : Not use the corresponding BdID

Remarks

This function is provided to programming method using library only. It is not provided to Protocol progamming method.

Example

Refer to FAS_Connect library.

See Also

FAS_Connect

FAS_IsIPAddressExist

This function checks whether the drive is connected.

Syntax

BOOL FAS_IsIPAddressExist(BYTE sb1, BYTE sb2, BYTE sb3, BYTE sb4, int iBdID);

Parameters

sb1, sb2, sb3, sb4

IP Address. (Ex,192.168.0.10 → sb1:192, sb2:168, sb3:0, sb4:10)

iBdID

The ID number of the board. ibdID set by FAS_Connect function.

Return Value

TRUE : Use the corresponding IP Address

FALSE : Not use the corresponding IP Address

Remarks

This function is provided to programming method using library only. It is not provided to Protocol progamming method

Example

Refer to FAS_Connect library.

See Also

FAS_Connect

FAS_EnableLog

This function controls Log out related to communication error.

Syntax

```
void FAS_EnableLog(BOOL bEnable);
```

Parameters

bEnable

Log output setting.

Remarks

Control the Log output which occurs while Ezi-MOTION Ethernet function in the current process. This setting does not affect the Log outout of other processes or programs.
Log starts from FAS_Connect. When FAS_Close is used to the currently connected drive, Log output is terminated. The default setting for the Log output is TRUE.

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcDisableLog()
{
        FAS_EnableLog(FALSE);
         // After this, the Log of the functions are not printed.

        BYTE sb1 = 192, sb2 = 168, sb3 = 0, sb4 = 2; // IP :192.168.0.2
        BYTE iBdID = 0; // A unique board number of 192.168.0.2

        // Try to connect.
        if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
        {
                // Connection failed.

                return;
        }

        // Terminate the connection.
        FAS_Close(iBdID);
}
```

See Also

FAS_SetLogPath

FAS_SetLogPath

This function sets folder path of Log output files.

Syntax

BOOL FAS_SetLogPath(LPCTSTR lpPath);

Parameters

lpPath

Folder path character string of Log output file.

Return Value

If the folder name does not exist or is not accessible, FALSE will return.

Remarks

This function must be called before using FAS_Connect library.
If the IpPath value is NULL or the length of character string is 0, the Log path is selected to Ezi-MOTION Ethernet Library folder. The default value for Log path is NULL, in which the current library and the program exist.

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcEnableLog()
{
        BYTE sb1 = 192, sb2 = 168, sb3 = 0, sb4 = 2; // IP :192.168.0.2
        BYTE iBdID = 0; // A unique board number of 192.168.0.2

        // Log output
        FAS_EnableLog(TRUE);   // Do not need to use it.

        if (!FAS_SetLogPath(_T("C:₩₩Logs₩₩"))) // C:₩ Logs folder must exist.
        {
                // Log path does not exist.
                Return;
        }

        // Logs of all functions are displa yed in C: Logs folder.

        // Try to connect.
        if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
        {
                // Connction fail.

                return;
        }

        // Connection close
        FAS_Close(iBdID);
}
```

See Also

FAS_EnableLog

FAS_SetLogLevel

This function sets the path to save the output Log.

Syntax

BOOL FAS_SetLogLevel(enum LOG_LEVEL level);

Parameters

level

Log output range setting

Return Value

If a value other than the step setting value is entered, FALSE is returned.

Remarks

LOG_LEVEL_COMM : Only Logs related to commun ication errors are displayed

LOG_LEVEL_PARAM : The parameter setting function Log is additionally output to the above Log output.

LOG_LEVEL_MOTION : The motion command function Log is additionally output to the above Log output.

LOG_LEVEL_ALL : All Logs t hat can be outputted are displayed.

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcEnableLog()
{
        BYTE sb1 = 192, sb2 = 168, sb3 = 0, sb4 = 2; // IP :192.168.0.2
        int iBdID = 0 // A unique board number of 192.168.0.2

        // Log output
        FAS_EnableLog(TRUE);   // Do not need to use it

        FAS_SetLogLevel(LOG_LEVEL_ALL); // Log output range setting.

        // Try to connect.
        if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
        {
                // Connection fail

                return;
        }

        // Connection close
        FAS_Close(iBdID);
}
```

See Also

FAS_EnableLog

FAS_PrintCustomLog

This function sets the path to save the output Log.

Syntax

```
BOOL FAS_PrintCustomLog(
        int iBdID,
        enum LOG_LEVEL level,
        LPCTSTR lpszMsg
        );
```

Parameters

iBdID

The ID number of the board. iBdID set by FAS_Connect function.

level

Log output range setting

lpszMsg

String of Log to be output

Return Value

If a value other than the step setting value is entered, FALSE is returned.

Remarks

The Level is equal to the set value(range) of FAS_SetLogLevel()
Used to output the log at a specific location (function) in the program
Or, used to output the log differently from the setting value of FAS_SetLogLevel () in the program.

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcCustomLog()
{
        int iBdID = 0 // A unique board number of 192.168.0.2
        int level = LOG_LEVEL_PARAM;

        ////Communication error and parameter setting function Log output setting
        FAS_PrintCustomLog (iBdID, level, lpszMsg );

}
```

See Also

FAS_ SetLogLevel

2 - 3 . A/D Conversion Function

Function Name	Description
FAS_SetADConfig	Sets parameters related to A/D conversion.
FAS_GetADConfig	Loads parameter related to A/D conversion.
FAS_ReadADValue	Loads A/D conversion value of the corresponding CH.
FAS_ReadADAllValue	Loads A/D conversion value of the 8CH.
FAS_LoadADConfig	Loads parameters related to A/D conversion from ROM.
FAS_SaveADConfig	Saves parameter related to A/D conversion to ROM.

FAS_SetADConfig

This function sets parameters related to AD conversion.

Syntax

```
int FAS_SetADConfig (
    BYTE iBdID,
    BYTE channel,
    enum AD_DATA_TYPE type,
    long value,
    long* recv
);
```

Parameters

iBdID

The ID number of the board. iBdID set by FAS_Connect function.

channel

CH select number corresponding to the CH

(Ex.) In case of CH No.1 : 0, In case of CH No.5 : 4)

type

No. of pararmeter to set (0 :Analog input range, 1: Filter buffer length, 2: AD conversion offset)

value

Value of parameter number to set

recv

Variable pointer which will read the value applied to parameter and be saved. (Analog input range may not be changed by the function depending on the external switch setting. When setting, check whether recv is applied as the setting value.)

Return Value

FMM_OK : Command has benn performed normally.

FMM_INVALID_SLAVE_NUM : The board of iBdID doest not exist.

Remarks

Example

```
#include "FAS_ EziMOTIONPlusE.h"
```

```
void funcInputStatus()
{
        BYTE sb1 = 192, sb2 = 168, sb3 = 0, sb4 = 2; // IP :192.168.0.2
        BYTE iBdID = 0;;      //   Unique number of Board
        BYTE channel = 0;                       // CH No.1
        BYTE type = TYPE_AD_RANGE;          // Ananlog input range
        long value = 1;                         // -5~5[V]
        long recv;
        int nRtn;

        // Try to connect.
        if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
        {
                // Connection fail.

                return;
        }

        nRtn = FAS_SetADConfig (iBdID, channel, type, value, &recv);

        _ASSERT(nRtn == FMM_OK);

        // Disconnect.
        FAS_Close(iBdID);
}
```

See Also

FAS_GetADConfig

This function loads data related to A/D conversion.

Syntax

```
int FAS_GetADConfig(
    BYTE iBdID,
    BYTE channel,
    enum AD_DATA_TYPE type,
    long value
);
```

Parameters

iBdID

The ID number of the board. iBdID set by FAS_Connect function.

channel

CH select number corresponding to the CH

(Ex.) In case of CH No.1 : 0, In case of CH No.5 : 4)

type

No. of parameter to load

(0 : Analog input range, 1: Filter buffer length, 2: AD conversion offset)

value

Value of parameter number to load

Return Value

FMM_OK : Command has been performed normally.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID d oes not exist

Remarks

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcInputStatus()
{
        BYTE sb1 = 192, sb2 = 168, sb3 = 0, sb4 = 2; // IP :192.168.0.2
        BYTE iBdID = 0;;      // A unique board number
        unsigned long uLatchMask = 0x0000FFFF;   // (Latch 0~15 Clear)
```

```
    BYTE channel = 0;                    // CH No.1
    BYTE type = TYPE_AD_RANGE;           // Analog input range
    long value;
    int nRtn;

    // Try to connect.
    if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
    {
        // Connection failed.

        return;
    }

    nRtn = FAS_GetADConfig (iBdID, channel, type, &value);

    _ASSERT(nRtn == FMM_OK);

    // Disconnect.
    FAS_Close(iBdID);
}
```

See Also

FAS_ReadADValue

This function reads A/D conversion value of the corresponding CH.

Syntax

```
int FAS_ReadADValue(
    BYTE iBdID,
    BYTE channel,
    short* advalue
);
```

Parameters

iBdID

The ID number of the board. iBdID set by FAS_Connect function.

channel

CH select number corresponding to the CH

(Ex.) In case of CH No.1 : 0, In case of CH No.5 : 4)

advalue

Variable pointer where A/D conversion value of the corresponding CH will be saved.

Return Value

FMM_OK : Command has been performed normally.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcInputStatus()
{
        BYTE sb1 = 192, sb2 = 168, sb3 = 0, sb4 = 2; // IP :192.168.0.2
        BYTE iBdID = 0;;      // A unique board number

        BYTE channel = 0;                    // CH No.1
        short advalue;

        int nRtn;

        // Try to connect.
        if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
```

```
        {
                // Connection failed.

                return;
        }

        nRtn = FAS_ReadADValue (iBdID, channel, &advalue);

        _ASSERT(nRtn == FMM_OK);

        // Disconnect.
        FAS_Close(iBdID);
}
```

See Also

FAS_ReadADAllValue

This function reads entire Latch Count. (No. 0~15)

Syntax

```
int FAS_ReadADAllValue(
    BYTE iBdID,
    BYTE offset,
    AD_BUFFER* adbuffer
);
```

Parameters

iBdID

The ID number of the board. iBdID set by FAS_Connect function.

offset

Offest of 8 CH (0 : CH No. 1~8, 1 :CH No. 9~16)

adbuffer

Variable pointer where A/D conversion value of 8 CH will be saved.

Return Value

FMM_OK : Command has been performed normally.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcInputStatus()
{
        BYTE sb1 = 192, sb2 = 168, sb3 = 0, sb4 = 2; // IP :192.168.0.2
        BYTE iBdID = 0;;      // A unique board number

        BYTE offset = 0;
        AD_BUFFER adbuffer;

        int nRtn;

        // Try to connect.
        if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
        {
```

```
                    // Connection fail.

                    return;
                }

                nRtn = FAS_ FAS_ReadADAllValue (iBdID,   offset, & adbuffer);

                _ASSERT(nRtn == FMM_OK);

                // Disconnect.
                FAS_Close(iBdID);
        }
```

See Also

FAS_LoadADConfig

This function loads data related to A/D conversion from ROM.

Syntax

```
int FAS_LoadADConfig(
    BYTE iBdID,
);
```

Parameters

iBdID

The ID number of the board. iBdID set by FAS_Connect function.

Return Value

FMM_OK : Command has been performed normally.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcModifyParameter ()
{
        BYTE sb1 = 192, sb2 = 168, sb3 = 0, sb4 = 2; // IP :192.168.0.2
        BYTE iBdID = 0;;      // A unique board number
        int nRtn;

        // Try to connect.
        if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
        {
                // Connection fail.

                return;
        }

        nRtn = FAS_LoadADConfig(iBdID);

        _ASSERT(nRtn == FMM_OK);

        // Disconnect.
```

```
        FAS_Close(iBdID);
    }
```

See Also

FAS_SaveADConfig

This function saves data related to AD conversion to ROM.

Syntax

```
int FAS_SaveADConfig(
    BYTE iBdID,
);
```

Parameters

iBdID

The ID number of the board. iBdID set by FAS_Connect function.

Return Value

FMM_OK : Command has been performed normally.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcModifyParameter ()
{
        BYTE sb1 = 192, sb2 = 168, sb3 = 0, sb4 = 2; // IP :192.168.0.2
        BYTE iBdID = 0;;      // A unique board number

        int nRtn;

        // Try to connect.
        if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
        {
                // Connection fail.

                return;
        }

        nRtn = FAS_SaveADConfig (iBdID);

        _ASSERT(nRtn == FMM_OK);
```

```
        // Discoonect.
        FAS_Close(iBdID);
    }
```

See Also

3 . Appedix – Network information setting using DHCP

3 - 1 . DHCP Function

1) What is DHCP(Dynamic Host Configuration Protocol)?

→ Standard Network Protocol used to configure Network information dynamically for performing TCP / IP communication such as IP address.
(Network information : Gateway, Subnet, IP address)

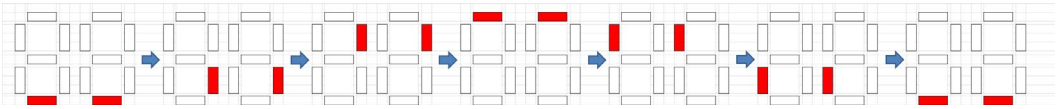
2) When not using DHCP

→ If Gateway, Subnet, IP address set as drive standard is not used, change the setting by using GUI, save and need to know about current Network information.

→ When using DHCP, Gateway, Subnet, IP address are automatically set in product. However, it is necessary to save the automatically set Network information using GUI.

3 - 2 . Network setting using DHCP (Ethernet series)

1) Set IP seting switch (SW1, SW2) as F,F
2) Connect Ethernet at Ethernet IN Connector
3) Power ON
4) 7-segment flashed as below (For product without 7-segmen, move to 6))



5) When the Network information is set, the IP address is displayed on the 7-segment. (After displaying aaa.bbb.ccc.ddd, Hex. value corresponding to ddd is displayed.)
6) Power OFF after accessing to GUI and saving the Network information
(Using Config Slave ID / IP Address)
7) Do not overlap[the value of IP setting switch (SW1, SW2) with Gateway at 1~254
8) Power ON(Setting finished)

- DHCP sets Network information dynamically, so ddd in IP address – aaa.bbb.ccc.ddd can be changed every time the power is applied. Therefore, 3-2. 6) must be performed after Network information is set by DHCP.

- **Network information can be set using DHCP, only when using a PC or a router with a DHCP server function.**

Fast, Accurate, Smooth Motion

FASTECH Co., Ltd

655, Pyeongcheon-ro Bucheon-si, Gyeonggi-do,
Rm #1202, 401-dong, Bucheon Techno-Park
(Postal code: 14502)

TEL: 032-234-6317 **FAX:** 032-234-6302

E-mail: sales@fastech-motions.com

Homepage: www.fastech-motions.com