Ezi-IO® RS-485 DIO

Input/Output Module

User Manual

Communication Function

(**Rev.03**)

FASTECH

Table of Contents

Table of Contents 2
1. Communication Protocol 3
1－1. Communication Functions 3
1－1－1. Communication Specifications 3
1－1－2. RS-485 Communication Protocol (Ver6) 3
1－1－3. Receive Frame Data Structure 4
1－1－4. CRC Calculation Example 5
1－1－5. Receive Frame Data Structure Communication Error (Ver6) 7
1－2. Structure of Frame 8
1－2－1. Frame type and Data Configuration 8
1－3. Program Method 17
2. Library for PC Program (Ver6) 18
2－1. Library Configuration 18
2－2. Board Connection Function 20
2－3. Active Level Related Functions 28
2－4. Input Control Function 35
2－5. Output Control Function 46
3. PLC Program Protocol 57

1 . Communication Protocol

1 - 1 . Communication Functions

Up to 16 Ezi-IO RS-485 products can be controlled by Daisy-Chain link on RS-485 (2-wire).

Caution	**If Windows goes to the stand-by mode, serial communication is basically disconnected. So, after recovering from the stand-by mode, the user should connect communication again. This content is equally applied to the library provided with the product.**

1 - 1 - 1 . Communication Specifications

Item	Specifications
Communication Type	Asynchronous
	Half-duplex
Baud rate [bps]	9600, 19200, 38400,57600, 115200, 230400,460800,921600
Data Type	8bit Binary ASCII Code, HEX
Parity bit	No
Stop bit	1bit
CRC Check	Yes
Maximum Cabling Length (Converter<->Drive at the end)	30 m
Minimum Cable Length between boards	More than 60 cm
Maximum Number of Connected boards	16 axes (No. 0~F)

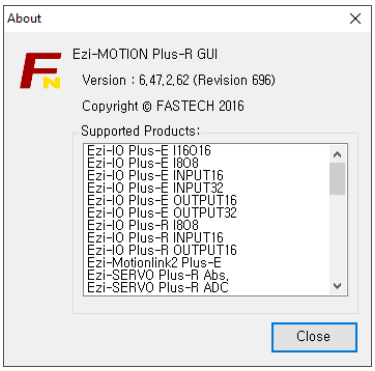
1 - 1 - 2 . RS-485 Communication Protocol (Ver6)

Thers are 2 kinds of program version for Plus-R products. This manual support for Version 6 level.

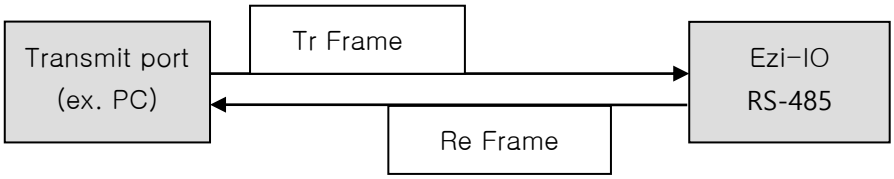
Firmware Version	Compatible with	GUI Version
V06 (V06.0x.0xx.xx)	<->	Version 6 (6.xx.x.xxx)
V08 (V08.xx.0xx.xx)	<->	Version 8 (8.xx.x.xxx)

The version currently in use can be checked as follows.

Click 'Help' Menu and select 'About Ezi-MOTION Plus-R GUI'.



1) Overview of communication FRAME



2) Basic structure of FRAME

Header	*Frame Data*	Tail
0xAA 0xCC	4~252 bytes	0xAA 0xEE

① 0xAA : Delimited byte

② 0xAA 0xCC : Displays that the Frame locates in header.

③ 0xAA 0xEE : Displays that the Frame locates in tail.

④ If any of the Frame data is '0xAA', '0xAA' should be added right after it. (byte stuffing *1)

⑤ If any data following '0xAA' is not '0xAA', '0xCC' or '0xEE', it displays that an error has occurred.

1 - 1 - 3 . Receive Frame Data Structure

The detailed configuration of the receiving ***frame data*** is as follows:

Slave ID	Frame type	Data	CRC	
1 byte	1 byte	0 ~ 248 bytes.	2 bytes	
			Low byte	High byte

① Slave ID : Module number (0~15) connected to the PC communication port.

② Frame type : To designate command type of relevant frames.
For the command type, refer to 「Frame Type and Data Configuration」section.

③ Data : Data structure and length is set according to Frame type.
For more information, refer to 「Frame Type and Data Configuration」section.

④ CRC : To check that an error occurs during communication, '0xA001' of a polynomial factor in CRC16(Cyclic Redundancy Check) is used. Or 'X16+X15+X2+1' of a polynomial factor in CRC-16-IBM(Cyclic Redundancy Check) is used. CRC calculation is performed for all items (Slave ID, Frame type, Data) prior to CRC item.

1 - 1 - 4 . CRC Calculation Example

The following program source is included in a file (file name : CRC_Checksum.c) provided with the product.

1) How to use '0xA001' of CRC16

```
const unsigned short TABLE_CRCVALUE[] =
{
  0X0000, 0XC0C1, 0XC181, 0X0140, 0XC301, 0X03C0, 0X0280, 0XC241,
  0XC601, 0X06C0, 0X0780, 0XC741, 0X0500, 0XC5C1, 0XC481, 0X0440,
  0XCC01, 0X0CC0, 0X0D80, 0XCD41, 0X0F00, 0XCFC1, 0XCE81, 0X0E40,
  0X0A00, 0XCAC1, 0XCB81, 0X0B40, 0XC901, 0X09C0, 0X0880, 0XC841,
  0XD801, 0X18C0, 0X1980, 0XD941, 0X1B00, 0XDBC1, 0XDA81, 0X1A40,
  0X1E00, 0XDEC1, 0XDF81, 0X1F40, 0XDD01, 0X1DC0, 0X1C80, 0XDC41,
  0X1400, 0XD4C1, 0XD581, 0X1540, 0XD701, 0X17C0, 0X1680, 0XD641,
  0XD201, 0X12C0, 0X1380, 0XD341, 0X1100, 0XD1C1, 0XD081, 0X1040,
  0XF001, 0X30C0, 0X3180, 0XF141, 0X3300, 0XF3C1, 0XF281, 0X3240,
  0X3600, 0XF6C1, 0XF781, 0X3740, 0XF501, 0X35C0, 0X3480, 0XF441,
  0X3C00, 0XFCC1, 0XFD81, 0X3D40, 0XFF01, 0X3FC0, 0X3E80, 0XFE41,
  0XFA01, 0X3AC0, 0X3B80, 0XFB41, 0X3900, 0XF9C1, 0XF881, 0X3840,
  0X2800, 0XE8C1, 0XE981, 0X2940, 0XEB01, 0X2BC0, 0X2A80, 0XEA41,
  0XEE01, 0X2EC0, 0X2F80, 0XEF41, 0X2D00, 0XEDC1, 0XEC81, 0X2C40,
  0XE401, 0X24C0, 0X2580, 0XE541, 0X2700, 0XE7C1, 0XE681, 0X2640,
  0X2200, 0XE2C1, 0XE381, 0X2340, 0XE101, 0X21C0, 0X2080, 0XE041,
  0XA001, 0X60C0, 0X6180, 0XA141, 0X6300, 0XA3C1, 0XA281, 0X6240,
  0X6600, 0XA6C1, 0XA781, 0X6740, 0XA501, 0X65C0, 0X6480, 0XA441,
  0X6C00, 0XACC1, 0XAD81, 0X6D40, 0XAF01, 0X6FC0, 0X6E80, 0XAE41,
  0XAA01, 0X6AC0, 0X6B80, 0XAB41, 0X6900, 0XA9C1, 0XA881, 0X6840,
  0X7800, 0XB8C1, 0XB981, 0X7940, 0XBB01, 0X7BC0, 0X7A80, 0XBA41,
  0XBE01, 0X7EC0, 0X7F80, 0XBF41, 0X7D00, 0XBDC1, 0XBC81, 0X7C40,
  0XB401, 0X74C0, 0X7580, 0XB541, 0X7700, 0XB7C1, 0XB681, 0X7640,
  0X7200, 0XB2C1, 0XB381, 0X7340, 0XB101, 0X71C0, 0X7080, 0XB041,
  0X5000, 0X90C1, 0X9181, 0X5140, 0X9301, 0X53C0, 0X5280, 0X9241,
  0X9601, 0X56C0, 0X5780, 0X9741, 0X5500, 0X95C1, 0X9481, 0X5440,
  0X9C01, 0X5CC0, 0X5D80, 0X9D41, 0X5F00, 0X9FC1, 0X9E81, 0X5E40,
  0X5A00, 0X9AC1, 0X9B81, 0X5B40, 0X9901, 0X59C0, 0X5880, 0X9841,
  0X8801, 0X48C0, 0X4980, 0X8941, 0X4B00, 0X8BC1, 0X8A81, 0X4A40,
  0X4E00, 0X8EC1, 0X8F81, 0X4F40, 0X8D01, 0X4DC0, 0X4C80, 0X8C41,
  0X4400, 0X84C1, 0X8581, 0X4540, 0X8701, 0X47C0, 0X4680, 0X8641,
  0X8201, 0X42C0, 0X4380, 0X8341, 0X4100, 0X81C1, 0X8081, 0X4040
};

unsigned short CalcCRC(unsigned char* pDataBuffer, unsigned long usDataLen)
{
  unsigned char nTemp;
  unsigned short wCRCWord = 0xFFFF;
```

```
    while (usDataLen--)
    {
                nTemp = wCRCWord ^ *(pDataBuffer++);
                wCRCWord >>= 8;
                wCRCWord ^= TABLE_CRCVALUE[nTemp];
    }
    return wCRCWord;
  }
```

2) How to use 'X16+X15+X2+1' of CRC-16-IBM

```
unsigned short CalcCRCbyAlgorithm(unsigned char* pDataBuffer, unsigned long usDataLen)
{
    const unsigned short POLYNOMIAL = 0xA001;
    unsigned short wCrc;
    int iByte, iBit;

    /* Initialize CRC */
    wCrc = 0xffff;

    for (iByte = 0; iByte < usDataLen; iByte++)
    {
        /* Exclusive-OR the byte with the CRC */
        wCrc ^= *(pDataBuffer + iByte);

        /* Loop through all 8 data bits */

        for (iBit = 0; iBit <= 7; iBit++)
        {
            /* If the LSB is 1, shift the CRC and XOR the polynomial mask with the CRC */

          // Note - the bit test is performed before the rotation, so can't move the << here
                if (wCrc & 0x0001)
                {
                        wCrc >>= 1;
                        wCrc ^= POLYNOMIAL;
                }
                else
                {
                        // Just rotate it
                        wCrc >>= 1;
                }
        }
    }
    return wCrc;
    }
```

1 - 1 - 5 . Receive Frame Data Structure Communication Error (Ver6)

The basic structure of the frame of the receiver to the transmitter's command is the same. However, the difference in the ***Frame Data*** section below is as follows with 'Communication Status' added.

Slave ID	Frame type	Data		CRC	
1 byte	1 byte	1 byte	0 ~ 247bytes	2 bytes	
		Communication Status	Receive date	Low byte	High byte

① Slave ID : Same as receive frame.
(If it is not same, it is recognized as the error status.)

② Frame type : Same as receive frame.
(If it is not same, it is recognized as the error status.)

③ Data : The data of this item may not be included for sending simple execution commands. However, the reply contains 1 byte of data indicating the communication status (error / normal status).

The following table describes the byte values that indicate the communication status.

Hexa value	Decimal value	Description
0x00	0	Communication is normal.
0x80	128	Frame type Error : Cannot recognize the received Frame type command.
0x81	129	Data Error, ROM Data Read/Write Error : Receive data value is out of range.
0x82	130	Receive Frame Error : Received Frame does not meet the standard.
0xAA	170	CRC Error : CRC error occurred because the received frame data was affected by ambient noise. In this case, the DLL library on the sending side automatically tries to communicate once more..

Caution	1) If 'Header' and 'Slave ID' values in the transmit Frame are abnormal, there is no response from the product. 2) If the communication status is displayed to '130', the size of response data is '0' byte.

1 - 2 . Structure of Frame

1 - 2 - 1 . Frame type and Data Configuration

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

<table>
<tr><th>Frame type</th><th>Library Name</th><th>Contents Description</th></tr>
<tr><td>0x01
(1)</td><td>FAS_
GetSlaveInfo</td><td>Connected slave type and program version information are required.

Transmitting : 0 byte

Receiving : 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
- 150 : Ezi-IO-RS-I16 series
- 155 : Ezi-IO-RS-I8O8 series
- 160 : Ezi-IO-RS-O16 series</td></tr>
<tr><td>0xC0
(192)</td><td>FAS_GetInput</td><td>Read the Input / Latch status.

Transmitting : 0 byte

Receiving : 9 bytes

<table>
<tr><td>1 byte</td><td>4 bytes</td><td>4 bytes</td></tr>
<tr><td>Communication Status</td><td>Input Status</td><td>Latch Status</td></tr>
</table>

Receiving DATA Structure

MSB LSB
<table>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>63~56</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>55~48</td></tr>
<tr><td>Latch15</td><td>Latch14</td><td>Latch13</td><td>Latch12</td><td>Latch11</td><td>Latch10</td><td>Latch9</td><td>Latch8</td><td>47~40</td></tr>
<tr><td>Latch7</td><td>Latch6</td><td>Latch5</td><td>Latch4</td><td>Latch3</td><td>Latch2</td><td>Latch1</td><td>Latch0</td><td>39~32</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>31~24</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>23~16</td></tr>
<tr><td>Input15</td><td>Input14</td><td>Input13</td><td>Input12</td><td>Input11</td><td>Input10</td><td>Input9</td><td>Input8</td><td>15~8</td></tr>
<tr><td>Input7</td><td>input6</td><td>Input5</td><td>Input4</td><td>Input3</td><td>Input2</td><td>Input1</td><td>Input0</td><td>7~0</td></tr>
</table>

0: OFF
1: ON
*Ezi-IO-RS-I16 series : Read the status of Input 0~15 & Latch 0~15.
*Ezi-IO-RS-I8O8 series : Read the status of Input 0~7 & Latch 0~7.</td></tr>
</table>

<table>
<tr>
<td>0xC1
(193)</td>
<td>FAS_ClearLatch</td>
<td>
Clear(Reset) the Latch of the corresponding bit.

Transmitting : 4 bytes
<table>
<tr><td>4 bytes</td></tr>
<tr><td>Data</td></tr>
</table>

Transmitting Data Structure
<table>
<tr><td colspan="4">MSB</td><td colspan="4"></td><td>LSB</td><td></td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>31~24</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>23~16</td></tr>
<tr><td>Latch15</td><td>Latch14</td><td>Latch13</td><td>Latch12</td><td>Latch11</td><td>Latch10</td><td>Latch9</td><td>Latch8</td><td>15~8</td></tr>
<tr><td>Latch7</td><td>Latch6</td><td>Latch5</td><td>Latch4</td><td>Latch3</td><td>Latch2</td><td>Latch1</td><td>Latch0</td><td>7~0</td></tr>
</table>
0 : Keep current status
1 : Latch Clear(Reset)
*Ezi-IO-RS-I16 series : Latch 0~15
*Ezi-IO-RS-I8O8 series : Latch 0~7

Receiving : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>Communication Status</td></tr>
</table>
</td>
</tr>
<tr>
<td>0xC2
(194)</td>
<td>FAS_GetLatchCount</td>
<td>
Read Latch Count

Trasmitting : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>Data</td></tr>
</table>
Data : Latch bit Number
*Ezi-IO-RS-I16 series : 0~15
*Ezi-IO-RS-I8O8 series : 0~7

Receiving : 5 bytes
<table>
<tr><td>1 byte</td><td>4 bytes</td></tr>
<tr><td>Communication Status</td><td>Data</td></tr>
</table>
Data : 0~2^{32}
</td>
</tr>
</table>

<table>
<tr>
<td>0xC3
(195)</td>
<td>FAS_GetLatchCount
All</td>
<td>
Read all Latch Count

Transmitting : 0 byte

Receiving : 65 bytes
<table>
<tr><td>1 byte</td><td>64 bytes</td></tr>
<tr><td>Communication Status</td><td>Data</td></tr>
</table>

Receiving Data Structure
<table>
<tr><td colspan="3">MSB</td><td colspan="3"></td><td colspan="2">LSB</td></tr>
<tr><td>COUNT15</td><td>COUNT14</td><td>COUNT13</td><td>・・・</td><td>・・・</td><td>COUNT2</td><td>COUNT1</td><td>COUNT0</td></tr>
<tr><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td><td></td><td></td><td>4 bytes</td><td>4 bytes</td><td>4</td></tr>
</table>
*Ezi-IO-RS-I16 series : COUNT 0~15
*Ezi-IO-RS-I8O8 series : COUNT 0~7
</td>
</tr>
<tr>
<td>0xC4
(196)</td>
<td>FAS_ClearLatch
Count</td>
<td>
Reset the Latch Count of the corresponding bit.

Transmitting : 4 bytes
<table>
<tr><td>4 bytes</td></tr>
<tr><td>Data</td></tr>
</table>

Transmitting Data Structure
<table>
<tr><td colspan="3">MSB</td><td colspan="5"></td><td>LSB</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>31~24</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>23~16</td></tr>
<tr><td>Latch15</td><td>Latch14</td><td>Latch13</td><td>Latch12</td><td>Latch11</td><td>Latch10</td><td>Latch9</td><td>Latch8</td><td>15~8</td></tr>
<tr><td>Latch7</td><td>Latch6</td><td>Latch5</td><td>Latch4</td><td>Latch3</td><td>Latch2</td><td>Latch1</td><td>Latch0</td><td>7~0</td></tr>
</table>
*Ezi-IO-RS-I16 series : Latch 0~15
*Ezi-IO-RS-I8O8 series : Latch 0~7
Receiving : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>Communication Status</td></tr>
</table>
</td>
</tr>
</table>

<table>
<tr>
<td>0xC5
(197)</td>
<td>FAS_GetOutput</td>
<td>
Read Output / Trigger(Run/Stop) status

Transmitting : 0 byte

Receiving : 9 bytes
<table>
<tr><td>1 byte</td><td>8 bytes</td></tr>
<tr><td>Communication Status</td><td>Data</td></tr>
</table>

Receving Data Structure
<table>
<tr><td>MSB</td><td></td><td></td><td></td><td></td><td></td><td></td><td>LSB</td><td></td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>63~56</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>55~48</td></tr>
<tr><td>Run/Stop15</td><td>Run/Stop14</td><td>Run/Stop13</td><td>Run/Stop12</td><td>Run/Stop11</td><td>Run/Stop10</td><td>Run/Stop9</td><td>Run/Stop8</td><td>47~40</td></tr>
<tr><td>Run/Stop7</td><td>Run/Stop6</td><td>Run/Stop5</td><td>Run/Stop4</td><td>Run/Stop3</td><td>Run/Stop2</td><td>Run/Stop1</td><td>Run/Stop0</td><td>39~32</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>31~24</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>23~16</td></tr>
<tr><td>Output15</td><td>Output14</td><td>Output13</td><td>Output12</td><td>Output11</td><td>Output10</td><td>Output9</td><td>Output8</td><td>15~8</td></tr>
<tr><td>Output7</td><td>Output6</td><td>Output5</td><td>Output4</td><td>Output3</td><td>Output2</td><td>Output1</td><td>Output0</td><td>7~0</td></tr>
</table>
0 : OFF (Trigger Stop)
1 : ON (Trigger Run)

*Ezi-IO-RS-O16 series : Read the status of Output 0~15, Trigger(Run/Stop) 0~15
*Ezi-IO-RS-I8O8 series : Read the status of Output 8~15, Trigger(Run/Stop) 8~15
</td>
</tr>
</table>

<table>
<tr>
<td>0xC6
(198)</td>
<td>FAS_SetOutput</td>
<td>
Configure the Output. (Set : On, Reset : Off)

Transmitting : 8 bytes
<table>
<tr><td>4 bytes</td><td>4 bytes</td></tr>
<tr><td>Set Output</td><td>Reset Output</td></tr>
</table>

Transmitting Data Structure
<table>
<tr><td>MSB</td><td colspan="7"></td><td>LSB</td><td></td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>63~56</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>55~48</td></tr>
<tr><td>Reset Output15</td><td>Reset Output14</td><td>Reset Output13</td><td>Reset Output12</td><td>Reset Output11</td><td>Reset Output10</td><td>Reset Output9</td><td>Reset Output8</td><td>47~40</td></tr>
<tr><td>Reset Output7</td><td>Reset Output6</td><td>Reset Output5</td><td>Reset Output4</td><td>Reset Output3</td><td>Reset Output2</td><td>Reset Output1</td><td>Reset Output0</td><td>39~32</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>31~24</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>23~16</td></tr>
<tr><td>Output15</td><td>Output14</td><td>Output13</td><td>Output12</td><td>Output11</td><td>Output10</td><td>Output9</td><td>Output8</td><td>15~8</td></tr>
<tr><td>Output7</td><td>Output6</td><td>Output5</td><td>Output4</td><td>Output3</td><td>Output2</td><td>Output1</td><td>Output0</td><td>7~0</td></tr>
</table>
1 : Execute ‘Set Output’ or ‘Reset Output’.
(When the same bit of Set Output and Reset Output is 1, it is executed as Reset)
0 : ‘Set Output’ or ‘Reset Output’ is not executed.
*Ezi-IO-RS-O16 series : Can configure Output 0~15, Reset Output 0~15 optional
*Ezi-IO-RS-I8O8 series : Can configure Output 8~15, Reset Output 8~15 optional
Receiving : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>Communication Status</td></tr>
</table>
</td>
</tr>
</table>

<table>
<tr>
<td>0xC7
(199)</td>
<td>FAS_SetTrigger</td>
<td>
Configure Trigger Out.

Transmitting : 13 bytes
<table>
<tr><td>13 bytes</td></tr>
<tr><td>Data</td></tr>
</table>

Transmitting Data Structure
<table>
<tr><td>MSB</td><td></td><td></td><td></td><td></td><td>LSB</td></tr>
<tr><td>Count</td><td>Blank</td><td>On time</td><td>Blank</td><td>Period</td><td>Output No.</td></tr>
<tr><td>4 bytes</td><td>2 bytes</td><td>2 bytes</td><td>2 bytes</td><td>2 bytes</td><td>1 byte</td></tr>
</table>
Output No. : Trigger setting Output No.
*Ezi-IO-RS-O16 series : 0~15
*Ezi-IO-RS-I8O8 series : 8~15
Period : Trigger output cycle, the unit is [ms],
The value should be larger than On time value (2~65,535).
On time : Trigger output On time, the unit is [ms],
The value should be smaller than Period value (1~65,534).
Count : Trigger output repeat count (1~4,294,967,295)

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

<table>
<tr>
<td>0xC8
(200)</td>
<td>FAS_SetRunStop</td>
<td>
Run or Stop the Trigger output of the corresponding bit.
If ‘Stop bit’ is on during ‘Run’, it stops even if it doesn’t output as many as count.

Transmitting : 13 bytes
<table>
<tr><td>13 bytes</td></tr>
<tr><td>Data</td></tr>
</table>

Transmitting Data Structure
<table>
<tr><td>MSB</td><td></td><td></td><td></td><td></td><td></td><td></td><td>LSB</td><td></td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>63~56</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>55~48</td></tr>
<tr><td>Stop15</td><td>Stop14</td><td>Stop13</td><td>Stop12</td><td>Stop11</td><td>Stop10</td><td>Stop9</td><td>Stop8</td><td>47~40</td></tr>
<tr><td>Stop7</td><td>Stop6</td><td>Stop5</td><td>Stop4</td><td>Stop3</td><td>Stop2</td><td>Stop1</td><td>Stop0</td><td>39~32</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>31~24</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>23~16</td></tr>
<tr><td>Run15</td><td>Run14</td><td>Run13</td><td>Run12</td><td>Run11</td><td>Run10</td><td>Run9</td><td>Run8</td><td>15~8</td></tr>
<tr><td>Run7</td><td>Run6</td><td>Run5</td><td>Run4</td><td>Run3</td><td>Run2</td><td>Run1</td><td>Run0</td><td>7~0</td></tr>
</table>
*Ezi-IO-RS-O16 series : Run 0~15, Stop 0~15
*Ezi-IO-RS-I8O8 series : Run 8~15, Stop 8~15
Receiving : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>Communication Status</td></tr>
</table>
</td>
</tr>
<tr>
<td>0xC9
(201)</td>
<td>FAS_GetTrigger</td>
<td>
Read current trigger output count.

Transmitting : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>Data</td></tr>
</table>
Data : Trigger output bit number
*Ezi-IO-RS-O16 series : Output Bit No. 0~15
*Ezi-IO-RS-I8O8 series : Output Bit No. 8~15

Receiving : 5 bytes
<table>
<tr><td>1 byte</td><td>4 bytes</td></tr>
<tr><td>Communication Status</td><td>Data</td></tr>
</table>
Data : 0~2^{32}
</td>
</tr>
</table>

<table>
<tr>
<td>0xCA
(202)</td>
<td>FAS_GetIOLevel</td>
<td>
Read Input or Output Active Level.

Transmitting : 0 byte

Receiving : 5 bytes
<table>
<tr><td>1 byte</td><td>4 bytes</td></tr>
<tr><td>Communication Status</td><td>Data</td></tr>
</table>

Receiving Data Structure
<table>
<tr><td>MSB</td><td></td><td></td><td></td><td></td><td></td><td></td><td>LSB</td><td></td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>31~24</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>23~16</td></tr>
<tr><td>LEVEL15</td><td>LEVEL14</td><td>LEVEL13</td><td>LEVEL12</td><td>LEVEL11</td><td>LEVEL10</td><td>LEVEL9</td><td>LEVEL8</td><td>15~8</td></tr>
<tr><td>LEVEL7</td><td>LEVEL6</td><td>LEVEL5</td><td>LEVEL4</td><td>LEVEL3</td><td>LEVEL2</td><td>LEVEL1</td><td>LEVEL0</td><td>7~0</td></tr>
</table>
0 : Low Active Level (Latch : Falling edge)
1 : High Active Level (Latch : Rising edge)

*Ezi-IO-RS-O16 series : Read Onput Actcive Level 0~15
*Ezi-IO-RS-I16 series : Read Iutput Actcive Level 0~15
*Ezi-IO-RS-I8O8 series : Read Input Actcive Level 0~7 & Output Actcive Level 8~15
</td>
</tr>
<tr>
<td>0xCB
(203)</td>
<td>FAS_SetIOLevel</td>
<td>
Configure Input or Output Active Level.

Transmitting : 4 bytes

<table>
<tr><td>4 bytes</td></tr>
<tr><td>Data</td></tr>
</table>

Transmitting Data Structure
<table>
<tr><td>MSB</td><td></td><td></td><td></td><td></td><td></td><td></td><td>LSB</td><td></td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>31~24</td></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>23~16</td></tr>
<tr><td>LEVEL15</td><td>LEVEL14</td><td>LEVEL13</td><td>LEVEL12</td><td>LEVEL11</td><td>LEVEL10</td><td>LEVEL9</td><td>LEVEL8</td><td>15~8</td></tr>
<tr><td>LEVEL7</td><td>LEVEL6</td><td>LEVEL5</td><td>LEVEL4</td><td>LEVEL3</td><td>LEVEL2</td><td>LEVEL1</td><td>LEVEL0</td><td>7~0</td></tr>
</table>
0 : Low Active Level (Latch : Falling edge)
1 : High Active Level (Latch : Rising edge)

*Ezi-IO-RS-O16 series : Configure Onput Actcive Level 0~15
*Ezi-IO-RS-I16 series : Configure Iutput Actcive Level 0~15
*Ezi-IO-RS-I8O8 series : Configure Input Actcive Level 0~7, Output Actcive Level 8~15
Receiving : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>Communication Status</td></tr>
</table>
</td>
</tr>
</table>

<table>
<tr>
<td>0xCC
(204)</td>
<td>FAS_
LoadIOLevel</td>
<td>Read ROM Memory Active Level.
Change the data in RAM of the board to the data of ROM.

Transmitting : 0 byte

Receiving : 1 byte

<table>
<tr><td>1 byte</td></tr>
<tr><td>Communication Status</td></tr>
</table>
</td>
</tr>
<tr>
<td>0xCD
(205)</td>
<td>FAS_
SaveIOLevel</td>
<td>Save the currently set active level value in ROM memory. This ensures that the values are stored even when the power is turned off.

Transmitting : 0 byte

Receiving : 1 byte

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

1 - 3 . Program Method

There are two ways of programming to Ezi-IO RS-485.

The first is to use the Visual C ++ language in the window system of a PC.
In this case, use the library supplied with the product (see 「2. Library for PC Program」).

The second method is to send a command character directly without using a library function. Like protocol test program, user should write low level protocol program by himself / herself. It is mainly applied when PLC is used as host controller.

Please refer to the 「3. PLC Program Protocol 」for details.
You can test with a user GUI program called 'ProtocolTest_PlusR.exe'.

2 . Library for PC Program (Ver6)

2 - 1 . Library Configuration

To use this library, C++ header file(*.h) and library file(*.lib or *.dll) are required. These files are included in "₩₩FASTECH₩₩Ezi-MOTION Plus-R V6₩₩include₩₩" The following contents should be included in a source file for development.

#include "₩₩FASTECH₩₩ Ezi-MOTION Plus-R V6₩₩include₩₩FAS_EziMotionPlusR.h"
#include "₩₩FASTECH₩₩ Ezi-MOTION Plus-R V6₩₩include₩₩ReturnCodes_Define.h"
#include "₩₩FASTECH₩₩ Ezi-MOTION Plus-R V6₩₩include₩₩MOTION_DEFINE.h"
#include "₩₩FASTECH₩₩ Ezi-MOTION Plus-R V6₩₩include₩₩COMM_Define.h"

The library files are as follows :
"₩₩FASTECH₩₩Ezi-MOTION Plus-R V6₩₩include₩₩EziMotionPlusR.lib"
"₩₩FASTECH₩₩Ezi-MOTION Plus-R V6₩₩include₩₩EziMotionPlusR.dll"

Sample program source using this library is included in
"₩₩FASTECH₩₩ Ezi-MOTION Plus-R V6₩₩Examples₩₩"

(1) The following table describes the values returned when using each library function. These return values can only be seen in library (DLL) functions and are not supported in programs using the protocol.

Type	Name	Return value	Description
Normal	FMM_OK	0	The function has normally performed the command.
Input Error	FMM_NOT_OPEN	1	Wrong port number is input.
	FMM_INVALID_PORT_NUM	2	Selected port number is not connected.
	FMM_INVALID_SLAVE_NUM	3	Wrong slave number is input.
Connection Error	FMC_DISCONNECTED	5	The corresponding drive is disconnected.
	FMC_TIMEOUT_ERROR	6	There is no response for the specified time. (100 msec)
	FMC_CRCFAILED_ERROR	7	Checksum error occurred during communication.
	FMC_RECVPACKET_ERROR	8	A protocol level error has occurred in a packet received from the drive.

(2) The following table shows the return values that are common to all libraries. You can check the results (communication status and operation status) checked by the drive. It is supported for both using libraries (DLL) and programming using protocols.

Type	Name	Return value	Description
Normal	FMP_OK	0	Communication has been normally performed.
Input Error	FMP_FRAMETYPEERROR	128	The board cannot recognize the command.
	FMP_DATAERROR	129	Input data is out of the range.
Connection Error	FMP_PACKETERROR	130	A protocol level error has occurred in a packet received by the drive.
	FMP_PACKETCRCERROR	170	The CRC calculations of the packets received by the board do not match.

2 - 2 . Board Connection Function

Function Name	Description
FAS_Connect	Attempt to connect with the board using the UDP Protocol. : Returns TRUE if the connection was successful and FALSE if the connection failed.
FAS_Close	Attempt to terminate communication with the board.
FAS_GetSlaveInfo	Reads board type and program version. : Returns board type and version information.
FAS_IsSlaveExist	Check the existence of the drive. : Returns TRUE if present, FALSE if connection failed.
FAS_EnableLog	Controls output of log related to communication error.
FAS_SetLogPath	Set the path to save the output log. : Returns TRUE if the path exists, FALSE if path does not exist or is inaccessible.

FAS_Connect

FAS_Connect is a function that connects Ezi-IO RS-485.

Syntax

```
BOOL FAS_Connect(
    BYTE nPortNo,
    DWORD dwBaud
);
```

Parameters

nPortNo

Select Serial Port number to connect.

dwBaud

Enter Baudrate of Serial Port.

Return Value

Returns TRUE if the connection was successful and FALSE if the connection failed.

Remarks

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcInit()
{
        BYTE nPortNo = 1;  // COMM Port number
        DWORD dwBaudrate = 115200;        // Baudrate value. (Can be changed by settings)
        BYTE iSlaveNo = 0;  // Slave No (0 ~ 15)
        char lpBuff[256];
        int nBuffSize = 256;
        BYTE nType;
        int nRtn;

        // Try to connect
        if (FAS_Connect(nPortNo, dwBaudrate) == FALSE)
        {
                // Connection failed.
                // Port might not be connected or Baudrate might not match.
                return;
        }

        if (FAS_IsSlaveExist(nPortNo, iSlaveNo) == FALSE)
```

```
        {
                // Corresponding slave number does not exist.
                // Check slave number of Ezi-IO GATE Plus-R
                return;
        }

        nRtn = FAS_GetSlaveInfo(nPortNo, iSlaveNo, &nType, lpBuff, nBuffSize);
        if (nRtn != FMM_OK)
        {
                // The command did not run successfully.
                // Refer to ReturnCodes_Define.h
        }

        printf("Port : %d (Slave %d) ₩n", nPortNo, iSlaveNo);
        printf("₩tType : %d ₩n", nType);
        printf("₩tVersion : %d ₩n", lpBuff);

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

See Also

FAS_Close

FAS_Close

Disconnect the serial port in use.

Syntax

```
void FAS_Close(
    BYTE nPortNo
);
```

Parameters

nPortNo

Port number to disconnect.

Remarks

Example

Refer to FAS_Connect library.

See Also

FAS_Connect

FAS_GetSlaveInfo

Get version information string of the relevant board.

Syntax

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

Parameters

nPortNo

Port number of the board.

iSlaveNo

Slave number of the board.

pType

Type number of the board.

lpBuff

Buffer Pointer to receive Version information string.

nBuffSize

Memory allocation size value of lpBuff.

Return Value

FMM_OK : The command ran successfully.

FMM_NOT_OPEN : The board is not connected yet.

FMM_INVALID_PORT_NUM : nPort does not exist among connected ports.

FMM_INVALID_SLAVE_NUM : The board of corresponding iSlaveNo does not exist.

Remarks

Example

Refer to FAS_Connect library.

See Also

FAS_IsSlaveExist

Check if the board is currently connected.

Syntax

```
BOOL FAS_IsSlaveExist(
    BYTE nPortNo,
    BYTE iSlaveNo
);
```

Parameters

nPortNo

Port number of the board.

iSlaveNo

Slave number of the board.

Return Value

TRUE : Connected.

FALSE : Disconnected.

Remarks

This function is provided only in the library(DLL) and is not supported for protocol programming.

Example

Refer to FAS_Connect library.

See Also

FAS_Connect

FAS_EnableLog

Determines whether to output the communication error related log.

Syntax

void FAS_EnableLog(BOOL bEnable);

Parameters

bEnable

Set whether to output the log.

Remarks

Controls log output that occurs while using Ezi-MOTION Plus-R function in the current process.
This setting does not affect the log output of other processes or programs.
Log is generated from FAS_Connect.
When FAS_Close is used to disconnect the currently connected port, log output is terminated. The default setting for Log output is TRUE.

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcDisableLog()
{
        BYTE nPortNo = 1;   // COMM Port number

        FAS_EnableLog(FALSE);

        // After this, the log of the functions will not be output.

        // Try to connect.
        if (FAS_Connect(nPortNo, 115200) == FALSE)
        {
                // Connection failed.
                // Port might not be connected or Baudrate might not match.
                return;
        }

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

See Also

FAS_SetLogPath

FAS_SetLogPath

Set the path to save the output log.

Syntax

BOOL FAS_SetLogPath(LPCTSTR lpPath);

Parameters

lpPath

Folder path Character string of Log output file.

Return Value

If the path entered does not exist or cannot be accessed, FALSE is returned.

Remarks

This function must be called before using the FAS_Connect function.
If the lpPath value is NULL or a zero-length string is entered, the log path is set to the folder where the program currently using Ezi-MOTION Plus-R Library exists.

The default value for the log path is NULL, which is the folder where the current program exists.

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcEnableLog()
{
	BYTE nPortNo = 1;  // COMM Port number

	// Output Log.
	FAS_EnableLog(TRUE);   // It is not necessary to use.

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

	// Logs of all functions are saved in the C: ₩ Logs folder.

	// Try to connect.
	if (FAS_Connect(nPortNo, 115200) == FALSE)
	{
		// Connection failed.
		// Port might not be connected or Baudrate might not match.
		return;
	}

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

See Also

FAS_EnableLog

2 - 3 . Active Level Related Functions

Function Name	Description
FAS_GetIOLevel	Read Input or Output Active Level
FAS_SetIOLevel	Set Input or Output Active Level
FAS_ SaveIOLevel	Save current Input Active Level or Output Active Level to ROM Data will be preserved even when the power is OFF.
FAS_LoadIOLevel	Read Input Active Level or Output Active Level from ROM and write it to RAM.

FAS_GetIOLevel

Read Input or Output Active Level.

Syntax

```
int FAS_GetInputLevel(
    BYTE nPortNo,
    BYTE iSlaveNo,
    unsigned long* ulOLevel
);
```

Parameters

nPortNo

Port number of the board.

iSlaveNo

Slave number of the board.

ulOLevel

Value to set Active Level.

Return Value

FMM_OK : The command ran successfully.

FMM_NOT_OPEN : The board is not connected yet.

FMM_INVALID_PORT_NUM : nPort does not exist among connected ports.

FMM_INVALID_SLAVE_NUM : The board of corresponding iSlaveNo does not exist.

Remarks

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcIOStatus()
{
        BYTE nPortNo = 1;   // COMM Port number
        BYTE iSlaveNo = 0;  // Slave No (0 ~ 15)
        unsigned long uInputLevel;

        int nRtn;

        // Try to connect
        if (FAS_Connect(nPortNo, 115200) == FALSE)
        {
                // Connection failed.
```

```
                // Port might not be connected or Baudrate might not match.
                return;
            }

            nRtn = FAS_GetInputLevel(m_nPortNo, iSlaveNo, &uIOLevel);

            _ASSERT(nRtn == FMM_OK);

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

See Also

FAS_SetIOLevel

Set Input or Output Active Level.

Syntax

```
int FAS_SetIOLevel(
    BYTE nPortNo,
    BYTE iSlaveNo,
    unsigned long ulIOLevel
);
```

Parameters

nPortNo

Port number of the board.

iSlaveNo

Slave number of the board.

ulOLevel

Value to set Active Level.

Return Value

FMM_OK : The command ran successfully.
FMM_NOT_OPEN : The board is not connected yet.
FMM_INVALID_PORT_NUM : nPort does not exist among connected ports.
FMM_INVALID_SLAVE_NUM : The board of corresponding iSlaveNo does not exist.

Remarks

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcInputStatus()
{
        BYTE nPortNo = 1;   // COMM Port number
        BYTE iSlaveNo = 0;  // Slave No (0 ~ 15)
        unsigned long ulIOLevel = 0x0000FF00;
                                    // 0~7: low active,   8~15 : high active

        int nRtn;

        // Try to connect
        if (FAS_Connect(nPortNo, 115200) == FALSE)
        {
```

```
            // Connection failed.
            // Port might not be connected or Baudrate might not match.
            return;
        }

        nRtn = FAS_SetIOLevel(m_nPortNo, iSlaveNo, uIOLevel);

        _ASSERT(nRtn == FMM_OK);

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

See Also

FAS_SaveIOLevel

Save current Input Active Level or Output Active Level to ROM.

Syntax

```
Int FAS_SaveIOLevel(
    BYTE nPortNo,
    BYTE iSlaveNo
);
```

Parameters

nPortNo

Port number of the board.

iSlaveNo

Slave number of the board.

Return Value

FMM_OK : The command ran successfully.

FMM_NOT_OPEN : The board is not connected yet.

FMM_INVALID_PORT_NUM : nPort does not exist among connected ports.

FMM_INVALID_SLAVE_NUM : The board of corresponding iSlaveNo does not exist.

Remarks

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcModifyParameter()
{
        BYTE nPortNo = 1;   // COMM Port number
        BYTE iSlaveNo = 0;  // Slave No (0 ~ 15)

        int nRtn;

        // Try to connect
        if (FAS_Connect(nPortNo, 115200) == FALSE)
        {
                // Connection failed.
                // Port might not be connected or Baudrate might not match.
                return;
        }
        // ROM Save on.
        nRtn = FAS_SavetIOLevel(nPortNo, iSlaveNo)
        _ASSERT(nRtn == FMM_OK); // If the command did not complete successfully, it stops.

        FAS_Close(nPortNo);
}
```

See Also

FAS_LoadIOLevel

Read Input or Output Active Level from ROM.

Syntax

```
int FAS_LoadIOLevel(
    BYTE nPortNo,
    BYTE iSlaveNo,
);
```

Parameters

nPortNo

Port number of the board.

iSlaveNo

Slave number of the board.

Return Value

FMM_OK : The command ran successfully.

FMM_NOT_OPEN : The board is not connected yet.

FMM_INVALID_PORT_NUM : nPort does not exist among connected ports.

FMM_INVALID_SLAVE_NUM : The board of corresponding iSlaveNo does not exist.

Remarks

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcModifyParameter()
{
        BYTE nPortNo = 1;   // COMM Port number
        BYTE iSlaveNo = 0;  // Slave No (0 ~ 15)

        int nRtn;

        // Try to connect
        if (FAS_Connect(nPortNo, 115200) == FALSE)
        {
                // Connection failed.
                // Port might not be connected or Baudrate might not match.
                return;
        }

         // Read Input Active Level or Output Active Level from ROM and write it to RAM.
        nRtn = FAS_LoadIOLevel(nPortNo, iSlaveNo)
        _ASSERT(nRtn == FMM_OK); // If the command did not complete successfully, it stops.

        FAS_Close(nPortNo);
}
```

See Also

2 - 4 . Input Control Function

Function Name	Description
FAS_GetInput	Read Input / Latch status.
FAS_ClearLatch	Clear the Latch of corresponding bit.
FAS_GetLatchCount	Read the Latch Cound of corresponding bit.
FAS_GetLatchCountAll	Read the Latch Count of all channels.
FAS_ClearLatchCount	Reset(0) the Latch Count of corresponding bit.

*Ezi-IO-RS-I16 series : 0~15

*Ezi-IO-RS-I8O8 series : 0~7

FAS_GetInput

Read Input / Latch status.

(Ezi-IO-RS-I16 series : 0~15 /Ezi-IO-RS-I8O8 series : 0~7)

Syntax

```
int FAS_GetInput(
    BYTE nPortNo,
    BYTE iSlaveNo
    unsigned long* uInput,
    unsigned long* uLatch
);
```

Parameters

nPortNo

Port number of the board.

iSlaveNo

Slave number of the board.

uInput

Parameter pointer to save Input value.

uLatch

Parameter pointer to save Latch value.

Return Value

FMM_OK : The command ran successfully.

FMM_INVALID_PORT_NUM : nPort does not exist among connected ports.

FMM_INVALID_SLAVE_NUM : The board of corresponding iSlaveNo does not exist.

Remarks

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcInputStatus()
{
        BYTE nPortNo = 1;   // COMM Port number
        BYTE iSlaveNo = 0;  // Slave No (0 ~ 15)
        unsigned long uInput;
        unsigned long uLatch;

        int nRtn;
```

```
        // Try to connect
        if (FAS_Connect(nPortNo, 115200) == FALSE)
        {
                // Connection failed.
                // Port might not be connected or Baudrate might not match.
                return;
        }

        nRtn = FAS_GetInput(m_nPortNo, iSlaveNo, &uInput, &uLatch);

        _ASSERT(nRtn == FMM_OK);

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

See Also

FAS_ClearLatch

Clear the Latch of corresponding bit.

(Ezi-IO-RS-I16 series : 0~15 /Ezi-IO-RS-I8O8 series : 0~7)

Syntax

```
int FAS_ClearLatch(
    BYTE nPortNo,
    BYTE iSlaveNo,
    unsigned long uLatchMask
);
```

Parameters

nPortNo

Port number of the board.

iSlaveNo

Slave number of the board.

uLatchMask

Bitmask value to clear the Latch.

Return Value

FMM_OK : The command ran successfully.

FMM_NOT_OPEN : The board is not connected yet.

FMM_INVALID_PORT_NUM : nPort does not exist among connected ports.

FMM_INVALID_SLAVE_NUM : The board of corresponding iSlaveNo does not exist.

Remarks

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcInputStatus()
{
        BYTE nPortNo = 1;   // COMM Port number
        BYTE iSlaveNo = 0;  // Slave No (0 ~ 15)
        unsigned long uLatchMask = 0x0000FFFF;   // (Latch 0~15 Clear)

        int nRtn;

        // Try to connect
        if (FAS_Connect(nPortNo, 115200) == FALSE)
        {
```

```
            // Connection failed.
            // Port might not be connected or Baudrate might not match.
            return;
        }

        nRtn = FAS_ClearLatch(m_nPortNo, iSlaveNo, uLatchMask);

        _ASSERT(nRtn == FMM_OK);

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

See Also

FAS_GetLatchCount

Read the Latch Count.

(Ezi-IO-RS-I16 series : 0~15 /Ezi-RS-PR-I8O8 series : 0~7)

Syntax

```
int FAS_ClearLatchCount(
    BYTE nPortNo,
    BYTE iSlaveNo,
    unsigned char iInputNo,
    unsigned long* uCount
);
```

Parameters

nPortNo

Port number of the board.

iSlaveNo

Slave number of the board.

iInputNo

Latch number to read Latch count.

uCount

Parameter pointer to save Latch count value.

Return Value

FMM_OK : The command ran successfully.

FMM_NOT_OPEN : The board is not connected yet.

FMM_INVALID_PORT_NUM : nPort does not exist among connected ports.

FMM_INVALID_SLAVE_NUM : The board of corresponding iSlaveNo does not exist.

Remarks

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcInputStatus()
{
        BYTE nPortNo = 1;   // COMM Port number
        BYTE iSlaveNo = 0;  // Slave No (0 ~ 15)
        unsigned iInputNo =0;   // (Latch count No., 0~15)
        unsigned long* uCount;
```

```
        int nRtn;

        // Try to connect
        if (FAS_Connect(nPortNo, 115200) == FALSE)
        {
                // Connection failed.
                // Port might not be connected or Baudrate might not match.
                return;
        }

        nRtn = FAS_GetLatchCount(m_nPortNo, iSlaveNo, iInputNo, &uCount);

        _ASSERT(nRtn == FMM_OK);

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

See Also

FAS_GetLatchCountAll

Read the Latch Count of all channels.

(Ezi-IO-RS-I16 series : 0~15 /Ezi-IO-RS-I8O8 series : 0~7)

Syntax

```
int FAS_GetLatchCountAll(
    BYTE nPortNo,
    BYTE iSlaveNo,
    unsigned long** ppuAllCount
);
```

Parameters

nPortNo

Port number of the board.

iSlaveNo

Slave number of the board.

ppuAllCount

Parameter pointer to save Latch count value(0~15).

Return Value

FMM_OK : The command ran successfully.

FMM_NOT_OPEN : The board is not connected yet.

FMM_INVALID_PORT_NUM : nPort does not exist among connected ports.

FMM_INVALID_SLAVE_NUM : The board of corresponding iSlaveNo does not exist.

Remarks

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcInputStatus()
{
        BYTE nPortNo = 1;   // COMM Port number
        BYTE iSlaveNo = 0;  // Slave No (0 ~ 15)
        unsigned long Latch_All_Count[16] = 0 ; //Parameter pointer to save Latch count 0~16

        int nRtn;

        // Try to connect
        if (FAS_Connect(nPortNo, 115200) == FALSE)
        {
```

```
            // Connection failed.
            // Port might not be connected or Baudrate might not match.
            return;
        }

        nRtn = FAS_GetLatchCountAll(m_nPortNo, iSlaveNo,
                (unsigned long**) Latch_All_Count));

        _ASSERT(nRtn == FMM_OK);

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

See Also

FAS_ClearLatchCount

Reset(0) the Latch Count of corresponding bit.

(Ezi-IO-RS-I16 series : 0~15 /Ezi-IO-RS-I8O8 series : 0~7)

Syntax

```
int FAS_ClearLatchCount(
    BYTE nPortNo,
    BYTE iSlaveNo,
    unsigned long uInputMask
);
```

Parameters

nPortNo

Port number of the board.

iSlaveNo

Slave number of the board.

uInputmask

Bitmask value of Latch No. to Reset(0) Latch count

Return Value

FMM_OK : The command ran successfully.

FMM_NOT_OPEN : The board is not connected yet.

FMM_INVALID_PORT_NUM : nPort does not exist among connected ports.

FMM_INVALID_SLAVE_NUM : The board of corresponding iSlaveNo does not exist.

Remarks

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcInputStatus()
{
        BYTE nPortNo = 1;   // COMM Port number
        BYTE iSlaveNo = 0;  // Slave No (0 ~ 15)
        unsigned long uInputmask = 0x000000FF ;   //Reset Latch count 0~7

        int nRtn;

        // Try to connect
        if (FAS_Connect(nPortNo, 115200) == FALSE)
```

```
        {
                // Connection failed.
                // Port might not be connected or Baudrate might not match.
                return;
        }

        nRtn = FAS_ClearLatchCount (m_nPortNo, iSlaveNo, uInputMask);

        _ASSERT(nRtn == FMM_OK);

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

See Also

2 - 5 . Output Control Function

Function Name	Description
FAS_GetOutput	Read Output and Trigger(Run or Stop) status.
FAS_SetOutput	Configure Output.
FAS_SetTrigger	Configure Trigger setting.
FAS_SetRunStop	Configure 'Run and Stop' setting of Trigger.
FAS_GetTriggerCount	Read Trigger count of the selected output.

*Ezi-IO-RS-O16 : 0~15

*Ezi-IO-RS-I8O8 series : 8~15

FAS_GetOutput

Read Output and Trigger(Run or Stop) status.

(Ezi-IO-RS-O16 : 0~15 /Ezi-IO-RS-I8O8 series : 8~15)

Syntax

```
int FAS_GetOutput(
    BYTE nPortNo,
    BYTE iSlaveNo
    unsigned long* uOutput,
    unsigned long* uStatus
);
```

Parameters

nPortNo

Port number of the board.

iSlaveNo

Slave number of the board.

uOutput

Parameter pointer to save Output value.

uStatus

Parameter pointer to save Trigger Run/Stop value.

Return Value

FMM_OK : The command ran successfully.

FMM_INVALID_PORT_NUM : nPort does not exist among connected ports.

FMM_INVALID_SLAVE_NUM : The board of corresponding iSlaveNo does not exist.

Remarks

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcInputStatus()
{
        BYTE nPortNo = 1;   // COMM Port number
        BYTE iSlaveNo = 0;  // Slave No (0 ~ 15)
        unsigned long uOutput;
        unsigned long uStatus;

        int nRtn;
```

```
        // Try to connect
        if (FAS_Connect(nPortNo, 115200) == FALSE)
        {
                // Connection failed.
                // Port might not be connected or Baudrate might not match.
                return;
        }

        nRtn = FAS_GetOutput(m_nPortNo, iSlaveNo, &uOutput, &uStatus);

        _ASSERT(nRtn == FMM_OK);

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

See Also

FAS_SetOutput

Configure Output.

(Ezi-IO-RS-O16 : 0~15 /Ezi-IO-RS-I8O8 series : 8~15)

Syntax

```
int FAS_SetOutput(
    BYTE nPortNo,
    BYTE iSlaveNo
    unsigned long uSet,
    unsigned long uClear
);
```

Parameters

nPortNo

Port number of the board.

iSlaveNo

Slave number of the board.

uSet

Bitmask value to turn on Output.

uClear

Bitmask value to turn off Output.

Return Value

FMM_OK : The command ran successfully.

FMM_INVALID_PORT_NUM : nPort does not exist among connected ports.

FMM_INVALID_SLAVE_NUM : The board of corresponding iSlaveNo does not exist.

Remarks

If uSet and uClear set the same bit, it is treated as uClear.

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcInputStatus()
{
        BYTE nPortNo = 1;   // COMM Port number
        BYTE iSlaveNo = 0;  // Slave No (0 ~ 15)
        unsigned long uSet = 0x0000FF00; (Turn on Output No. 8~15.)
        unsigned long uClear = 0x000000FF; (Turn off Output No. 0~7.)
```

```
        int nRtn;

        // Try to connect
        if (FAS_Connect(nPortNo, 115200) == FALSE)
        {
                // Connection failed.
                // Port might not be connected or Baudrate might not match.
                return;
        }

        nRtn = FAS_SetOutput(m_nPortNo, iSlaveNo, uSet, uClear);

        _ASSERT(nRtn == FMM_OK);

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

See Also

FAS_SetTrigger

Configure Trigger Setting..

(Ezi-IO-RS-O16 : 0~15 /Ezi-RS-PR-I8O8 series : 8~15)

Syntax

```
int FAS_SetTrigger(
    BYTE nPortNo,
    BYTE iSlaveNo,
    unsigned char uOutputNo
    TRIGGER_INFO* pTrigger
);
```

Parameters

nPortNo

Port number of the board.

iSlaveNo

Slave number of the board.

uOutputNo

Output No. to set Trigger

pTrigger

Structure variable to set Trigger.(Count, OnTime, Period)

Return Value

FMM_OK : The command ran successfully.

FMM_NOT_OPEN : The board is not connected yet.

FMM_INVALID_PORT_NUM : nPort does not exist among connected ports.

FMM_INVALID_SLAVE_NUM : The board of corresponding iSlaveNo does not exist.

Remarks

Ontime is in [ms].

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcInputStatus()
{
        BYTE nPortNo = 1;   // COMM Port number
        BYTE iSlaveNo = 0;  // Slave No (0 ~ 15)
        TRIGGER_INFO Trg_info;
        unsigned char uOutputNo = 0;

        int nRtn;
```

```
        Trg_info.wCount   = 100; //Trigger output count setting
        Trg_info.wOnTime   = 10; // 'On' time setting
        Trg_info.wPeriod   = 30 ; // Period settong, for each 30[ms] On : 10[ms] Off : 20[ms]
        Trg_info.wReserved1 = 0;
        Trg_info.wReserved2 = 0;

        // Try to connect
        if (FAS_Connect(nPortNo, 115200) == FALSE)
        {
                // Connection failed.
                // Port might not be connected or Baudrate might not match.
                return;
        }

        nRtn = FAS_SetTrigger(m_nPortNo, iSlaveNo, uOutputNo, &Trg_info);

        _ASSERT(nRtn == FMM_OK);

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

See Also

FAS_SetRunStop

Configure 'Run and Stop' setting of Trigger.

(Ezi-IO-RS-O16 : 0~15 /Ezi-IO-RS-I8O8 series : 8~15)

Syntax

```
int FAS_SetRunStop(
    BYTE nPortNo,
    BYTE iSlaveNo
    unsigned long uRun,
    unsigned long uStop
);
```

Parameters

nPortNo

Port number of the board.

iSlaveNo

Slave number of the board.

uRun

Bitmask value to run Trigger.

uStop

Bitmask value to stop Trigger.

Return Value

FMM_OK : The command ran successfully.

FMM_INVALID_PORT_NUM : nPort does not exist among connected ports.

FMM_INVALID_SLAVE_NUM : The board of corresponding iSlaveNo does not exist.

Remarks

If uRun and uStop set the same bit, it is treated as uStop.

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcInputStatus()
{
        BYTE nPortNo = 1;   // COMM Port number
        BYTE iSlaveNo = 0;  // Slave No (0 ~ 15)
        unsigned long uRun = 0x0000FF00; (Run Trigger No. 8~15.)
        unsigned long uStop = 0x000000FF; (Stop Trigger No. 0~7.)
```

```
        int nRtn;

        // Try to connect
        if (FAS_Connect(nPortNo, 115200) == FALSE)
        {
                // Connection failed.
                // Port might not be connected or Baudrate might not match.
                return;
        }

        nRtn = FAS_SetRunStop(m_nPortNo, iSlaveNo, uRun, uStop);

        _ASSERT(nRtn == FMM_OK);

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

See Also

FAS_GetTriggerCount

Read Trigger count of the selected output.

(Ezi-IO-RS-O16 : 0~15 /Ezi-IO-RS-I8O8 series : 8~15)

Syntax

```
int FAS_GetTriggerCount(
    BYTE nPortNo,
    BYTE iSlaveNo
    unsigned char uOutputNo,
    unsigned long* uCount
);
```

Parameters

nPortNo

Port number of the board.

iSlaveNo

Slave number of the board.

uOutputNo

Output No. to read Trigger count.

uCount

Parameter pointer to save Trigger Count value.

Return Value

FMM_OK : The command ran successfully.

FMM_INVALID_PORT_NUM : nPort does not exist among connected ports.

FMM_INVALID_SLAVE_NUM : The board of corresponding iSlaveNo does not exist.

Remarks

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcInputStatus()
{
        BYTE nPortNo = 1;   // COMM Port number
        BYTE iSlaveNo = 0;  // Slave No (0 ~ 15)
        unsigned char uOutputNo = 1; (Output No. 0~15)
        unsigned long* uCount;

        int nRtn;
```

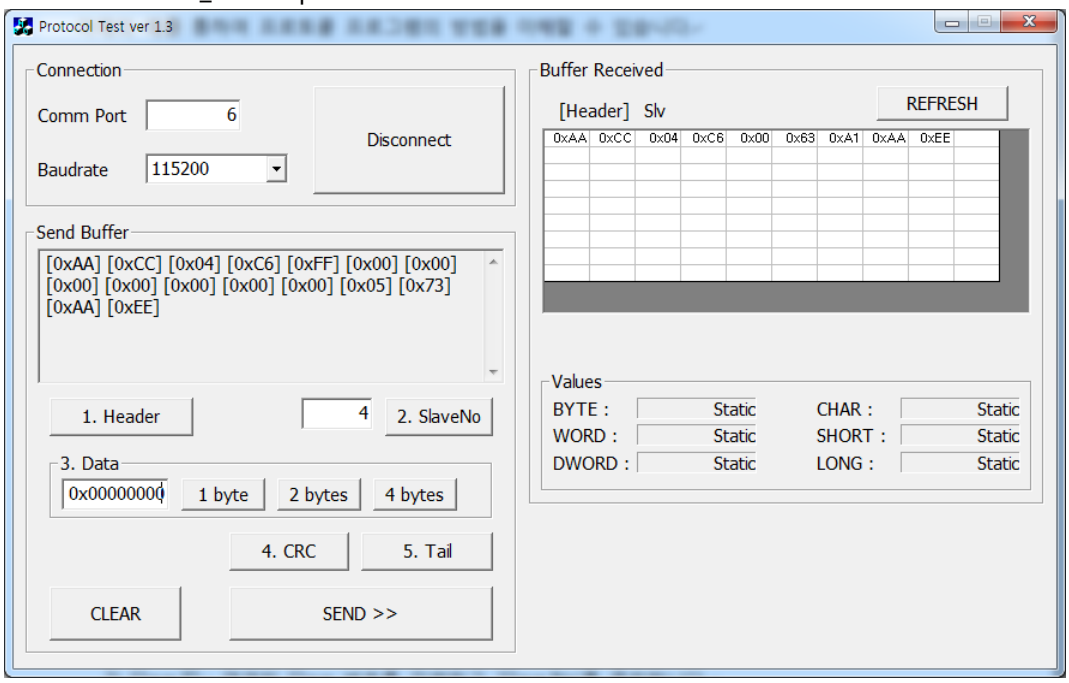
```
        // Try to connect
        if (FAS_Connect(nPortNo, 115200) == FALSE)
        {
                // Connection failed.
                // Port might not be connected or Baudrate might not match.
                return;
        }

        nRtn = FAS_GetTriggerCount(m_nPortNo, iSlaveNo, uOutputNo, &uCount);

        _ASSERT(nRtn == FMM_OK);

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

See Also

3 . PLC Program Protocol

When Protocol Test_Plus R.exe is executed in the folder where the user program (GUI) is installed, the following window is activated.
You can learn the method of the protocol program through the example below.

(1) Protocol for FAS_SetOutput command



All protocols should include information of Header, Slave No., Frame type, Data, CRC and Tail.

1) After selecting 'Comm Port' and 'Baudrate', click 'Connect'.
2) Header : Click 'Header' and '[0xAA][0xCC]' will appear on 'Send Buffer' window.
3) Slave ID : Enter the connected slave number and clidk 'Slave No'.
4) Frame type : Select the type of command.
'You can check the contents of FAS-SetOutput command in 1-2-1. Frame type and Data configuration as listed below.

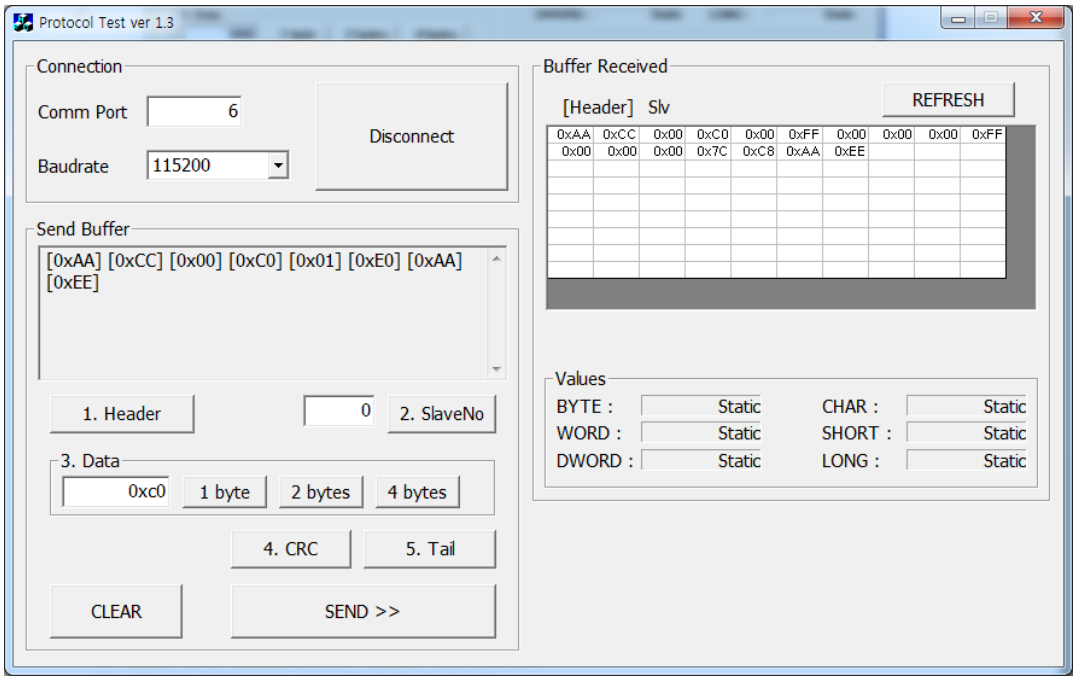
Frame type	DLL Library name	Data
198 (0xC6)	FAS_SetOutput	Refer to 「1-2-1. Frame type and Data Configuration」

Enter '0xC6', and click '1 byte' as transmit frame type is 1 byte.

5) Data : Enter Set Output data (0x000000ff) and click 4bytes, enter Reset Output data (0x00000000) and click 4bytes.
6) CRC : If you click this, checksum value is automatically calculated and shown as 2 bytes.
7) Tail : If you click 'Tail', '[0xAA][0xEE]' is displayed.

8) Lastly, click 'Send' and data entered in 'Send Buffer' window is sent to Ezi-IO RS-485. You can check the output with LED lighting.
9) The command is normally sent to Ezi-IO RS-485, and the content normally received from Ezi-IO RS-485 is displayed in the 'Buffer Received' window.

(2) Protocol for FAS_GetInput command



1) Header
2) Slave No.
3) Frame type : Enter '0xC0' for 'FAS_GetInput' command. Click 1byte.
4) CRC
5) Tail
6) Send : You can read the current input status and check it with the contents displayed in the Buffer Received window.

(3) PLC Programming

The difference between the actual protocol program and the above example is as follows. The following functions are automatically performed in the above test GUI program.

1) The 'Byte stuffing' technique is used to clearly distinguish the header from the tail. Please refer to 「1-1-2. RS-485 Communication Protocol」.
2) Use CRC function to check communication error.
3) Refer to 「1-1-4. CRC Calculation Example」.

Fast, Accurate, Smooth Motion

FASTECH Co., Ltd.

Rm#1202, 401-dong, Bucheon Techno-Park,
655, Pyeongcheon-ro, Bucheon-si Gyeonggi-do,
Republic of Korea (Postal Code: 14502)
TEL : +82-32-234-6317 FAX : +82-32-234-6302
E-mail : sales@fastech-motions.com
Homepage : www.fastech-motions.com

- It is prohibited to unauthorized or reproduced in whole or in part described in the User's Guide.
- If you need a user manual to the loss or damage, etc., please contact us or your nearest distributor.
- User manual are subject to change without notice to improve the product or quantitative changes in specifications and user's manual.
- Ezi-IO RS-485 is registered trademark of FASTECH Co., Ltd in the national registration

ⓒ Copyright 2021 FASTECH Co.,Ltd. Dec 28, 2021 Rev.03