

Ezi-STEP[®] Plus-R

Micro Stepping System
with Network based Motion Controller

Ezi-STEP[®] Plus-R MINI

Micro Stepping System
with Network Based Motion Controller

Ezi-STEP[®] ALL

Step Motors with Integrated
Drive and Controller

User Manual

Communication Function_Ver6

(Rev.08.05.029)



– Table of Contents –

1. Communication Protocols.....	6
1-1. Communication Functions	6
1-1-1. Communication Specifications	6
1-1-2. RS-485 Communication Protocol(Ver6).....	6
1-1-3. CRC Calculation Example.....	7
1-1-4. Response Frame Structure and Communication Error(Ver6)	10
1-2. Structure of Frame type(Ver6)	11
1-2-1. Frame type and Data Configuration.....	11
1-2-2. Parameter Lists	25
1-2-3. Setup bit of Output pin.....	26
1-2-4. Setup bit of Input pin.....	26
1-2-5. Bit setup of Status Flag.....	27
1-2-6. Position Table Item.....	27
1-2-7. Information of Motors.....	28
1-3. Program Method.....	29
2. Library for PC Program.....	30
2-1. Library Configuration	30
2-2. Communication Status Window.....	31
2-3. Drive Link Function.....	35
FAS_Connect.....	36
FAS_Close.....	38
FAS_GetSlaveInfo.....	39
FAS_GetMotorInfo.....	40

FAS_IsSlaveExist.....	41
FAS_EnableLog.....	42
FAS_SetLogPath.....	43
2-4. Parameter Control Function.....	44
FAS_SaveAllParameters.....	45
FAS_SetParameter.....	47
FAS_GetParameter.....	48
FAS_GetROMParameter.....	49
2-5. Servo Control Function.....	50
FAS_StepAlarmReset.....	51
2-6. Control I/O Function.....	52
FAS_SetIOInput.....	53
FAS_GetIOInput.....	55
FAS_SetIOOutput.....	56
FAS_GetIOOutput.....	57
FAS_GetIOAssignMap.....	58
FAS_SetIOAssignMap.....	60
FAS_IOAssignMapReadROM.....	61
2-7. Position Control Function.....	62
FAS_SetCommandPos.....	63
FAS_SetActualPos.....	65
FAS_GetCommandPos.....	66
FAS_GetActualPos.....	68
FAS_GetPosError.....	69
FAS_GetActualVel.....	70
FAS_ClearPosition.....	71
2-8. Drive Status Control Function.....	72
FAS_GetIOAxisStatus.....	73
FAS_GetMotionStatus.....	74


FAS_GetAllStatus.....	75
FAS_GetAxisStatus.....	76
2-9. Running Control Function	77
FAS_MoveStop.....	78
FAS_EmergencyStop.....	79
FAS_MoveOriginSingleAxis.....	80
FAS_MoveSingleAxisAbsPos.....	81
FAS_MoveSingleAxisIncPos.....	83
FAS_MoveToLimit.....	84
FAS_MoveVelocity.....	85
FAS_PositionAbsOverride.....	86
FAS_PositionIncOverride.....	88
FAS_VelocityOverride.....	89
FAS_AllMoveStop.....	90
FAS_AllEmergencyStop.....	91
FAS_AllMoveOriginSingleAxis.....	92
FAS_AllMoveSingleAxisAbsPos.....	93
FAS_AllMoveSingleAxisIncPos.....	94
FAS_MoveSingleAxisAbsPosEx.....	95
FAS_MoveSingleAxisIncPosEx.....	97
FAS_MoveVelocityEx.....	98
2-10. Position Table Control Function	100
FAS_PosTableReadItem.....	101
FAS_PosTableWriteItem.....	103
FAS_PosTableWriteROM.....	104
FAS_PosTableReadROM.....	105
FAS_PosTableRunItem.....	106
FAS_PosTableReadOneItem.....	107
FAS_PosTableWriteOneItem.....	108

2-11. Other Control Function	109
FAS_TriggerOutput_RunA.....	110
FAS_TriggerOutput_Status.....	
3. Protocol for PLC Program.....	112

1. Communication Protocols

1-1. Communication Functions

Ezi-STEP Plus-R can control up to 16 axis by Daisy-Chain link at RS-485(two-wire).

 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

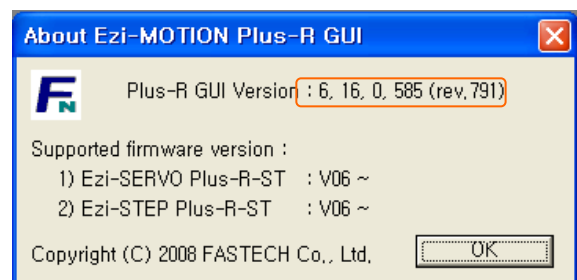
Specification	RS-485
Communication Type	Asynchronous
	Half-duplex
Baudrate [bps]	19200, 38400, 57600, 115200, 230400, 460800, 921600
Data Type	8bit ASCII Code, HEX
Parity	No
Stop Bit	1bit
CRC Check	Yes
Max Cabling Length (Converter ↔ Drive)	30 m
Min Cable length between drive	More than 60 cm
Number of Connected Axis	16 axis (No. 0~F)

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

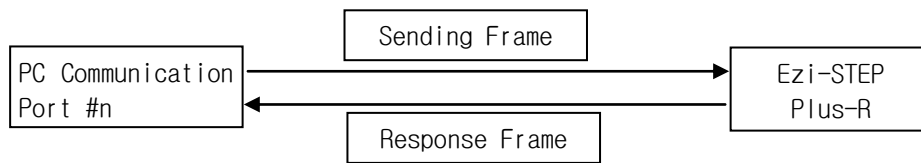
There are 2 kinds of program version for STEP Plus-R. This manual supports for **Version 6** level.

Type	Firmware version	compatibility	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 checked 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	5~252 bytes	0xAA 0xEE

- ① 0xAA : Delimited byte
- ② 0xAA 0xCC: Indicate header of the frame.
- ③ 0xAA 0xEE: Indicate tail of the frame.
- 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 indicates an error.

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: Dive module number (0~15) connected to the PC communication port.
- ② Frame type: Designate command type of relevant frames. For the command type, refer to 「Frame Type and Data Configuration」 section.
- ③ Data: Data structure and length is set according to Frame type. For more information, refer to 「Frame Type and Data Configuration」 section.
- ④ CRC: To check an error which occurs during communication, '0xA001' of a polynomial factor in **CRC(Cyclic Redundancy Check)**. '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 identical. 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, need to recognize as the error status.)
- ② Frame type: Same to sending Frame.
(When this is not same to sending data, need to recognize 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 included to the display of communication status (error / normal status).

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 : Responded data value is aside from 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) Servo is OFF status 4) try to Z-pulse Origin without external encoder
0x86	134	RESET Failure : The user has tried to execute new running commands in wrong condition as follows. 1) STEP is ON status 2) Already reset status by external input signal
0xAA	170	CRC Error : When received frame data is error by external noise, sending side of DLL Library is automatically trying to send 1 more time of communication signal.

	<ul style="list-style-type: none"> 1) If 'Header' and 'Slave ID' values in the sending Frame are abnormal, there is no response from the drive. 2) If the communication status is displayed to '130', the size of response data is '0' byte.
---	--

1-2. Structure of Frame type(Ver6)

1-2-1. Frame type and Data Configuration

(1) The following table explains the content and configuration by frame type of data.

Frame Type	Library Name	Contents						
0x01 (1)	FAS_ GetSlaveInfo	<p>Connected slave type and program version information are required.</p> <p>Sending : 0 byte Response : 1~248 bytes</p> <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 bytes)</td></tr></table> <p>◆ Slave type : 20 : Ezi-STEP Plus-R ST 60 : Ezi-STEP Plus-R MINI 1 : Ezi-SERVO Plus-R ST</p>	1 byte	1 bytes	0~246 bytes	Communication status	Slave type	ACII string with NULL byte (strlen() + 1 bytes)
1 byte	1 bytes	0~246 bytes						
Communication status	Slave type	ACII string with NULL byte (strlen() + 1 bytes)						
0x05 (5)	FAS_ GetMotorInfo	<p>Connected motor type and manufacturer information are required.</p> <p>Sending : 0 byte Response : 1~248 bytes</p> <table><tr><td>1 byte</td><td>1 bytes</td><td>0~246 bytes</td></tr><tr><td>Communication status</td><td>Motor type (1~255)</td><td>ACII string with NULL byte (strlen() + 1 bytes)</td></tr></table> <p>◆ Motor type: refer to 「1-1-7.Information of Motors」</p>	1 byte	1 bytes	0~246 bytes	Communication status	Motor type (1~255)	ACII string with NULL byte (strlen() + 1 bytes)
1 byte	1 bytes	0~246 bytes						
Communication status	Motor type (1~255)	ACII string with NULL byte (strlen() + 1 bytes)						
0x10 (16)	FAS_ SaveAllParameters	<p>Save currently set parameters & assigned signals in the ROM of the drive. Even the drive is powered off, saving these data & parameters are possible.</p> <p>Values set at 'FAS_SetParameter' & 'FAS_SetIOAssignMap' are saved together.</p> <p>Sending : 0 byte Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table>	1 byte	Communication status				
1 byte								
Communication status								
0x11 (17)	FAS_ GetROMParameter	<p>Specific parameter values in the ROM are recognized.</p> <p>Sending : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Parameter number (0~28)</td></tr></table> <p>Response : 5 bytes</p> <table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication status</td><td>Parameter value</td></tr></table> <p>Refer to 「1-2-2.Parameter List」</p>	1 byte	Parameter number (0~28)	1 byte	4 bytes	Communication status	Parameter value
1 byte								
Parameter number (0~28)								
1 byte	4 bytes							
Communication status	Parameter value							

0x12 (18)	FAS_ SetParameter	<p>Specific parameter values are saved to the RAM.</p> <p>Sending : 5 bytes</p> <table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Parameter number (0~28)</td><td>Parameter value</td></tr></table> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table> <p>Refer to 「1-2-2.Parameter List」</p>	1 byte	4 bytes	Parameter number (0~28)	Parameter value	1 byte	Communication status
1 byte	4 bytes							
Parameter number (0~28)	Parameter value							
1 byte								
Communication status								
0x13 (19)	FAS_ GetParameter	<p>Specific parameter values in the RAM are recognized</p> <p>Sending : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Parameter number (0~28)</td></tr></table> <p>Response : 5 bytes</p> <table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication status</td><td>Parameter value</td></tr></table> <p>Refer to 「1-2-2.Parameter List」</p>	1 byte	Parameter number (0~28)	1 byte	4 bytes	Communication status	Parameter value
1 byte								
Parameter number (0~28)								
1 byte	4 bytes							
Communication status	Parameter value							
0x20 (32)	FAS_ SetI0Output	<p>Set output signal level of the control output port.</p> <p>Sending : 8 bytes</p> <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> <p>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」 .</p> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table>	4 bytes	4 bytes	I/O set mask value	I/O clear mask value	1 byte	Communication status
4 bytes	4 bytes							
I/O set mask value	I/O clear mask value							
1 byte								
Communication status								
0x21 (33)	FAS_ SetI0Input	<p>Set input signal level of the control input port.</p> <p>Sending : 8 bytes</p> <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> <p>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」 .</p> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table>	4 bytes	4 bytes	I/O set mask value	I/O clear mask value	1 byte	Communication status
4 bytes	4 bytes							
I/O set mask value	I/O clear mask value							
1 byte								
Communication status								

0x22 (34)	FAS_ GetI0Input	<p>Current input signal status of the control input port is recognized.</p> <p>Sending : 0 byte</p> <p>Response : 5 byte</p> <table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication status</td><td>Input status value</td></tr></table> <p>Relevant bit by each input signal , refer to 「1-2-4. Bit setup of Input Pin」 .</p>	1 byte	4 bytes	Communication status	Input status value				
1 byte	4 bytes									
Communication status	Input status value									
0x23 (35)	FAS_ GetI0Output	<p>Current output signal status of the control output port is recognized.</p> <p>Sending : 0 byte</p> <p>Response : 5 byte</p> <table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication status</td><td>Output status value</td></tr></table> <p>Relevant bit by each output signal , refer to 「1-2-3.Bit setup of Output Pin」 .</p>	1 byte	4 bytes	Communication status	Output status value				
1 byte	4 bytes									
Communication status	Output status value									
0x24 (36)	FAS_ SetI0AssignMap	<p>Assign I/O signal to the pin of CN1 port and set signal level simultaneously. By running ‘FAS_SaveAllParameters’ , you can save the setting value to the ROM.</p> <p>Sending : 6 bytes</p> <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> <p>◆ I/O number: ‘0~11’ corresponds to ‘Limit+, Limit-, Org, IN1, ..., IN9’ respectively, and ‘12~22’ corresponds to ‘COMP, OUT1, ..., OUT9’ respectively.</p> <p>◆ I/O pin masking data: Refer to 「1-2-4. Bit setup of Input Pin」 .</p> <p>◆ Level Setting: 0:Active Low, 1:Active High</p> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table>	1 byte	4 bytes	1 byte	I/O number	I/O pin masking data	Setting level	1 byte	Communication status
1 byte	4 bytes	1 byte								
I/O number	I/O pin masking data	Setting level								
1 byte										
Communication status										
0x25 (37)	FAS_ GetI0AssignMap	<p>Recognize pin setting status of CN1 port from RAM.</p> <p>Sending : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>I/O number</td></tr></table> <p>◆ I/O number: ‘0~11’ corresponds to ‘Limit+, Limit-, Org, IN1, ..., IN9’ respectively, and ‘12~22’ corresponds to ‘COMP, OUT1, ..., OUT9’ respectively.</p> <p>Response : 6 bytes</p> <table><tr><td>1 byte</td><td>4 bytes</td><td>1 byte</td></tr><tr><td>Communication status</td><td>I0 pin masking status</td><td>Level status</td></tr></table> <p>For more information, refer to ‘0x24’ Frame type.</p>	1 byte	I/O number	1 byte	4 bytes	1 byte	Communication status	I0 pin masking status	Level status
1 byte										
I/O number										
1 byte	4 bytes	1 byte								
Communication status	I0 pin masking status	Level status								

0x26 (38)	FAS_ IOAssignMapReadROM	<p>Recognize setting status of I/O and setting value of signal level from ROM area. These values are loaded to RAM.</p> <p>Sending : 0 byte</p> <p>Response : 2 bytes</p> <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>	1 byte	1 byte	Communication status	Command performing status (0 : complete, values except 0: error)												
1 byte	1 byte																	
Communication status	Command performing status (0 : complete, values except 0: error)																	
0x27 (39)	FAS_ TriggerOutput_RunA	<p>Start/Stop command for 'Compare Out' signal</p> <p>Sending : 18 bytes</p> <table><tr><td>1 byte</td><td>4 bytes</td><td>4 byte</td></tr><tr><td>Output start/stop (1:start 0:stop)</td><td>Pulse start position [pulse]</td><td>Pulse period [pulse]</td></tr></table> <table><tr><td>4 byte</td><td>1 bytes</td><td>4 byte</td></tr><tr><td>Pulse width [msec]</td><td>Output pin number (fix to 0)</td><td>spare</td></tr></table> <p>Response : 2 byte</p> <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>	1 byte	4 bytes	4 byte	Output start/stop (1:start 0:stop)	Pulse start position [pulse]	Pulse period [pulse]	4 byte	1 bytes	4 byte	Pulse width [msec]	Output pin number (fix to 0)	spare	1 byte	1 byte	Communication status	Command performing status (0 : complete, values except 0: error)
1 byte	4 bytes	4 byte																
Output start/stop (1:start 0:stop)	Pulse start position [pulse]	Pulse period [pulse]																
4 byte	1 bytes	4 byte																
Pulse width [msec]	Output pin number (fix to 0)	spare																
1 byte	1 byte																	
Communication status	Command performing status (0 : complete, values except 0: error)																	
0x28 (40)	FAS_ TriggerOutput_Status	<p>Command to check if the trigger output pulse is working or not.</p> <p>Sending : 0 byte</p> <p>Response : 2 byte</p> <table><tr><td>1 byte</td><td>1 bytes</td></tr><tr><td>Communication status</td><td>Status (1:output ON, 0 :output OFF)</td></tr></table>	1 byte	1 bytes	Communication status	Status (1:output ON, 0 :output OFF)												
1 byte	1 bytes																	
Communication status	Status (1:output ON, 0 :output OFF)																	
0x2A (42)	FAS_ StepEnable	<p>Step ON/OFF status is set.</p> <p>Sending : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>0:OFF, 1:ON</td></tr></table> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table>	1 byte	0:OFF, 1:ON	1 byte	Communication status												
1 byte																		
0:OFF, 1:ON																		
1 byte																		
Communication status																		

0x2C (44)	FAS_ StepAlarmReset	<p>Reset STEP alarm status or release reset. To make a reset, send 'reset ON' and 'reset release' sequentially.</p> <p>Sending : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Reset ON(1) Reset release(0)</td></tr></table> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table>	1 byte	Reset ON(1) Reset release(0)	1 byte	Communication status
1 byte						
Reset ON(1) Reset release(0)						
1 byte						
Communication status						
0x2E (46)	FAS_ ServoAlarmtype	<p>To request the Alarm type</p> <p>Sending: 0 byte</p> <p>Response: 2 byte</p> <table><tr><td>1 byte</td><td>1 bytes</td></tr><tr><td>Communication status</td><td>Alarm type (1~)</td></tr></table> <p>◆ Alarm type: No alarm (0) OverCurrent(1) OverSpeed(2) StepOut(3) OverTemperature(5) BackEMF(6) MotorConnect(7) MotorPower(9) Inposition(10)</p>	1 byte	1 bytes	Communication status	Alarm type (1~)
1 byte	1 bytes					
Communication status	Alarm type (1~)					

0x31 (49)	FAS_ MoveStop	Request to stop motor currently operates Sending : 0 byte Response : 1 byte <table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table>	1 byte	Communication status				
1 byte								
Communication status								
0x32 (50)	FAS_ EmergencyStop	Request emergency stop of the motor. Sending : 0 byte Response : 1 byte <table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table>	1 byte	Communication status				
1 byte								
Communication status								
0x33 (51)	FAS_ MoveOriginSingleAxis	Request the motor to return to origin under current setting parameter condition Sending : 0 byte Response : 1 byte <table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table>	1 byte	Communication status				
1 byte								
Communication status								
0x34 (52)	FAS_ MoveSingleAxisAbsPos	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>	4 bytes	4 bytes	Absolute position value	Running speed [pps]	1 byte	Communication status
4 bytes	4 bytes							
Absolute position value	Running speed [pps]							
1 byte								
Communication status								
0x35 (53)	FAS_ MoveSingleAxisIncPos	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>	4 bytes	4 bytes	Incremental position value	Running speed [pps]	1 byte	Communication status
4 bytes	4 bytes							
Incremental position value	Running speed [pps]							
1 byte								
Communication status								

0x36 (54)	FAS_ MoveToLimit	<p>Request the motor to start limit motion under current setting parameter condition</p> <p>Sending : 5 bytes</p> <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> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table>	4 bytes	1 byte	Running speed [pps]	Running direction (0: -Limit 1: +Limit)	1 byte	Communication status
4 bytes	1 byte							
Running speed [pps]	Running direction (0: -Limit 1: +Limit)							
1 byte								
Communication status								
0x37 (55)	FAS_ MoveVelocity	<p>Request the motor to start jog motion at the current setting parameter condition</p> <p>Sending : 5 bytes</p> <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> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table>	4 bytes	1 byte	Running speed [pps]	Running direction (0: -Jog 1: +Jog)	1 byte	Communication status
4 bytes	1 byte							
Running speed [pps]	Running direction (0: -Jog 1: +Jog)							
1 byte								
Communication status								
0x38 (56)	FAS_ PositionAbsOverride	<p>Request the motor to change the target absolute position value[pulse] while it is in running.</p> <p>Sending : 4 bytes</p> <table><tr><td>4 bytes</td></tr><tr><td>Changed command position value [pulse]</td></tr></table> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table>	4 bytes	Changed command position value [pulse]	1 byte	Communication status		
4 bytes								
Changed command position value [pulse]								
1 byte								
Communication status								
0x39 (57)	FAS_ PositionIncOverride	<p>Request the motor to change the target incremental position value[pulse] during operation.</p> <p>Sending : 4 bytes</p> <table><tr><td>4 bytes</td></tr><tr><td>Changed command position value [pulse]</td></tr></table> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table>	4 bytes	Changed command position value [pulse]	1 byte	Communication status		
4 bytes								
Changed command position value [pulse]								
1 byte								
Communication status								
0x3A (58)	FAS_ VelocityOverride	<p>Request the motor to change the running speed value[pps] during operation.</p> <p>Sending : 4 bytes</p> <table><tr><td>4 bytes</td></tr><tr><td>Changed running speed [pps]</td></tr></table> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table>	4 bytes	Changed running speed [pps]	1 byte	Communication status		
4 bytes								
Changed running speed [pps]								
1 byte								
Communication status								

0x3B (59)	FAS_ AllMoveStop	Request stop for all motor that connected in same port during operation. Sending : 0 byte (Slave number must be '99') Response : no response (All slaves do not send response because cannot receive response from all slaves simultaneously.)								
0x3C (60)	FAS_ AllEmergencyStop	Request emergency stop for all motor that connected in same port during operation. Sending : 0 byte (Slave number must be '99') Response : no response (All slaves do not send response because cannot receive response from all slaves simultaneously.)								
0x3D (61)	FAS_All MoveOriginSingleAxis	Request return to origin under current setting parameter condition for all drives that connected in same port. Sending : 0 byte (Slave number must be '99') Response : no response (All slaves do not send response because cannot receive response from all slaves simultaneously.)								
0x3E (62)	FAS_All SingleAxisAbsPos	Request move its position as much as the absolute value[pulse] for all drives 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 (All slaves do not send response because cannot receive response from all slaves simultaneously.)	4 bytes	4 bytes	Absolute position value	Running speed [pps]				
4 bytes	4 bytes									
Absolute position value	Running speed [pps]									
0x3F (63)	FAS_All SingleAxisIncPos	Request move its position as much as the incremental value[pulse] for all drives 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 (All slaves do not send response because cannot receive response from all slaves simultaneously.)	4 bytes	4 bytes	incremental position value	Running speed [pps]				
4 bytes	4 bytes									
incremental position value	Running speed [pps]									
0x80 (128)	FAS_ MoveSingleAxisAbsPos Ex	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>	4 bytes	4 bytes	4 bytes	2 bytes	Absolute position value	Running speed [pps]	Flag option	Custom Accel. Time (1~9999)
4 bytes	4 bytes	4 bytes	2 bytes							
Absolute position value	Running speed [pps]	Flag option	Custom Accel. Time (1~9999)							

		<table><tr><td>2 bytes</td><td>24 bytes</td></tr><tr><td>Custom Decel. Time (1~9999)</td><td>Reserved</td></tr></table> <p>Flag ooption : 0x0001 : reserved 0x0002 : Custom Accel. Time is used. 0x0004 : Custom Decel. Time is used.</p> <p>If the Flag bit is OFF status(0), Accel./Decel. time value is used that saved in controller.</p> <p>Response: 1 byte</p>	2 bytes	24 bytes	Custom Decel. Time (1~9999)	Reserved								
2 bytes	24 bytes													
Custom Decel. Time (1~9999)	Reserved													
0x81 (129)	FAS_ MoveSingleAxisIncPos Ex	<p>Request the motor to move its position as much as the absolute value[pulse] with Custom Accel. / Decel. Time[msec]</p> <p>Sending: 40 bytes</p> <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> <p>Flag ooption : 0x0001 : reserved 0x0002 : Custom Accel. Time is used. 0x0004 : Custom Decel. Time is used.</p> <p>If the Flag bit is OFF status(0), Accel./Decel. time value is used that saved in controller.</p> <p>Response: 1 byte</p>	4 bytes	4 bytes	4 bytes	2 bytes	incremental position value	Running speed [pps]	Flag option	Custom Accel. Time (1~9999)	2 bytes	24 bytes	Custom Decel. Time (1~9999)	Reserved
4 bytes	4 bytes	4 bytes	2 bytes											
incremental position value	Running speed [pps]	Flag option	Custom Accel. Time (1~9999)											
2 bytes	24 bytes													
Custom Decel. Time (1~9999)	Reserved													
0x82 (130)	FAS_ MoveVelocityEx	<p>Request the motor to start jog motion at the current setting parameter condition with custom Accel/Decel time value[msec].</p> <p>Sending: 37 bytes</p> <table><tr><td>4 bytes</td><td>1 bytes</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> <p>Flag ooption : 0x0001 : reserved 0x0002 : Custom Accel./Decel. Time is used.</p> <p>If the Flag bit is OFF status(0), Accel./Decel. time value is used that saved in controller.</p> <p>Response : 1 byte</p>	4 bytes	1 bytes	4 bytes	Running speed [pps]	Running direction (0: -Jog 1: +Jog)	Flag option	2 bytes	26 bytes	Custom Accel./Decel. Time (1~9999)	Reserved		
4 bytes	1 bytes	4 bytes												
Running speed [pps]	Running direction (0: -Jog 1: +Jog)	Flag option												
2 bytes	26 bytes													
Custom Accel./Decel. Time (1~9999)	Reserved													

	FAS_MoveLinearAbsPos	Fulfill Linear Interpolation for multi-drives connected in same port. Position value is absolute value [pulse] unit and refer to 「2. Library for PC Program」.												
	FAS_MoveLinearIncPos	Fulfill Linear Interpolation for multi-drives connected in same port. Position value is incremental value [pulse] unit and refer to 「2. Library for PC Program」.												
0x40 (64)	FAS_GetAxisStatus	<p>Request the flag value indicates operation status</p> <p>Sending : 0 byte Response : 5 bytes</p> <table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication status</td><td>Status flag value</td></tr></table> <p>Assign bit related to each Flag, refer to 「1-2-5. Bit setup of Status Flag」.</p>	1 byte	4 bytes	Communication status	Status flag value								
1 byte	4 bytes													
Communication status	Status flag value													
0x41 (65)	FAS_GetIOAxisStatus	<p>Request the I/O status and the running Flag status. (Frame type 0x22, 0x23, and 0x40 are packed.)</p> <p>Sending : 0 byte Response : 13 bytes</p> <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>	1 byte	4 bytes	4 bytes	4 bytes	Communication status	Input status value	Output status value	Status flag value				
1 byte	4 bytes	4 bytes	4 bytes											
Communication status	Input status value	Output status value	Status flag value											
0x42 (66)	FAS_GetMotionStatus	<p>Request the current operation progress status and its Position Table number (Frame type 0x51, 0x53, 0x54, and 0x55 are packed.)</p> <p>Sending : 0 byte Response : 21 bytes</p> <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> <p>*Actual Position value : when external encoder is connected</p>	1 byte	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	Communication status	Command position value	Actual Position value	Position Difference value	Running speed value	Current running PT number
1 byte	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes									
Communication status	Command position value	Actual Position value	Position Difference value	Running speed value	Current running PT number									

0x43 (67)	FAS_ GetAllStatus	<p>Request all data including the current running status (Frame type 0x41, and 0x42 are packed.)</p> <p>Sending : 0 byte Response : 33 bytes</p> <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>Current running PT number</td></tr></table> <p>*Actual Position value : when external encoder is connected</p>	1 byte	4 bytes	4 bytes	4 bytes	Communication status	Input status value	Output status value	Status flag value	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	Command position value	Actual position value	Position Difference value	Running speed value	Current running PT number
1 byte	4 bytes	4 bytes	4 bytes																	
Communication status	Input status value	Output status value	Status flag value																	
4 bytes	4 bytes	4 bytes	4 bytes	4 bytes																
Command position value	Actual position value	Position Difference value	Running speed value	Current running PT number																
0x50 (80)	FAS_ SetCommandPos	<p>User can set the command position value before it starts and then can check how the command position value is changed.</p> <p>Sending : 4 bytes</p> <table><tr><td>4 bytes</td></tr><tr><td>Command position setting count value</td></tr></table> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table>	4 bytes	Command position setting count value	1 byte	Communication status														
4 bytes																				
Command position setting count value																				
1 byte																				
Communication status																				
0x51 (81)	FAS_ GetCommandPos	<p>Request the command position value[pulse] being tracked.</p> <p>Sending : 0 byte Response : 5 bytes</p> <table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication status</td><td>Command position value</td></tr></table>	1 byte	4 bytes	Communication status	Command position value														
1 byte	4 bytes																			
Communication status	Command position value																			
0x52 (82)	FAS_ SetActualPos	<p>When external encoder is connected to drive, the actual position value is continuously renewed while the motor is operating. User can set the actual position value before it starts and then can check how the actual position value is changed.</p> <p>Sending : 4 bytes</p> <table><tr><td>4 bytes</td></tr><tr><td>Actual position count value</td></tr></table> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table>	4 bytes	Actual position count value	1 byte	Communication status														
4 bytes																				
Actual position count value																				
1 byte																				
Communication status																				
0x53 (83)	FAS_ GetActualPos	<p>Request the current actual position value[pulse].</p> <p>* When external encoder is connected</p> <p>Sending : 0 byte Response : 5 bytes</p> <table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication status</td><td>Actual position value</td></tr></table>	1 byte	4 bytes	Communication status	Actual position value														
1 byte	4 bytes																			
Communication status	Actual position value																			

0x54 (84)	FAS_ GetPosError	<p>Request the difference[pulse] between the command position value and the actual position value.</p> <p>* When external encoder is connected</p> <p>Sending : 0 byte Response : 5 bytes</p> <table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication status</td><td>Position difference value</td></tr></table>	1 byte	4 bytes	Communication status	Position difference value		
1 byte	4 bytes							
Communication status	Position difference value							
0x55 (85)	FAS_ GetActualVel	<p>Request the current running speed value [pps]</p> <p>Sending : 0 byte Response : 5 bytes</p> <table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication status</td><td>Speed value</td></tr></table>	1 byte	4 bytes	Communication status	Speed value		
1 byte	4 bytes							
Communication status	Speed value							
0x56 (86)	FAS_ ClearPosition	<p>User can set the command position and actual position value as '0' before it starts to operate and can check how the command position value is changed.</p> <p>Sending : 0 byte Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table> <p>*Actual Position value : when external encoder is connected</p>	1 byte	Communication status				
1 byte								
Communication status								
0x58 (88)	FAS_ MovePause	<p>To request the pause start and pause end of motor motioning.</p> <p>Sending : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>0:pause release, 1:pause start</td></tr></table> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table>	1 byte	0:pause release, 1:pause start	1 byte	Communication status		
1 byte								
0:pause release, 1:pause start								
1 byte								
Communication status								
0x60 (96)	FAS_ PosTableReadItem	<p>To read Position Table values in the RAM of the drive.</p> <p>Sending : 2 bytes</p> <table><tr><td>2 bytes</td></tr><tr><td>Readable PT number (0~255)</td></tr></table> <p>Response : 65 bytes</p> <table><tr><td>1 byte</td><td>64 bytes</td></tr><tr><td>Communication status</td><td>Relevant PT values</td></tr></table> <p>For items by each PT, refer to 「1-2-6. Position Table Item」.</p>	2 bytes	Readable PT number (0~255)	1 byte	64 bytes	Communication status	Relevant PT values
2 bytes								
Readable PT number (0~255)								
1 byte	64 bytes							
Communication status	Relevant PT values							

0x61 (97)	FAS_ PosTableWriteItem	<p>To save Position Table values to the RAM of the drive.</p> <p>Sending : 66 bytes</p> <table><tr><td>2 bytes</td><td>64 bytes</td></tr><tr><td>PT number (0-255)</td><td>Relevant PT value</td></tr></table> <p>For items by each PT, refer to 「1-2-6. Position Table Item」 .</p> <p>Response : 2 bytes</p> <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>	2 bytes	64 bytes	PT number (0-255)	Relevant PT value	1 byte	1 byte	Communication status	Command performing status (values except 0 : complete, 0: error)
2 bytes	64 bytes									
PT number (0-255)	Relevant PT value									
1 byte	1 byte									
Communication status	Command performing status (values except 0 : complete, 0: error)									
0x62 (98)	FAS_ PosTableReadROM	<p>To read all Position Table values (256 ea) in the ROM of the drive</p> <p>Sending : 0 byte</p> <p>Response : 2 bytes</p> <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>	1 byte	1 byte	Communication status	Command performing status (0 : complete, values except 0: error)				
1 byte	1 byte									
Communication status	Command performing status (0 : complete, values except 0: error)									
0x63 (99)	FAS_ PosTableWriteROM	<p>To save all Position Table value(256 ea) to the ROM of the drive.</p> <p>Sending : 0 byte</p> <p>Response : 2 bytes</p> <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>	1 byte	1 byte	Communication status	Command performing status (0 : complete, values except 0: error)				
1 byte	1 byte									
Communication status	Command performing status (0 : complete, values except 0: error)									
0x64 (100)	FAS_ PosTableRunItem	<p>To start the position table operation from the designated Position Table number</p> <p>Sending : 2 bytes</p> <table><tr><td>2 bytes</td></tr><tr><td>PT Number (0~255)</td></tr></table> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table>	2 bytes	PT Number (0~255)	1 byte	Communication status				
2 bytes										
PT Number (0~255)										
1 byte										
Communication status										
0x6A (106)	FAS_ PosTableReadOneItem	<p>To read one of Position Table values in the RAM of the drive.</p> <p>Sending: 4 byte</p> <table><tr><td>2 byte</td><td>2 byte</td></tr><tr><td>PT Number (0-255)</td><td>Offset value(0-40)</td></tr></table> <p>Refer to 「1-2-6. Position Table Item」 for Offset value</p> <p>Response: 5 byte</p> <table><tr><td>1 byte</td><td>4 byte</td></tr><tr><td>Communication status</td><td>Relevant one of PT value</td></tr></table>	2 byte	2 byte	PT Number (0-255)	Offset value(0-40)	1 byte	4 byte	Communication status	Relevant one of PT value
2 byte	2 byte									
PT Number (0-255)	Offset value(0-40)									
1 byte	4 byte									
Communication status	Relevant one of PT value									

0x6B (107)	FAS_ PosTableWriteOneItem	To save one of Position Table values to the RAM of the drive.		
		Sending: 8 byte		
		2 byte	2 byte	4 byte
		PT Number (0~255)	Offset value (0~40)	Relevant one of PT value
		Refer to 「1-2-6. Position Table Item」 for Offset value		
		Response: 2 byte		
		1 byte	1 byte	
		Communication status	Command performing status (values except 0 : complete, 0: error)	

* Frame Type '0x65' ~ '0x69' , '0x0E' ~ '0x0F' are assigned for internal use.

* PT Number : 0~255 for **Ezi-STEP-PR**
 0~63 for **Ezi-STEP-PR-MI/Ezi-STEP-ALL**

1-2-2. Parameter Lists

No.	Name	Unit	Lower Limit	Upper Limit	Default
0	Pulse per Revolution		0	15	10
1	Axis Max Speed	[pps]	1	500,000	500,000
2	Axis Start Speed	[pps]	1	35,000	1
3	Axis Acc Time	[msec]	1	9,999	100
4	Axis Dec Time	[msec]	1	9999	100
5	Speed Override	[%]	1	500	100
6	Jog Speed	[pps]	1	500,000	5,000
7	Jog Start Speed	[pps]	1	35,000	1
8	Jog Acc Dec Time	[msec]	1	9,999	100
9	Servo Alarm Logic		0	1	0
10	Servo ON Logic		0	1	0
11	Servo Alarm Reset Logic		0	1	0
12	S/W Limit Plus Value	[pulse]	-134,217,727	+134,217,727	+134,217,727
13	S/W Limit Minus Value	[pulse]	-134,217,727	+134,217,727	-134,217,727
14	S/W Limit Stop Method		0	1	1
15	H/W Limit Stop Method		0	1	1
16	Limit Sensor Logic		0	1	0
17	Org Speed	[pps]	1	500,000	5,000
18	Org Search Speed	[pps]	1	500,000	1,000
19	Org Acc Dec Time	[msec]	1	9,999	50
20	Org Method		0	2	0
21	Org Dir		0	1	0
22	Org Offset	[pulse]	-134,217,727	+134,217,727	0
23	Org Position Set	[pulse]	-134,217,727	+134,217,727	0
24	Org Sensor Logic		0	1	0
25	Stop current	[%]	20	100	50
26	Motion Dir		0	1	0
27	Limit Sensor Dir		0	1	0
28	Encoder Multiply Value		0	3	0
29	EncoderDir		0	1	0
30	Pos. Value Counting Method		0	1	0

1-2-3. Setup bit of Output pin

Here is detail description of '0x20' frame type.

This command is only applicable only to 9 signals of 'User Output 0' ~ 'User Output 8' out of 24 signal types in the control output port. The rest of 15 output signals cannot be operated by the user's disposal. When any relevant situation occurs while the drive operates, they will be indicated. 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 Output 1	0x00010000
reserved	0x00000002	reserved	0x00000200	User Output 2	0x00020000
Alarm	0x00000004	reserved	0x00000400	User Output 3	0x00040000
Moving	0x00000008	reserved	0x00000800	User Output 4	0x00080000
Acc/Dec	0x00000010	PT Output 0	0x00001000	User Output 5	0x00100000
ACK	0x00000020	PT Output 1	0x00002000	User Output 6	0x00200000
END	0x00000040	PT Output 2	0x00004000	User Output 7	0x00400000
AlarmBlink	0x00000080	User Output 0	0x00008000	User Output 8	0x00800000

【Example 1】 Sending data to turn ON the User Output 5.

4 bytes (I/O set mask value)	4 bytes (I/O clear mask value)
0x00100000	0x00000000

【Example 2】 Sending data to turn OFF the User Output 5.

4 bytes (I/O set mask value)	4 bytes (I/O clear mask value)
0x00000000	0x00100000

1-2-4. Setup bit of Input pin

Here is detail description of '0x21' frame type.

This command is only applicable to 32 signals in the control input port. User can use signals for testing 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	AlarmReset	0x00010000	JPT input 2	0x01000000
Limit-	0x00000002	PT A5	0x00000200	reserved	0x00020000	JPT Start	0x02000000
Origin	0x00000004	PT A6	0x00000400	Pause	0x00040000	User Input 0	0x04000000
Clear Position	0x00000008	PT A7	0x00000800	Org Search	0x00080000	User Input 1	0x08000000
PT A0	0x00000010	PT Start	0x00001000	Teaching	0x00100000	User Input 2	0x10000000
PT A1	0x00000020	Stop	0x00002000	E-stop	0x00200000	User Input 3	0x20000000
PT A2	0x00000040	Jog+	0x00004000	JPT input 0	0x00400000	User Input 4	0x40000000
PT A3	0x00000080	Jog-	0x00008000	JPT input 1	0x00800000	User Input 5	0x80000000

【Example 1】 Sending data to turn ON the Pause port

4 bytes (I/O set mask value)	4 bytes (I/O clear mask value)
0x00040000	0x00000000

【Example 2】 Sending data to turn OFF the Pause port

4 bytes (I/O set mask value)	4 bytes (I/O clear mask value)
0x00000000	0x00040000

1-2-5. Bit setup of Status Flag

Refer to 'EZISTEP_AXISSTATUS' structure of 'motion_define.h' of include folder.

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
reserved		0x00000020
reserved		0x00000040
FFLAG_ERRSTEPALARM	One or more error of STEP alarm(8 ea) occurs.	0x00000080
FFLAG_ERROVERCURRENT	The motor driving device is under over-current	0x00000100
FFLAG_ERROVERSPEED	The motor speed exceeded 3000[rpm].	0x00000200
FFLAG_ERRSPEED	The motor is not tracked normally by pulse input.	0x00000400
reserved		0x00000800
FFLAG_ERROVERHEAT	The internal temperature of the drive exceeds 55° C.	0x00001000
FFLAG_ERRREVPOWER	A counter electromotive force of the motor exceeds 70V.	0x00002000
FFLAG_ERRMOTORPOWER	The motor voltage is abnormal.	0x00004000
FFLAG_ERRLOWPOWER	The drive voltage is abnormal.	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
reserved		0x00080000
reserved		0x00100000
FFLAG_ALARMRESET	AlarmReset has run.	0x00200000
FFLAG_PTSTOPED	Position Table operation has been finished.	0x00400000
FFLAG_ORIGINSENSOR	The origin sensor is ON.	0x00800000
FFLAG_ZPULSE	The z-pulse signal of motor is ON status.	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 not running as Acceleration/Deceleration but as constant speed of operation.	0x80000000

1-2-6. Position Table Item

Refer to 'motion_define.h' of include files.

Name	Name of Structure Parameter	Number of Bytes	Offset position	Unit	Low Limit	Upper Limit
Position	lPosition	4 (signed)	0	[pulse]	-134217728	+134217728
Low Speed	dwStartSpd	4 (unsigned)	4	[pps]	0	500000
High Speed	dwMoveSpd	4 (unsigned)	8	[pps]	0	500000
Accel. Time	wAccelRate	2 (unsigned)	12	[msec]	1	9999
Decel. Time	wDecelRate	2 (unsigned)	14	[msec]	1	9999
Command	wCommand	2 (unsigned)	16		0	9
Wait time	wWaitTime	2 (unsigned)	18	[msec]	0	600000
Continuous Action	wContinuous	2 (unsigned)	20		0	1
Jump Table No.	wBranch	2 (unsigned)	22		0 10000	255 10255
Jump PT 0	wCond_branch0	2 (unsigned)	24		0 10000	255 10255
Jump PT 1	wCond_branch1	2 (unsigned)	26		0 10000	255 10255
Jump PT 2	wCond_branch2	2 (unsigned)	28		0 10000	255 10255
Loop Count	wLoopCount	2 (unsigned)	30		0	100
Loop Jump Table No.	wBranchAfterLoop	2 (unsigned)	32		0 10000	255 10255
PT set	wPTSet	2 (unsigned)	34		0	15
Loop Counter Clear	wLoopCountCLR	2 (unsigned)	36		0	255
Compare Position	lTriggerPos	4 (signed)	38	[pulse]	-134217728	+134217728
Compare Width	wTriggerOnTime	2 (unsigned)	42	[msec]	1	9999
Blank		20 (unsigned)	44	0x00		

For the setting method by each item, [refer to other manual 「User Manual_Position Table」](#).

Please refer to separate manual 「User Manual_Position Table」 for setting method per each time.

1-2-7. Information of Motors

First 2 digits of number and 1~2 characters indicate the motor size and length.

【Example】 56XL : Motor Flange size is 56mm and Extra large size

Other part indicates the motor manufacturer information as below.

Display	Maker
blank	JapanServo
SD	Sanyo Denki
POR	Portescap
NPM	NPM
FUL	Fulling
YK	Yunkong
MIN	Minebia
Lin	Linear Step ㄱ

1 – 3. Program Method

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

The first is generally used method with using Visual C++ language under window system of PC. Library that serviced together with Ezi-STEP Plus-R have to be used. Please refer to [「2. Library for PC Program」](#)

The second method is sending command (characters) directly to Ezi-STEP Plus-R. User has to prepare low-level protocol programming like 'Protocol Test' program and this method is applied when use higher-level control unit as like PLC.

For more programming method details, please exercise 'ProtocolTest_PlusR.exe' is serviced together with GUI.

Please refer to [「3. Protocol for PLC Program」](#).

2. Library for PC Program

2-1. Library Configuration

To use this library, C++ header file(*.h) and library file(*.lib or *.dll) are required. These files locate in "[WWFASTECHWW EziMOTION PlusR WWincludeWW](#)". And the following contents should be included in a source file for development.

```
#include "WWFASTECHWW EziMOTION PlusR WWincludeWWFAS\_EziMotionPlusR.h"
#include "WWFASTECHWW EziMOTION PlusR WWincludeWWCOMM\_Define.h"
#include "WWFASTECHWW EziMOTION PlusR WWincludeWWMOTION\_DEFINE.h"
#include "WWFASTECHWW EziMOTION PlusR WWincludeWWReturnCodes\_Define.h"
```

Also, library files are as follows:

```
"WWFASTECHWW EziMOTION PlusR WWincludeWW EziMotionPlusR.lib"
"WWFASTECHWW EziMOTION PlusR WWincludeWW EziMotionPlusR.dll"
```

A sample program source of with using these libraries locate at

["WWFASTECHWW EziMOTION PlusR WWExamplesWW"](#) folder.

(1) The following table explains values returned when each library(DLL) function is used. The user **can only check the values returned at the library(DLL) function**. Low level programming method does not support following table.

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 drive 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 Drive.

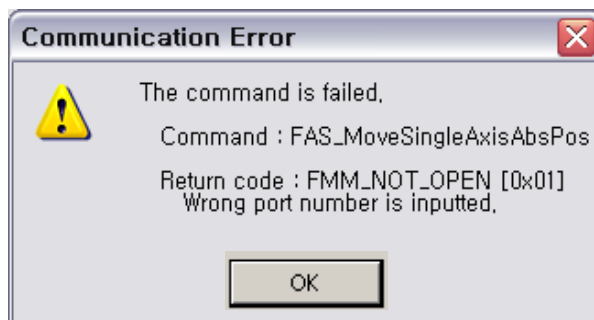
(2) The following table indicates return values included commonly in all libraries and these functions offer to check the result (communication status, running status) judged by the drive. These functions are available for using library (DLL) and protocol.

Item	Description	Returned Value	Description
Normal	FMP_OK	0	Communication has been normally performed.
Input Error	FMP_FRAMETYPEERROR	128	The drive cannot recognize the command.
	FMP_DATAERROR	129	Input data is out of the range.
Operation Error	FMP_BUSYMOTOR	133	The motor is already running or not prepared for running.
Connection Error	FMP_PACKETERROR	130	Protocol level error occurs in packet that Drive' s received.
	FMP_PACKETCRCERROR	170	CRC value is not correct in packet that Drive' s received.

2-2. Communication Status Window

Above communication status is divided by 3 groups.

(1) Communication Error



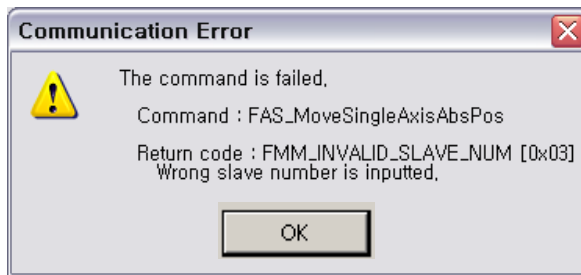
FMM_NOT_OPEN,

COM Port is not connected. (This error cannot be occurred in GUI.)



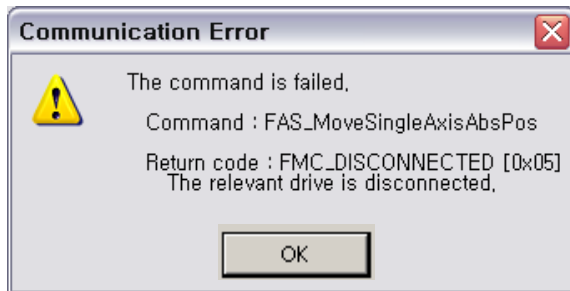
FMM_INVALID_PORT_NUM,

COM Port number does not exist. Checking the 'Device Manager' window in Window OS. (This error cannot be occurred in GUI.)



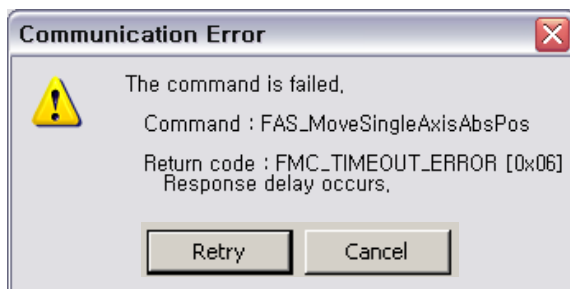
FMM_INVALID_SLAVE_NUM,

Slave number does not exist. Checking the ID value of the drive.
(This error cannot be occurred in GUI.)



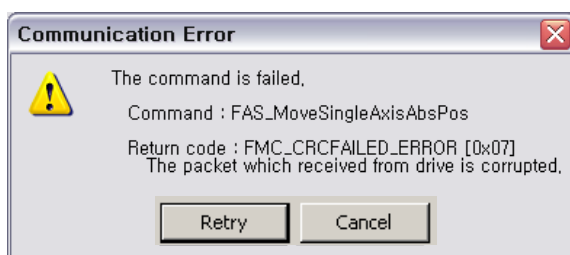
FMC_DISCONNECTED = 5,

COM Port is disconnected during communication. Checking the communication cable
Or Power of the drive.



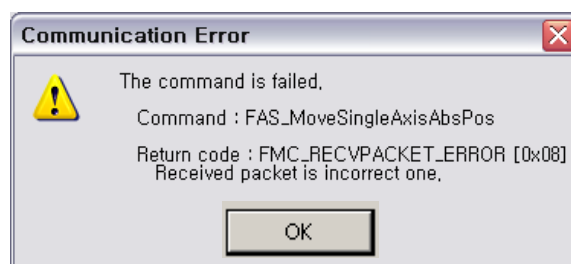
FMC_TIMEOUT_ERROR,

There is no response from the drive.



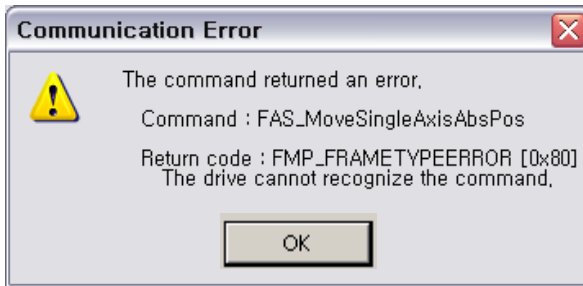
FMC_CRCFAILED_ERROR,

CRC value of communication packet from the drive is not correct. Checking the
Possibility of noise on communication cable.



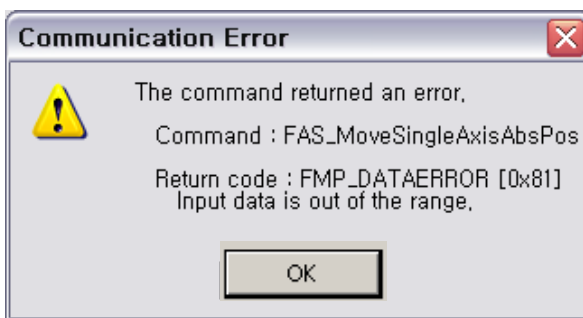
FMC_RECVPACKET_ERROR,

The length of received packet is not correct. Checking the possibility of noise on communication cable.



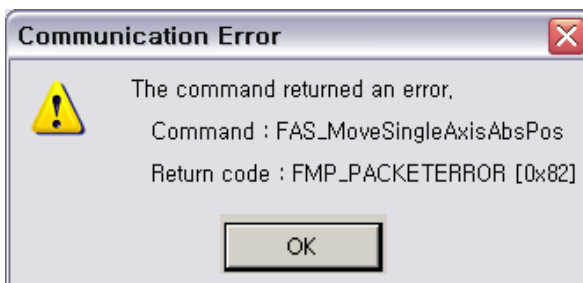
FMP_FRAMETYPEERROR = 0x80,

Drive does not recognize the command or wrong command is sent.
Checking the command value that you want to send to the drive.



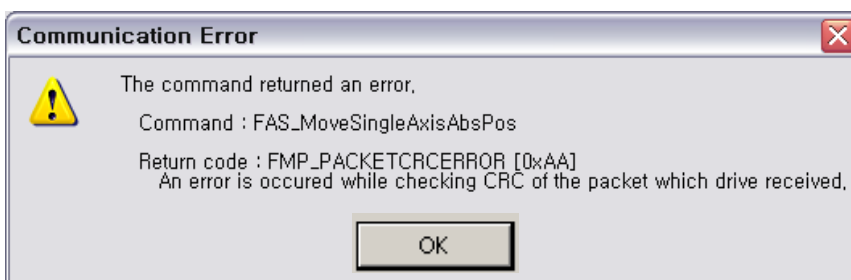
FMP_DATAERROR,

The value of the sent data is out of the proper range of the drive.
Checking the value that you want to send to the drive.



FMP_PACKETERROR,

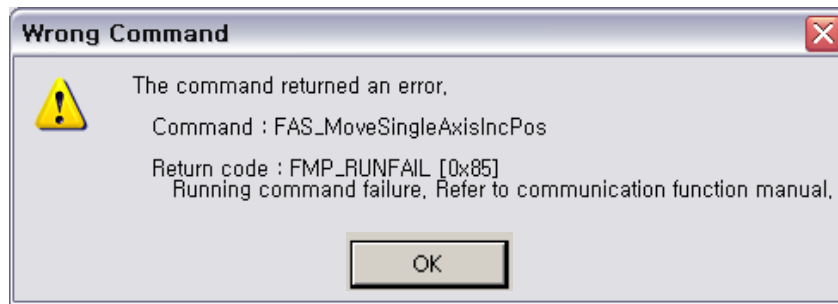
The length of received packet on drive is not correct. Checking the possibility of noise on communication cable.



FMP_PACKETCRCERROR = 0xAA,

The incorrect CRC value of packet sent to the drive. Checking the possibility of noise on communication cable.

(2) Wrong Command



FMP_RUNFAIL = 0x85,

Fail on motion command : Tried to new motion under following status.

- . The motor is already running
- . The motor is under stop command
- . Try to Z-pulse Origin without external encoder (only for Ezi-STEP)

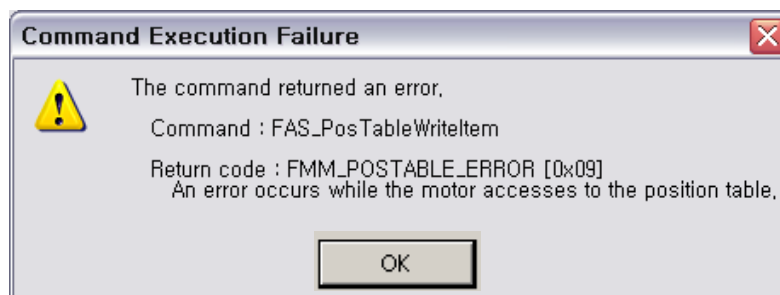


FMP_RESETFAIL,

Fail on reset command : Tried to new motion under following status.

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

(3) Command Execution Error



FMM_POSTABLE_ERROR,

The execution of DLL library for 'Position Table' is failed.

2-3. Drive Link Function

Function Name	Description
FAS_Connect	The drive tries to connect communication with the drive module: When it is successfully connected, TRUE will be returned. Otherwise, FALSE will be returned.
FAS_Close	The drive tries to disconnect communication with the drive module.
FAS_GetSlaveInfo	The drive reads drive type and program version: Drive type and version information will be returned.
FAS_GetMotorInfo	The drive reads motor type and manufacturer information: Motor type and maker information will be returned.
FAS_IsSlaveExist	Check the existence of the relevant drive: When it exists, TRUE will be returned. Otherwise, FALSE will be returned.
FAS_EnableLog	To select the communication error log function ON/OFF : 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 connection Ezi-STEP Plus-R.

Syntax

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

Parameters

nPortNo

Select a serial port number to be connected.

dwBaud

Input the Baudrate of the serial port.

Return Value

When it is successfully connected, TRUE will be returned. Otherwise, FALSE will be returned.

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-STEP 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) Wn", nPortNo, iSlaveNo);
    printf("WtType : %d Wn", nType);
    printf("WtVersion : %d Wn", 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 be disconnected

Remarks

Example

Refer to 'FAS_Connect' library.

See Also

FAS_Connect

FAS_GetSlaveInfo

To get the version information string of the relevant drive

Syntax

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

Parameters

nPortNo

Port number of relevant drive

iSlaveNo

Slave number of relevant drive

pType

Type number of relevant drive

lpBuff

Buffer pointer will get version information string

nBuffSize

Memory allocation size of lpBuff

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

Remarks

Example

Refer to 'FAS_Connect' library.

See Also

FAS_GetMotorInfo

To get the motor information string of the relevant drive

Syntax

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

Parameters

nPortNo

Port number of relevant drive

iSlaveNo

Slave number of relevant drive

pType

Type number of relevant motor

lpBuff

Buffer pointer to get version information string

nBuffSize

Memory allocation size of lpBuff

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

Remarks

Example

Refer to 'FAS_Connect' library.

See Also

FAS_IsSlaveExist

Check connection status of the drive

Syntax

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

Parameters

nPortNo

Port number of relevant drive

iSlaveNo

Slave number of relevant drive

Return Value

TRUE : The drive is connected.

FALSE : The drive 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 select 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-MOTION Plus-R DLL function used. This setup

Do not effect th other process or other program.

Log function start from 'FAS_Connect' function, the Log output is end when the 'FAS_Close' is excuted.

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcDisableLog()
{
    BYTE nPortNo = 1;

    FAS_EnableLog(FALSE);

    // 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

Folder path Character string of Log output file.

Return Value

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

Remarks

This function have to be called before 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:\\WLogs\\WW" ))) // C:\\WLogs folder exist.
    {
        // Log path does not exist.
        Return;
    }

    // All Log output is stored in C:\\WLogs 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	Save current status of parameters to the ROM: Even after the drive is powered OFF, parameters related to operating speed, acceleration/deceleration time, and origin return need to be preserved.
FAS_SetParameter	Save designated parameter to the RAM: Specific parameter is saved.
FAS_GetParameter	Read designated parameter from the RAM: Specific parameter is read.
FAS_GetROMParameter	Read designated parameter from the ROM: Specific parameter is read from the ROM.

FAS_SaveAllParameters

Save all edited parameters up to now and assigned I/O signals to the ROM area.

Syntax

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

Parameters

nPortNo

Port number of relevant drive

iSlaveNo

Slave number of relevant drive

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no drive 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 IParamVal;
    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, STEP_AXISSTARTSPEED, &IParamVal);
    if (nRtn != FMM_OK)
    {
        // Command has not been performed properly.
        // Refer to ReturnCodes_Define.h.
        _ASSERT(FALSE);
    }
    else
    {
        // Parameter value saved in Ezi-STEP Plus-R.
        printf("Parameter [before] : Start Speed = %d Wn", IParamVal);
    }
}
```

```

    // Change Axis Start Speed parameter as 200 then read it again.
    nRtn = FAS_SetParameter(nPortNo, iSlaveNo, STEP_AXISSTARTSPEED, 200);
    _ASSERT(nRtn == FMM_OK);    // You have to check if the command didn't execute
correctly.

```

```

    nRtn = FAS_GetParameter(nPortNo, iSlaveNo, STEP_AXISSTARTSPEED, &IParamVal);
    _ASSERT(nRtn == FMM_OK);
    printf("Parameter [after] : Start Speed = %d Wn", IParamVal);

```

```

    // Check the value saved in the ROM.
    nRtn = FAS_GetROMParameter(nPortNo, iSlaveNo, STEP_AXISSTARTSPEED, &IParamVal);
    _ASSERT(nRtn == FMM_OK);    // You have to check if the command didn't execute
correctly.
    printf("Parameter [ROM] : Start Speed = %d Wn", IParamVal);

```

```

    // Edit the parameter value then save it in the ROM.
    nRtn = FAS_SetParameter(nPortNo, iSlaveNo, STEP_AXISSTARTSPEED, 100);
    _ASSERT(nRtn == FMM_OK);    // You have to check 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 drive

iSlaveNo

Slave number of relevant drive

iParamNo

Parameter number to be edited

lParamValue

Parameter value to be edited

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

FMM_INVALID_PARAMETER_NUM : There is no parameter of designated iParamNo.

Remarks

The function operates for only one parameter designated.

Parameters in the drive 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 value of the drive

Syntax

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

Parameters

nPortNo

Port number of relevant drive

iSlaveNo

Slave number of relevant drive

iParamNo

Parameter number to be brought

IParamValue

Parameter values

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

FMM_INVALID_PARAMETER_NUM : There is no parameter of designated iParamNo.

Remarks

The function operates for only one parameter designated.

Parameters in the drive 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* IROMParam  
);
```

Parameters

nPortNo

Port number of relevant drive

iSlaveNo

Slave number of relevant drive

iParamNo

Parameter number to be brought

IROMParam

Parameter values saved in the ROM

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no drive 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_StepAlarmReset	Release alarm of the drive generated alarm: Troubleshoot root cause of the alarm prior to use this function.
FAS_AlarmType	Read the Alarm type of the drive.

FAS_StepAlarmReset

To send AlarmReset command

Syntax

```
int FAS_StepAlarmReset(  
    BYTE nPortNo,  
    BYTE iSlaveNo  
    BYTE bReset  
);
```

Parameters

nPortNo

Port number of relevant drive

iSlaveNo

Slave number of relevant drive

bReset

Reset command (1: reset, 0:reset release)

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

Remarks

Before sending this command, troubleshoot root cause of the alarm.

For alarm cause, [refer to 'User Manual_Text'](#) .

Two times commands are needed for clearing the alarm status.

This command have to be executed sequentially '1' and '0' for the value Of 'bReset' . If you are execute only '1' value, the motor will be 'unlock' Status.

Example

See Also

2-6. Control I/O Function

Function Name	Description
FAS_SetI0Input	To set the input signal level of the control input port : Set input signal [ON] or [OFF] status.
FAS_GetI0Input	To read the current input signal status of the control input port : The signal status returns by bit for each input signal.
FAS_SetI0Output	To set the output signal level of the control input port : Set output signal [ON] or [OFF] status.
FAS_GetI0Output	To read the current input signal status of the control output port : The signal status returns by bit for each output signal.
FAS_GetI0AssignMap	To read the pin of setting status of the CN1 port : The setting status for each 9 variable signals returns by bit to the Input and Output port.
FAS_SetI0AssignMap	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_I0AssignMapReadROM	To load the pin of setting status of CN1 port from ROM area to RAM area.

FAS_SetIOInput

To set I/O input. For more information, refer to '1-1-5. Frame Type and Data Configuration'.

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

dwIOSetMask

Input bitmask value to be set

dwIOCLRMask

Input bitmask value to be cleared

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no drive 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 & STEP_IN_BITMASK_LIMITP)
    {
        // Limit + input is ON.
    }
}
```

```

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

    // Turning ON 'Clear Position' and 'User Input 1' inputs and turning off 'Jog +' input.
    nRtn = FAS_SetIIOInput(nPortNo, iSlaveNo, STEP_IN_BITMASK_CLEARPOSITION |
STEP_IN_BITMASK_USERIN1, STEP_IN_BITMASK_PJOG);
    _ASSERT(nRtn == FMM_OK);

    // Check I/O output.
    nRtn = FAS_GetIIOOutput(nPortNo, iSlaveNo, &dwOutput);
    _ASSERT(nRtn == FMM_OK);
    if (dwOutput & STEP_OUT_BITMASK_USEROUT0)
    {
        // User Output 0 is ON.
    }

    // Turn off User Output 1 and 2 signals.
    nRtn = FAS_SetIIOOutput(nPortNo, iSlaveNo, 0, STEP_OUT_BITMASK_USEROUT1 |
STEP_OUT_BITMASK_USEROUT2);
    _ASSERT(nRtn == FMM_OK);

    // Disconnect.
    FAS_Close(nPortNo);
}

```

See Also

FAS_GetIIOInput

FAS_GetIOInput

To read I/O input values. For more information, refer to '1-1-5. Frame Type and Data Configuration'.

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

dwIOInput

Parameter pointer where input values will be saved

Return Value

FMM_OK : Command has been successfully performed.

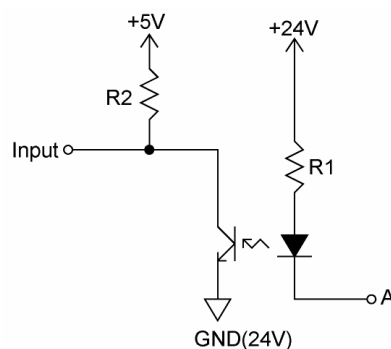
FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

Remarks

There are 12 input pins in Ezi-STEP Plus-R. The user can select and use 9 input pins of them. This function can read the input port status as 32bit. All of them are insulated by a photocoupler. (Refer to the figure.)



If voltage from an external input, is 24V at Port A, 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 '1-1-5. Frame Type and Data Configuration'.

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

dwIOSetMask

Output bitmask value to be set (ON status)

dwIOCLRMask

Output bitmask value be cleared (OFF status)

Return Value

FMM_OK : Command has been successfully performed.

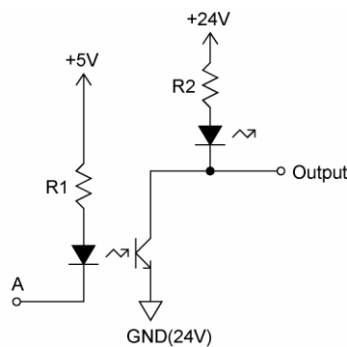
FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

Remarks

There are 10 output pins in Ezi-STEP Plus-R. The user can select and use 9 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 '1-1-5. Frame Type and Data Configuration'.

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

dwIOInput

Parameter pointer where the output value will be saved.

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no drive 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 '1-1-5. Frame Type and Data Configuration'.

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

iIOPinNo

I/O pin number to be read

dwIOLogicMask

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

bLevel

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

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

Remarks

For dwIOLogicMask, 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 E-Stop 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, STEP_IN_BITMASK_ESTOP,
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,
STEP_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 '1-1-5. Frame Type and Data Configuration'.

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

iIOPinNo

I/O Pin number to be read

dwIOLogicMask

Logic mask value to be assigned to the relevant pin

bLevel

Active Level value of the relevant logic

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

FMM_INVALID_PARAMETER_NUM : Designated iIOPinNo or dwIOLogicMask 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_IOWAssignMapReadROM

To load the status of CN1 assignment I/O setting status and signal level in ROM area

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

Remarks

Example

See Also

FAS_ GetIOWAssignMap

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 of the motor

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

lCmdPos

Command position value to be set.

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no drive 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 coordinates that customer wants.

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 of the motor

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

lActPos

Actual position value to be set.

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

Remarks

Can be used when external encoder is connected.

The user sets the encoder feedback counter value to the value what customer 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 drive

iSlaveNo

Slave number of relevant drive

lCmdPos

Parameter pointer where command position value will be saved

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no drive 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 Ezi-STEP Plus-R.
    nRtn = FAS_GetCommandPos(nPortNo, iSlaveNo, &lValue);
    _ASSERT(nRtn == FMM_OK);
    printf("CMDPOS : %d Wn", lValue);
    nRtn = FAS_GetActualVel(nPortNo, iSlaveNo, &lValue);
    _ASSERT(nRtn == FMM_OK);
    printf("ACTVEL : %d Wn", 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 drive.

iSlaveNo

Slave number of relevant drive.

lActPos

Parameter pointer where the actual position value will be saved.

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

Remarks

Can be used when external encoder is connected.

When the user decides the motor position and checks its actual position, this function is generally used.

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 drive.

iSlaveNo

Slave number of relevant drive.

lPosErr

Parameter pointer where the position error value will be saved

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

Remarks

Can be used when external encoder is connected.

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 drive.

iSlaveNo

Slave number of relevant drive.

lActVel

Parameter pointer where the actual velocity value will be saved

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no drive 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 position value of the motor to '0'

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no drive 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

2-8. Drive 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 be returned.
FAS_GetMotionStatus	To read the current running progress status and its PT number : The command position value, the actual position value, the speed value will be returned.
FAS_GetAllStatus	To read all status includes the current I/O 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 drive

FAS_GetIOAxisStatus

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

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

dwInStatus

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

dwOutStatus

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

dwAxisStatus

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

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no drive 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 drive.

iSlaveNo

Slave number of relevant drive.

lCmdPos

Parameter pointer where the command position value will be saved

lActPos

Parameter pointer where the actual position value will be saved.

lPosErr

Parameter pointer where the position error value will be saved

lActVel

Parameter pointer where the actual velocity value will be saved

wPosItemNo

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

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no drive 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 drive, the motor Axis Status, the motor motion status at one time.

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 drive.

iSlaveNo
Slave number of relevant drive.

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

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

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

lCmdPos
Parameter pointer where the command position value will be saved

lActPos
Parameter pointer where the actual position value will be saved

lPosErr
Parameter pointer where the position error value will be saved

lActVel
Parameter pointer where the actual velocity value will be saved

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

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no drive 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 '1-1-5. Frame Type and Data Configuration' .

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

dwAxisStatus

Parameter pointer where the axis status value of the relevant motor

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no drive 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	Stop the motor in running with deceleration.
FAS_EmergencyStop	Stop the motor in running immediately without deceleration.
FAS_MoveOriginSingleAxis	Start operation to return origin.
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	Changed the target absolute position value [pulse] of the motor in running.
FAS_PositionIncOverride	Changed the target incremental position value [pulse] of the motor in running.
FAS_VelocityOverride	Changed the running velocity value [pps] of the motor in running.
FAS_AllMoveStop	Stop all motors connected in same port with deceleration.
FAS_AllEmergencyStop	Stop all motors connected in same port immediately without deceleration.
FAS_AllMoveOriginSingleAxis	Start operation to return all motors in same port to origin position.
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_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.

FAS_MoveStop

To stop the motor

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no drive 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 drive.

iSlaveNo

Slave number of relevant drive.

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no drive 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 drive.

iSlaveNo

Slave number of relevant drive.

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no drive 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 value

Syntax

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

Parameters

nPortNo
Port number of relevant drive.

iSlaveNo
Slave number of relevant drive.

lAbsPos
Absolute coordinate where position to move

lVelocity
Velocity when the motor moves

Return Value

FMM_OK : Command has been successfully performed.
 FMM_NOT_OPEN : The drive has not been connected yet.
 FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
 FMM_INVALID_SLAVE_NUM : There is no drive 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;
    EZISTEP_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 status.
    nRtn = FAS_GetAxisStatus(nPortNo, iSlaveNo, &dwAxisStatus);
    _ASSERT(nRtn == FMM_OK);
    stAxisStatus.dwValue = dwAxisStatus;

    //if (dwAxisStatus & 0x00000001)
    if (stAxisStatus.FFLAG_ERRORALL)
```

```

        FAS_StepAlarmReset(nPortNo, iSlaveNo);

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

    if (dwInput & (STEP_IN_BITMASK_STOP | STEP_IN_BITMASK_PAUSE | STEP_IN_BITMASK
_ESTOP))
        FAS_SetIOInput(nPortNo, iSlaveNo, 0, STEP_IN_BITMASK_STOP | STEP_IN_
BITMASK_PAUSE | STEP_IN_BITMASK_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 drive.

iSlaveNo

Slave number of relevant drive.

lIncPos

Incremental coordinate where position to move

lVelocity

Velocity when the motor moves

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no drive 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 drive.

iSlaveNo

Slave number of relevant drive.

lVelocity

Velocity when the motor moves

iLimitDir

Limit direction of the motor moves (0: -Limit, 1: +Limit)

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no drive 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 available for Jog motion.

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

lVelocity

Velocity when the motor moves

iVelDir

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

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no drive 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 drive.

iSlaveNo

Slave number of relevant drive.

lOverridePos

Absolute coordinate position value to be changed

Return Value

FMM_OK : Command has been successfully performed.

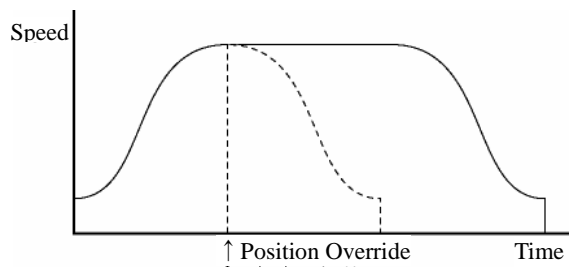
FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

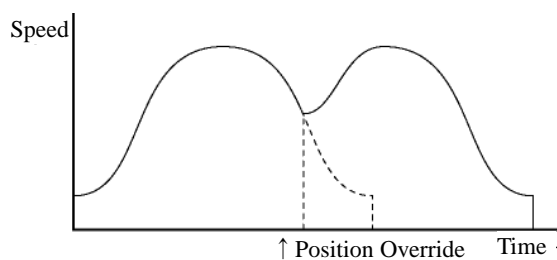
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

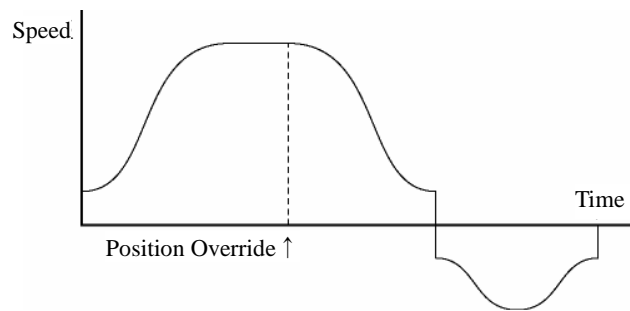
- 1) If the target position is set to the farther coordinate than the original target position while the motor moves under acceleration or constant velocity, the motor moves to the velocity pattern until then and stops at the target position.



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

iSlaveNo

Slave number of relevant drive.

lOverridePos

Incremental coordinate position value to be changed

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

Remarks

Refer to 'FAS_PositionAbsOverride' library.

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 IVelocity
);
```

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

IVelocity

Velocity to be changed in [pps]

Return Value

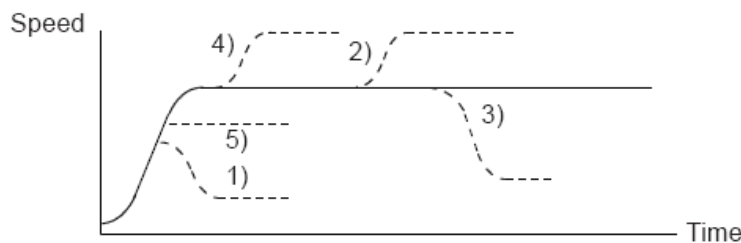
FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

Remarks



- 1) In case of $((\text{change speed}) < (\text{speed before change}))$, the motor reaches to the change speed through acceleration/deceleration using a new velocity pattern.
- 5) In case of $((\text{change speed}) \geq (\text{speed before change}))$, the motor reaches to the change speed through acceleration/deceleration without any new velocity pattern.
- 4) The motor reaches to the 'speed before change' without change of the velocity pattern and then it reaches to 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_AIIMoveStop

To stop all motors that connected in same port.

Syntax

```
int FAS_AIIMoveStop(  
    BYTE nPortNo  
);
```

Parameters

nPortNo

Port number of relevant drive.

Return Value

No response

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_AllEmergencyStop

To stop all motors that connected in same port without deceleration

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

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_AIIMoveOriginSingleAxis

To search the origin of system for all motor those are connected in same port. For more information, refer to [‘User Manual_Text 9.3 Origin Return’](#) .

Syntax

```
int FAS_AIIMoveOriginSingleAxis(  
    BYTE nPortNo  
);
```

Parameters

nPortNo

Port number of relevant drive.

Return Value

No response

Remarks

Example

Refer to ‘FAS_MoveSingleAxisAbsPos’ library.

See Also

FAS_AllMoveSingleAxisAbsPos

To move all motors that connected in same port to the absolute coordinate

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

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_AIIMoveSingleAxisIncPos

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

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

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_MoveSingleAxisAbsPosEx

To move the motor to the absolute coordinate

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

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 drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no drive 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;
    EZISTEP_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
    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 motioning 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 motioning 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

Syntax

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

Parameters

nPortNo
Port number of relevant drive.

iSlaveNo
Slave number of relevant drive.

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 drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no drive 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_MoveSingleAxisIncPos(
    BYTE nPortNo,
    BYTE iSlaveNo,
    DWORD lVelocity,
    int iVelDir,
    VELOCITY_OPTION_EX* lpExOption
);
```

Parameters

nPortNo
Port number of relevant drive.

iSlaveNo
Slave number of relevant drive.

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 drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

FMM_INVALID_SLAVE_NUM : There is no drive 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	To read items of RAM area in the specific position table
FAS_PosTableWriteItem	To save specific position table to RAM area
FAS_PosTableWriteROM	To save all of position table values to ROM area : Total 256 PT values are saved.
FAS_PosTableReadROM	To read position table values in ROM area : Total 256 PT values are read.
FAS_PosTableRunItem	The motor starts to run from the designated position table in sequence.
FAS_PosTableReadOneItem	To read items of RAM area in the specific one item of position table
FAS_PosTableWriteOneItem	To save specific item of specific position table to RAM area

FAS_PosTableReadItem

To read a specific item in the position table

Syntax

```
int FAS_PosTableReadItem(
    BYTE nPortNo,
    BYTE iSlaveNo,
    WORD wItemNo,
    LPITEM_NODE lpItem
);
```

Parameters

nPortNo
Port number of relevant drive.

iSlaveNo
Slave number of relevant drive.

wItemNo
Item number to be read

lpItem
Item structure pointer where item value is saved

Return Value

FMM_OK : Command has been successfully performed.
 FMM_NOT_OPEN : The drive has not been connected yet.
 FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
 FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.
 FMM_INVALID_PARAMETER_NUM : wItemNo is out of range.

Remarks

Example

```
#include "FAS_EziMOTIONPlusR.h"

void funcPosTable()
{
    BYTE nPortNo = 1; // COMM Port Number
    BYTE iSlaveNo = 0; // Slave No (0 ~ 15)
    WORD wItemNo;
    ITEM_NODE nodeItem;
    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;
    }

    // Read No.20 Position table value and edit the position value.
    wItemNo = 20;
    nRtn = FAS_PosTableReadItem(nPortNo, iSlaveNo, wItemNo, &nodeItem);
    _ASSERT(nRtn == FMM_OK);

    nodeItem.lPosition = 260000; // Change the position value to 260000.
    nodeItem.wBranch = 23;      // Set next command to 23.
```

```
        nodeItem.wContinuous = 1;           // Next command should be connected without
deceleration.

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

        // Call the value in the ROM regardless of edited position table data.
        nRtn = FAS_PosTableReadROM(nPortNo, iSlaveNo);
        _ASSERT(nRtn == FMM_OK);

        // Save edited position table data in the ROM.
        nRtn = FAS_PosTableWriteROM(nPortNo, iSlaveNo);
        _ASSERT(nRtn == FMM_OK);

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

See Also

FAS_PosTableWriteItem

FAS_PosTableWriteItem

To edit specific items in the position table

Syntax

```
int FAS_PosTableWriteItem(  
    BYTE nPortNo,  
    BYTE iSlaveNo,  
    WORD wItemNo,  
    LPITEM_NODE lpItem  
);
```

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

wItemNo

Item number to be edited

lpItem

Item structure pointer to be edited

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

FMC_POSTABLE_ERROR : An error occurs while position table is being written.

FMM_INVALID_PARAMETER_NUM : wItemNo is out of range.

Remarks

Position Table data is saved to RAM / ROM area. This function activates to save data to RAM area. When power is off, data is deleted.

Example

See Also

FAS_PosTableReadItem

FAS_PosTableWriteROM

To save all current position table items to ROM area

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

FMC_POSTABLE_ERROR : An error occurs while position table is being saved.

Remarks

Position table data is saved to RAM / ROM area. This function activates to save data to ROM area. Even though power is off, data is preserved.

Example

See Also

FAS_PosTableReadROM

FAS_PosTableReadROM

To read position table items being saved in ROM area

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

FMC_POSTABLE_ERROR : An error occurs while position table is being read.

Remarks

Example

See Also

FAS_PosTableWriteROM

FAS_PosTableRunItem

To perform command from a specific item in the position table

Syntax

```
int FAS_PosTableRunItem(  
    BYTE nPortNo,  
    BYTE iSlaveNo,  
    WORD wItemNo  
);
```

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

wItemNo

Item number to start motion

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

FMM_INVALID_PARAMETER_NUM : wItemNo is out of range.

Remarks

Example

See Also

FAS_GetAllStatus

FAS_MoveStop

FAS_EmergencyStop

FAS_PosTableReadOneItem

To read specific item in the specific position table

Syntax

```
int FAS_PosTableReadOneItem(  
    BYTE nPortNo,  
    BYTE iSlaveNo,  
    WORD wItemNo,  
    WORD wOffset,  
    long* lPosItemVal  
);
```

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

wItemNo

Item number to be read

wOffset

offset value which will be read from PT items. (Refer to [‘1-2-6. Position Table Item’](#))

lPosItemVal

Parameter pointer where PT item data value will be saved

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

FMM_INVALID_PARAMETER_NUM : wItemNo is out of range.

Remarks

Example

See Also

FAS_PosTableWriteOneItem

FAS_PosTableWriteOneItem

To edit specific item in the specific position table

Syntax

```
int FAS_PosTableWriteOneItem(  
    BYTE nPortNo,  
    BYTE iSlaveNo,  
    WORD wItemNo,  
    WORD wOffset,  
    long lPosItemVal  
);
```

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

wItemNo

Item number to be edited

wOffset

offset value which will be saved from PT items. (Refer to [‘1-2-6. Position Table Item’](#))

lPosItemVal

PT item data value to be set

Return Value

FMM_OK : Command has been successfully performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

FMC_POSTABLE_ERROR : An error occurs while position table is being written.

FMM_INVALID_PARAMETER_NUM : wItemNo is out of range.

Remarks

Example

See Also

FAS_PosTableReadOneItem

2–11. Other Control Function

Function Name	Description
FAS_TriggerOutput_RunA	To Start/Stop command for 'Compare Out' signal
FAS_TriggerOutput_Status	To check if the trigger output pulse 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 drive.

iSlaveNo
Slave number of relevant drive.

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

long lStartPos
Output start position [pulse]

DWORD dwPeriod
Period of output signal [pulse]

DWORD dwPulseTime
Width of output signal [msec]

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

FMM_INVALID_PARAMETER_NUM : wItemNo is out of range.

Remarks

Example

See Also

FAS_TriggerOutput_Status

FAS_TriggerOutput_Status

To check if the trigger output is working or not.

Syntax

```
int FAS_TriggerOutput_Status(  
    BYTE nPortNo,  
    BYTE iSlaveNo,  
    BYTE* bTriggerStatus  
);
```

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

bTriggerStatus

Current status of signal output.

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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


Remarks

Example

See Also

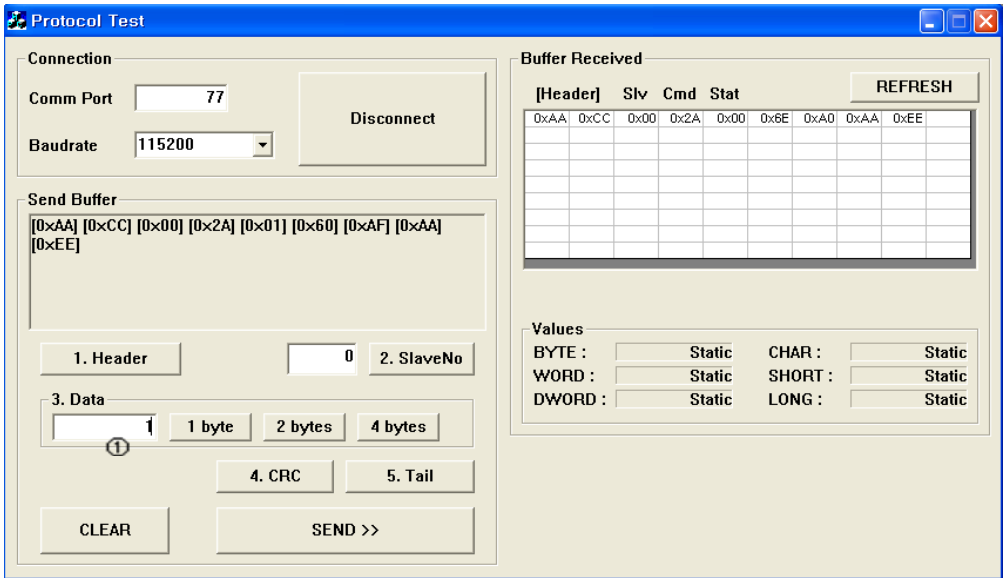
FAS_TriggerOutput_RunA

3. Protocol for PLC Program

Next window activates when you click  icon in User Program(GUI) installed folder.

Next test procedure will help you to understand the protocol programming.

- (1) Servo ON/OFF command purpose of command
 - * In case of Ezi-STEP Plus-R : Jump to next step('(2)Motion Command'), because the motor is ready to move status after Power ON.



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

- 1) Select 'Comm Port' number and 'Baudrate' , and click 'Connect' button.
- 2) Header: Click 'Header' and you can see '[0xAA][0xCC]' on 'Send Buffer' window.
- 3) Slave ID : Insert your connected slave number(above example is '0') and click 'SlaveNo' .
- 4) Frame type : Select 'Frame type' .
You can find next table information in '1-2-1. Frame Type and Data Configuration' on UserManual(Ezi-SERVO Plus-R)_Communication Function about Servo ON/OFF command.

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

- Insert '42' in ❶ area and click '1 byte' because the size of Frame Type is 1 byte.
- 5) Data: To make Servo ON status, the data is '1' so insert '1' in ❶ area and click '1 byte' .
- 6) CRC: Click 'CRC' and the automatically calculated result value (2 bytes) is displayed on 'Send Buffer' window.

- 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-SERVO Plus-R.
You can check the motor torque and LED flash for Servo ON status.
- 9) After sending command, you can check the answering information from Ezi-SERVO Plus-R on 'Buffer Received' window.

(2) Motion command purpose of command

The screenshot shows the 'Protocol Test' window with the following sections:

- Connection:** Comm Port: 77, Baudrate: 115200, Disconnect button.
- Send Buffer:**
 - Header: 1. Header (0), 2. SlaveNo (0)
 - 3. Data: 5000, 1 byte, 2 bytes, 4 bytes (selected)
 - 4. CRC, 5. Tail
 - CLEAR, SEND >> buttons
- Buffer Received:**
 - Table with columns: [Header], Slv, Cmd, Stat, and a REFRESH button.
 - Table content:

[Header]	Slv	Cmd	Stat
0xAA	0xCC	0x00	0x35
0x00	0x00	0x66	0x90
0xAA	0xEE		
- Values:**
 - BYTE: Static, CHAR: Static
 - WORD: Static, SHORT: Static
 - DWORD: Static, LONG: Static

- 1) Header
- 2) Slave No.
- 3) Frame type: insert '53' in 1 byte size for 'Incremental Move' command.
- 4) Data (Position value): insert '10000' and click '4byte'.
- 5) Data (Running speed): insert '5000' and click '4 byte'.
- 6) CRC
- 7) Tail
- 8) Send: When parameter sets as 'default' value, motor rotates as one revolution. '53' command is incremental move command so once click 'Send', motor will rotate again as same distance.

(3) PLC Programming

In 'Protocol test GUI' automatically calculate the 'Byte stuffing' and 'CRC' data. For protocol programming in PLC, you have to add the function of 'Byte stuffing' and 'CRC' calculation.

For 'Byte stuffing' refer to '[1-1-2. RS-485 Communication Protocol](#)' and for 'CRC' refer to '[1-1-3. CRC Calculation Example](#)' on UserManual(Ezi-STEP Plus-R)_Communication Function.

**FASTECH Co., Ltd.**

Rm #1202, Bucheon Technopark 401 Dong, Yakdae-dong,
Wonmi-Gu, Bucheon-si, Gyeonggi-do, Rep. Of Korea (Zip:420-734)
TEL : 82-32-234-6300, 6301 FAX : 82-32-234-6302
Email : fastech@fastech.co.kr Homepage : www.fastech.co.kr

- Please note that the specifications are subject to change without notice due to product improvements.

© Copyright 2008 FASTECH Co.,Ltd.

All Rights Reserved. Dec 17, 2017 rev.08.05.029