Ezi-MOTIONLINK® Plus-R

Network based Motion Controller Plug-in to Servo Drives

User Manual

Communication Function Ver_6

(Rev.04)

FASTECH

- Table of Contents -

1 . Communication Protocols

1－1. Communication Functions

Ezi-MOTIONLINK-Plus-R can control up to 16 axes by multidrop link at RS-485(two-wire) using Daisy-Chain.

Caution	**Pay attention that when Windows goes into standby or power-save mode, serial communication is basically disconnected. When the system is recovered from standby mode, it should be connected again with serial communication. This is also applicable to the library provided.**

1－1－1. Communication Specifications

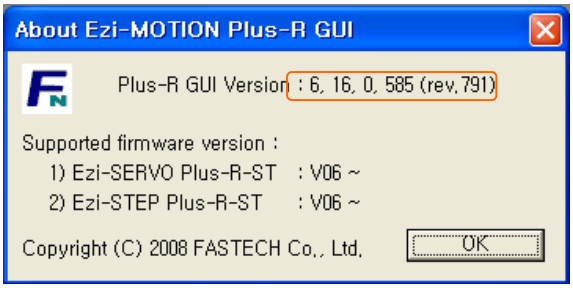
Specification	RS-485
Communication Type	Asynchronous
	Half-duplex
Baud Rate [bps]	19200, 38400, 57600, 115200, 230400,460800,921600
Data Type	8bit Binary ASCII Code, HEX
Parity	No
Stop Bit	1bit
CRC Check	Yes(CRC-16)
Max Cabling Length (Converter ↔ End of Ezi-MOTIONLINK-Plus-R)	Within 30 [m]
Min Cable length between Ezi-MOTIONLINK-Plus-R	More than 60 [cm]
Number of Connected Axes	16 axes (No. 0~F)

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

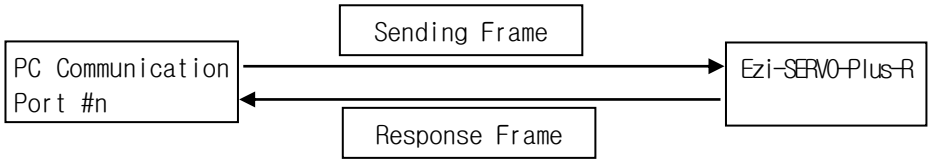
Thers are 2 kinds of GUI version. This manual support for 'Version 6' level.

Type	Firmware version	compatability	User Program(GUI) version
1	Level 6 (V06.0x.0xx.xx)	<->	Level 6 (6.xx.x.xxx)
2	Level 8 (V08.xx.0xx.xx)	<->	Level 8 (8.xx.x.xxx)

After connect the User Program(GUI),
Version number can be check in
'About Plus-R GUI...'menu in 'Help' menu.



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

Detailed ***Frame Data*** is configured as follows:

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

① Slave ID : Ezi-MOTIONLINK-Plus-R 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 「1-2-1. Frame Type and Data Configuration」section.
③ Data : Data structure and length is set according to Frame type. For more information, refer to 「1-2-1. 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.

1－1－3. CRC Calculation Example

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

1)'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) '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;
 }
```

1－1－4. Response Frame Structure and Communication Error (Ver6)

When any command is sent, the basic structure of Frame at the response side is same. However, there is a difference in case of ***Frame Data,*** which 'communication status' is added as shown below.

Slave ID	Frame Type	Data		CRC	
1 byte	1 byte	1 byte	0~247 bytes	2 bytes	
		Communication status	Response data	Low byte	High byte

① Slave ID : Same to sending Frame.
(When this is not same to sending data, it should be recognized as the error status.)

② Frame type : Same to sending Frame.

(When this is not same to sending data, it should be recognized as the error status.)

③ Data : When simple executive instructions are sent, this data cannot be read. However, in case of response, 1 byte is added to display the communication status (error / normal).

The code by bytes means the '**Communication status'** as follows.

Hexa Code	Decimal Code	Description
0x00	0	Communication is normal.
0x80	128	Frame Type Error : Responded Frame type cannot be recognized.
0x81	129	Data error, ROM data read/write error : Data value responded is without the given range.
0x82	130	Received Frame Error : Frame data received is out of this specification.
0x85	133	Running Command Failure : The user has tried to execute new running commands in wrong condition as follows. 1) currently motor is running 2) currently motor is stopping 3) currently Servo is OFF status 4) try to Z-pulse Origin without external encoder 5) other wrong motion command
0x86	134	RESET Failure : The user has tried to execute new running in condition as follows. 1) While the servo is ON 2) Already RESET in ON by external input signal
0x87	135	Servo ON Failure ① : While an alarm occurs, the user has tried to execute Servo ON command.
0x88	136	Servo ON Failure ② : While Emergency Stop occurs, the user has tried to execute Servo ON command.
0x89	137	Servo ON Failure ③ : 'ServoON' signal is assigned to external input signal already. Servo ON/OFF can execute by external input signal only.
0xAA	170	CRC Error : Frame data received is occurred CRC error by noise influence, etc. In this case, DLL Library of sending side automatically try to send 1 more time.

Caution	**1) If 'Header' and 'Slave ID' values in the sending 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</th></tr>
<tr><td>0x01
(1)</td><td>FAS_
GetSlaveInfo</td><td>Connected slave type and program version information are required.

Sending : 0 byte
Response : 1~248 bytes

<table>
<tr><td>1 byte</td><td>1 bytes</td><td>0~246 bytes</td></tr>
<tr><td>Communication status</td><td>Slave type</td><td>ACII string with NULL byte (strlen() + 1 byes)</td></tr>
</table>

◆ Slave type : 11 : Ezi-MOTIONLINK-Plus-R</td></tr>
<tr><td>0x10
(16)</td><td>FAS_
SaveAllParameters</td><td>Current setting parameters & assign of IO signals are saved in the ROM of Ezi-MOTIONLINK-Plus-R. Even though the product is powered off, saving these must be possible.
(Values set at 'FAS_SetParameter' &'FAS_SetIOAssignMap' are saved together.)

Sending : 0 byte
Response : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>Communication status</td></tr>
</table></td></tr>
<tr><td>0x11
(17)</td><td>FAS_
GetRomParameter</td><td>Specific parameter values in the ROM are read.

Sending : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>Parameter number (0~32)</td></tr>
</table>

Response : 5 bytes
<table>
<tr><td>1 byte</td><td>4 bytes</td></tr>
<tr><td>Communication status</td><td>Parameter value</td></tr>
</table>

Refer to 「1-2-2. Parameter List」</td></tr>
<tr><td>0x12
(18)</td><td>FAS_
SetParameter</td><td>Specific parameter values are saved to the RAM of Ezi-MOTIONLINK-Plus-R.

Sending : 5 bytes
<table>
<tr><td>1 byte</td><td>4 bytes</td></tr>
<tr><td>Parameter number (0~32)</td><td>Parameter value</td></tr>
</table>

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

Refer to 「1-2-2. Parameter List」</td></tr>
</table>

<table>
<tr><td></td><td></td><td></td></tr>
<tr><td>0x13
(19)</td><td>FAS_
GetParameter</td><td>Specific parameter values in the RAM are read

Sending : 1 byte<table><tr><td>1 byte</td></tr><tr><td>Parameter number (0~32)</td></tr></table>
Response : 5 bytes<table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication status</td><td>Parameter value</td></tr></table>Refer to 「1-2-2. Parameter List」</td></tr>
<tr><td>0x20
(32)</td><td>FAS_
SetIOOutput</td><td>Output signal level of the control output port is set.

Sending : 8 bytes<table><tr><td>4 bytes</td><td>4 bytes</td></tr><tr><td>I/O set mask value</td><td>I/O clear mask value</td></tr></table>
When specific bit of the set mask is ‘1’, the relevant output port signal is set to [ON].
When specific bit of the clear mask is ‘1’, the relevant output port signal is set to [OFF].
For more information, refer to 「1-2-3. Bit setup of Output Pin」.

Response : 1 byte<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x21
(33)</td><td>FAS_
SetIOInput</td><td>Input signal level of the control input port is set.

Sending : 8 bytes<table><tr><td>4 bytes</td><td>4 bytes</td></tr><tr><td>I/O set mask value</td><td>I/O clear mask value</td></tr></table>
When specific bit of the set mask is ‘1’, the relevant input port signal is set to [ON].
When specific bit of the clear mask is ‘1’, the relevant input port signal is set to [OFF].
For more information, refer to「1-2-4. Bit setup of Input Pin」.

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

<table>
<tr><td>0x22
(34)</td><td>FAS_
GetIOInput</td><td>Current input signal status of the control input port is read.

Sending : 0 byte
Response : 5 byte<table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication status</td><td>Input status value</td></tr></table>
Relevant bit by each input signal, refer to 「1-2-4. Bit setup of Input Pin」.</td></tr>
<tr><td>0x23
(35)</td><td>FAS_
GetIOOutput</td><td>Current output signal status of the control output port is read.

Sending : 0 byte
Response : 5 byte<table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication status</td><td>Output status value</td></tr></table>
Relevant bit by each output signal, refer to 「1-2-3. Bit setup of Output Pin」.</td></tr>
<tr><td>0x24
(36)</td><td>FAS_
SetIOAssignMap</td><td>To assign control I/O signals to the pin of CN1 port and set the signal level. By running 'FAS_SaveAllParameters', you can save the setting value to the ROM.

Sending : 6 bytes<table><tr><td>1 byte</td><td>4 bytes</td><td>1 byte</td></tr><tr><td>I/O number</td><td>I/O pin masking data</td><td>Setting level</td></tr></table>
◆I/O number: '0~11' corresponds to 'Limit+, Limit-, Org, IN1,..., IN9' respectively, and '12~22' corresponds to 'COMP, OUT1,..., OUT9' respectively.
◆I/O pin masking data: Refer to 「1-2-4. Bit setup of Input Pin」.
◆Level Setting: 0: Active Low, 1:Active High

Response : 1 byte<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x25
(37)</td><td>FAS_
GetIOAssignMap</td><td>I/O setting status of CN3 is read.

Sending : 1 byte<table><tr><td>1 byte</td></tr><tr><td>I/O number</td></tr></table>
◆I/O number: '0~7' corresponds to 'Limit+, Limit-, Org, IN1, ..., IN5' respectively, and '8~11' corresponds to 'COMP, OUT1, ..., OUT3' respectively.

Response : 6 bytes<table><tr><td>1 byte</td><td>4 bytes</td><td>1 byte</td></tr><tr><td>Communication status</td><td>IO pin masking status</td><td>Level status</td></tr></table>
For more information, refer to '0x24'Frame type.</td></tr>
</table>

<table>
<tr><td>0x26
(38)</td><td>FAS_
IOAssignMapReadROM</td><td>Control I/O setting status and signal level setting value are read from ROM area.

Sending : 0 byte
Response : 2 bytes
<table><tr><td>1 byte</td><td>1 byte</td></tr><tr><td>Communication status</td><td>Command performing status
(0 : complete, values except 0: error)</td></tr></table></td></tr>
<tr><td>0x27
(39)</td><td>FAS_
TriggerOutput_RunA</td><td>To occur a command for ‘Compare Out’ signal

Sending : 18 bytes
<table><tr><td>1 byte</td><td>4 bytes</td><td>4 bytes</td></tr><tr><td>Output start/stop command
(1:start 0:stop)</td><td>Pulse start position [pulse]</td><td>Pulse period [pulse]</td></tr></table>
<table><tr><td>4 bytes</td><td>1 byte</td><td>4 bytes</td></tr><tr><td>Pulse width [msec]</td><td>Output pin number
(fix to 0)</td><td>spare</td></tr></table>◆ Pulse start position: Setting the start position of first pulse output.
(-134,217,727 ~134,217,727)
◆ Pulse period: Setting the pulse period.
(0: pulse output only 1 time in pulse start position
1~134,217,727 : pulse output repeatedly depends on setting)
◆ Pulse width: Setting the pulse width. (1~1000)

Response : 2 bytes
<table><tr><td>1 byte</td><td>1 byte</td></tr><tr><td>Communication status</td><td>Command performing status
(0: complete, values except 0: error)</td></tr></table></td></tr>
<tr><td>0x28
(40)</td><td>FAS_
TriggerOutput_Status</td><td>Command to check if the trigger output pulse is working or not.

Sending : 0 byte
Response : 2 bytes
<table><tr><td>1 byte</td><td>1 byte</td></tr><tr><td>Communication status</td><td>Status (1 :output ON, 0 :output OFF)</td></tr></table></td></tr>
<tr><td>0x2A
(42)</td><td>FAS_
ServoEnable</td><td>Set Servo ON/OFF status.

Sending : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>0:OFF, 1:ON</td></tr></table>
Response : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
</table>

<table>
<tr><td>0x2B
(43)</td><td>FAS_
ServoAlarmReset</td><td>Reset Servo alarm status.

Sending : 0 byte
Response : 1 byte<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x31
(49)</td><td>FAS_
MoveStop</td><td>To request to stop running the motor

Sending : 0 byte
Response : 1 byte<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x32
(50)</td><td>FAS_
EmergencyStop</td><td>To request the running motor to stop emergently

Sending : 0 byte
Response : 1 byte<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x33
(51)</td><td>FAS_
MoveOriginSingleAxis</td><td>To request the motor to return to the origin at the current setting parameter condition

Sending : 0 byte
Response : 1 byte<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x34
(52)</td><td>FAS_
MoveSingleAxisAbsPos</td><td>To request the motor to move its position as much as the absolute value[pulse]

Sending : 8 bytes<table><tr><td>4 bytes</td><td>4 bytes</td></tr><tr><td>Absolute position value</td><td>Running speed [pps]</td></tr></table>
Response : 1 byte<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x35
(53)</td><td>FAS_
MoveSingleAxisIncPos</td><td>To request the motor to move its position as much as the incremental value[pulse]
Sending : 8 bytes<table><tr><td>4 bytes</td><td>4 bytes</td></tr><tr><td>Incremental position value</td><td>Running speed [pps]</td></tr></table>
Response : 1 byte<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
</table>

<table>
<tr><td>0x36
(54)</td><td>FAS_
MoveToLimit</td><td>To request the motor to start limit motion at the current setting parameter condition

Sending : 5 bytes<table><tr><td>4 bytes</td><td>1 byte</td></tr><tr><td>Running speed [pps]</td><td>Running direction (0: -Limit 1: +Limit)</td></tr></table>
Response : 1 byte<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x37
(55)</td><td>FAS_
MoveVelocity</td><td>To request the motor to start Jog motion at the current setting parameter condition

Sending : 5 bytes<table><tr><td>4 bytes</td><td>1 byte</td></tr><tr><td>Running speed [pps]</td><td>Running direction (0: -Jog 1: +Jog)</td></tr></table>
Response : 1 byte<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x38
(56)</td><td>FAS_
PositionAbsOverrid
e</td><td>To request the motor to change the target absolute position value[pulse] while it is in running.

Sending : 4 bytes<table><tr><td>4 bytes</td></tr><tr><td>Changed command position value [pulse]</td></tr></table>
Response : 1 byte<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x39
(57)</td><td>FAS_
PositionIncOverride</td><td>To request the motor to change the target incremental position value[pulse] while it is in running.

Sending : 4 bytes<table><tr><td>4 bytes</td></tr><tr><td>Changed command position value [pulse]</td></tr></table>
Response : 1 byte<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
</table>

<table>
<tr><td>0x3A
(58)</td><td>FAS_
VelocityOverride</td><td>To request the motor to change the running speed value[pps] while it is in running.
Sending : 4 bytes<table><tr><td>4 bytes</td></tr><tr><td>Changed running speed [pps]</td></tr></table>The accel/decel time is assigned to 'Axis Acc Time' and 'Axis Dec Time' value in parameter lists.

Response : 1 byte<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x3B
(59)</td><td>FAS_
AllMoveStop</td><td>To request stop for all motor that connected in same port.

Sending : 0 byte
(Slave number must be '99')
Response : no response
(Can not get a reply from all slave at the same time.
So all slave does not send a response.)</td></tr>
<tr><td>0x3C
(60)</td><td>FAS_
AllEmergencyStop</td><td>To request emergency stop for all motor that connected in same port.

Sending : 0 byte
(Slave number must be '99')
Response : no response
(Can not get a reply from all slave at the same time.
So all slave does not send a response.)</td></tr>
<tr><td>0x3D
(61)</td><td>FAS_All
MoveOriginSingleAxis</td><td>To request return to the origin at the current setting parameter condition for all motors that connected in same port.

Sending : 0 byte
(Slave number must be '99')
Response : no response
(Can not get a reply from all slave at the same time.
So all slave does not send a response.)</td></tr>
<tr><td>0x3E
(62)</td><td>FAS_All
SingleAxisAbsPos</td><td>To request move its position as much as the absolute value[pulse] for all motors that connected in same port.

Sending : 8 bytes (Slave number must be '99')<table><tr><td>4 bytes</td><td>4 bytes</td></tr><tr><td>Absolute position value</td><td>Running speed [pps]</td></tr></table>
Response : no response
(Can not get a reply from all slave at the same time.
So all slave does not send a response.)</td></tr>
</table>

<table>
<tr>
<td>0x3F
(63)</td>
<td>FAS_All
SingleAxisIncPos</td>
<td>To request move its position as much as the incremental value[pulse] for all motors that connected in same port.

Sending : 8 bytes (Slave number must be '99')
<table>
<tr><td>4 bytes</td><td>4 bytes</td></tr>
<tr><td>Incremental position value</td><td>Running speed [pps]</td></tr>
</table>

Response : no response
(Can not get a reply from all slave at the same time.
So all slave does not send a response.)</td>
</tr>
<tr>
<td>0x80
(128)</td>
<td>FAS_
MoveSingleAxisAbsPosEx</td>
<td>Request the motor to move its position as much as the absolute value[pulse] with Custom Accel. / Decel. Time[msec]

Sending: 40 bytes
<table>
<tr><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td><td>2 bytes</td></tr>
<tr><td>Absolute position value</td><td>Running speed [pps]</td><td>Flag option</td><td>Custom Accel. Time (1~9999)</td></tr>
</table>
<table>
<tr><td>2 bytes</td><td>24 bytes</td></tr>
<tr><td>Custom Decel. Time (1~9999)</td><td>Reserved</td></tr>
</table>

Flag ooption : 0x0001 : reserved
0x0002 : Custom Accel. Time is used.
0x0004 : Custom Decel. Time is used.

If the Flag bit is OFF status(0), Accel./Decel. Time value is used that saved in controller.

Response: 1 byte</td>
</tr>
<tr>
<td>0x81
(129)</td>
<td>FAS_
MoveSingleAxisIncPosEx</td>
<td>Request the motor to move its position as much as the absolute value[pulse] with Custom Accel. / Decel. Time[msec]

Sending: 40 bytes
<table>
<tr><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td><td>2 bytes</td></tr>
<tr><td>incremental position value</td><td>Running speed [pps]</td><td>Flag option</td><td>Custom Accel. Time (1~9999)</td></tr>
</table>
<table>
<tr><td>2 bytes</td><td>24 bytes</td></tr>
<tr><td>Custom Decel. Time (1~9999)</td><td>Reserved</td></tr>
</table>

Flag option : 0x0001 : reserved
0x0002 : Custom Accel. Time is used.
0x0004 : Custom Decel. Time is used.
If the Flag bit is OFF status(0), Accel./Decel. Time value is used that saved in controller.

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

<table>
<tr><td>0x82
(130)</td><td>FAS_
MoveVelocityEx</td><td>Request to start Jog movement with acceleration value[msec].

Sending: 37 bytes
<table><tr><td>4 bytes</td><td>1 byte</td><td>4 bytes</td></tr><tr><td>Running speed [pps]</td><td>Running direction (0: -Jog 1: +Jog)</td><td>Flag option</td></tr></table>
<table><tr><td>2 bytes</td><td>26 bytes</td></tr><tr><td>Custom Accel./Decel. Time (1~9999)</td><td>Reserved</td></tr></table>
Flag option : 0x0001 : reserved
0x0002 : Custom Accel./Decel. Time is used.

If the Flag bit is OFF status(0), Accel./Decel. Time value is used that saved in controller.

Response : 1 byte</td></tr>
<tr><td>0x70</td><td>FAS_MoveLinearAbsPos</td><td>To request Linear Interpolation move its position as much as the absolute value[pulse] for more than 2 motors that connected in same port.
The position value is ins absolute value[pulse].
Refer to 「2. Library for PC program(Ver6)」.</td></tr>
<tr><td>0x71</td><td>FAS_MoveLinearIncPos</td><td>To request Linear Interpolation move its position as much as the incremental value[pulse] for more than 2 motors that connected in same port.
The position value is ins incremental value[pulse].
Refer to 「2. Library for PC program(Ver6)」.</td></tr>
<tr><td>0x40
(64)</td><td>FAS_
GetAxisStatus</td><td>To request the Flag value of displaying the running status

Sending : 0 byte
Response : 5 bytes

<table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication status</td><td>Status flag value</td></tr></table>
For bit related to each Flag, refer to 「1-2-5. Bit setup of Status Flag」.</td></tr>
<tr><td>0x41
(65)</td><td>FAS_
GetIOAxisStatus</td><td>To request the I/O status and the running Flag status.
(Frame type 0x22, 0x23, and 0x40 are packed.)

Sending : 0 byte
Response : 13 bytes

<table><tr><td>1 byte</td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td></tr><tr><td>Communication status</td><td>Input status value</td><td>Output status value</td><td>Status flag value</td></tr></table></td></tr>
</table>

<table>
<tr><td>0x42
(66)</td><td>FAS_
GetMotionStatus</td><td>To request the current running progress status and its PT number
(Frame type 0x51, 0x53, 0x54, and 0x55 are packed.)

Sending : 0 byte
Response : 21 bytes

<table>
<tr><td>1 byte</td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td></tr>
<tr><td>Communication status</td><td>Command position value</td><td>Actual Position value</td><td>Position Difference value</td><td>Running speed value</td><td>Current running PT number</td></tr>
</table>
</td></tr>
<tr><td>0x43
(67)</td><td>FAS_
GetAllStatus</td><td>To request all data including the current running status
(Frame type 0x41, and 0x42 are packed.)

Sending : 0 byte
Response : 33 bytes
<table>
<tr><td>1 byte</td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td></tr>
<tr><td>Communication status</td><td>Input status value</td><td>Output status value</td><td>Status flag value</td></tr>
</table>

<table>
<tr><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td></tr>
<tr><td>Command position value</td><td>Actual position value</td><td>Position Difference value</td><td>Running speed value</td><td>Reserved</td></tr>
</table>
</td></tr>
<tr><td>0x50
(80)</td><td>FAS_
SetCommandPos</td><td>The command position value is continuously updated during operation. It is allowed to check the change of the command position value after setting this command position before starting operation.

Sending : 4 bytes
<table>
<tr><td>4 bytes</td></tr>
<tr><td>Command position setting count value</td></tr>
</table>

Response : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>Communication status</td></tr>
</table>
</td></tr>
<tr><td>0x51
(81)</td><td>FAS_
GetCommandPos</td><td>To request the command position value[pulse] being tracked.

Sending : 0 byte
Response : 5 bytes

<table>
<tr><td>1 byte</td><td>4 bytes</td></tr>
<tr><td>Communication status</td><td>Command position value</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>0x52
(82)</td><td>FAS_
SetActualPos</td><td>If Encoder feedback is used, the actual position value is continuously updated during operation. The actual position can be set to this value before starting operation to check the actual position value.

Sending : 4 bytes<table><tr><td>4 bytes</td></tr><tr><td>Actual position count value</td></tr></table>
Response : 1 byte<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x53
(83)</td><td>FAS_
GetActualPos</td><td>To request the current actual position value[pulse].

Sending : 0 byte

Response : 5 bytes<table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication status</td><td>Actual position value</td></tr></table></td></tr>
<tr><td>0x54
(84)</td><td>FAS_
GetPosError</td><td>To request the difference[pulse] between the command position value and the actual position value.

Sending : 0 byte
Response : 5 bytes

<table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication status</td><td>Position difference value</td></tr></table>
By this value, the user can check the current running status (how much inposition is tracked).</td></tr>
<tr><td>0x55
(85)</td><td>FAS_
GetActualVel</td><td>To request the current running speed value [pps]

Sending : 0 byte

Response : 5 bytes<table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication status</td><td>Speed value</td></tr></table></td></tr>
<tr><td>0x56
(86)</td><td>FAS_
ClearPosition</td><td>Ezi-MOTIONLINK Plus-R is closed-loop controlled, so the command position value is continuously updated during operation. This command position and actual position value set (Actual position) to '0' before starting operation.

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><td>0x58
(88)</td><td>FAS_
MovePause</td><td>To request the pause start and pause end of motor motioning.

Sending : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>0:pause release, 1:pause start</td></tr></table>
Response : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x60
(96)</td><td>FAS_
PosTableReadItem</td><td>To read PT values in the RAM of the drive.

Sending : 2 bytes
<table><tr><td>2 bytes</td></tr><tr><td>Readable PT No. (0~255)</td></tr></table>
Response : 65 bytes
<table><tr><td>1 byte</td><td>64 bytes</td></tr><tr><td>Communication Status</td><td>Relevant PT value</td></tr></table>
For items by each PT, refer to 「1-2-6. Position Table Item」.</td></tr>
<tr><td>0x61
(97)</td><td>FAS_
PosTableWriteItem</td><td>To save PT values to the RAM of the drive.

Sending : 66 bytes
<table><tr><td>2 bytes</td><td>64 bytes</td></tr><tr><td>PT No. (0~255)</td><td>Relevant PT value</td></tr></table>
For items by each PT, refer to 「1-2-6. Position Table Item」.

Response : 2 bytes
<table><tr><td>1 byte</td><td>1 byte</td></tr><tr><td>Communication Status</td><td>Command performing status
(values except 0 : complete, 0: error)</td></tr></table></td></tr>
<tr><td>0x62
(98)</td><td>FAS_
PosTableReadROM</td><td>To read all PT values (256) in the ROM of the drive.

Sending : 0 byte

Response : 2 bytes
<table><tr><td>1 byte</td><td>1 byte</td></tr><tr><td>Communication
Status</td><td>Command performing status
(0: complete, values except 0: error)</td></tr></table></td></tr>
<tr><td>0x63
(99)</td><td>FAS_
PosTableWriteROM</td><td>To save all PT value(256) to the ROM of the drive.

Sending : 0 byte

Response : 2 bytes</td></tr>
</table>

<table>
<tr><td></td><td></td><td>
<table>
<tr><td>1 byte</td><td>1 byte</td></tr>
<tr><td>Communication Status</td><td>Command performing status
(0: complete, values except 0: error)</td></tr>
</table>
</td></tr>
<tr><td>0x64
(100)</td><td>FAS_
PosTableRunItem</td><td>To start the position table operation from the designated PT number.

Sending : 2 bytes
<table>
<tr><td>2 bytes</td></tr>
<tr><td>PT No. (0~255)</td></tr>
</table>
Response : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>Communication Status</td></tr>
</table>
</td></tr>
<tr><td>0x6A
(106)</td><td>FAS_
PosTableReadOneItem</td><td>To read specific values of PT items in the RAM of the drive.

Sending : 4 bytes
<table>
<tr><td>2 bytes</td><td>2 bytes</td></tr>
<tr><td>PT No. to read
(0~255)</td><td>Offset value of the specific item to read
(0~40)</td></tr>
</table>
Refer to 「1-2-6. Position Table Item」 for Offset value.

Response : 5 bytes
<table>
<tr><td>1 byte</td><td>4 bytes</td></tr>
<tr><td>Communication Status</td><td>Relevant one of PT value</td></tr>
</table>
</td></tr>
<tr><td>0x6B
(107)</td><td>FAS_
PosTableWriteOneItem</td><td>To save specific values of PT items in the RAM of the drive.

Sending : 8 bytes
<table>
<tr><td>2 bytes</td><td>2bytes</td><td>4 bytes</td></tr>
<tr><td>PT No. To save(0~255)</td><td>Offset value of the specific item to save
(0~40)</td><td>Stored value</td></tr>
</table>
Refer to 「1-2-6. Position Table Item」 for Offset value.

Response : 2 bytes
<table>
<tr><td>1 byte</td><td>1 byte</td></tr>
<tr><td>Communication Status</td><td>Command performing status
(values except 0: complete, 0: error)</td></tr>
</table>
</td></tr>
</table>

* Frame Type '0x65'~ '0x69', '0x90'~ '0x92' are allotted for internal use.

1－2－2. Parameter Lists

No.	Name	Unit	Lower	Upper	Default
0	Encoder Multiply		0	3	3
1	Axis Max Speed	[pps]	1	2,500,000	500,000
2	Axis Start Speed	[pps]	1	2,500,000	1
3	Axis Acc Time	[msec]	1	9,999	100
4	Axis Dec Time	[msec]	1	9,999	100
5	Speed Override	[%]	1	500	100
6	Jog Speed	[pps]	1	2,500,000	5,000
7	Jog Start Speed	[pps]	1	2,500,000	1
8	Jog Acc Dec Time	[msec]	1	9,999	100
9*1	S/W Limit Plus Value	[pulse]	-134,217,728	134,217,727	134,217,727
10*1	S/W Limit Minus Value	[pulse]	-134,217,728	134,217,727	-134,217,728
11	S/W Limit Stop Method		0	1	0
12	H/W Limit Stop Method		0	1	0
13	Limit Sensor Logic		0	1	0
14	Org Speed	[pps]	1	2,500,000	5,000
15	Org Search Speed	[pps]	1	2,500,000	1,000
16	Org Acc Dec Time	[msec]	1	9,999	50
17	Org Method		0	5	0
18	Org Direction		0	1	1
19*1	Org OffSet	[pulse]	-134,217,728	134,217,727	0
20*1	Org Position Set	[pulse]	-134,217,728	134,217,727	0
21	Org Sensor Logic		0	1	0
22	Limit Sensor Direction		0	1	0
23	Pulse Type		0	1	1
24	Encoder Direction		0	1	0
25	Motion Direction		0	1	0
26	Servo Alarmreset Logic		0	1	0
27	Servo On Output Logic		0	1	0
28	Servo Alarm Logic		0	1	1
29	Servo Inposition Logic		0	1	0
30*2	Servo Alarmreset On Time	msec	10	1000	10
31*2	Use Motion Queue		0	1	0
32*2	Motion Profile		0	1	0
33*2	Origin Search OK Flag Off		0	3	0

*1 The parameter range differs from the product version, listed as below.
V06.03.0**4**x.xx : -134,217,728 ~134,217,727
V06.03.0**5**x.xx : - 2,147,483,648 ~ 2,147,483,647

*2 The parameter is available from the product version V06.03.0xx.**28** and above.

1－2－3. Bit setting of the control output pin

This displays the detailed description for 0x20 Frame type.
This command is applicable only to 3 signals of 'User Output 0' ~ 'User Output 2' out of 24 signal types in the control output port. The rest (15 output signals) of them cannot be operated by the user's disposal. When any relevant situation occurs while Ezi-MOTIONLINK-Plus-R operates, they are displayed. The following table shows bit mask values by each signal.

Signal Name	Relevant Bit Position	Signal Name	Relevant Bit Position	Signal Name	Relevant Bit Position
Compare Out	0x00000001	Origin Search OK	0x00000100	User OUT 1	0x00010000
Inposition	0x00000002	Reserved	0x00000200	User OUT 2	0x00020000
Alarm	0x00000004	Reserved	0x00000400	Reserved	0x00040000
Moving	0x00000008	Reserved	0x00000800	Reserved	0x00080000
Acc/Dec	0x00000010	Reserved	0x00001000	Reserved	0x00100000
Reserved	0x00000020	Reserved	0x00002000	Reserved	0x00200000
Reserved	0x00000040	Reserved	0x00004000	Reserved	0x00400000
Reserved	0x00000080	User OUT 0	0x00008000	Reserved	0x00800000

【Example 1】 Sending data to turn ON the User Output 2 port.

4 bytes (I/O set mask value)	4 bytes (I/O clear mask value)
0x00020000	0x00000000

【Example 2】 Sending data to turn OFF the User Output 2 port

4 bytes (I/O set mask value)	4 bytes (I/O clear mask value)
0x00000000	0x00020000

1－2－4. Bit setting of the control input pin

This displays the detailed description for 0x21 Frame type.

This command is applicable to 32 signals in the control input port. The user can use signals for test as if they are inputted without actual input signal. The following table shows bit mask values by each signal.

Signal Name	Relevant Bit Position	Signal Name	Relevant Bit Position	Signal Name	Relevant Bit Position	Signal Name	Relevant Bit Position
Limit+	0x00000001	PT A4	0x00000100	Alarm Reset	0x00010000	JPT input2	0x01000000
Limit-	0x00000002	PT A5	0x00000200	ServoON	0x00020000	JPT Start	0x02000000
Origin	0x00000004	PT A6	0x00000400	Pause	0x00040000	User IN 0	0x04000000
Clear Position	0x00000008	PT A7	0x00000800	Org Search	0x00080000	User IN 1	0x08000000
PT A0	0x00000010	PT Start	0x00001000	Teaching	0x00100000	User IN 2	0x10000000
PT A1	0x00000020	Stop	0x00002000	E-stop	0x00200000	User IN 3	0x20000000
PT A2	0x00000040	Jog+	0x00004000	JPT input0	0x00400000	User IN 4	0x40000000
PT A3	0x00000080	Jog-	0x00008000	JPT input1	0x00800000	Reserved	0x80000000

【Example 1】 Sending data to turn ON the Clear Position port

4 bytes (I/O set mask value)	4 bytes (I/O clear mask value)
0x00000004	0x00000000

【Example 2】 Sending data to turn OFF the Clear Position port

4 bytes (I/O set mask value)	4 bytes (I/O clear mask value)
0x00000000	0x00000004

1－2－5. Bit setting of Status Flag

Refer to 'motion_define.h' of include files.

Name of Flag Define	Contents	Relevant Bit Position
FFLAG_ERRORALL	One or more error occurs.	0X00000001
FFLAG_HWPOSILMT	'+' direction limit sensor turns ON.	0X00000002
FFLAG_HWNEGALMT	'-' direction limit sensor turns ON.	0X00000004
FFLAG_SWPOGILMT	'+' direction program limit is exceeded.	0X00000008
FFLAG_SWNEGALMT	'-' direction program limit is exceeded.	0X00000010
Reserved1		0X00000020
Reserved2		0X00000040
Reserved3		0X00000080
Reserved4		0X00000100
Reserved5		0X00000200
Reserved6		0X00000400
Reserved7		0X00000800
Reserved8		0X00001000
Reserved9		0X00002000
Reserved10		0X00004000
Reserved11		0X00008000
FFLAG_EMGSTOP	The motor is under emergency stop.	0X00010000
FFLAG_SLOWSTOP	The motor is under general stop.	0X00020000
FFLAG_ORIGINRETURNING	The motor is returning to the origin.	0X00040000
FFLAG_INPOSITION	Inposition has been finished.	0X00080000
FFLAG_SERVOON	The motor is under Servo ON.	0X00100000
FFLAG_ALARMRESET	AlarmReset has run.	0X00200000
Reserved12		0X00400000
FFLAG_ORIGINSENSOR	The origin sensor is ON.	0X00800000
FFLAG_ZPULSE	Operation of z-pulse method during zero return operation	0X01000000
FFLAG_ORIGINRETOK	Origin return operation has been finished.	0X02000000
FFLAG_MOTIONDIR	To display the motor operating direction (+: OFF, -: ON)	0X04000000
FFLAG_MOTIONING	The motor is running.	0X08000000
FFLAG_MOTIONPAUSE	The motor in running is stopped by Pause command.	0X10000000
FFLAG_MOTIONACCEL	The motor is operating to the acceleration section.	0X20000000
FFLAG_MOTIONDECEL	The motor is operating to the deceleration section.	0X40000000
FFLAG_MOTIONCONST	The motor is operating to the normal speed, not acceleration / deceleration sections.	0X80000000

1－2－6. Bit setting of Status Flag

Refer to 'motion_define.h' of include files.

Refer to 'motion_define.h' among the include file.

Name	Name of structure paramater	Number of Byte	Offset value	Unit	Low Limit	Upper Limit
Position*1	lPosition	4 (signed)	0	[pulse]	-2,147,483,648	2,147,483,647
Low Speed	dwStartSpd	4 (unsigned)	4	[pps]	0	2,500,000
High Speed	dwMoveSpd	4 (unsigned)	8	[pps]	0	2,500,000
Accel. Time	wAccelRate	2 (unsigned)	12	[msec]	1	9,999
Decel. Time	wDecelRate	2 (unsigned)	14	[msec]	1	9,999
Command	wCommand	2 (unsigned)	16		0	10
Wait time	wWaitTime	2 (unsigned)	18	[msec]	0	600,000
Continuous Action	wContinuous	2 (unsigned)	20		0	1
Jump Table No.	wBranch	2 (unsigned)	22		0 10,000	255 10,255
Jump PT 0	wCond_branch0	2 (unsigned)	24		0 10,000	255 10,255
Jump PT 1	wCond_branch1	2 (unsigned)	26		0 10,000	255 10,255
Jump PT 2	wCond_branch2	2 (unsigned)	28		0 10,000	255 10,255
Loop Count	wLoopCount	2 (unsigned)	30		0	100
Loop Jump Table No.	wBranchAfterLoop	2 (unsigned)	32		0 10,000	255 10,255
PT set	wPTSet	2 (unsigned)	34		0	15
Loop Counter Clear	wLoopCountCLR	2 (unsigned)	36		0	255
Check Inposition	bCheckInpos	2 (unsigned)	38		0	1
Compare Position	lTriggerPos	4 (signed)	40	[pulse]	-134,217,728	+134,217,727

Compare Width	wTriggerOnTime	2 (unsigned)	44	[msec]	1	9,999
Reserved		2 (unsigned)	46	[%]	20	90
Reserved		4 (unsigned)	48	[pps]	0	33,333
Reserved		4 (signed)	52	[pulse]	-134,217,728	+134,217,727
Reserved		2 (unsigned)	56		0	10,000
Blank		6 (unsigned)	58	0x00		

*1 The parameter range differs from the product version, listed as below.
V06.03.04x.xx : -134,217,728 ~134,217,727
V06.03.05x.xx : - 2,147,483,648 ~ 2,147,483,647

For the setting method by each item, refer to 「User Manual-Position Table」.

1－3. Program Method

There are 2 method of programming for Ezi-MOTIONLINK-Plus-R.

The first and commonly used method is using Visual C++ language under window system of PC.

Library that offered together with Ezi-MotionLink-PR has to be used. Refer to 「2. Library for PC Program(Ver6)」

The second method can be accomplished by sending command characters directly to Ezi-MOTIONLINK-Plus-R. The user has to prepare low level protocol programming like 'Protocol Test' program. This method is normally used for PLC system.

The specific program method can be tested using the user GUI program called 'ProtocolTest_PlusR.exe' provided with the product. Refer to 「 3. Protocol for PLC Program」.

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₩₩EziMOTION PlusR₩₩include₩₩ The following contents should be included in a source file for development.

#include "₩₩FASTECH₩₩ EziMOTION PlusR ₩₩include₩₩FAS_EziMotionPlusR.h"
#include "₩₩FASTECH₩₩ EziMOTION PlusR ₩₩include₩₩COMM_Define.h"
#include "₩₩FASTECH₩₩ EziMOTION PlusR ₩₩include₩₩MOTION_DEFINE.h"
#include "₩₩FASTECH₩₩ EziMOTION PlusR ₩₩include₩₩ReturnCodes_Define.h"

Also, library files are as follows:
"₩₩FASTECH₩₩ EziMOTION PlusR ₩₩include₩₩EziMotionPlusR.lib"
"₩₩FASTECH₩₩ EziMOTION PlusR₩₩include₩₩EziMotionPlusR.dll"

A sample program source of using library is included in a "₩₩FASTECH₩₩EziMOTION PlusR ₩₩Examples₩₩" folder.

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

Item	Definition	Returned Value	Description
Normal	FMM_OK	0	The function has normally performed the command.
Input Error	FMM_NOT_OPEN	1	Wrong port number is inputted.
	FMM_INVALID_PORT_NUM	2	The port that is not connected.
	FMM_INVALID_SLAVE_NUM	3	Wrong slave number is inputted.
Operation Error	FMM_POSTABLE_ERROR	9	An error occurs while the motor accesses to the position table.
Connection Error	FMC_DISCONNECTED	5	The relevant product is disconnected.
	FMC_TIMEOUT_ERROR	6	Response delay(100 msec) occurs.
	FMC_CRCFAILED_ERROR	7	Checksum error occurs.
	FMC_RECVPACKET_ERROR	8	Protocol level error occurs in packet that comes from Product.

(2) The following table shows return values included commonly in all libraries. The user can check the result (communication status, running status) judged by the product. When the user develops programs by using protocols or libraries(DLL), they are available as well.

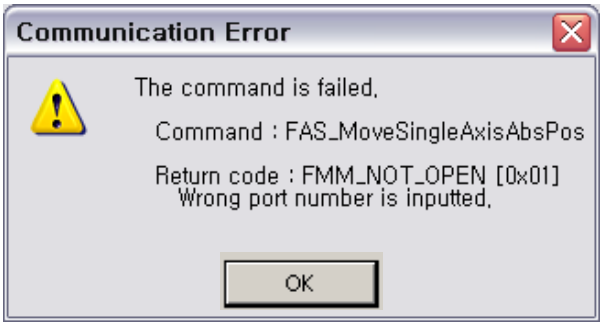
Item	Description	Returned Value	Description
Normal	FMP_OK	0	Communication has been normally performed.
Input Error	FMP_FRAMETYPEERROR	128	The product cannot recognize the command.
	FMP_DATAERROR	129	Input data is out of the range.

Operation Error	FMP_ RUNFAIL	133	The motor is already running or not prepared for running. Other wrong motion command.
	FMP_RESETFAIL	134	The user cannot execute AlarmReset command while the servo is ON.
	FMP_SERVOONFAIL1	135	An alarm has occurred.
	FMP_SERVOONFAIL2	136	The motor is under Emergency Stop.
	FMP_SERVOONFAIL3	137	'ServoON'signal is already assigned to input pin.
Connection Error	FMP_PACKETERROR	130	Protocol level error occurs in packet that Product's received.
	FMP_PACKETCRCERROR	170	CRC value is not correct in packet that Product's received.

2－2. Communication Status Window

Above communication status is divide by 3 groups.

(1) Communication Error



FMM_NOT_OPEN,

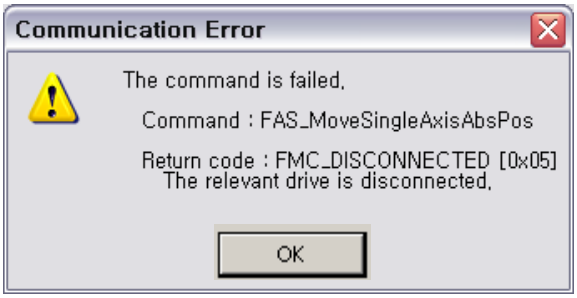
COM Port is not connected. (Does not occur on GUI)

FMM_INVALID_PORT_NUM,

COM Port number of inputted number is not a connected Port. (Does not occur on GUI)

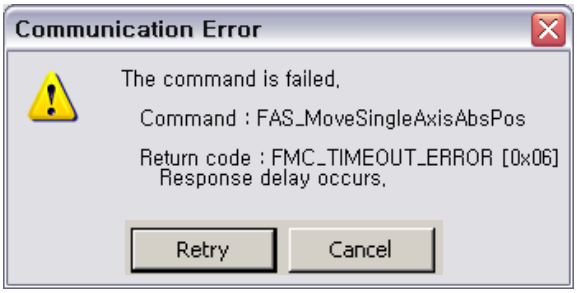
FMM_INVALID_SLAVE_NUM,

Slave number does not exist. (Does not occur on GUI)



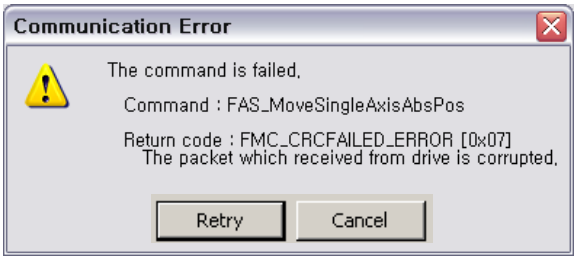
FMC_DISCONNECTED = 5,

COM Port is a disconnect.



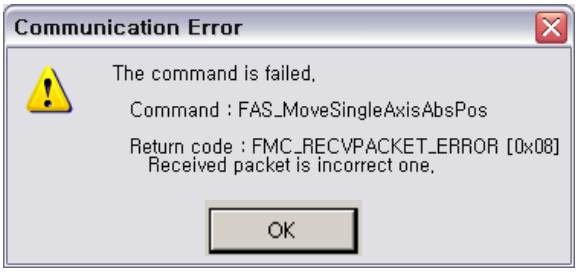
FMC_TIMEOUT_ERROR,

There is no response from the product.



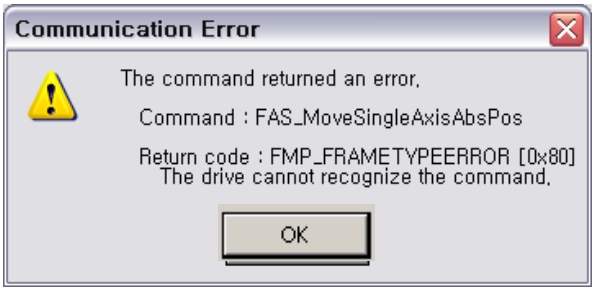
FMC_CRCFAILED_ERROR,

CRC value of communication packet from the product is not correct.



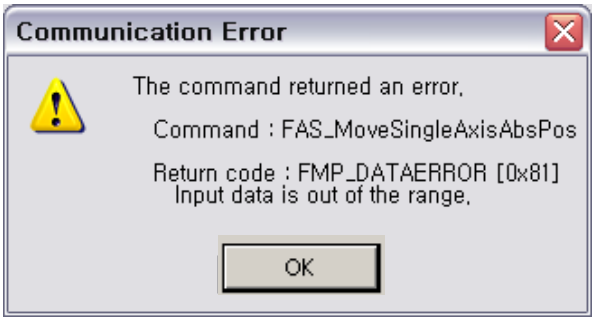
FMC_RECVPACKET_ERROR,

The length of received packet is not correct.



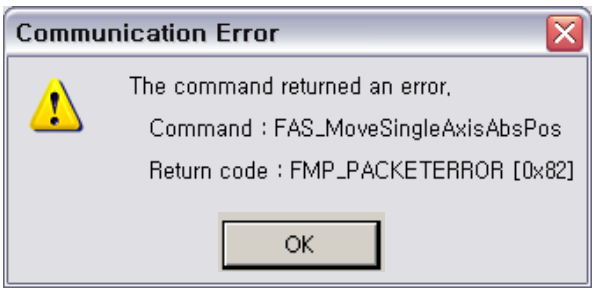
FMP_FRAMETYPEERROR = 0x80,

Product do not recognize the command or wrong command is sended.



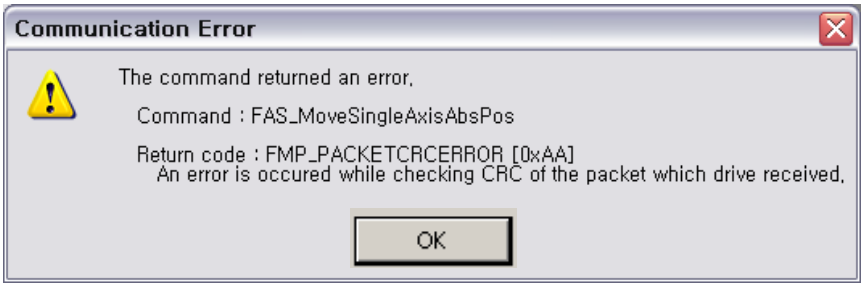
FMP_DATAERROR,

The value of the sent data is out of the proper range for product.



FMP_PACKETERROR,

Received Frame is the data which not suit as standard.(The length of received packet on product is not correct.)



FMP_PACKETCRCERROR = 0xAA,

CRC calculation of Packet sent to Drive does not correct.

(2) Wrong Command

FMP_RUNFAIL = 0x85,

Fail on motion command : The motor can not run on next status.

-. The motor is already running

-. The motor is under stop command

-. Servo OFF status

-. Try to Z-pulse Origin without external encoder (only for Ezi-STEP)

-. other wrong motion command

FMP_RESETFAIL,

Fail on reset command : The motor can not reset on next status.

-. Servo ON status

-. Already 'Reset'status by external input signal.

FMP_SERVOONFAIL1,

Wrong 'Servo ON' command during Alarm happens.

FMP_SERVOONFAIL2,

Wrong 'Servo ON' command during E-Stop happens.

FMP_SERVOONFAIL3,

'Servo ON' Signal is assigned by external input pin. In this case Servo ON command by DLL library is not working.

2－3. Communication Link Function

Function Name	Description
FAS_Connect	The product tries to connect communication with Ezi-MOTIONLINK-Plus-R**:** When it is successfully connected, TRUE will return. Otherwise, FALSE will return.
FAS_Close	The product tries to disconnect communication with Ezi-MOTIONLINK-Plus-R.
FAS_GetSlaveInfo	Ezi-MOTIONLINK-Plus-R reads product type and program version: Product type and version information will return.
FAS_IsSlaveExist	Ezi-MOTIONLINK-Plus-R checks(with FAS_Connect) whether there is the relevant Ezi-MOTIONLINK-Plus-R. When it exists, TRUE will return. Otherwise, FALSE will return.
FAS_EnableLog	To control the communication error log function : When it exists, TRUE will return. Otherwise, FALSE will return.
FAS_SetLogPath	To set the saved folder name of error log file : When folder exists, TRUE will return. Otherwise, FALSE will return.

FAS_Connect

FAS_Connect is the function of connecting Ezi-MotionLink-PR.

Syntax

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

Parameters

nPortNo
Select a serial port to be connected.
dwBaud
Input the Baudrate of the serial port.

Return Value

When it is successfully connected, TRUE will returns. Otherwise, FALSE will return.

Remarks

Example

```
#include "FAS_EziMOTIONPlusR.h"

void funcInit()
{
        BYTE nPortNo = 1;  // COMM Port Number
        DWORD dwBaudrate = 115200;          // Baudrate. (Be variable by setting)
        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.
                // The port is not connected or the baudrate may be wrong.
                return;
        }

        if (FAS_IsSlaveExist(nPortNo, iSlaveNo) == FALSE)
        {
                // There is no relevant slave number.
                // Check the slave number of Ezi-SERVO Plus-R.
                return;
        }

        nRtn = FAS_GetSlaveInfo(nPortNo, iSlaveNo, &nType, lpBuff, nBuffSize);
```

```
        if (nRtn != FMM_OK)
        {
                // Command has not been performed properly.
                // 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

To disconnect the serial port being used

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

To get the version information string of the relevant product

Syntax

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

Parameters

nPortNo

Port number of relevant product

iSlaveNo

Slave number of relevant product

pType

Relevant product type number

lpBuff

Buffer pointer to get version information string

nBuffSize

lpBuff memory allocation size

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no product of iSlaveNo in the relevant port.

Remarks

Example

Refer to 'FAS_Connect' library.

See Also

FAS_IsSlaveExist

To check that the product is connected

Syntax

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

Parameters

nPortNo

Port number of relevant product

iSlaveNo

Slave number of relevant product

Return Value

TRUE : The product is connected.

FALSE : The product is disconnected.

Remarks

This function is provided from the library only and it is inapplicable to the protocol program mode.

Example

Refer to 'FAS_Connect' library.

See Also

FAS_Connect

FAS_EnableLog

To control the save function of communication error log file.

Syntax

void FAS_EnableLog(BOOL bEnable);

Parameters

bEnable

Select output of Log.

Remarks

Select the Log output during Ezi-MOITON Plus-R function used. This setup
Do not affect the other process or other program.
Log function start from 'FAS_Connect' function, the Log output is end when the 'FAS_Close' is excuted. Basig setting value of Log output is TRUE.

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcDisableLog()
{
        BYTE nPortNo = 1;       // COMM Port No.

        FAS_EnableLog(FALSE);

        // After this, the Logs of the functions are not output.

        // Try to connect.
        if (FAS_Connect(nPortNo, 115200) == FALSE)
        {
                // connection fail.
                // cab be different Port or different Baudrate.
                return;
        }

        // Connection close.
```

```
        FAS_Close(nPortNo);
}
```

See Also

FAS_SetLogPath

FAS_SetLogPath

Setup the folder path of Log output files.

Syntax

BOOL FAS_SetLogPath(LPCTSTR lpPath);

Parameters

lpPath

Absolute path string where log will be saved.

Return Value

If the folder name does not exist or can not access, return FALSE.

Remarks

This function has to be called before using FAS_Connect library.
If the lpPath value is NULL or the length is 0, the Log path is selected to Ezi-MOTION Plus-R Library folder. The default value for Log path is NULL that the current library and program exist folder.

Example

```
#include "FAS_ EziMOTIONPlusR.h"
```

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

	// Log output.
	FAS_EnableLog(TRUE);

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

	// All Log output is stored in C:₩Logs folder.

	// Try to connect.
	if (FAS_Connect(nPortNo, 115200) == FALSE)
	{
		// Connection fail.
		// cab be different Port or different Baudrate.
		return;
	}

	// Close connect.
	FAS_Close(nPortNo);
}
```

See Also

FAS_EnableLog

2－4. Parameter Control Function

Function Name	Description
FAS_SaveAllParameters	Current parameters are saved to the ROM: Even after the product is powered OFF, parameters related to operating speed, acceleration/deceleration time, and origin return need to be preserved.
FAS_SetParameter	The designated parameter is saved to the RAM: Specific parameter is saved.
FAS_GetParameter	The designated parameter is read from the RAM: Specific parameter is read.
FAS_GetROMParameter	The designated parameter is read from the ROM: Specific parameter is read from the ROM.

FAS_SaveAllParameters

All parameters edited up to now & assign status of In/Out signals are saved in the ROM area.

Syntax

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

Parameters

nPortNo

Port number of relevant product

iSlaveNo

Slave number of relevant product

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no product of iSlaveNo in the relevant port.

Remarks

Parameter values set to 'FAS_SetIOAssignMap' library as well as current parameter values are saved to the ROM.

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcModifyParameter()
{
        BYTE nPortNo = 1;   // COMM Port Number
        BYTE iSlaveNo = 0;  // Slave No (0 ~ 15)
        long lParamVal;
        int nRtn;

        // Try to connect
        if (FAS_Connect(nPortNo, 115200) == FALSE)
        {
                // Connection failed.
                // The port is not connected or the baudrate may be wrong.
                return;
        }
```

```
	// Check Axis Start Speed Parameter.
	nRtn = FAS_GetParameter(nPortNo, iSlaveNo, SERVO_AXISSTARTSPEED, &lParamVal);
	if (nRtn != FMM_OK)
	{
		// Command has not been performed properly.
		// Refer to ReturnCodes_Define.h.
		_ASSERT(FALSE);
	}
	else
	{
		// Parameter value saved in current product.
		printf("Parameter [before] : Start Speed = %d ₩n", lParamVal);
	}


	// Change Start Speed parameter as 200 then read it again.
	nRtn = FAS_SetParameter(nPortNo, iSlaveNo, SERVO_AXISSTARTSPEED, 200);
	_ASSERT(nRtn == FMM_OK);   // It will stop if the command didn't execute correctly.

	nRtn = FAS_GetParameter(nPortNo, iSlaveNo, SERVO_AXISSTARTSPEED, &lParamVal);
	_ASSERT(nRtn == FMM_OK);
	printf("Parameter [after] : Start Speed = %d ₩n", lParamVal);


	// Check the value saved in the ROM.
	nRtn = FAS_GetROMParameter(nPortNo, iSlaveNo, SERVO_AXISSTARTSPEED, &lParamVal);
	_ASSERT(nRtn == FMM_OK);  // It will stop if the command didn't execute correctly.
	printf("Parameter [ROM] : Start Speed = %d ₩n", lParamVal);


	// Edit the parameter value then save it in the ROM.
	nRtn = FAS_SetParameter(nPortNo, iSlaveNo, SERVO_AXISSTARTSPEED, 100);
	_ASSERT(nRtn == FMM_OK);   // It will stop if the command didn't execute correctly.
	nRtn = FAS_SaveAllParameters(nPortNo, iSlaveNo);
	_ASSERT(nRtn == FMM_OK);

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

See Also

FAS_GetRomParameter

FAS_SetParameter

Edit the relevant parameter value and then save it to the RAM.

Syntax

```
int FAS_SetParameter(
    BYTE nPortNo,
    BYTE iSlaveNo,
    BYTE iParamNo,
    long lParamValue
);
```

Parameters

nPortNo

Port number of relevant product

iSlaveNo

Slave number of relevant product

iParamNo

Parameter number to be edited

lParamValue

Parameter value to be edited

Return Value

FMM_OK : Command has been normally performed.
FMM_NOT_OPEN : The product has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no product of iSlaveNo in the relevant port.
FMM_INVALID_PARAMETER_NUM : There is no parameter of designated iParamNo.

Remarks

The function operates only for one parameter designated.
Parameters in the product are saved to 2 memory areas. That is, when power is off, the ROM saves parameters permanently. When power is on, parameters in the ROM are copied to the DSP RAM and used. When the user changes parameters, it changes not parameters in the ROM but parameter in the RAM. This function is to set the parameter number designated from the RAM to the relevant value.

Example

Refer to 'FAS_SaveAllParameter' library.

See Also

FAS_GetParameter

FAS_GetParamater

To call specific parameter values of the product

Syntax

```
int FAS_GetParameter(
    BYTE nPortNo,
    BYTE iSlaveNo,
    BYTE iParamNo,
    long* lParamValue
);
```

Parameters

nPortNo

Port number of relevant product

iSlaveNo

Slave number of relevant product

iParamNo

Parameter number to be imported

lParamValue

Parameter values

Return Value

FMM_OK : Command has been normally performed.
FMM_NOT_OPEN : The product has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no product of iSlaveNo in the relevant port.
FMM_INVALID_PARAMETER_NUM : There is no parameter of designated iParamNo.

Remarks

The function operates only for one parameter designated.
Parameters in the product are saved to 2 memory areas. That is, when power is off, the ROM saves parameters permanently. When power is on, parameters in the ROM are copied to the DSP RAM and used. When the user changes parameters, it changes not parameters in the ROM but parameter in the RAM. This function reads the parameter number designated to the RAM.

Example

Refer to 'FAS_SaveAllParameter' library.

See Also

FAS_SetParameter

FAS_GetROMParameter

To call parameters saved in the ROM

Syntax

```
int FAS_GetROMParameter(
    BYTE nPortNo,
    BYTE iSlaveNo,
    BYTE iParamNo,
    long* lRomParam
);
```

Parameters

nPortNo

Port number of relevant product

iSlaveNo

Slave number of relevant product

iParamNo

Parameter number to be imported

lRomParam

Parameter values saved in the ROM

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no product of iSlaveNo in the relevant port.

FMM_INVALID_PARAMETER_NUM : There is no parameter of designated iParamNo.

Remarks

To call parameter values saved in the ROM

Even though this function runs, the value in the RAM is not changed. For this, run FAS_SetParameter.

Example

Refer to 'FAS_SaveAllParameter' library.

See Also

FAS_SaveAllParameters

2－5. Servo Control Function

Function Name	Description
FAS_ServoEnable	The Servo status of the product designated turns ON/OFF.
FAS_ServoAlarmReset	The product which an alarm occurs is released: Troubleshoot the alarm cause and use this function.

FAS_ServoEnable

To turn ON/OFF the product servo

Syntax

```
int FAS_ServoEnable(
    BYTE nPortNo,
    BYTE iSlaveNo,
    BOOL bOnOff
);
```

Parameters

nPortNo
Port number of relevant product
iSlaveNo
Slave number of relevant
bOnOff
Enable or Disable.

Return Value

FMM_OK : Command has been normally performed.
FMM_NOT_OPEN : The product has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no product of iSlaveNo in the relevant port.

Remarks

The given time is required until Servo ON flag in the axis status turns on after enable.

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcAxisStatus()
{
        BYTE nPortNo = 1;   // COMM Port Number
        BYTE iSlaveNo = 0;  // Slave No (0 ~ 15)
        EZISERVO_AXISSTATUS AxisStatus;
        int nRtn;

        // Try to connect
        if (FAS_Connect(nPortNo, 115200) == FALSE)
        {
                // Connection failed.
                // The port is not connected or the baudrate may be wrong.
                return;
        }

        nRtn = FAS_GetAxisStatus(nPortNo, iSlaveNo, &(AxisStatus.dwValue));
        _ASSERT(nRtn == FMM_OK);

        // If SERVO_ON flag turns off, the servo turns on.
        if (AxisStatus.FFLAG_SERVOON == 0)
        {
```

```
                nRtn = FAS_ServoEnable(nPortNo, iSlaveNo, TRUE);
                _ASSERT(nRtn == FMM_OK);
        }

        // If there is an alarm, AlarmReset runs.
        if (AxisStatus.FFLAG_ERRORALL || AxisStatus.FFLAG_ERROVERCURRENT ||
AxisStatus.FFLAG_ERROVERLOAD)
        {
                nRtn = FAS_ServoAlarmReset(nPortNo, iSlaveNo);
                _ASSERT(nRtn == FMM_OK);
        }

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

See Also

FAS_ServoAlarmReset

FAS_ServoAlarmReset

To send AlarmReset command

Syntax

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

Parameters

nPortNo

Port number of relevant product

iSlaveNo

Slave number of relevant product

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no product of iSlaveNo in the relevant port.

Remarks

Before sending this command, troubleshoot the alarm cause.

For alarm cause, refer to 'User Manual_Text'.

Example

Refer to 'FAS_ServoEnable' library

See Also

FAS_ServoEnable

2－6. Control I/O Function

Function Name	Description
FAS_SetIOInput	To set the input signal level of the control input port : Input signal is set to [ON] or [OFF].
FAS_GetIOInput	To read the current input signal status of the control input port : The signal status returns by bit for each input signal.
FAS_SetIOOutput	To set the output signal level of the control output port : Output signal is set to [ON] or [OFF].
FAS_GetIOOutput	To read the current output signal status of the control output port : The signal status returns by bit for each output signal.
FAS_GetIOAssignMap	To read the pin setting status of the CN1 port : The setting status for each 9 variable signals returns by bit to the Input and Output port.
FAS_SetIOAssignMap	To assign the control I/O signal to CN1 port pin and also set the signal level : Setting for each 9 variable signals is assigned to the Input and Output port.
FAS_IOAssignMapReadROM	To load the pin setting status of CN1 port from ROM area to RAM area.

FAS_SetIOInput

To set I/O input. For more information, refer to '1-2. Structure of Frame'.

Syntax

```
int FAS_SetIOInput(
    BYTE nPortNo,
    BYTE iSlaveNo,
    DWORD dwIOSetMask,
    DWORD dwIOCLRMask
);
```

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product.

dwIOSetMask

Input bitmask value to be set(ON)

dwIOCLRMask

Input bitmask value to be cleared(OFF)

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no product of iSlaveNo in the relevant port.

Remarks

Be careful that dwIOSetMask bit and dwIOCLRMask bit are not duplicated.

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcIO()
{
        BYTE nPortNo = 1;   // COMM Port Number
        BYTE iSlaveNo = 0;  // Slave No (0 ~ 15)
        DWORD dwInput, dwOutput;
        int nRtn;

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

```
	{
		// Connection failed.
		// The port is not connected or the baudrate may be wrong.
		return;
	}

	// Check I/O input.
	nRtn = FAS_GetIOInput(nPortNo, iSlaveNo, &dwInput);
	_ASSERT(nRtn == FMM_OK);
	if (dwInput & SERVO_IN_BITMASK_LIMITP)
	{
		// Limit + input is ON.
	}

	if (dwInput & SERVO_IN_BITMASK_USERIN0)
	{
		// User Input 0 is ON.
	}

	// Turning ON 'Clear Position' and 'User Input 1' inputs and turning off 'Jog +' input.
	nRtn = FAS_SetIOInput(nPortNo, iSlaveNo, SERVO_IN_BITMASK_CLEARPOSITION |
SERVO_IN_BITMASK_USERIN1, SERVO_IN_BITMASK_PJOG);
	_ASSERT(nRtn == FMM_OK);

	// Check I/O output.
	nRtn = FAS_GetIOOutput(nPortNo, iSlaveNo, &dwOutput);
	_ASSERT(nRtn == FMM_OK);
	if (dwOutput & SERVO_OUT_BITMASK_USEROUT0)
	{
		// User Output 0 is ON.
	}

	// Turn off User Output 1 and 2 signals.
	nRtn = FAS_SetIOOutput(nPortNo, iSlaveNo, 0, SERVO_OUT_BITMASK_USEROUT1 |
SERVO_OUT_BITMASK_USEROUT2);
	_ASSERT(nRtn == FMM_OK);

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

See Also

FAS_GetIOInput

FAS_GetIOInput

To read I/O input values. For more information, refer to '1-2. Structure of Frame'.

Syntax

```
int FAS_GetIOInput(
    BYTE nPortNo,
    BYTE iSlaveNo,
    DWORD* dwIOInput
);
```

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product.

dwIOInput

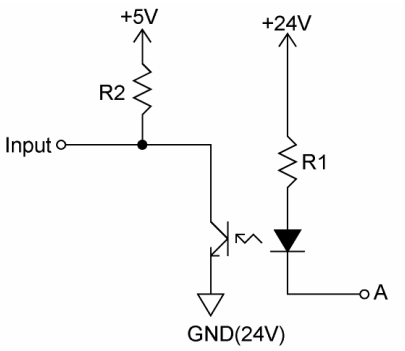
Parameter pointer which input values will be saved

Return Value

FMM_OK : Command has been normally performed.
FMM_NOT_OPEN : The product has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

Remarks

There are 8 input pins in Ezi-MOTIONLINK-Plus-R. The user can select and use 5 input pins of them. This function can read the input port status by 32bit. All of them are insulated by a photocoupler. (Refer to the figure.)



When Port A is supplied 24V from an external input port, the input is recognized to 5V(High).

Example

Refer to 'FAS_SetIOInput' library.

See Also

FAS_SetIOInput

FAS_SetIOOutput

To set I/O output values. For more information, refer to '1-2. Structure of Frame'.

Syntax

```
int FAS_SetIOOutput(
    BYTE nPortNo,
    BYTE iSlaveNo,
    DWORD dwIOSetMask,
    DWORD dwIOCLRMask
);
```

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product.

dwIOSetMask

Output bitmask value to be set(ON status)

dwIOCLRMask

Output bitmask value be cleared(OFF status)

Return Value

FMM_OK : Command has been normally performed.

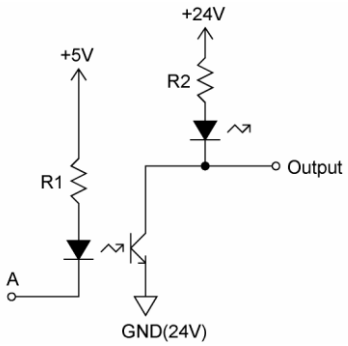
FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

Remarks

There are 4 output pins in Ezi-MOTIONLINK-Plus-R. The user can select and use 3 output pins of them.



When output data is '1', Port A becomes 0V. When it is '0', Port A becomes +5V.

Be careful that dwIOSetMask bit and dwIOCLRMask bit are not duplicated.

Example

Refer to FAS_SetIOInput.

See Also

FAS_GetIOOutput

FAS_GetIOOutput

To read I/O output values. For more information, refer to '1-2. Structure of Frame'.

Syntax

```
int FAS_GetIOOutput(
    BYTE nPortNo,
    BYTE iSlaveNo,
    DWORD* dwIOOutput
);
```

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product.

dwIOInput

Parameter pointer which the output value will be saved.

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

Remarks

Example

Refer to 'FAS_SetIOInput' library

See Also

FAS_SetIOOutput

FAS_GetIOAssignMap

To read I/O Assign Map. For more information, refer to '1-2. Structure of Frame'.

Syntax

```
int FAS_GetIOAssignMap(
    BYTE nPortNo,
    BYTE iSlaveNo,
    BYTE iIOPinNo,
    BYTE* nIOLogic,
    BYTE* bLevel
);
```

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product.

iIOPinNo

I/O pin number to be read

nIOLogic

Parameter pointer which the logic value assigned to a relevant pin will be saved

bLevel

Parameter pointer which the active level of relevant logic will be saved

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

Remarks

For nIOLogic, refer to 'Motion_define.h'.

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcIOAssign()
{
        BYTE nPortNo = 1;  // COMM Port Number
        BYTE iSlaveNo = 0;  // Slave No (0 ~ 15)
        BYTE iPinNo;
        DWORD dwLogicMask;
        BYTE bLevel;
```

```
	BYTE i;
	int nRtn;

	// Try to connect
	if (FAS_Connect(nPortNo, 115200) == FALSE)
	{
		// Connection failed.
		// The port is not connected or the baudrate may be wrong.
		return;
	}

	// Check assigned information of input pin.
	for (i=0; i</*Input Pin Count*/12; i++)
	{
		nRtn = FAS_GetIOAssignMap(nPortNo, iSlaveNo, i, &dwLogicMask, &bLevel);
		_ASSERT(nRtn == FMM_OK);

		if (dwLogicMask != IN_LOGIC_NONE)
			printf("Input Pin %d : Logic Mask 0x%08X (%s)₩n", i, dwLogicMask,
((bLevel == LEVEL_LOW_ACTIVE) ? "Low Active" : "High Active"));
		else
			printf("Input Pin %d : Not assigned₩n", i);
	}

	// Assign SERVOON Logic (Low Active) to input pin 3.
	iPinNo = 3;		// 0 ~ 11 value is available (Caution : 0 ~ 2 is fixed.)
	nRtn = FAS_SetIOAssignMap(nPortNo, iSlaveNo, iPinNo, SERVO_IN_BITMASK_SERVOON,
LEVEL_LOW_ACTIVE);
	_ASSERT(nRtn == FMM_OK);

	// Check assign information of output pin.
	for (i=0; i<10/*Output Pin Count*/; i++)
	{
		nRtn = FAS_GetIOAssignMap(nPortNo, iSlaveNo, 12/*Input Pin Count*/ + i,
&dwLogicMask, &bLevel);
		_ASSERT(nRtn == FMM_OK);

		if (dwLogicMask != OUT_LOGIC_NONE)
			printf("Output Pin %d : Logic Mask 0x%08X (%s)₩n", i, dwLogicMask,
((bLevel == LEVEL_LOW_ACTIVE) ? "Low Active" : "High Active"));
		else
			printf("Output Pin %d : Not assigned₩n", i);
	}

	// Assign ALARM Logic (High Active) to output pin 9.
	iPinNo = 9;		// 0 ~ 9 value is available (Caution : 0 is fixed to COMPOUT.)
```

```
        nRtn = FAS_SetIOAssignMap(nPortNo, iSlaveNo, 12/*Input Pin Count*/ + iPinNo,
SERVO_OUT_BITMASK_ALARM, LEVEL_HIGH_ACTIVE);
        _ASSERT(nRtn == FMM_OK);

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

See Also

FAS_SetIOAssignMap

FAS_SetIOAssignMap

To set I/O Assign Map. For more information, refer to '1-2. Structure of Frame'.

Syntax

```
int FAS_SetIOAssignMap(
    BYTE nPortNo,
    BYTE iSlaveNo,
    BYTE iIOPinNo,
    BYTE nLogicNo,
    BYTE bLevel
);
```

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product.

iIOPinNo

I/O Pin number to be read

nIOLogic

Logic value to be assigned to the relevant pin

bLevel

Active Level value of the relevant logic

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

FMM_INVALID_PARAMETER_NUM : Designated iIOPinNo or nIOLogic value is out of range.

Remarks

To save current setting values to the ROM memory, 'FAS_SaveAllParameters' library should be run.

Example

Refer to 'FAS_GSetIOAssignMap' library

See Also

FAS_GetIOAssignMap

FAS_IOAssignMapReadROM

To load the I/O setting status and signal level value being saved in ROM area

Syntax

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

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product.

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

FMC_POSTABLE_ERROR : While reading position table, error occurred

Remarks

Example

See Also

FAS_ GetIOAssignMap

2－7. Position Control Function

Function Name	Description
FAS_SetCommandPos	To set the command position value
FAS_SetActualPos	To set the current position to the actual position value
FAS_GetCommandPos	To read the current command position value
FAS_GetActualPos	To read the current actual position value
FAS_GetPosError	To read the difference between the actual position value and the command position value
FAS_GetActualVel	To read the actual running speed value while the motor is moving
FAS_ClearPosition	To set the command position and actual position value to '0'

FAS_SetCommandPos

To set the command position value to the motor

Syntax

```
int FAS_SetCommandPos(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long lCmdPos
);
```

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product.

lCmdPos

Commnad position value to be set.

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

Remarks

The user sets the position command (pulse output counter) value.

This function is generally used when the user sets the current position to desired coordinates.

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcClearPosition()
{
        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.
                // The port is not connected or the baudrate may be wrong.
                return;
        }
```

```
        // Initialize Command Position and Actual Position values to 0.
        nRtn = FAS_SetCommandPos(nPortNo, iSlaveNo, 0);
        _ASSERT(nRtn == FMM_OK);
        nRtn = FAS_SetActualPos(nPortNo, iSlaveNo, 0);
        _ASSERT(nRtn == FMM_OK);

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

See Also

FAS_SetActualPos

FAS_SetActualPos

To set the actual position value to the motor

Syntax

```
int FAS_SetActualPos(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long lActPos
);
```

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product.

lActPos

Actual position value to be set.

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

Remarks

The user sets the encoder feedback counter value to the value that he wants.

Example

Refer to 'FAS_GetActualPos' library.

See Also

FAS_SetCommandPos

FAS_GetCommandPos

To read the command position of the current motor

Syntax

```
int FAS_GetCommandPos(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long* lCmdPos
);
```

Parameters

nPortNo

Port number of relevant product

iSlaveNo

Slave number of relevant product

lCmdPos

Parameter pointer that command position value will be saved

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

Remarks

To read the position command (pulse output counter) value.

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcDisplayStatus()
{
        BYTE nPortNo = 1;   // COMM Port Number
        BYTE iSlaveNo = 0;  // Slave No (0 ~ 15)
        long lValue;
        int nRtn;

        // Try to connect
        if (FAS_Connect(nPortNo, 115200) == FALSE)
        {
                // Connection failed.
                // The port is not connected or the baudrate may be wrong.
                return;
```

```
        }

        // Check position information of product.
        nRtn = FAS_GetCommandPos(nPortNo, iSlaveNo, &lValue);
        _ASSERT(nRtn == FMM_OK);
        printf("CMDPOS : %d ₩n", lValue);
        nRtn = FAS_GetActualPos(nPortNo, iSlaveNo, &lValue);
        _ASSERT(nRtn == FMM_OK);
        printf("ACTPOS : %d ₩n", lValue);
        nRtn = FAS_GetPosError(nPortNo, iSlaveNo, &lValue);
        _ASSERT(nRtn == FMM_OK);
        printf("POSERR : %d ₩n", lValue);
        nRtn = FAS_GetActualVel(nPortNo, iSlaveNo, &lValue);
        _ASSERT(nRtn == FMM_OK);
        printf("ACTVEL : %d ₩n", lValue);

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

See Also

FAS_GetActualPos

FAS_GetActualPos

To read the actual position value of the motor

Syntax

```
int FAS_GetActualPos(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long* lActPos
);
```

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product.

lActPos

Parameter pointer which the actual position value will be saved.

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

Remarks

It is mainly used to confirm the actual position after the completion of positioning.

Example

Refer to 'FAS_GetCOmmandPosition' library.

See Also

FAS_GetCommandPos

FAS_GetPosError

To read the position error of the motor

Syntax

```
int FAS_GetPosError(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long* lPosErr
);
```

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product.

lPosErr

Parameter pointer which the position error value will be saved

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

Remarks

Example

Refer to 'FAS_GetCOmmandPosition' library.

See Also

FAS_GetCommandPos,

FAS_GetActualPos

FAS_GetActualVel

To read the actual velocity of the motor

Syntax

```
int FAS_GetActualVel(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long* lActVel
);
```

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product.

lActVel

Parameter pointer which the actual velocity value will be saved

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

Remarks

Example

Refer to 'FAS_GetCOmmandPosition' library.

See Also

FAS_ClearPosition

To set the command position value and actual value to '0'

Syntax

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

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product.

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

Remarks

The user sets the position command (pulse output counter) value.

This function is generally used when the user sets the current position to initial values.

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcClearPosition()
{
        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.
		// The port is not connected or the baudrate may be wrong.
		return;
	}

	// Initialize Command Position and Actual Position values to 0.
	nRtn = FAS_ClearPosition(nPortNo, iSlaveNo);
	_ASSERT(nRtn == FMM_OK);

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

See Also

FAS_SetActualPos, FAS_SetCommandPos

2－8. Status Control Function

Function Name	Description
FAS_GetIOAxisStatus	To read control I/O status, running status Flag value : The current input status value, the output setting status value, and the running status Flag value will return.
FAS_GetMotionStatus	To read the current running progress status : The command position value, the actual position value, the speed value will return.
FAS_GetAllStatus	To read all status including the current status at one time : This function is to combine 'FAS_GetIOAxisStatus' function and 'FAS_GetMotionStatus' function.
FAS_GetAxisStatus	To read the running status Flag value of the relevant Ezi-MOTIONLINK-Plus-R

FAS_GetIOAxisStatus

To read I/O Input and Output values of the relevant product, and the motor Axis Status

Syntax

```
int FAS_GetIOAxisStatus(
    BYTE nPortNo,
    BYTE iSlaveNo,
    DWORD* dwInStatus,
    DWORD* dwOutStatus,
    DWORD* dwAxisStatus
);
```

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product.

dwInStatus

Parameter pointer which the I/O input value will be saved.

dwOutStatus

Parameter pointer which the I/O output value will be saved.

dwAxisStatus

Parameter pointer which the axis status value of the relevant motor will be saved

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_GetMotionStatus

To read the motion status of current motor at one time

Syntax

```
int FAS_GetMotionStatus(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long* lCmdPos,
    long* lActPos,
    long* lPosErr,
    long* lActVel,
    WORD* wPosItemNo
);
```

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product.

lCmdPos

Parameter pointer which the command position value will be saved

lActPos

Parameter pointer which the actual position value will be saved.

lPosErr

Parameter pointer which the position error value will be saved

lActVel

Parameter pointer which the actual velocity value will be saved

wPosItemNo

Parameter pointer which current running item number in the Position Table will be saved

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_GetAllStatus

To read I/O Input and Output values of the relevant product, the motor Axis Status, the motor motion status

Syntax

```
int FAS_GetAllStatus(
    BYTE nPortNo,
    BYTE iSlaveNo,
    DWORD* dwInStatus,
    DWORD* dwOutStatus,
    DWORD* dwAxisStatus,
    long* lCmdPos,
    long* lActPos,
    long* lPosErr,
    long* lActVel,
    WORD* wPosItemNo
);
```

Parameters

nPortNo
Port number of relevant product.
iSlaveNo
Slave number of relevant product.
dwInStatus
Parameter pointer which the I/O input value will be saved.
dwOutStatus
Parameter pointer which the I/O output value will be saved.
dwAxisStatus
Parameter pointer which the axix status value of the relevant motor will be saved
lCmdPos
Parameter pointer which the command position value will be saved
lActPos
Parameter pointer which the actual position value will be saved
lPosErr
Parameter pointer which the position error value will be saved
lActVel
Parameter pointer which the actual velocity value will be saved
wPosItemNo
Parameter pointer which current running item number in the Position Table will be saved

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_GetAxisStatus
FAS_GetMotionStatus

FAS_GetAxisStatus

To read the motor Axis Status value. For status Flag, refer to '1-2. Structure of Frame '.

Syntax

```
int FAS_GetAxisStatus(
    BYTE nPortNo,
    BYTE iSlaveNo,
    DWORD* dwAxisStatus
);
```

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product.

dwAxisStatus

Parameter pointer which the axis status value of the relevant motor

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

2－9. Running Control Function

Function Name	Description
FAS_MoveStop	The motor in running is decelerate and stopped.
FAS_EmergencyStop	The motor in running stops directly without deceleration
FAS_MoveOriginSingleAxis	The motor starts the origin return.
FAS_MoveSingleAxisAbsPos	The motor moves as much as the given absolute position value.
FAS_MoveSingleAxisIncPos	The motor moves as much as the given incremental position value.
FAS_MoveToLimit	The motor moves up to the position that the limit sensor is detected.
FAS_MoveVelocity	The motor moves to the given velocity and direction: This function is available to Jog motion.
FAS_PositionAbsOverride	While the motor is running, the target absolute position value [pulse] is changed.
FAS_PositionIncOverride	While the motor is running, the target incremental position value [pulse] is changed.
FAS_VelocityOverride	While the motor is running, the running velocity value [pulse] is changed.
FAS_AllMoveStop	All running motors that connected in same port are decelerate and stopped.
FAS_AllEmergencyStop	All running motors that connected in same port are directly stop without deceleration.
FAS_AllMoveOriginSingleAxis	All motors that connected in same port are starts the origin return.
FAS_AllMoveSingleAxisAbsPos	All motors that connected in same port moves as much as the given absolute position value.
FAS_AllMoveSingleAxisIncPos	All motors that connected in same port moves as much as the given incremental position value.
FAS_MoveLinearAbsPos	Between 2 motors that connected in the same port, start to Linear Interpolation operation as much as the given absolute position value.
FAS_MoveLinearIncPos	Between 2 motors that connected in the same port, start to Linear Interpolation operation as much as the given incremental position value.
FAS_MoveLinearAbsPos2[*1]	Improved version of **FAS_MoveLinearAbsPos.** Acceleration and Deceleration were improved.
FAS_ MoveLinearIncPos2[*1]	Improved version of **FAS_MoveLinearIncPos.** Acceleration and Deceleration were improved.
FAS_MoveSingleAxisAbsPosEx	The motor moves as much as the given absolute position value

	with custom accel/decel time value.
FAS_MoveSingleAxisIncPosEx	The motor moves as much as the given incremental position value with custom accel/decel time value.
FAS_MoveVelocityEx	The motor moves to the given velocity and direction: This function is available to Jog motion with custom accel/decel time value.
FAS_MovePause	The motor starts pause in runing or the motor starts again In pause status. The function operates normally only when operated by the position command which has the acceleration time and deceleration time with the same value.

*1 These functions are available from Firmware [ver.6.1.20.18] or higher version.

FAS_MoveStop

To stop the motor

Syntax

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

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product.

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_EmergencyStop

To stop the motor without deceleration

Syntax

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

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product.

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

Remarks

This function does not include deceleration phase. So, the user must be careful so that the machine cannot be impacted.

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_MoveOriginSingleAxis

To search the origin of system.

For more information, refer to 'User Manual_Text 9.3 Origin Return'.

Syntax

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

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product.

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_MoveSingleAxisAbsPos

To move the motor to the absolute coordinate

Syntax

```
int FAS_MoveSingleAxisAbsPos(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long lAbsPos,
    DWORD lVelocity,
    );
```

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product.

lAbsPos

Absolute coordinate of position to move

lVelocity

Velocity when the motor moves

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

Remarks

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcMove()
{
        BYTE nPortNo = 1;   // COMM Port Number
        BYTE iSlaveNo = 0;  // Slave No (0 ~ 15)
        DWORD dwAxisStatus, dwInput;
        EZISERVO_AXISSTATUS stAxisStatus;
        long lAbsPos, lIncPos, lVelocity;
        int nRtn;

        // Try to connect
```

```
	if (FAS_Connect(nPortNo, 115200) == FALSE)
	{
		// Connection failed.
		// The port is not connected or the baudrate may be wrong.
		return;
	}

	// Check error and Servo ON status.
	nRtn = FAS_GetAxisStatus(nPortNo, iSlaveNo, &dwAxisStatus);
	_ASSERT(nRtn == FMM_OK);
	stAxisStatus.dwValue = dwAxisStatus;

	//if (dwAxisStatus & 0x00000001)
	if (stAxisStatus.FFLAG_ERRORALL)
		FAS_ServoAlarmReset(nPortNo, iSlaveNo);
	//if ((dwAxisStatus & 0x00100000) == 0x00)
	if (stAxisStatus.FFLAG_SERVOON == 0)
		FAS_ServoEnable(nPortNo, iSlaveNo, TRUE);

	// Check input status.
	nRtn = FAS_GetIOInput(nPortNo, iSlaveNo, &dwInput);
	_ASSERT(nRtn == FMM_OK);

	if (dwInput & (SERVO_IN_LOGIC_STOP | SERVO_IN_LOGIC_PAUSE | SERVO_IN_LOGIC_ESTOP))
		FAS_SetIOInput(nPortNo, iSlaveNo, 0, SERVO_IN_LOGIC_STOP | SERVO_IN_LOGIC_PAUSE | SERVO_IN_LOGIC_ESTOP);

	// Increase the motor to 15000 pulse.
	lIncPos = 15000;
	lVelocity = 30000;
	nRtn = FAS_MoveSingleAxisIncPos(nPortNo, iSlaveNo, lIncPos, lVelocity);
	_ASSERT(nRtn == FMM_OK);

	// Stand by until motion command is completely finished.
	do
	{
		Sleep(1);

		nRtn = FAS_GetAxisStatus(nPortNo, iSlaveNo, &dwAxisStatus);
		_ASSERT(nRtn == FMM_OK);
		stAxisStatus.dwValue = dwAxisStatus;
	}
	while (stAxisStatus.FFLAG_MOTIONING);

	// Move the motor to '0'.
```

```
        lAbsPos = 0;
        lVelocity = 20000;
        nRtn = FAS_MoveSingleAxisAbsPos(nPortNo, iSlaveNo, lAbsPos, lVelocity);
        _ASSERT(nRtn == FMM_OK);

        // Stand by until motion command is completely finished
        do
        {
                Sleep(1);

                nRtn = FAS_GetAxisStatus(nPortNo, iSlaveNo, &dwAxisStatus);
                _ASSERT(nRtn == FMM_OK);
                stAxisStatus.dwValue = dwAxisStatus;
        }
        while (stAxisStatus.FFLAG_MOTIONING);

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

See Also

FAS_MoveSingleAxisIncPos

To move the motor to the incremental coordinate value

Syntax

```
int FAS_MoveSingleAxisIncPos(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long lIncPos,
    DWORD lVelocity
    );
```

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product.

lIncPos

Incremental coordinate of position to move

lVelocity

Velocity when the motor moves

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_MoveToLimit

To give the motor a command to search the limit sensor

Syntax

```
int FAS_MoveToLimit(
    BYTE nPortNo,
    BYTE iSlaveNo,
    DWORD lVelocity,
    int iLimitDir,
    );
```

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product.

lVelocity

Velocioty when the motor moves

iLimitDir

Limit direction which the motor moves

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_MoveVelocity

To move the motor to the relevant direction and velocity.

This function is also available for Jog motion.

Syntax

```
int FAS_MoveVelocity(
    BYTE nPortNo,
    BYTE iSlaveNo,
    DWORD lVelocity,
    int iVelDir
);
```

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product.

lVelocity

Velocity when the motor moves

iVelDir

Direction which the motor moves (0: -Jog, 1: +Jog)

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_PositionAbsOverride

To change the absolute position value set while the motor moves to the absolute position

Syntax

```
int FAS_PositionAbsOverride(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long lOverridePos
);
```

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product.

lOverridePos

Absolute coordinate position value to be changed

Return Value

FMM_OK : Command has been normally performed.

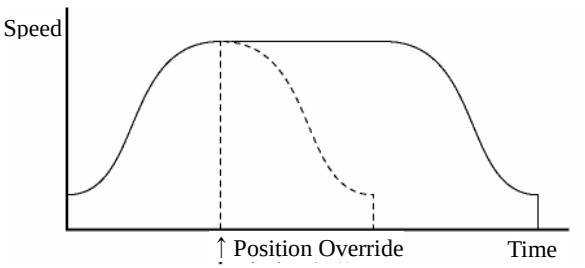
FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

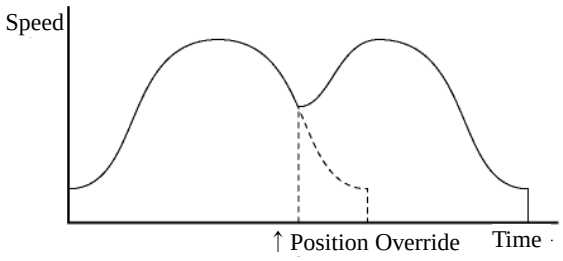
FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

Remarks

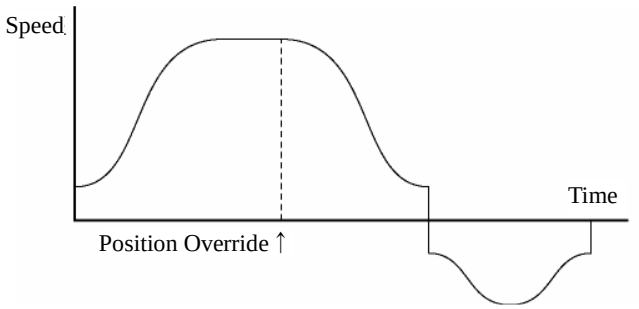
1) If the target position is set to the farther coordinate than the original target position while the motor moves to the accelerated or uniform velocity, the motor moves to the velocity pattern until then and stops the target position.



2) If the target position is changed while the motor is decelerated, it is again accelerated up to the uniform velocity and then stops to the target position.



3) If the changed target position is set to the closer coordinate than the original target position, the motor move to the changed target position.



Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_PositionIncOverride

FAS_PositionIncOverride

To change the incremental position value set while the motor moves to the incremental position

Syntax

```
int FAS_PositionIncOverride(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long lOverridePos
);
```

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product.

lOverridePos

Incremental coordinate position value to be changed

Return Value

FMM_OK : Command has been normally performed.
FMM_NOT_OPEN : The product has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

Remarks

1) Refer to 'FAS_PositionAbsOverride' library.
2) Can not use with FAS_PositionIncOverride library at the same time.
   Can not use with FAS_VelocityOverride library at the same time.

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_PositionAbsOverride

FAS_VelocityOverride

To change the velocity set while the motor moves

Syntax

```
int FAS_VelocityOverride(
    BYTE nPortNo,
    BYTE iSlaveNo,
    DWORD lVelocity
);
```

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product.

lVelocity

Velocity to be changed in [pps]

Return Value

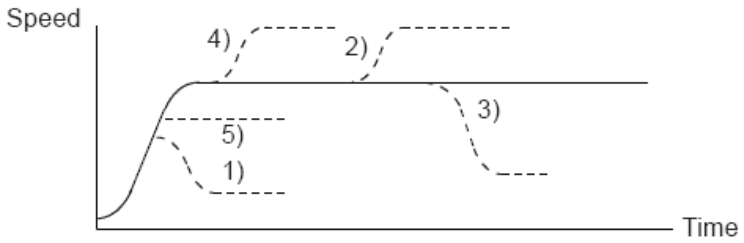
FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

Remarks



1) In case of ((change speed) < (speed before change)), the motor reaches the change speed through acceleration/deceleration using a new velocity pattern.

5) In case of ((change speed) ≥ (speed before change)), the motor reaches the change speed through acceleration/deceleration without any new velocity pattern.

4) The motor reaches the 'speed before change'without a change of the velocity pattern and then it reaches the 'change speed'by a new velocity pattern.

2),3) After acceleration/deceleration is finished, the motor reaches the change speed corresponding to the velocity pattern of the 'change speed'.

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_AllMoveStop

To stop the motor that connected in same port.

Syntax

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

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product. (must be '99')

Return Value

No response

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_AllEmergencyStop

To stop the motor that connected in same port without deceleration

Syntax

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

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product. (must be '99')

Return Value

No response

Remarks

This function does not include deceleration phase. So, the user must be careful so that the machine cannot be impacted.

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_AllMoveOriginSingleAxis

To search the origin of all motor that is connected in same port.

For more information, refer to 'User Manual_Text 9.3 Origin Return'.

Syntax

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

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product. (must be '99')

Return Value

No response

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_MoveLinearAbsPos2

Improved version of FAS_MoveLinearAbsPos.

Acceleration and Deceleration were improved.

Syntax

```
int FAS_MoveLinearAbsPos2(
    BYTE nNoOfBds,
    int* iBdID,
    long* lplAbsPos,
    DWORD lFeedreat,
    DWORD wAcceltime
);
```

Parameters

nNoOfBds

Number of drives to execute linear motion

iBdID

ID array of Drives

lplAbsPos

Movement position's arrangement of Drives

lFeedrate

Linear velocity value during movement

wAcceltime

Time value of acceleration / deceleration section during movement

Return Value

FMM_OK : The command ran successfully.

FMM_NOT_OPEN : The drive it not connected yet.

FMM_INVALID_SLAVE_NUM : The drive of corresponding iBdID does not exist.

Remarks

FAS_ MoveLinearIncPos2

Improved version of FAS_MoveLinearIncPos.

Acceleration and Deceleration were improved.

Syntax

```
int FAS_MoveLinearIncPos2(
    BYTE nNoOfBds,
    int* iBdID,
    long* lplIncPos,
    DWORD lFeedreat,
    DWORD wAcceltime
);
```

Parameters

nNoOfBds

Number of drives to execute linear motion

iBdID

ID array of Drives

lplAbsPos

Movement position's arrangement of Drives

lFeedrate

Linear velocity value during movement

wAcceltime

Time value of acceleration / deceleration section during movement

Return Value

FMM_OK : The command ran successfully.

FMM_NOT_OPEN : The drive it not connected yet.

FMM_INVALID_SLAVE_NUM : The drive of corresponding iBdID does not exist.

Remarks

FAS_AllMoveSingleAxisAbsPos

To move the motor that connected in same port to the absolute coordinate

Syntax

```
int FAS_AllMoveSingleAxisAbsPos(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long lAbsPos,
    DWORD lVelocity,
    );
```

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product. (must be '99')

lAbsPos

Absolute coordinate of position to move

lVelocity

Velocity when the motor moves

Return Value

No response

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_AllMoveSingleAxisIncPos

To move the motor that connected in same port to the incremental coordinate value

Syntax

```
int FAS_AllMoveSingleAxisIncPos(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long lIncPos,
    DWORD lVelocity
    );
```

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product. (must be '99')

lIncPos

Incremental coordinate of position to move

lVelocity

Velocity when the motor moves

Return Value

No response

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_MoveLinearAbsPos

To move(Linear Interpolation) more than 2 motors that connected in same port to the absolute coordinate.

Syntax

```
int FAS_MoveLinearAbsPos(
    BYTE nPortNo,
    BYTE nNoOfSlaves,
    BYTE *iSlaveNo,
    long *lAbsPos,
    DWORD lFeedrate,
    WORD wAccelTime
    );
```

Parameters

nPortNo

Port number of relevant product.

nNoOfSlaves

Slave numbers for Linear motioning.

iSlaveNo

Array of Slave numbers.

lAbsPos

Array of position value for each slave.

lFeedrate

Speed value for motioning.

wAccelTime

Acceleration & deceleration time value.

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

Remarks

FAS_MoveLinearIncPos

To move(Linear Interpolation) more than 2 motors that connected in same port to the incremental coordinate.

Syntax

```
int FAS_MoveLinearIncPos(
    BYTE nPortNo,
    BYTE nNoOfSlaves,
    BYTE *iSlaveNo,
    long *lIncPos,
    DWORD lFeedrate,
    WORD wAccelTime
    );
```

Parameters

nPortNo

Port number of relevant product.

nNoOfSlaves

Slave numbers for Linear motioning.

iSlaveNo

Array of Slave numbers.

lIncPos

Array of position value for each slave.

lFeedrate

Speed value for motioning.

wAccelTime

Acceleration & deceleration time value.

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

Remarks

FAS_MoveSingleAxisAbsPosEx

To move the motor to the absolute coordinate
(Available to set drive acc/dec time)

Syntax

```
int FAS_MoveSingleAxisAbsPosEx(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long lAbsPos,
    DWORD lVelocity,
    MOTION_OPTION_EX* lpExOption
);
```

Parameters

nPortNo
Port number of relevant product.

iSlaveNo
Slave number of relevant product.

lAbsPos
Absolute coordinate of position to move

lVelocity
Velocity when the motor moves

lpExOption
Custom option.

Return Value

FMM_OK : Command has been normally performed.
FMM_NOT_OPEN : The product has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

Remarks

Refer to MOTION_OPTION_EX struct.

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcMoveEx()
{
        BYTE nPortNo = 1;   // COMM Port Number
        BYTE iSlaveNo = 0;  // Slave No (0 ~ 15)
        DWORD dwAxisStatus, dwInput;
        EZISERVO_AXISSTATUS stAxisStatus;
        long lAbsPos, lIncPos, lVelocity;
```

```
MOTION_OPTION_EX opt = {0};
int nRtn;

// Try to connect
if (FAS_Connect(nPortNo, 115200) == FALSE)
{
	// Connection failed.
	// The port number may be wrong, or incorrect Baudrate.
	return;
}

// Moving motor with different acc/dec time : FAS_MoveSingleAxisIncPosEx
lIncPos = 15000;
lVelocity = 30000;

opt.flagOption.BIT_USE_CUSTOMACCEL = 1;
opt.flagOption.BIT_USE_CUSTOMDECEL = 1;

opt.wCustomAccelTime = 50;
opt.wCustomDecelTime = 200;

nRtn = FAS_MoveSingleAxisIncPosEx(nPortNo, iSlaveNo, lIncPos, lVelocity,
&opt);
_ASSERT(nRtn == FMM_OK);

// Waiting until motion command is done.
do
{
	Sleep(1);

	nRtn = FAS_GetAxisStatus(nPortNo, iSlaveNo, &dwAxisStatus);
	_ASSERT(nRtn == FMM_OK);
	stAxisStatus.dwValue = dwAxisStatus;
}
while (stAxisStatus.FFLAG_MOTIONING);

// Moving motor to position 0.
lAbsPos = 0;
lVelocity = 20000;
nRtn = FAS_MoveSingleAxisAbsPos(nPortNo, iSlaveNo, lAbsPos, lVelocity);
_ASSERT(nRtn == FMM_OK);

// Waiting until motion command is done.
do
{
	Sleep(1);
```

```
            nRtn = FAS_GetAxisStatus(nPortNo, iSlaveNo, &dwAxisStatus);
            _ASSERT(nRtn == FMM_OK);
            stAxisStatus.dwValue = dwAxisStatus;
        }
        while (stAxisStatus.FFLAG_MOTIONING);

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

See Also

FAS_MoveSingleAxisIncPosEx

To move the motor to the Incremental coordinate
(Available to set drive acc/dec time)

Syntax

```
int FAS_MoveSingleAxisIncPosEx(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long lIncPos,
    DWORD lVelocity,
    MOTION_OPTION_EX* lpExOption
);
```

Parameters

nPortNo
Port number of relevant product.

iSlaveNo
Slave number of relevant product.

lIncPos
Incremental coordinate of position to move

lVelocity
Velocity when the motor moves

lpExOption
Custom option.

Return Value

FMM_OK : Command has been normally performed.
FMM_NOT_OPEN : The product has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

Remarks

Example

See Also

FAS_MoveVelocityEx

To move the motor to the relevant direction and velocity.

This function is also available for Jog motion.

Syntax

```
int FAS_MoveVelocityEx(
    BYTE nPortNo,
    BYTE iSlaveNo,
    DWORD lVelocity,
    int iVelDir,
    VELOCITY_OPTION_EX* lpExOption
);
```

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product.

lVelocity

Velocity when the motor moves

iVelDir

Direction which the motor moves (0: -Jog, 1: +Jog)

lpExOption

Custom option.

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

Remarks

Refer to VELOCITY_OPTION_EX struct.

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcMoveVelocityEx()
{
        BYTE nPortNo = 1;      // COMM Port Number
        BYTE iSlaveNo = 0;     // Slave No (0 ~ 15)
        long lVelocity;
        VELOCITY_OPTION_EX opt = {0};
```

```
	int nRtn;

	// Try to connect
	if (FAS_Connect(nPortNo, 115200) == FALSE)
	{
		// Connection failed.
		// The port number may be wrong, or incorrect Baudrate.
		return;
	}

	// Moving motor with different acc/dec time : FAS_MoveSingleAxisIncPosEx
	lVelocity = 30000;

	opt.flagOption.BIT_USE_CUSTOMACCDEC = 1;
	opt.wCustomAccDecTime = 300;

	nRtn = FAS_MoveVelocityEx(nPortNo, iSlaveNo, lVelocity, DIR_INC, &opt);
	_ASSERT(nRtn == FMM_OK);

	Sleep(5000);
	FAS_MoveStop(nPortNo, iSlaveNo);
}
```

See Also

2－10. Position Table Control Function

Function Name	Description
FAS_PosTableReadItem	Read item values of desired position table from RAM area.
FAS_PosTableWriteItem	Stores the item value of a desired position table in RAM area.
FAS_PosTableWriteROM	Store all position table values in ROM area : All 256 PT values are saved.
FAS_PosTableReadROM	Read position table values in ROM area : Read all 256 PT values.
FAS_PosTableRunItem	Starts operation sequentially from the designated position table.
FAS_PosTableReadOneItem	Read desired item value of desired position table from RAM area.
FAS_PosTableWriteOneItem	Store desired item value of desired position table in RAM area.

FAS_PosTableReadItem

Read desired item values of position table.

Syntax

```
int FAS_PosTableReadItem(
    int iBdID,
    WORD wItemNo,
    LPITEM_NODE lpItem
);
```

Parameters

iBdID

ID number of the drive. IBdID set by the FAS_Connect function.

wItemNo

Item number to read.

lpItem

Item structure pointer to store Item value.

Return Value

FMM_OK : The command ran successfully.

FMM_NOT_OPEN : The drive it not connected yet.

FMM_INVALID_SLAVE_NUM : The drive of corresponding iBdID does not exist.

FMM_INVALID_PARAMETER_NUM : wItemNo is out of range.

Remarks

Example

```
#include "FAS_EziMOTIONPlusE.h"

void funcPosTable()
{

        BYTE sb1 = 192, sb2 = 168, sb3=0, sb4=2 // IP :192.168.0.2
        int iBdID = 0;          //   Drive number
        WORD wItemNo;
        ITEM_NODE nodeItem;
        int nRtn;

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

```
{
        // Connection failed.

        return;
}

// Read position table value of item No. 20 and change the position value.
wItemNo = 20;
nRtn = FAS_PosTableReadItem(iBdID, wItemNo, &nodeItem);
_ASSERT(nRtn == FMM_OK);

nodeItem.lPosition = 260000;  // Change the position value to 260000.
nodeItem.wBranch = 23;                  // Set next command to No. 23.
nodeItem.wContinuous = 1;               // Set the next command to
continue immediately without deceleration.

nRtn = FAS_PosTableWriteItem(iBdID, wItemNo, &nodeItem);
_ASSERT(nRtn == FMM_OK);

// The ROM value is loaded, ignoring the currently modified position table
data.
nRtn = FAS_PosTableReadROM(iBdID);
_ASSERT(nRtn == FMM_OK);

// Save current modified position table data to ROM.
nRtn = FAS_PosTableWriteROM(iBdID);
_ASSERT(nRtn == FMM_OK);

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

See Also

FAS_PosTableWriteItem

FAS_PosTableWriteItem

Modify the desired item in the position table.

Syntax

```
int FAS_PosIableWriteItem(
    int iBdID,
    WORD wItemNo,
    LPITEM_NODE lpItem
);
```

Parameters

iBdID

ID number of the drive. IBdID set by the FAS_Connect function.

wItemNo

Item number to change

lpItem

Item structure pointer to modify.

Return Value

FMM_OK : The command ran successfully.
FMM_NOT_OPEN : The drive it not connected yet.
FMM_INVALID_SLAVE_NUM : The drive of corresponding iBdID does not exist.
FMC_POSTABLE_ERROR : An error occurred while writing the position table.
FMM_INVALID_PARAMETER_NUM : wItemNo is out of range.

Remarks

The area where position table data is stored is RAM and ROM.
This function is to save data in RAM area. When power is off, data is deleted.

Example

See Also

FAS_PosTableWriteROM

Saves all current Position Table Items to ROM area.

Syntax

```
int FAS_PosTableWriteROM(
    int iBdID
```

```
);
```

Parameters

iBdID

ID number of the drive. IBdID set by the FAS_Connect function.

Return Value

FMM_OK : The command ran successfully.
FMM_NOT_OPEN : The drive it not connected yet.
FMM_INVALID_SLAVE_NUM : The drive of corresponding iBdlD does not exist.
FMC_POSTABLE_ERROR : An error occurred while saving the position table.

Remarks

The area where position table data is stored is RAM and ROM.
This function saves the data in the ROM area. The data is saved even when the power is turned off.

Example

See Also

FAS_PosTableReadROM

FAS_PosTableReadROM

Reat the Position Table Item values currently saved in the ROM area.

Syntax

```
int FAS_PosTableReadROM(
    int iBdID
);
```

Parameters

iBdID

ID number of the drive. IBdID set by the FAS_Connect function.

Return Value

FMM_OK : The command ran successfully.
FMM_NOT_OPEN : The drive it not connected yet.
FMM_INVALID_SLAVE_NUM : The drive of corresponding iBdlD does not exist.
FMC_POSTABLE_ERROR : An error occurred while reading the position table.

Remarks

Example

See Also

FAS_PosTableWriteROM

FAS_PosTableRunItem

Starts operation sequentially from the designated position table.

Syntax

```
int FAS_PosTableRunItem(
    int iBdID,
    WORD wItemNo
);
```

Parameters

iBdID

ID number of the drive. IBdID set by the FAS_Connect function.

wItemNo

Item number to start operation.

Return Value

FMM_OK : The command ran successfully.

FMM_NOT_OPEN : The drive it not connected yet.

FMM_INVALID_SLAVE_NUM : The drive of corresponding iBdID does not exist.

FMM_INVALID_PARAMETER_NUM : wItemNo is out of range.

Remarks

Example

See Also

FAS_GetAllStatus

FAS_MoveStop

FAS_EmergencyStop

FAS_PosTableReadOneItem

Read desired item value of position table.

Syntax

```
int FAS_PosTableReadOneItem(
    int iBdID,
    WORD wItemNo,
    WORD wOffset,
    long* lPosItemVal
);
```

Parameters

iBdID

ID number of the drive. IBdID set by the FAS_Connect function.

wItemNo

Item number to read.

wOffset

Offset value of the item to read. (Refer to 「1-2-6. Position Table Item」)

Return Value

FMM_OK : The command ran successfully.

FMM_NOT_OPEN : The drive it not connected yet.

FMM_INVALID_SLAVE_NUM : The drive of corresponding iBdID does not exist.

FMM_INVALID_PARAMETER_NUM : wItemNo is out of range.

Remarks

Example

See Also

FAS_PosTableReadItem

FAS_PosTableWriteOneItem

FAS_PosTableWriteOneItem

Modify the desired Item value in the Position Table.

Syntax

```
int FAS_PosIableWriteOneItem(
int iBdID,
WORD wItemNo,
WORD wOffset,
long lPosItemVal
);
```

Parameters

iBdID

ID number of the drive. IBdID set by the FAS_Connect function.

wItemNo

Item number to change

wOffset

Offset value of the item to read. (Refer to 「1-2-6. Position Table Item」)

Return Value

FMM_OK : The command ran successfully.

FMM_NOT_OPEN : The drive it not connected yet.

FMM_INVALID_SLAVE_NUM : The drive of corresponding iBdID does not exist.

FMC_POSTABLE_ERROR : An error occurred while writing the position table.

FMM_INVALID_PARAMETER_NUM : wItemNo is out of range.

Remarks

Example

See Also

FAS_PosTableWriteItem

FAS_PosTableReadOneItem

2－11. Other Control Function

Function Name	Description
FAS_TriggerOutput_RunA	To generate an output signal at a specific position.
FAS_TriggerOutput_Status	To check if the trigger output pulse(COMP) is working or not.

FAS_TriggerOutput_RunA

To start/stop the digital output signal(Compare Out pin) when reaching the specific Taregt position.

Syntax

```
int FAS_TriggerOutput_RunA(
BYTE nPortNo,
BYTE iSlaveNo,
BOOL bStartTrigger,
long lStartPos,
DWORD dwPeriod,
DWORD dwPulseTime,

);
```

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product.

bStartTrigger

Output start/stop command (1:start, 0:stop)

long lStartPos

Output start position [pulse]

DWORD dwPeriod

Period of output signal [pulse]

DWORD dwPulseTime

Pulse width of output signal [msec]

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

Remarks

Example

See Also

FAS_TriggerOutput_Status

FAS_TriggerOutput_Status

To check if the trigger output function is working or not.

Syntax

```
int FAS_TrggerOutput_Status(
BYTE nPortNo,
BYTE iSlaveNo,
BYTE* bTriggerStatus

);
```

Parameters

nPortNo

Port number of relevant product.

iSlaveNo

Slave number of relevant product.

bTriggerStatus

Current status of signal output.

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The product has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no Slave of iSlaveNo in the relevant port.

Remarks

Example

See Also

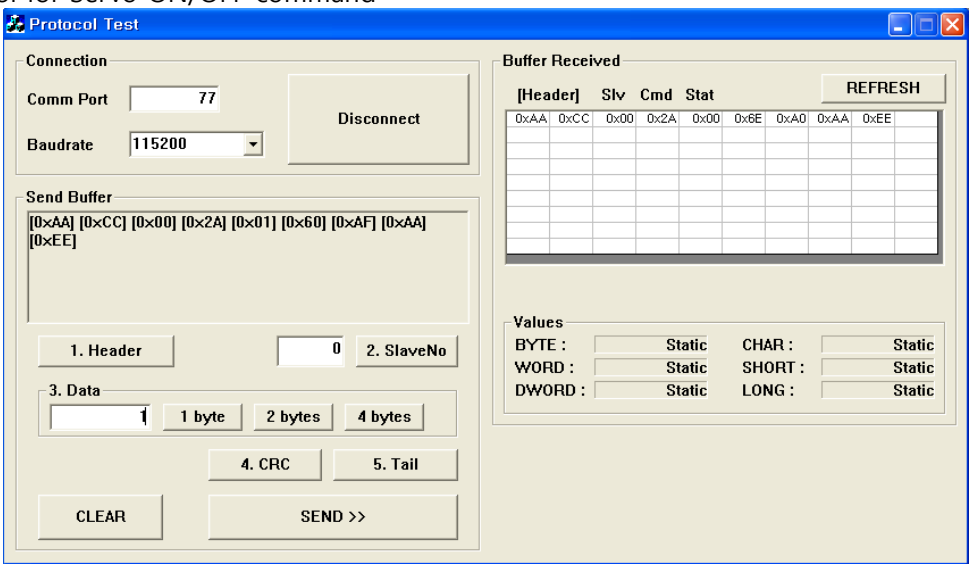
FAS_ TriggerOutput_RunA

3 . Protocol for PLC Program

As follows window is open when you click in User Program(GUI) installed folder.

The example below will help you to understand the protocol programming.

(1)Protocol for Servo ON/OFF command



The header and tail information are needed for protocol programming.
Additionally Frame Data (Slave ID, Frame type, Data and CRC) is also needed in every protocol with header and tail.

1)Select 'Comm Port','Baudrate' and then click 'Connect' button.

2)Header : Click 'Header' and you can see '[0XAA][0xCC]' on 'Send Buffer' window.

3)Slave ID : Insert connected slave number(above example is '0') and click 'SlaveNo'.

4)Frame type : Select 'Frame type'.
You can check as below table information at '1-2-1. Frame Type and Data Configuration'.

Frame type	DLL Library name	Data
42 (0x2A)	FAS_ServoEnable	Setting the Servo ON/OFF status. Sending : 1 byte 1 byte 0:OFF, 1:ON

Input '42' and the size of receive frame type is 1byte, so click '1 byte'.

5)Data : To make Servo ON status, the designated command is '1'. Input '1' and click'1 byte'.

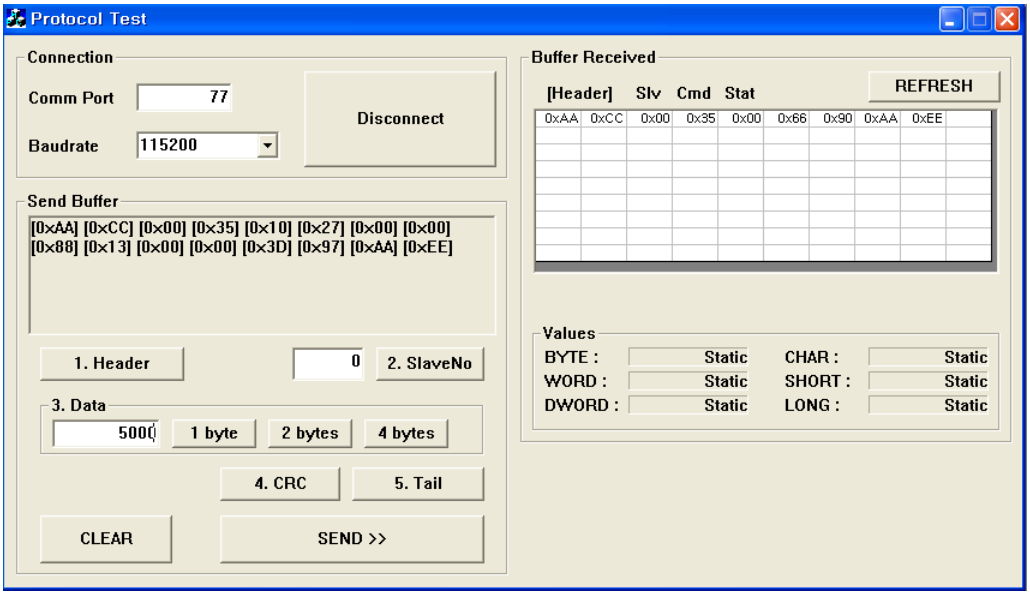
6)CRC : If click 'CRC', calculate checksum value automatically and expressed as 2 byte size.

7)Tail : Click 'Tail' and you can see '[0XAA][0xEE]' on 'Send Buffer' window.

8)Finally click 'Send' button to send command characters to Ezi-MOTIONLINK Plus-R.
You can check the motor torque and LED flash for Servo ON status.

9) A command is sent to Ezi-MOTIONLINK Plus-R normally and information which is received normally from Ezi-MOTIONLINK Plus-R is expressed at 'Buffer Received' window.

(2)Protocol for motor motion command



1) Header
2) Slave No.
3) Frame type : Input '53' for 'Incremental Move'command.
4) Data(Position value) : Input'10000'and click '4byte'.
5) Data(Running speed) : Input'5000'and click '4 byte'.
6) CRC
7) Tail
8) Send : If parameter is 'default', the motor rotates 360 degree. Command '53' is an incremental move command, so if tou click 'Send" again, the motor rotates same amount of distance.

(3)PLC Programming

Between the actual protocol program and the above example have difference as below. In GUI program for a test, a function below is carried out automatically.

1) For clear separation between Header and Tail, use 'Byte stuffing' technique.
   Refer to 「1-1-2. RS-485 Communication Protocol(Ver6)」.
2) Use CRC function to check communication error.
3) Refer to 「1-1-3. CRC Calculation Example」.

FASTECH
Fast, Accurate, Smooth Motion

FASTECH Co., Ltd.
Rm#1202, 401-dong, Bucheon Techno-Park,
655, Pyeongcheon-ro, Bucheon-si Gyeonggi-do,
Republic of Korea (Zip:14502)
TEL : +82-32-234-6300 FAX : +82-32-234-6302
E-mail : fastech@fastech.co.kr
Homepage : www.fastech.co.kr