

# **Ezi-SERVO<sup>®</sup> ALL**

Closed Loop Stepping System



## User Manual

Communication Function\_Ver6

( Rev.01.01.04 )



## - Table of Contents -

<b>1. Communication Protocols .....</b>	<b>6</b>
<b>1-1. Communication Functions .....</b>	<b>6</b>
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
<b>1-2. Structure of Frame type(Ver6) .....</b>	<b>11</b>
1-2-1. Frame type and Data Configuration .....	11
1-2-2. Parameter Lists .....	25
1-2-3. Bit setup of Output pin .....	26
1-2-4. Bit setup of Input pin .....	26
1-2-5. Bit setup of Status Flag .....	27
1-2-6. Position Table Item .....	29
1-2-7. Information of Motors .....	29
<b>1-3. Program Method.....</b>	<b>30</b>
<b>2. Library for PC Program(Ver6) .....</b>	<b>31</b>
<b>2-1. Library Configuration .....</b>	<b>31</b>
<b>2-2. Communication Status Window .....</b>	<b>32</b>
<b>2-3. Drive Link Function .....</b>	<b>37</b>
FAS_Connect .....	38
FAS_Close .....	40
FAS_GetSlaveInfo .....	41
FAS_GetMotorInfo .....	42
FAS_IsSlaveExist .....	43

<b>2-4. Parameter Control Function .....</b>	<b>44</b>
FAS_SaveAllParameters .....	45
FAS_SetParameter .....	47
FAS_GetParameter .....	48
FAS_GetROMParameter .....	49
<b>2-5. Servo Control Function .....</b>	<b>50</b>
FAS_ServoEnable .....	51
FAS_ServoAlarmReset .....	53
<b>2-6. Control I/O Function .....</b>	<b>54</b>
FAS_SetIOInput .....	55
FAS_GetIOInput .....	57
FAS_SetIOOutput .....	58
FAS_GetIOOutput .....	59
FAS_GetIOAssignMap .....	60
FAS_SetIOAssignMap .....	62
FAS_IOAssignMapReadROM .....	63
<b>2-7. Position Control Function .....</b>	<b>64</b>
FAS_SetCommandPos .....	65
FAS_SetActualPos .....	67
FAS_GetCommandPos .....	68
FAS_GetActualPos .....	70
FAS_GetPosError .....	71
FAS_GetActualVel .....	72
FAS_ClearPosition .....	73
<b>2-8. Drive Status Control Function .....</b>	<b>74</b>
FAS_GetIOAxisStatus .....	75
FAS_GetMotionStatus .....	76
FAS_GetAllStatus .....	77
FAS_GetAxisStatus .....	78


<b>2-9. Running Control Function .....</b>	<b>79</b>
FAS_MoveStop.....	80
FAS_EmergencyStop.....	81
FAS_MoveOriginSingleAxis.....	82
FAS_MoveSingleAxisAbsPos.....	83
FAS_MoveSingleAxisIncPos.....	85
FAS_MoveToLimit.....	86
FAS_MoveVelocity.....	87
FAS_PositionAbsOverride.....	88
FAS_PositionIncOverride.....	90
FAS_VelocityOverride.....	91
FAS_AllMoveStop.....	92
FAS_AllEmergencyStop.....	93
FAS_AllMoveOriginSingleAxis.....	94
FAS_AllMoveSingleAxisAbsPos.....	95
FAS_AllMoveSingleAxisIncPos.....	96
FAS_MoveLinearAbsPos / FAS_MoveLinearIncPos.....	97
FAS_MoveSingleAxisAbsPosEx.....	98
FAS_MoveSingleAxisIncPosEx.....	100
FAS_MoveVelocityEx.....	101
<b>2-10. Position Table Control Function .....</b>	<b>103</b>
FAS_PosTableReadItem.....	104
FAS_PosTableWriteItem.....	106
FAS_PosTableWriteROM.....	107
FAS_PosTableReadROM.....	108
FAS_PosTableRunItem.....	109
FAS_PosTableReadOneItem.....	110
FAS_PosTableWriteOneItem.....	111
<b>2-11. Other Control Function .....</b>	<b>112</b>
FAS_TriggerOutput_RunA.....	113

FAS_TriggerOutput_Status.....	114
FAS_MovePush.....	115
FAS_GetPushStatus.....	117
<b>3. Protocol for PLC Program .....</b>	<b>118</b>

# 1. Communication Protocols

## 1-1. Communication Functions

Ezi-SERVO ALL can control up to 16 axes by multidrop link at RS-485(two-wire).

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

### 1-1-1. Communication Specifications

Specification	RS-485
Communication Type	Asynchronous
	Half-duplex
Baud Rate [bps]	19200, 38400, 57600, 115200, 230400, 460800, 921600
Data Type	8bit Binary
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 Axes	16 axes (No. 0~F)

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

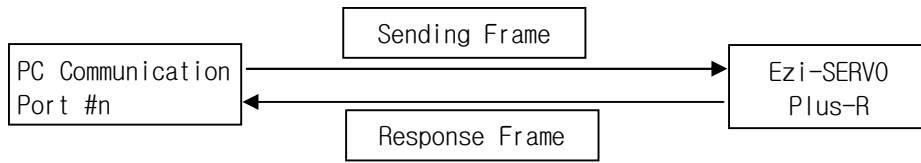
There are 2 kinds of program version for SERVO ALL. This manual support for **Version 6** level.

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 check in  
'About Plus-R GUI...' menu in 'Help' menu.



1) Overview of communication FRAME



2) Basic structure of Frame

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

- ① 0xAA : Delimited byte
- ② 0xAA 0xCC : Displays that the Frame locates in header.
- ③ 0xAA 0xEE : Displays that the Frame locates in tail.
- ④ If any of the Frame data is '0xAA' , '0xAA' should be added right after it. (byte stuffing)
- ⑤ If any data following '0xAA' is not '0xAA' , '0xCC' or '0xEE' , it displays that an error has occurred.

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

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

- ① Slave ID : Dive module number (0~15) connected to the PC communication port.
- ② Frame type : To designate command type of relevant frames. For the command type, refer to 「Frame Type and Data Configuration」 section.
- ③ Data : Data structure and length is set according to Frame type. For more information, refer to 「Frame Type and Data Configuration」 section.
- ④ CRC : To check that an error occurs during communication, '0xA001' of a polynomial factor in **CRC(Cyclic Redundancy Check)** is used. Or 'X<sup>16</sup>+X<sup>15</sup>+X<sup>2</sup>+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, 0XEBO1, 0X2BC0, 0X2A80, 0XEA41,

```

```

0XEE01, 0X2EC0, 0X2F80, 0XEF41, 0X2D00, 0XEDC1, 0XEC81, 0X2C40,
0XE401, 0X24C0, 0X2580, 0XE541, 0X2700, 0XE7C1, 0XE681, 0X2640,
0X2200, 0XE2C1, 0XE381, 0X2340, 0XE101, 0X21C0, 0X2080, 0XE041,
0XA001, 0X60C0, 0X6180, 0XA141, 0X6300, 0XA3C1, 0XA281, 0X6240,
0X6600, 0XA6C1, 0XA781, 0X6740, 0XA501, 0X65C0, 0X6480, 0XA441,
0X6C00, 0XACC1, 0XAD81, 0X6D40, 0XAF01, 0X6FC0, 0X6E80, 0XAE41,
0XAA01, 0X6AC0, 0X6B80, 0XAB41, 0X6900, 0XA9C1, 0XA881, 0X6840,
0X7800, 0XB8C1, 0XB981, 0X7940, 0XBB01, 0X7BC0, 0X7A80, 0XBA41,
0XBE01, 0X7EC0, 0X7F80, 0XBF41, 0X7D00, 0XBDC1, 0XBC81, 0X7C40,
0XB401, 0X74C0, 0X7580, 0XB541, 0X7700, 0XB7C1, 0XB681, 0X7640,
0X7200, 0XB2C1, 0XB381, 0X7340, 0XB101, 0X71C0, 0X7080, 0XB041,
0X5000, 0X90C1, 0X9181, 0X5140, 0X9301, 0X53C0, 0X5280, 0X9241,
0X9601, 0X56C0, 0X5780, 0X9741, 0X5500, 0X95C1, 0X9481, 0X5440,
0X9C01, 0X5CC0, 0X5D80, 0X9D41, 0X5F00, 0X9FC1, 0X9E81, 0X5E40,
0X5A00, 0X9AC1, 0X9B81, 0X5B40, 0X9901, 0X59C0, 0X5880, 0X9841,
0X8801, 0X48C0, 0X4980, 0X8941, 0X4B00, 0X8BC1, 0X8A81, 0X4A40,
0X4E00, 0X8EC1, 0X8F81, 0X4F40, 0X8D01, 0X4DC0, 0X4C80, 0X8C41,
0X4400, 0X84C1, 0X8581, 0X4540, 0X8701, 0X47C0, 0X4680, 0X8641,
0X8201, 0X42C0, 0X4380, 0X8341, 0X4100, 0X81C1, 0X8081, 0X4040
};

```

```

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

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

```

## 2) 'X16+X15+X2+1' of CRC-16-IBM

```

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

    /* Initialize CRC */
    wCrc = 0xffff;

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

        /* Loop through all 8 data bits */

```



```

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

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

```

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


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

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

- ① Slave ID : Same to sending Frame.  
(When this is not same to sending data, it should be recognized as the error status.)
- ② Frame type : Same to sending Frame.  
(When this is not same to sending data, it should be recognized as the error status.)
- ③ Data : When simple executive instructions are sent, this data cannot be read. However, in case of response, 1 byte is added to display the communication status (error / normal).

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

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

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

## 1-2. Structure of Frame type(Ver6)

### 1-2-1. Frame type and Data Configuration

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

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 border="1"> <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 : 1 : Ezi-SERVO Plus-R ST    20 : Ezi-STEP Plus-R ST 50 : Ezi- SERVO Plus-R MINI 30 : Ezi-SERVO ALL</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 maker information are required.</p> <p>Sending : 0 byte Response : 1~248 bytes</p> <table border="1"> <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 「1-2-7.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>Current setting parameters &amp; assign of IO signals are saved in the ROM of the drive. Even though the drive is powered off, saving these must be possible.</p> <p>Values set at 'FAS_SetParameter' &amp; 'FAS_SetIOAssignMap' are saved together.</p> <p>Sending : 0 byte Response : 1 byte</p> <table border="1"> <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 read.</p> <p>Sending : 1 byte</p> <table border="1"> <tr> <td>1 byte</td> </tr> <tr> <td>Parameter number (0~31)</td> </tr> </table> <p>Response : 5 bytes</p> <table border="1"> <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 (0~31)	1 byte	4 bytes	Communication status	Parameter value
1 byte								
Parameter number (0~31)								
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 border="1" data-bbox="544 248 1153 327"> <tr> <td>1 byte</td> <td>4 bytes</td> </tr> <tr> <td>Parameter number (0~31)</td> <td>Parameter value</td> </tr> </table> <p>Response : 1 byte</p> <table border="1" data-bbox="544 405 860 483"> <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 (0~31)	Parameter value	1 byte	Communication status
1 byte	4 bytes							
Parameter number (0~31)	Parameter value							
1 byte								
Communication status								
0x13 (19)	FAS_ GetParameter	<p>Specific parameter values in the RAM are read</p> <p>Sending : 1 byte</p> <table border="1" data-bbox="544 669 831 786"> <tr> <td>1 byte</td> </tr> <tr> <td>Parameter number (0~32)</td> </tr> </table> <p>Response : 5 bytes</p> <table border="1" data-bbox="544 842 1048 920"> <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 (0~32)	1 byte	4 bytes	Communication status	Parameter value
1 byte								
Parameter number (0~32)								
1 byte	4 bytes							
Communication status	Parameter value							
0x20 (32)	FAS_ SetI0Output	<p>Output signal level of the control output port is set.</p> <p>Sending : 8 bytes</p> <table border="1" data-bbox="544 1099 1067 1160"> <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 border="1" data-bbox="544 1429 837 1489"> <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>Input signal level of the control input port is set.</p> <p>Sending : 8 bytes</p> <table border="1" data-bbox="544 1630 1032 1691"> <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 border="1" data-bbox="544 1960 828 2020"> <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 read.</p> <p>Sending : 0 byte Response : 5 byte</p> <table border="1" data-bbox="544 286 1090 342"> <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 read.</p> <p>Sending : 0 byte Response : 5 byte</p> <table border="1" data-bbox="544 573 1051 656"> <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>To assign control I/O signals to the pin of CN1 port and set the signal level. By running 'FAS_SaveAllParameters', you can save the setting value to the ROM.</p> <p>Sending : 6 bytes</p> <table border="1" data-bbox="544 925 1219 981"> <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> <ul style="list-style-type: none"> <li>◆ I/O number: '0~11' corresponds to 'Limit+, Limit-, Org, IN1, ..., IN9' respectively, and '12~22' corresponds to 'COMP, OUT1, ..., OUT9' respectively.</li> <li>◆ I/O pin masking data: Refer to 「1-2-4. Bit setup of Input Pin」.</li> <li>◆ Level Setting: 0:Active Low, 1:Active High</li> </ul> <p>Response : 1 byte</p> <table border="1" data-bbox="544 1223 829 1279"> <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>Pin setting status of CN1 port is read from RAM area.</p> <p>Sending : 1 byte</p> <table border="1" data-bbox="544 1386 798 1464"> <tr> <td>1 byte</td> </tr> <tr> <td>I/O number</td> </tr> </table> <ul style="list-style-type: none"> <li>◆ I/O number: '0~11' corresponds to 'Limit+, Limit-, Org, IN1, ..., IN9' respectively, and '12~22' corresponds to 'COMP, OUT1, ..., OUT9' respectively.</li> </ul> <p>Response : 6 bytes</p> <table border="1" data-bbox="544 1666 1321 1744"> <tr> <td>1 byte</td> <td>4 bytes</td> <td>1 byte</td> </tr> <tr> <td>Communication status</td> <td>I/O 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	I/O pin masking status	Level status
1 byte										
I/O number										
1 byte	4 bytes	1 byte								
Communication status	I/O pin masking status	Level status								

0x26 (38)	FAS_ IOAssignMapReadROM	<p>Pin setting status of CN1 port is loaded to RAM from ROM area.</p> <p>Sending : 0 byte</p> <p>Response : 2 bytes</p> <table border="1" data-bbox="544 315 1321 427"> <tr> <td data-bbox="544 315 821 353">1 byte</td> <td data-bbox="821 315 1321 353">1 byte</td> </tr> <tr> <td data-bbox="544 353 821 427">Communication status</td> <td data-bbox="821 353 1321 427">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 border="1" data-bbox="544 573 1350 689"> <tr> <td data-bbox="544 573 837 611">1 byte</td> <td data-bbox="837 573 1157 611">4 bytes</td> <td data-bbox="1157 573 1350 611">4 byte</td> </tr> <tr> <td data-bbox="544 611 837 689">Output start/stop (1:start 0:stop)</td> <td data-bbox="837 611 1157 689">Pulse start position [pulse]</td> <td data-bbox="1157 611 1350 689">Pulse period [pulse]</td> </tr> </table> <table border="1" data-bbox="544 725 1294 842"> <tr> <td data-bbox="544 725 742 763">4 byte</td> <td data-bbox="742 725 1061 763">1 bytes</td> <td data-bbox="1061 725 1294 763">4 byte</td> </tr> <tr> <td data-bbox="544 763 742 842">Pulse width [msec]</td> <td data-bbox="742 763 1061 842">Output pin number (fix to 0)</td> <td data-bbox="1061 763 1294 842">spare</td> </tr> </table> <p>Response : 2 byte</p> <table border="1" data-bbox="544 931 1326 1039"> <tr> <td data-bbox="544 931 831 969">1 byte</td> <td data-bbox="831 931 1326 969">1 byte</td> </tr> <tr> <td data-bbox="544 969 831 1039">Communication status</td> <td data-bbox="831 969 1326 1039">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 border="1" data-bbox="544 1234 1326 1312"> <tr> <td data-bbox="544 1234 831 1272">1 byte</td> <td data-bbox="831 1234 1326 1272">1 bytes</td> </tr> <tr> <td data-bbox="544 1272 831 1312">Communication status</td> <td data-bbox="831 1272 1326 1312">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_ ServoEnable	<p>Servo ON/OFF status is set.</p> <p>Sending : 1 byte</p> <table border="1" data-bbox="544 1480 799 1559"> <tr> <td data-bbox="544 1480 799 1518">1 byte</td> </tr> <tr> <td data-bbox="544 1518 799 1559">0:OFF, 1:ON</td> </tr> </table> <p>Response : 1 byte</p> <table border="1" data-bbox="544 1626 823 1704"> <tr> <td data-bbox="544 1626 823 1664">1 byte</td> </tr> <tr> <td data-bbox="544 1664 823 1704">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																		
0x2B (43)	FAS_ ServoAlarmReset	<p>Servo alarm status is reset.</p> <p>Sending : 0 byte</p> <p>Response : 1 byte</p> <table border="1" data-bbox="544 1850 834 1924"> <tr> <td data-bbox="544 1850 834 1888">1 byte</td> </tr> <tr> <td data-bbox="544 1888 834 1924">Communication status</td> </tr> </table>	1 byte	Communication status														
1 byte																		
Communication status																		

0x2E (46)	FAS_ ServoAlarmtype	<p>To request the Alarm type</p> <p>Sending: 0 byte Response: 2 byte</p> <table border="1" data-bbox="544 280 1114 358"> <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) OverLoad(4) OverTemperature(5) BackEMF(6) MotorConnect(7) EncoderConnect(8) MotorPower(9) Inposition(10) SystemHalt(11) ROMdevice(12) OverInputVoltage(14) Position Overflow(15)</p>	1 byte	1 bytes	Communication status	Alarm type ( 1~ )		
1 byte	1 bytes							
Communication status	Alarm type ( 1~ )							
0x31 (49)	FAS_ MoveStop	<p>To request to stop running the motor</p> <p>Sending : 0 byte Response : 1 byte</p> <table border="1" data-bbox="544 788 842 866"> <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	<p>To request the running motor to stop emergently</p> <p>Sending : 0 byte Response : 1 byte</p> <table border="1" data-bbox="544 1075 847 1153"> <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	<p>To request the motor to return to the origin at the current setting parameter condition</p> <p>Sending : 0 byte Response : 1 byte</p> <table border="1" data-bbox="544 1386 826 1456"> <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	<p>To request the motor to move its position as much as the absolute value[pulse]</p> <p>Sending : 8 bytes</p> <table border="1" data-bbox="544 1612 1099 1682"> <tr> <td>4 bytes</td> <td>4 bytes</td> </tr> <tr> <td>Absolute position value</td> <td>Running speed [pps]</td> </tr> </table> <p>Response : 1 byte</p> <table border="1" data-bbox="544 1753 831 1823"> <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	<p>To request the motor to move its position as much as the incremental value[pulse]</p> <p>Sending : 8 bytes</p> <table border="1" data-bbox="547 264 1099 365"> <tr> <td>4 bytes</td> <td>4 bytes</td> </tr> <tr> <td>Incremental position value</td> <td>Running speed [pps]</td> </tr> </table> <p>Response : 1 byte</p> <table border="1" data-bbox="547 398 831 465"> <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>To request the motor to start limit motion at the current setting parameter condition</p> <p>Sending : 5 bytes</p> <table border="1" data-bbox="547 611 1294 678"> <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 border="1" data-bbox="547 712 831 779"> <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>To request the motor to start jog motion at the current setting parameter condition</p> <p>Sending : 5 bytes</p> <table border="1" data-bbox="547 925 1305 992"> <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 border="1" data-bbox="547 1025 852 1093"> <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>To request the motor to change the target absolute position value[pulse] while it is in running.</p> <p>Sending : 4 bytes</p> <table border="1" data-bbox="547 1238 1035 1305"> <tr> <td>4 bytes</td> </tr> <tr> <td>Changed command position value [pulse]</td> </tr> </table> <p>Response : 1 byte</p> <table border="1" data-bbox="547 1339 842 1406"> <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>To request the motor to change the target incremental position value[pulse] while it is in running.</p> <p>Sending : 4 bytes</p> <table border="1" data-bbox="547 1552 1051 1619"> <tr> <td>4 bytes</td> </tr> <tr> <td>Changed command position value [pulse]</td> </tr> </table> <p>Response : 1 byte</p> <table border="1" data-bbox="547 1653 847 1742"> <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>To request the motor to change the running speed value[pps] while it is in running.</p> <p>Sending : 4 bytes</p> <table border="1" data-bbox="547 1865 943 1933"> <tr> <td>4 bytes</td> </tr> <tr> <td>Changed running speed [pps]</td> </tr> </table> <p>Response : 1 byte</p> <table border="1" data-bbox="547 1966 858 2054"> <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	To request stop for all motor that connected in same port.  Sending : 0 byte ( Slave number must be '99' )  Response : no response												
0x3C (60)	FAS_ AllEmergencyStop	To request emergency stop for all motor that connected in same port.  Sending : 0 byte ( Slave number must be '99' )  Response : no response												
0x3D (61)	FAS_All MoveOriginSingleAxis	To request return to the origin at the current setting parameter condition for all motors that connected in same port.  Sending : 0 byte ( Slave number must be '99' )  Response : no response												
0x3E (62)	FAS_All SingleAxisAbsPos	To request move its position as much as the absolute value [pulse] for all motors that connected in same port.  Sending : 8 bytes ( Slave number must be '99' ) <table border="1" style="margin-left: 20px;"> <tr> <td style="width: 50px;">4 bytes</td> <td style="width: 50px;">4 bytes</td> </tr> <tr> <td>Absolute position value</td> <td>Running speed [pps]</td> </tr> </table> Response : no response	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	To request move its position as much as the incremental value [pulse] for all motors that connected in same port.  Sending : 8 bytes ( Slave number must be '99' ) <table border="1" style="margin-left: 20px;"> <tr> <td style="width: 50px;">4 bytes</td> <td style="width: 50px;">4 bytes</td> </tr> <tr> <td>incremental position value</td> <td>Running speed [pps]</td> </tr> </table> Response : no response	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 border="1" style="margin-left: 20px;"> <tr> <td style="width: 50px;">4 bytes</td> <td style="width: 50px;">4 bytes</td> <td style="width: 50px;">4 bytes</td> <td style="width: 50px;">2 bytes</td> </tr> <tr> <td>Absolute position value</td> <td>Running speed [pps]</td> <td>Flag option</td> <td>Custom Accel. Time (1~9999)</td> </tr> </table> <table border="1" style="margin-left: 20px;"> <tr> <td style="width: 50px;">2 bytes</td> <td style="width: 50px;">24 bytes</td> </tr> <tr> <td>Custom Decel. Time (1~9999)</td> <td>Reserved</td> </tr> </table> Flag option : 0x0001 : reserved 0x0002 : Custom Accel. Time is used.	4 bytes	4 bytes	4 bytes	2 bytes	Absolute 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											
Absolute position value	Running speed [pps]	Flag option	Custom Accel. Time (1~9999)											
2 bytes	24 bytes													
Custom Decel. Time (1~9999)	Reserved													

		<p>0x0004 : Custom Decel. Time is used.          If the Flag bit is OFF status(0), Accel./Decel. Time value is used that saved in controller.</p> <p>Response: 1 byte</p>												
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 border="1"> <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 border="1"> <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 option : 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 border="1"> <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 border="1"> <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 option : 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													

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 border="1" data-bbox="547 353 1345 472"> <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 border="1" data-bbox="547 506 1094 618"> <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 option : 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	<p>To request Linear Interpolation move its position as much as the absolute value[pulse] for more than 2 motors that connected in same port. Refer to 「2. Library for PC program」 .</p>										
	FAS_MoveLinearIncPos	<p>To request Linear Interpolation move its position as much as the incremental value[pulse] for more than 2 motors that connected in same port. Refer to 「2. Library for PC program」 .</p>										
0x40 (64)	FAS_ GetAxisStatus	<p>To request the Flag value of displaying the running status</p> <p>Sending : 0 byte Response : 5 bytes</p> <table border="1" data-bbox="547 1485 1046 1563"> <tr> <td>1 byte</td> <td>4 bytes</td> </tr> <tr> <td>Communication status</td> <td>Status flag value</td> </tr> </table> <p>For 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>To 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 border="1" data-bbox="547 1865 1345 1977"> <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>To request the current running progress status and its PT number (Frame type 0x51, 0x53, 0x54, and 0x55 are packed.)</p> <p>Sending : 0 byte Response : 21 bytes</p> <table border="1" data-bbox="544 376 1406 528"> <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>	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>To 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 border="1" data-bbox="544 719 1305 819"> <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 border="1" data-bbox="544 846 1350 976"> <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>	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>Ezi-SERVO ALL is the closed loop control drive and so the command position value is continuously controlled while the motor is in running. The user sets it to the command position value before it starts to operate and then can check how the command position value is changed.</p> <p>Sending : 4 bytes</p> <table border="1" data-bbox="544 1245 1002 1323"> <tr> <td>4 bytes</td> </tr> <tr> <td>Command position setting count value</td> </tr> </table> <p>Response : 1 byte</p> <table border="1" data-bbox="544 1361 826 1435"> <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>To request the command position value[pulse] being tracked.</p> <p>Sending : 0 byte Response : 5 bytes</p> <table border="1" data-bbox="544 1581 1106 1659"> <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>Ezi-SERVO ALL is the closed loop control drive and so the actual position value is continuously controlled while the motor is in running. The user sets it to the actual position value before it starts to operate and then can check how the actual position value is changed.</p> <p>Sending : 4 bytes</p> <table border="1" data-bbox="544 405 1002 483"> <tr> <td>4 bytes</td> </tr> <tr> <td>Actual position count value</td> </tr> </table> <p>Response : 1 byte</p> <table border="1" data-bbox="544 517 863 595"> <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>To request the current actual position value[pulse].</p> <p>Sending : 0 byte</p> <p>Response : 5 bytes</p> <table border="1" data-bbox="544 734 1137 813"> <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>To request the difference [pulse] between the command position value and the actual position value.</p> <p>Sending : 0 byte</p> <p>Response : 5 bytes</p> <table border="1" data-bbox="544 1055 1161 1133"> <tr> <td>1 byte</td> <td>4 bytes</td> </tr> <tr> <td>Communication status</td> <td>Position difference value</td> </tr> </table> <p>By this value, the user can check the current running status (how much inposition is tracked).</p>	1 byte	4 bytes	Communication status	Position difference value
1 byte	4 bytes					
Communication status	Position difference value					
0x55 (85)	FAS_ GetActualVel	<p>To request the current running speed value [pps]</p> <p>Sending : 0 byte</p> <p>Response : 5 bytes</p> <table border="1" data-bbox="544 1391 1046 1469"> <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>Ezi-SERVO ALL is the closed loop control drive and so the command position value is continuously controlled while the motor is in running. The user sets the command position and actual position value to '0' before it starts to operate and then can check how the command position value is changed.</p> <p>Sending : 0 byte</p> <p>Response : 1 byte</p> <table border="1" data-bbox="544 1783 863 1861"> <tr> <td>1 byte</td> </tr> <tr> <td>Communication status</td> </tr> </table>	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 border="1" data-bbox="547 282 925 360"> <tr><td>1 byte</td></tr> <tr><td>0:pause release, 1:pause start</td></tr> </table> <p>Response : 1 byte</p> <table border="1" data-bbox="547 427 869 506"> <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 PT values in the RAM of the drive.</p> <p>Sending : 2 bytes</p> <table border="1" data-bbox="547 719 925 797"> <tr><td>2 bytes</td></tr> <tr><td>Readable PT number (0~63)</td></tr> </table> <p>Response : 65 bytes</p> <table border="1" data-bbox="547 835 1114 913"> <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 (0~63)	1 byte	64 bytes	Communication status	Relevant PT values		
2 bytes										
Readable PT number (0~63)										
1 byte	64 bytes									
Communication status	Relevant PT values									
0x61 (97)	FAS_ PosTableWriteItem	<p>To save PT values to the RAM of the drive.</p> <p>Sending : 66 bytes</p> <table border="1" data-bbox="547 1104 1046 1182"> <tr><td>2 bytes</td><td>64 bytes</td></tr> <tr><td>PT number (0~63)</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 border="1" data-bbox="547 1256 1323 1350"> <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 (0~63)	Relevant PT value	1 byte	1 byte	Communication status	Command performing status (values except 0 : complete, 0: error)
2 bytes	64 bytes									
PT number (0~63)	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 PT values (64 ea) in the ROM of the drive</p> <p>Sending : 0 byte</p> <p>Response : 2 bytes</p> <table border="1" data-bbox="547 1559 1323 1659"> <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 PT value(64 ea) to the ROM of the drive.</p> <p>Sending : 0 byte</p> <p>Response : 2 bytes</p> <table border="1" data-bbox="547 1895 1323 1995"> <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 PT number</p> <p>Sending : 2 bytes</p> <table border="1" data-bbox="544 264 799 342"> <tr><td>2 bytes</td></tr> <tr><td>PT Number (0~63)</td></tr> </table> <p style="text-align: right;">Response : 1 byte</p> <table border="1" data-bbox="544 376 836 454"> <tr><td>1 byte</td></tr> <tr><td>Communication status</td></tr> </table>	2 bytes	PT Number (0~63)	1 byte	Communication status																
2 bytes																						
PT Number (0~63)																						
1 byte																						
Communication status																						
0x6A (106)	FAS_ PosTableReadOneItem	<p>To read one of PT values in the RAM of the drive.</p> <p>Sending: 4 byte</p> <table border="1" data-bbox="544 566 1331 645"> <tr><td>2 byte</td><td>2 byte</td></tr> <tr><td>PT Number (0~63)</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 border="1" data-bbox="544 730 1139 808"> <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 (0~63)	Offset value(0~40)	1 byte	4 byte	Communication status	Relevant one of PT value												
2 byte	2 byte																					
PT Number (0~63)	Offset value(0~40)																					
1 byte	4 byte																					
Communication status	Relevant one of PT value																					
0x6B (107)	FAS_ PosTableWriteOneItem	<p>To save one of PT values to the RAM of the drive .</p> <p>Sending: 8 byte</p> <table border="1" data-bbox="544 947 1366 1025"> <tr><td>2 byte</td><td>2 byte</td><td>4 byte</td></tr> <tr><td>PT Number (0~63)</td><td>Offset value (0~40)</td><td>Relevant one of PT value</td></tr> </table> <p>Refer to 「1-2-6. Position Table Item」 for Offset value</p> <p>Response: 2 byte</p> <table border="1" data-bbox="544 1111 1347 1216"> <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 byte	2 byte	4 byte	PT Number (0~63)	Offset value (0~40)	Relevant one of PT value	1 byte	1 byte	Communication status	Command performing status (values except 0 : complete, 0: error)										
2 byte	2 byte	4 byte																				
PT Number (0~63)	Offset value (0~40)	Relevant one of PT value																				
1 byte	1 byte																					
Communication status	Command performing status (values except 0 : complete, 0: error)																					
0x78 (120)	FAS_ MovePush	<p>To request push motion(maintain specified motor torque) Command. This function is worked only with Ezi-SERVO-PR-MI and MC-28VA cylinder.</p> <p>Sending: 28 bytes</p> <table border="1" data-bbox="544 1469 1369 1585"> <tr><td>4 byte</td><td>4 bytes</td><td>4 byte</td><td>2 bytes</td><td>2 bytes</td></tr> <tr><td>Normal Start speed</td><td>Normal Move speed</td><td>Normal Position</td><td>Accel time</td><td>Decel time</td></tr> </table> <table border="1" data-bbox="544 1626 1235 1742"> <tr><td>2 byte</td><td>4 bytes</td><td>4 byte</td><td>2 byte</td></tr> <tr><td>Push torque ratio</td><td>Push Move speed</td><td>Push Position</td><td>Push mode</td></tr> </table> <p>Refer to 「User Manual Test 10-6. Push Motion」 .</p> <p>Response: 1 byte</p> <table border="1" data-bbox="544 1872 831 1951"> <tr><td>1 byte</td></tr> <tr><td>Communication status</td></tr> </table>	4 byte	4 bytes	4 byte	2 bytes	2 bytes	Normal Start speed	Normal Move speed	Normal Position	Accel time	Decel time	2 byte	4 bytes	4 byte	2 byte	Push torque ratio	Push Move speed	Push Position	Push mode	1 byte	Communication status
4 byte	4 bytes	4 byte	2 bytes	2 bytes																		
Normal Start speed	Normal Move speed	Normal Position	Accel time	Decel time																		
2 byte	4 bytes	4 byte	2 byte																			
Push torque ratio	Push Move speed	Push Position	Push mode																			
1 byte																						
Communication status																						

<p>0x79 (121)</p>	<p>FAS_ GetPushStatus</p>	<p>To request the current push motion status. This function is worked only with Ezi-SERVO-PR-MI and MC-28VA cylinder.</p> <p>Sending: 0 byte Response: 2 byte</p> <table border="1" data-bbox="544 394 1401 582"> <tr> <td data-bbox="544 394 735 432">1 byte</td> <td data-bbox="735 394 1401 432">1 bytes</td> </tr> <tr> <td data-bbox="544 432 735 582">Communication status</td> <td data-bbox="735 432 1401 582">           Push motion status            (0: normal Servo ON            1: push motioning but the work is not detected            2: work detected and torque is maintained )         </td> </tr> </table>	1 byte	1 bytes	Communication status	Push motion status (0: normal Servo ON 1: push motioning but the work is not detected 2: work detected and torque is maintained )
1 byte	1 bytes					
Communication status	Push motion status (0: normal Servo ON 1: push motioning but the work is not detected 2: work detected and torque is maintained )					

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



## 1-2-2. Parameter Lists

No.	Name	Unit	Lower Limit	Upper Limit	Default
0	Pulse per Revolution		0	9	9
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	Position Loop Gain		0	15	4
26	Inpos Value		0	15	0
27	Pos Tracking Limit	[pulse]	1	+134,217,727	5,000
28	Motion Dir		0	1	0
29	Limit Sensor Dir		0	1	0
30	Org Torque Ratio	[%]	10	100	50
31	Pos. Error Overflow Limit	[pulse]	1	+134,217,727	5,000

### 1-2-3. Bit setup of Output pin

This displays the detailed description for 0x20 Frame type.

This command is applicable only to 9 signals of 'User Output 0' ~ 'User Output 8' out of 24 signal types in the control output port. The rest (15 output signals) of them cannot be operated by the user's disposal. When any relevant situation occurs while the drive operates, they are displayed. The following table shows bit mask values by each signal.

Signal Name	Relevant Bit Position	Signal Name	Relevant Bit Position	Signal Name	Relevant Bit Position
Compare Out	0x00000001	Origin Search OK	0x00000100	User Output 1	0x00010000
Inposition	0x00000002	ServoReady	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 port.

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 port

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

### 1-2-4. Bit setup of Input pin

This displays the detailed description for 0x21 Frame type.

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

Signal Name	Relevant Bit Position	Signal Name	Relevant Bit Position	Signal Name	Relevant Bit Position	Signal Name	Relevant Bit Position
Limit+	0x00000001	PT A4	0x00000100	AlarmReset	0x00010000	JPT input 2	0x01000000
Limit-	0x00000002	PT A5	0x00000200	ServoON	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 'motion\_define.h' of include files.

It can be applied under the Firmware version of **06.03.043.8**

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
FFLAG_POSCNTOVER	Position error is higher than 'Pos Tracking Limit' value during position command run.	0X00000040
FFLAG_ERRSERVOALARM	One or more error of Servo 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
FFLAG_ERROVERLOAD	Load exceeding the max torque of the motor is loaded more than 5 seconds.	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 is not connected to drive correctly.	0X00004000
FFLAG_ERRINPOSITION	After operation is finished, a position error occurs for more than 3 seconds.	0X00008000
FFLAG_EMGSTOP	The motor is under emergency stop.	0X00010000
FFLAG_SLOWSTOP	The motor is under general stop.	0X00020000
FFLAG_ORIGINRETURNING	The motor is returning to the origin.	0X00040000
FFLAG_INPOSITION	Inposition has been finished.	0X00080000
FFLAG_SERVOON	The motor is under Servo ON.	0X00100000
FFLAG_ALARMRESET	AlarmReset has run.	0X00200000
FFLAG_PTSTOPED	Position Table operation has been finished.	0X00400000
FFLAG_ORIGINSENSOR	The origin sensor is ON.	0X00800000
FFLAG_ZPULSE	The motor operates to z-pulse type of origin return operations.	0X01000000
FFLAG_ORIGINRETOK	Origin return operation has been finished.	0X02000000
FFLAG_MOTIONDIR	To display the motor operating direction (+: Off, -: On)	0X04000000
FFLAG_MOTIONING	The motor is running.	0X08000000
FFLAG_MOTIONPAUSE	The motor in running is stopped by Pause command.	0X10000000
FFLAG_MOTIONACCEL	The motor is operating to the acceleration section.	0X20000000
FFLAG_MOTIONDECEL	The motor is operating to the deceleration section.	0X40000000
FFLAG_MOTIONCONST	The motor is operating to the normal speed, not acceleration / deceleration sections.	0X80000000

Next bit assignments can be applied over the Firmware version of **06.03.043.10**

Name of Flag Define	Contents	Relevant Bit Position
FFLAG_ERRORALL	One or more error occurs.	0X00000001
FFLAG_HWPOSILMT	'+' direction limit sensor turns ON.	0X00000002
FFLAG_HWNEGALMT	'-' direction limit sensor turns ON.	0X00000004
FFLAG_SWPOGILMT	'+' direction program limit is exceeded.	0X00000008
FFLAG_SWNEGALMT	'-' direction program limit is exceeded.	0X00000010
Reserved1		0X00000020
Reserved2		0X00000040
FFLAG_ERRPOSOVERFLOW	Position error is higher than 'Pos Error Overflow Limit' parameter after position command	0X00000080
FFLAG_ERROVERCURRENT	The motor driving device is under over-current.	0X00000100
FFLAG_ERROVERSPEED	The motor speed exceeded 3000[rpm].	0X00000200
FFLAG_ERRPOSTRACKING	Position error is higher than 'Pos Tracking Limit' parameter during position command run.	0X00000400
FFLAG_ERROVERLOAD	Load exceeding the max torque of the motor is loaded more than 5 seconds.	0X00000800
FFLAG_ERROVERHEAT	The internal temperature of the drive exceeds 55° C.	0X00001000
FFLAG_ERRBACKEMF	A counter electromotive force of the motor exceeds 70V.	0X00002000
FFLAG_ERRMOTORPOWER	The power supplied to the motor is less than low limited value.	0X00004000
FFLAG_ERRINPOSITION	After operation is finished, a position error occurs for more than 3 seconds.	0X00008000
FFLAG_EMGSTOP	The motor is under emergency stop.	0X00010000
FFLAG_SLOWSTOP	The motor is under general stop.	0X00020000
FFLAG_ORIGINRETURNING	The motor is returning to the origin.	0X00040000
FFLAG_INPOSITION	Inposition has been finished.	0X00080000
FFLAG_SERVOON	The motor is under Servo ON.	0X00100000
FFLAG_ALARMRESET	AlarmReset has run.	0X00200000
FFLAG_PTSTOPED	Position Table operation has been finished.	0X00400000
FFLAG_ORIGINSENSOR	The origin sensor is ON.	0X00800000
FFLAG_ZPULSE	The motor operates to z-pulse type of origin return operations.	0X01000000
FFLAG_ORIGINRETOK	Origin return operation has been finished.	0X02000000
FFLAG_MOTIONDIR	To display the motor operating direction (+: Off, -: On)	0X04000000
FFLAG_MOTIONING	The motor is running.	0X08000000
FFLAG_MOTIONPAUSE	The motor in running is stopped by Pause command.	0X10000000
FFLAG_MOTIONACCEL	The motor is operating to the acceleration section.	0X20000000
FFLAG_MOTIONDECEL	The motor is operating to the deceleration section.	0X40000000
FFLAG_MOTIONCONST	The motor is operating to the normal speed, not acceleration / deceleration sections.	0X80000000

## 1-2-6. Position Table Item

Refer to 'motion\_define.h' of include files.

Name	Name of Structure Parameter	Number of Bytes	Offset value	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
Check Inposition	bCheckInpos	2 (unsigned)	38		0	1
Compare Position	lTriggerPos	4 (signed)	40	[pulse]	-134217728	+134217728
Compare Width	wTriggerOnTime	2 (unsigned)	44	[msec]	1	9999
Push Ratio	wPushRatio	2 (unsigned)	46	[%]	10	90
Push Speed	dwPushSpeed	4 (unsigned)	48	[pps]	0	50000
Push Position	lPushPosition	4 (signed)	52	[pulse]	-134217728	+134217728
Push Mode	wPushMode	2 (unsigned)	56		0	1
Blank		6 (unsigned)	58	0x00		

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

## 1-2-7. Information of Motors

Firstly the number and 2~3 characters are display the motor size and length.

【Example 1】 56XL : Motor Flange size is 56mm and Extra long size

Second area is display the motor maker information like below.

Display	Maker
blank	JapanServo
SD	Sanyo Denki
POR	Portescap
NPM	NPM
FUL	Fulling

## 1 – 3. Program Method

There are 2 method of programming for Ezi-SERVO ALL.

The first is normally used method that using Visual C++ language under window system of PC. Library that serviced together with Ezi-SERVO ALL have to be used. Refer to [「2. Library for PC Program」](#)

The second method can be accomplished by sending command characters directly to Ezi-SERVO ALL. The user have to prepare low level protocol programming like ‘Protocol Test’ program. This method is normally used for PLC system. For excise the protocol programming, ‘ProtocolTest\_PlusR.exe’ GUI program is serviced together. Refer to [「3. Protocol for PLC Program」](#).

## 2. Library for PC Program(Ver6)

### 2-1. Library Configuration

To use this library, C++ header file(\*.h) and library file(\*.lib or \*.dll) are required. These files are included in [“WWFASTECHWW EziMOTION PlusR WWincludeWW”](#). 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 WWincludeWWEziMotionPlusR.lib”
“WWFASTECHWW EziMOTION PlusR WWincludeWWEziMotionPlusR.dll”
```

A sample program source of using library is included in a [“WWFASTECHWWEziMOTION PlusR WWExamplesWW”](#) folder.

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

Item	Definition	Returned Value	Description
Normal	FMM_OK	0	The function has normally performed the command.
Input Error	FMM_NOT_OPEN	1	Wrong port number is inputted.
	FMM_INVALID_PORT_NUM	2	The port that is not connected.
	FMM_INVALID_SLAVE_NUM	3	Wrong slave number is inputted.
Operation Error	FMM_POSTABLE_ERROR	9	An error occurs while the motor accesses to the position table.
Connection Error	FMC_DISCONNECTED	5	The relevant 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.

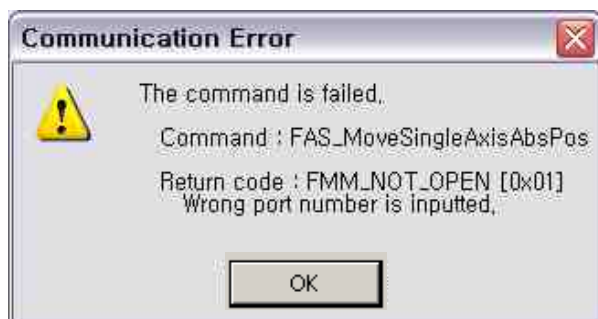
(2) The following table shows return values included commonly in all libraries. The user can check the result (communication status, running status) judged by the drive. When the user develops programs by using protocols without libraries (DLL), they are available as well.

Item	Description	Returned Value	Description
Normal	FMP_OK	0	Communication has been normally performed.
Input Error	FMP_FRAMETYPEERROR	128	The drive cannot recognize the command.
	FMP_DATAERROR	129	Input data is out of the range.
Operation Error	FMP_RUNFAIL	133	The motor is already running or not prepared for running. Other wrong motion command.
	FMP_RESETFAIL	134	The user cannot execute AlarmReset command while the servo is ON.
	FMP_SERVOONFAIL1	135	An alarm has occurred.
	FMP_SERVOONFAIL2	136	The motor is under Emergency Stop.
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 divide by 3 groups.

### (1) Communication Error



FMM\_NOT\_OPEN,

COM Port is not connected.



FMM\_INVALID\_PORT\_NUM,

COM Port number is not exist. Checking the 'Device Manager' window in Window OS.





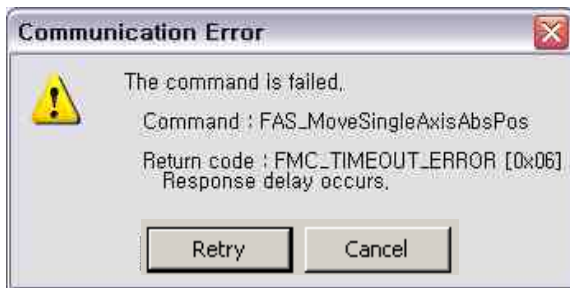
FMM\_INVALID\_SLAVE\_NUM,

Slave number is not exist. Checking the ID value of the drive.



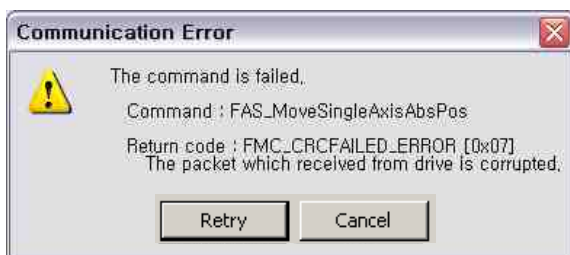
FMC\_DISCONNECTED = 5,

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



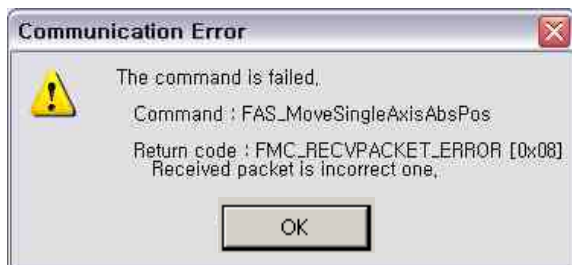
FMC\_TIMEOUT\_ERROR,

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.



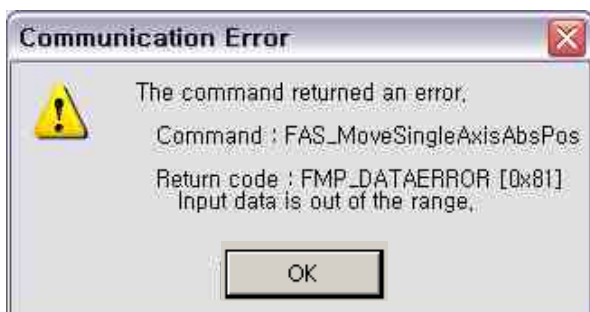
FMC\_RECVPACKET\_ERROR,

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



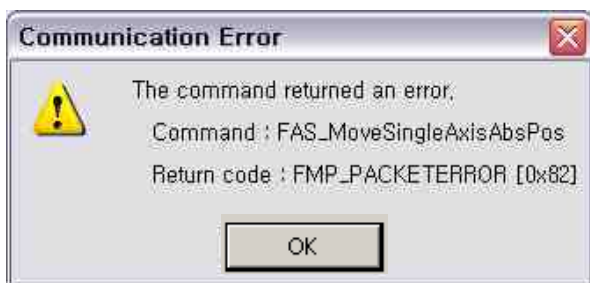
FMP\_FRAMETYPEERROR = 0x80,

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



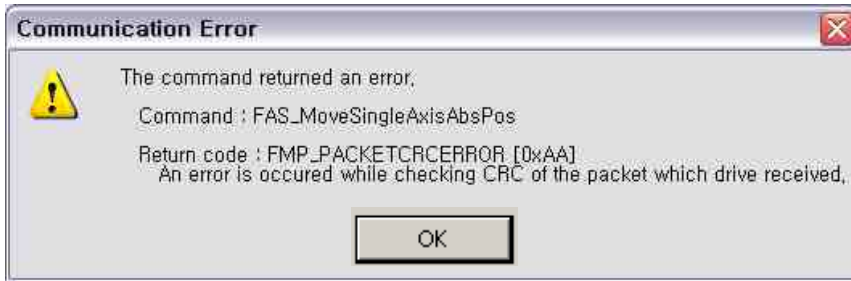
FMP\_DATAERROR,

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



FMP\_PACKETERROR,

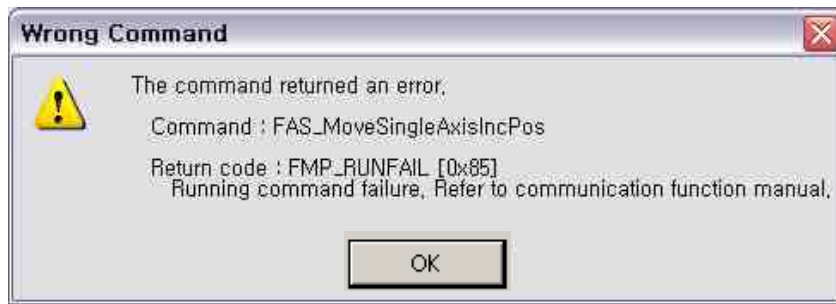
The length of received packet on drive is not correct. Checking the possibility of noise on communication cable.



FMP\_PACKETCRCERROR = 0xAA,

The CRC value on drive is not correct. Checking the possibility of noise on communication cable.

## (2) Wrong Command



FMP\_RUNFAIL = 0x85,

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

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



FMP\_RESETFAIL,

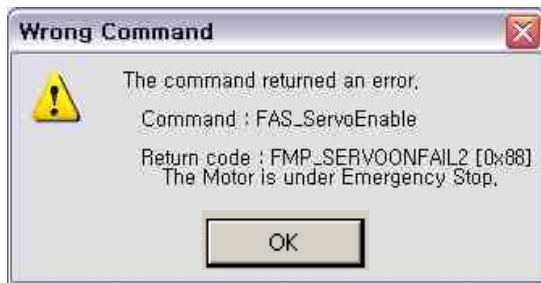
Fail on reset command : The motor can not reset on next status.

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



FMP\_SERVOONFAIL1,

Wrong 'Servo ON' command during Alarm happens.



FMP\_SERVOONFAIL2,

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



FMP\_SERVOONFAIL3,

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

### (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 return. Otherwise, FALSE will return.
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 return.
FAS_GetMotorInfo	The drive reads motor type and maker: Motor type and maker information will return.
FAS_IsSlaveExist	The drive checks whether there is the relevant drive: When it exists, TRUE will return. Otherwise, FALSE will return.

## FAS\_Connect

---

FAS\_Connect is the function of connecting Ezi-SERVO ALL.

Syntax

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

Parameters

*nPortNo*

Select a serial port to be connected.

*dwBaud*

Input the Baudrate of the serial port.

Return Value

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

Remarks

Example

```
#include "FAS_EziMOTIONPlusR.h"  
  
void funclnit()  
{  
    BYTE nPortNo = 1; // COMM Port Number  
    DWORD dwBaudrate = 115200; // Baudrate. (Be variable by setting)  
    BYTE iSlaveNo = 0; // Slave No (0 ~ 15)  
    char lpBuff[256];  
    int nBuffSize = 256;  
    BYTE nType;  
    int nRtn;  
  
    // Try to connect  
    if (FAS_Connect(nPortNo, dwBaudrate) == FALSE)  
    {  
        // Connection failed.  
        // The port is not connected or the baudrate may be wrong.  
        return;  
    }  
  
    if (FAS_IsSlaveExist(nPortNo, iSlaveNo) == FALSE)  
    {  
        // There is no relevant slave number.  
        // Check the slave number of Ezi-SERVO Plus-R.  
        return;  
    }  
  
    nRtn = FAS_GetSlaveInfo(nPortNo, iSlaveNo, &nType, lpBuff, nBuffSize);  
    if (nRtn != FMM_OK)  
    {  
        // Command has not been performed properly.  
        // Refer to ReturnCodes_Define.h.  
    }  
  
    printf("Port : %d (Slave %d) 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 disconnect

### 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*

Relevant drive type number

*lpBuff*

Buffer pointer to get version information string

*nBuffSize*

lpBuff memory allocation size

### Return Value

FMM\_OK : Command has been normally performed.

FMM\_NOT\_OPEN : The 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*

Relevant motor type number

*lpBuff*

Buffer pointer to get version information string

*nBuffSize*

lpBuff memory allocation size

Return Value

FMM\_OK : Command has been normally performed.

FMM\_NOT\_OPEN : The 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

---

To check that the drive is connected

### 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

## 2-4. Parameter Control Function

Function Name	Description
FAS_SaveAllParameters	Current parameters are saved 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	The designated parameter is saved to the RAM: Specific parameter is saved.
FAS_GetParameter	The designated parameter is read from the RAM: Specific parameter is read.
FAS_GetROMParameter	The designated parameter is read from the ROM: Specific parameter is read from the ROM.

## FAS\_SaveAllParameters

---

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

Syntax

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

Parameters

*nPortNo*  
Port number of relevant drive

*iSlaveNo*  
Slave number of relevant drive

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

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, SERVO_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-SERVO 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, SERVO_AXISSTARTSPEED, 200);
_ASSERT(nRtn == FMM_OK); // You have to check if the command didn't execute
correctly.

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

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

// Edit the parameter value then save it in the ROM.
nRtn = FAS_SetParameter(nPortNo, iSlaveNo, SERVO_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 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 : There is no parameter of designated iParamNo.

### Remarks

The function operates only for 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\_GetParameter

---

To call specific parameter values 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 imported

*IParamValue*

Parameter values

### 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 : There is no parameter of designated iParamNo.

### Remarks

The function operates only for 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* lRomParam  
);
```

### Parameters

*nPortNo*

Port number of relevant drive

*iSlaveNo*

Slave number of relevant drive

*iParamNo*

Parameter number to be imported

*lRomParam*

Parameter values saved in the ROM

### Return Value

FMM\_OK : Command has been normally performed.

FMM\_NOT\_OPEN : The 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_ServoEnable	The Servo of the drive designated turns ON/OFF.
FAS_ServoAlarmReset	The drive which an alarm occurs is released: Troubleshoot the alarm cause and use this function.
FAS_AlarmType	Read the Alarm type of the drive.

## FAS\_ServoEnable

---

To turn ON/OFF the drive servo

Syntax

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

Parameters

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

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

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

Example

```
#include "FAS_ EzIMOTIONPlusR.h"  
  
void funcAxisStatus()  
{  
    BYTE nPortNo = 1; // COMM Port Number  
    BYTE iSlaveNo = 0; // Slave No (0 ~ 15)  
    EZISERVO_AXISSTATUS AxisStatus;  
    int nRtn;  
  
    // Try to connect  
    if (FAS_Connect(nPortNo, 115200) == FALSE)  
    {  
        // Connection failed.  
        // The port is not connected or the baudrate may be wrong.  
        return;  
    }  
  
    nRtn = FAS_GetAxisStatus(nPortNo, iSlaveNo, &(AxisStatus.dwValue));  
    _ASSERT(nRtn == FMM_OK);  
  
    // If SERVO_ON flag turns off, the servo turns on..  
    if (AxisStatus.FFLAG_SERVOON == 0)  
    {  
        nRtn = FAS_ServoEnable(nPortNo, iSlaveNo, TRUE);  
        _ASSERT(nRtn == FMM_OK);  
    }  
  
    // If there is an alarm, AlarmReset runs.  
    if (AxisStatus.FFLAG_ERRORALL || AxisStatus.FFLAG_ERROVERCURRENT ||  
        AxisStatus.FFLAG_ERROVERLOAD)  
    {  
        nRtn = FAS_ServoAlarmReset(nPortNo, iSlaveNo);  
        _ASSERT(nRtn == FMM_OK);  
    }  
}
```

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

See Also

FAS\_ServoAlarmReset

## FAS\_ServoAlarmReset

---

To send AlarmReset command

Syntax

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

Parameters

*nPortNo*

Port number of relevant drive

*iSlaveNo*

Slave number of relevant drive

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

Before sending this command, troubleshoot the alarm cause.

For alarm cause, refer to 'User Manual\_Text' .

Example

Refer to 'FAS\_ServoEnable' library

See Also

FAS\_ServoEnable

## 2-6. Control I/O Function

Function Name	Description
FAS_SetI0Input	To set the input signal level of the control input port : Input signal is set to [ON] or [OFF].
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 output port : Output signal is set to [ON] or [OFF].
FAS_GetI0Output	To read the current output signal status of the control output port : The signal status returns by bit for each output signal.
FAS_GetI0AssignMap	To read the pin setting status of the CN1 port : The setting status for each 7 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 7 variable signals is assigned to the Input and Output port.
FAS_I0AssignMapReadROM	To load the pin setting status of CN1 port from ROM area to RAM area.

## FAS\_SetIOInput

---

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

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 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

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

Example

```
#include "FAS_ EzIMOTIONPlusR.h"  
  
void funcIO()  
{  
    BYTE nPortNo = 1; // COMM Port Number  
    BYTE iSlaveNo = 0; // Slave No (0 ~ 15)  
    DWORD dwInput, dwOutput;  
    int nRtn;  
  
    // Try to connect  
    if (FAS_Connect(nPortNo, 115200) == FALSE)  
    {  
        // Connection failed.  
        // The port is not connected or the baudrate may be wrong.  
        return;  
    }  
  
    // Check I/O input.  
    nRtn = FAS_GetIOInput(nPortNo, iSlaveNo, &dwInput);  
    _ASSERT(nRtn == FMM_OK);  
    if (dwInput & SERVO_IN_BITMASK_LIMITP)  
    {  
        // Limit + input is ON.  
    }  
}
```

```

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

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

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

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

// Disconnect.
FAS_Close(nPortNo);
}

```

See Also

FAS\_GetIOInput



## FAS\_GetIOInput

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

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 which input values will be saved

Return Value

FMM\_OK : Command has been normally performed .

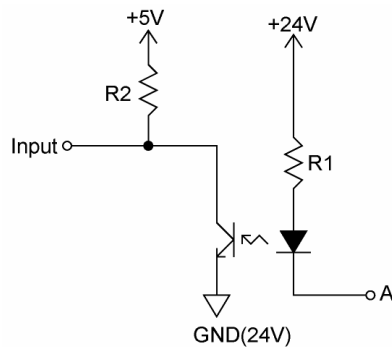
FMM\_NOT\_OPEN : The drive has not been connected yet .

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

There are 10 input pins in EziSERVO ALL. The user can select and use 7 input pins of them. This function can read the input port status by 32bit. All of them are insulated by a photocoupler. (Refer to the figure.)



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

Example

Refer to 'FAS\_SetIOInput' library.

See Also

FAS\_SetIOInput

## FAS\_SetIOOutput

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

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

*dwIOCLRMask*

Output bitmask value be cleared

Return Value

FMM\_OK : Command has been normally performed.

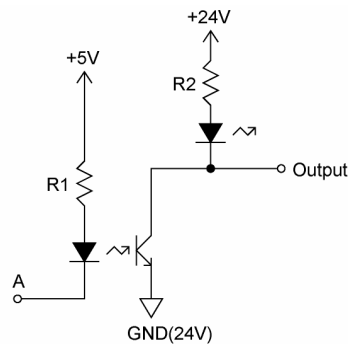
FMM\_NOT\_OPEN : The drive has not been connected yet .

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

There are 2 input pins in EziSERVO ALL. The user can select and use 1 output pins of them.



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

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

Example

Refer to FAS\_SetIOInput .

See Also

FAS\_GetIOOutput

## FAS\_GetIOOutput

---

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

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 which the output value will be saved .

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

Refer to 'FAS\_SetIOInput' library

See Also

FAS\_SetIOOutput

## FAS\_GetIOAssignMap

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

Syntax

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

Parameters

*nPortNo*

Port number of relevant drive.

*iSlaveNo*

Slave number of relevant drive.

*iIOPinNo*

I/O pin number to be read

*nIOLogic*

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

*bLevel*

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

Return Value

FMM\_OK : Command has been normally performed.

FMM\_NOT\_OPEN : The 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 nIOLogic, refer to 'Motion\_define.h' .

Example

```
#include "FAS_ EzIMOTIONPlusR.h"  
  
void funcIOAssign()  
{  
    BYTE nPortNo = 1; // COMM Port Number  
    BYTE iSlaveNo = 0; // Slave No (0 ~ 15)  
    BYTE iPinNo;  
    DWORD dwLogicMask;  
    BYTE bLevel;  
    BYTE i;  
    int nRtn;  
  
    // Try to connect  
    if (FAS_Connect(nPortNo, 115200) == FALSE)  
    {  
        // Connection failed.  
        // The port is not connected or the baudrate may be wrong.  
        return;  
    }  
  
    // Check assigned information of input pin.  
    for (i=0; i</*Input Pin Count*/12; i++)  
    {
```

```

        nRtn = FAS_GetIOAssignMap(nPortNo, iSlaveNo, i, &dwLogicMask, &bLevel);
        _ASSERT(nRtn == FMM_OK);

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

    // Assign SERV00N Logic (Low Active) to input pin 3.
    iPinNo = 3;          // 0 ~ 11 value is available (Caution : 0 ~ 2 is fixed.)
    nRtn = FAS_SetIOAssignMap(nPortNo, iSlaveNo, iPinNo, SERVO_IN_BITMASK_SERV00N,
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) Wn", i, dwLogicMask,
((bLevel == LEVEL_LOW_ACTIVE) ? "Low Active" : "High Active"));
        else
            printf("Output Pin %d : Not assigned Wn", i);
    }

    // Assign ALARM Logic (High Active) to output pin 9.
    iPinNo = 9;          // 0 ~ 9 value is available (Caution : 0 is fixed to COMPOUT.)
    nRtn = FAS_SetIOAssignMap(nPortNo, iSlaveNo, 12/*Input Pin Count*/ + iPinNo,
SERVO_OUT_BITMASK_ALARM, LEVEL_HIGH_ACTIVE);
    _ASSERT(nRtn == FMM_OK);

    // Disconnect.
    FAS_Close(nPortNo);
}

```

See Also

FAS\_SetIOAssignMap

## FAS\_SetIOAssignMap

---

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

Syntax

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

Parameters

*nPortNo*

Port number of relevant drive.

*iSlaveNo*

Slave number of relevant drive.

*iIOPinNo*

I/O Pin number to be read

*nIOLogic*

Logic value to be assigned to the relevant pin

*bLevel*

Active Level value of the relevant logic

Return Value

FMM\_OK : Command has been normally performed .

FMM\_NOT\_OPEN : The 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 nIOLogic value is out of range.

Remarks

To save current setting values to the memory, 'FAS\_SaveAllParameters' library should be run.

Example

Refer to 'FAS\_GSetIOAssignMap' library

See Also

FAS\_GetIOAssignMap

## FAS\_IOAssignMapReadROM

---

To load the status of CN1 assignment 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 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\_GetIOAssignMap

## 2-7. Position Control Function

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



## FAS\_SetCommandPos

---

To set the command position value to the motor

Syntax

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

Parameters

*nPortNo*

Port number of relevant drive.

*iSlaveNo*

Slave number of relevant drive.

*lCmdPos*

Command position value to be set.

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

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 he 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 to 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 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

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

Example

Refer to 'FAS\_GetActualPos' library.

See Also

FAS\_SetCommandPos

## FAS\_GetCommandPos

---

To read the command position of the current motor

Syntax

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

Parameters

*nPortNo*

Port number of relevant drive

*iSlaveNo*

Slave number of relevant drive

*lCmdPos*

Parameter pointer that command position value will be saved

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

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-SERVO Plus-R.  
    nRtn = FAS_GetCommandPos(nPortNo, iSlaveNo, &lValue);  
    _ASSERT(nRtn == FMM_OK);  
    printf("CMDPOS : %d Wn", lValue);  
    nRtn = FAS_GetActualPos(nPortNo, iSlaveNo, &lValue);  
    _ASSERT(nRtn == FMM_OK);  
    printf("ACTPOS : %d Wn", lValue);  
    nRtn = FAS_GetPosError(nPortNo, iSlaveNo, &lValue);  
    _ASSERT(nRtn == FMM_OK);  
    printf("POSERR : %d Wn", lValue);  
    nRtn = FAS_GetActualVel(nPortNo, iSlaveNo, &lValue);  
    _ASSERT(nRtn == FMM_OK);
```

```
printf("ACTVEL : %d Wn", IValue);  
  
// 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 which the actual position value will be saved .

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

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 which the position error value will be saved

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

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 which the actual velocity value will be saved

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

Refer to 'FAS\_GetCommandPosition' library.

See Also



## FAS\_ClearPosition

---

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

### Syntax

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

### Parameters

*nPortNo*

Port number of relevant drive.

*iSlaveNo*

Slave number of relevant drive.

### 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

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 return.
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 return.
FAS_GetAllStatus	To read all status including 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

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 which the I/O input value will be saved .

*dwOutStatus*

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

*dwAxisStatus*

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

Return Value

FMM\_OK : Command has been normally performed .

FMM\_NOT\_OPEN : The 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 which the command position value will be saved

*lActPos*

Parameter pointer which the actual position value will be saved .

*lPosErr*

Parameter pointer which the position error value will be saved

*lActVel*

Parameter pointer which the actual velocity value will be saved

*wPosItemNo*

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

### Return Value

FMM\_OK : Command has been normally performed .

FMM\_NOT\_OPEN : The 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

### 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 which the I/O input value will be saved .

*dwOutStatus*

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

*dwAxisStatus*

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

*lCmdPos*

Parameter pointer which the command position value will be saved

*lActPos*

Parameter pointer which the actual position value will be saved

*lPosErr*

Parameter pointer which the position error value will be saved

*lActVel*

Parameter pointer which the actual velocity value will be saved

*wPosItemNo*

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

### Return Value

FMM\_OK : Command has been normally performed .

FMM\_NOT\_OPEN : The 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 '1-2. Structure of Frame Type' .

### 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 which the axis status value of the relevant motor

### 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

Refer to 'FAS\_MoveSingleAxisAbsPos' library.

### See Also

## 2-9. Running Control Function

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

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 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

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 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

Refer to ‘FAS\_MoveSingleAxisAbsPos’ library.

### See Also

## FAS\_MoveSingleAxisAbsPos

---

To move the motor to the absolute coordinate

Syntax

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

Parameters

*nPortNo*

Port number of relevant drive.

*iSlaveNo*

Slave number of relevant drive.

*lAbsPos*

Absolute coordinate of position to move

*lVelocity*

Velocity when the motor moves

Return Value

FMM\_OK : Command has been normally performed .

FMM\_NOT\_OPEN : The 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;  
    EZISERVO_AXISSTATUS stAxisStatus;  
    long lAbsPos, lIncPos, lVelocity;  
    int nRtn;  
  
    // Try to connect  
    if (FAS_Connect(nPortNo, 115200) == FALSE)  
    {  
        // Connection failed.  
        // The port is not connected or the baudrate may be wrong.  
        return;  
    }  
  
    // Check error and Servo ON status.  
    nRtn = FAS_GetAxisStatus(nPortNo, iSlaveNo, &dwAxisStatus);  
    _ASSERT(nRtn == FMM_OK);  
    stAxisStatus.dwValue = dwAxisStatus;  
  
    //if (dwAxisStatus & 0x00000001)  
    if (stAxisStatus.FFLAG_ERRORALL)
```

```

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

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

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

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

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

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

// Move the motor to '0'.
lAbsPos = 0;
lVelocity = 20000;
nRtn = FAS_MoveSingleAxisAbsPos(nPortNo, iSlaveNo, lAbsPos, lVelocity);
_ASSERT(nRtn == FMM_OK);

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

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

// Disconnect.
FAS_Close(nPortNo);
}

```

See Also

## FAS\_MoveSingleAxisIncPos

---

To move the motor to the incremental coordinate value

Syntax

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

Parameters

*nPortNo*

Port number of relevant drive.

*iSlaveNo*

Slave number of relevant drive.

*lIncPos*

Incremental coordinate of position to move

*lVelocity*

Velocity when the motor moves

Return Value

FMM\_OK : Command has been normally performed .

FMM\_NOT\_OPEN : The 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 which the motor moves ( 0: -Limit, 1: +Limit)

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

Refer to 'FAS\_MoveSingleAxisAbsPos' library.

See Also

## FAS\_MoveVelocity

---

To move the motor to the relevant direction and velocity. This function is also available for Jog motion.

### Syntax

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

### Parameters

*nPortNo*

Port number of relevant drive.

*iSlaveNo*

Slave number of relevant drive.

*IVelocity*

Velocity when the motor moves

*iVelDir*

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

### Return Value

FMM\_OK : Command has been normally performed .

FMM\_NOT\_OPEN : The 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 normally performed .

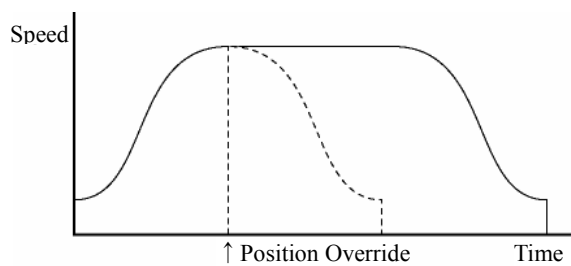
FMM\_NOT\_OPEN : The drive has not been connected yet .

FMM\_INVALID\_PORT\_NUM : There is no nPort in the connected ports .

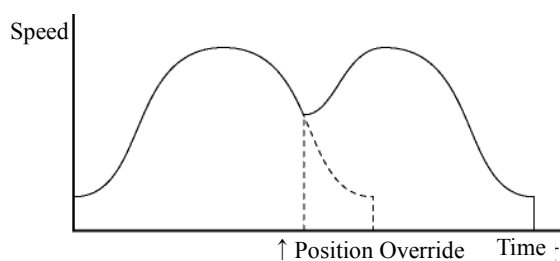
FMM\_INVALID\_SLAVE\_NUM : There is no drive of iSlaveNo in the relevant port .

Remarks

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

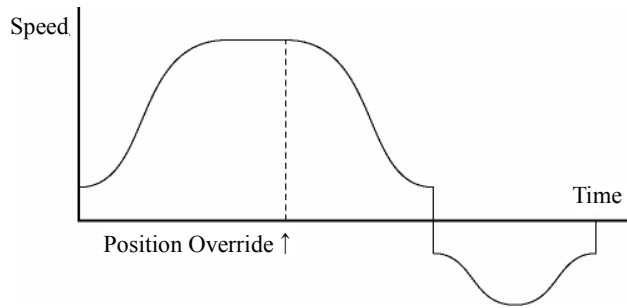


- 2) If the target position is changed while the motor is decelerated, it is again accelerated up to the uniform velocity and then stops to the target position.





- 3) If the changed target position is set to the closer coordinate than the original target position, the motor once stops to the position before change and then performs acceleration and deceleration to stop 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 IOverridePos  
);
```

### Parameters

*nPortNo*

Port number of relevant drive.

*iSlaveNo*

Slave number of relevant drive.

*IOverridePos*

Incremental coordinate position value to be changed

### 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 '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 lVelocity  
);
```

Parameters

*nPortNo*

Port number of relevant drive.

*iSlaveNo*

Slave number of relevant drive.

*lVelocity*

Velocity to be changed in [pps]

Return Value

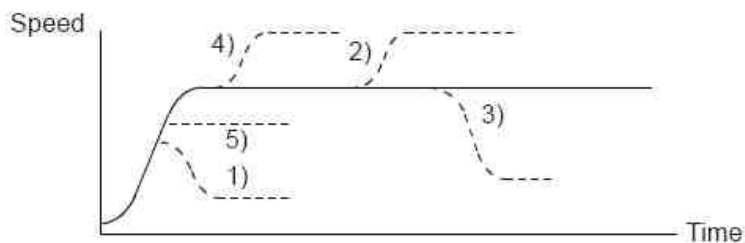
FMM\_OK : Command has been normally performed .

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



- 1) In case of  $((\text{change speed}) < (\text{speed before change}))$ , the motor reaches 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 the change speed through acceleration/deceleration without any new velocity pattern.
- 4) The motor reaches the 'speed before change' without a change of the velocity pattern and then it reaches the 'change speed' by a new velocity pattern.
- 2),3) After acceleration/deceleration is finished, the motor reaches the change speed corresponding to the velocity pattern of the 'change speed' .

Example

Refer to 'FAS\_MoveSingleAxisAbsPos' library.

See Also

## FAS\_AllMoveStop

---

To stop the motor that connected in same port.

### Syntax

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

### Parameters

*nPortNo*

Port number of relevant drive.

*iSlaveNo*

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

### Return Value

No response

### Remarks

### Example

Refer to 'FAS\_MoveSingleAxisAbsPos' library.

### See Also

## FAS\_AllEmergencyStop

---

To stop the motor that connected in same port without deceleration

### Syntax

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

### Parameters

*nPortNo*

Port number of relevant drive.

*iSlaveNo*

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

### Return Value

No response

### Remarks

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

### Example

Refer to 'FAS\_MoveSingleAxisAbsPos' library.

### See Also

## FAS\_AllMoveOriginSingleAxis

---

To search the origin of system for all motor that is connected in same port . For more information, refer to [‘User Manual\\_Text 9.3 Origin Return’](#) .

### Syntax

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

### Parameters

*nPortNo*

Port number of relevant drive.

*iSlaveNo*

Slave number of relevant drive. (must be ‘99’ )

### Return Value

No response

### Remarks

### Example

Refer to ‘FAS\_MoveSingleAxisAbsPos’ library.

### See Also

## FAS\_AIIMoveSingleAxisAbsPos

---

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

Syntax

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

Parameters

*nPortNo*

Port number of relevant drive.

*iSlaveNo*

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

*lAbsPos*

Absolute coordinate of position to move

*lVelocity*

Velocity when the motor moves

Return Value

No response

Remarks

Example

See Also

## FAS\_AIIMoveSingleAxisIncPos

---

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

Syntax

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

Parameters

*nPortNo*

Port number of relevant drive.

*iSlaveNo*

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

*lIncPos*

Incremental coordinate of position to move

*lVelocity*

Velocity when the motor moves

Return Value

No response

Remarks

Example

Refer to 'FAS\_MoveSingleAxisAbsPos' library.

See Also



## FAS\_MoveLinearAbsPos / FAS\_MoveLinearIncPos

---

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

### Syntax

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

### Parameters

*nPortNo*

Port number of relevant drive.

*nNoOfSlaves*

Slave numbers for Linear motioning.

*iSlaveNo*

Array of Slave numbers.

*lAbsPos*

Array of position value for each slave. (in case of incremental : Distance)

*lFeedrate*

Speed value for motioning.

*wAccelTime*

Acceleration & deceleration time value.

### 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

When this function is used, the 'Start speed' value must be designated to '1' .

## FAS\_MoveSingleAxisAbsPosEx

---

To move the motor to the absolute coordinate

Syntax

```
int FAS_MoveSingleAxisAbsPosEx(
    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;
    EZISERVO_AXISSTATUS stAxisStatus;
    long lAbsPos, lIncPos, lVelocity;
    MOTION_OPTION_EX opt = {0};
    int nRtn;

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

    // Moving motor with different acc/dec time
    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_MoveSingleAxisIncPosEx(  
    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_MoveVelocityEx(
    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_MoveSingleAxisIncPos Ex
    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 all items of position table
FAS_PosTableWriteItem	To save specific all items of position table items 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 64 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 one item of position table items 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 which item value is saved

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

```
#include "FAS_EzIMOTIONPlusR.h"  
  
void funcPosTable()  
{  
    BYTE nPortNo = 1; // COMM Port Number  
    BYTE iSlaveNo = 0; // Slave No (0 ~ 15)  
    WORD wItemNo;  
    ITEM_NODE nodelItem;  
    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, &nodelItem);  
    _ASSERT(nRtn == FMM_OK);  
  
    nodelItem.lPosition = 260000; // Change the position value to 260000.  
    nodelItem.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 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.

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 acts 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 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 .

FMC\_POSTABLE\_ERROR : An error occurs while position table is being saved.

Remarks

Position table data is saved to RAM / ROM area. This function acts 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 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 .

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 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\_GetAllStatus

FAS\_MoveStop

FAS\_EmergencyStop

## FAS\_PosTableReadOneItem

---

To read a one 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 in PT items . (Refer to ['1-2-6. Position Table Item'](#) )

### 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\_PosTableWriteOneItem

## FAS\_PosTableWriteOneItem

---

To edit one 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 save in PT items . (Refer to ['1-2-6. Position Table Item'](#) )

### 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.

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_MovePush	To request push motion(maintain specified motor torque) command
FAS_GetPushStatus	To request thecurrent push motion status



## 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

## FAS\_MovePush

---

To request push motion(maintain specified motor torque) command

Syntax

```
int FAS_MovePush(  
  BYTE nPortNo,  
  BYTE iSlaveNo,  
  DWORD dwStartSpd,  
  DWORD dwMoveSpd,  
  long lPosition,  
  WORD wAccel, WORD wDecel,  
  WORD wPushRate,  
  DWORD dwPushSpd,  
  long lEndPosition,  
  WORD wPushMode  
);
```

Parameters

*nPortNo*

Port number of relevant drive.

*iSlaveNo*

Slave number of relevant drive.

*DWORD dwStartSpd*

start speed of position command.

*DWORD dwMoveSpd*

move speed of position command.

*long lPosition*

absolute target position of position command.

*WORD wAccel*

Accel time of position command.

*WORD wDecel*

Deceleration time of position command.

*WORD wPushRate*

Torque ratio of Push motion.

*DWORD dwPushSpd*

Moce speed of Push motion.

*long lEndPosition*

absolute target position of push motion.

*WORD wPushMode*

Select the push mode (Stop or Non-stop)

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\_GetPushStatus

## FAS\_GetPushStatus

---

To request the current push motion status

Syntax

```
int FAS_MovePush(  
  BYTE nPortNo,  
  BYTE iSlaveNo,  
  BYTE* nPushStatus  
);
```

Parameters

*nPortNo*

Port number of relevant drive.

*iSlaveNo*

Slave number of relevant drive.

*BYTE\* nPushStatus*

Status value of push motion. (refer to ['1-2-1. FrameType and Data Configuration'](#) )

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 port s.

FMM\_INVALID\_SLAVE\_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

Example

See Also

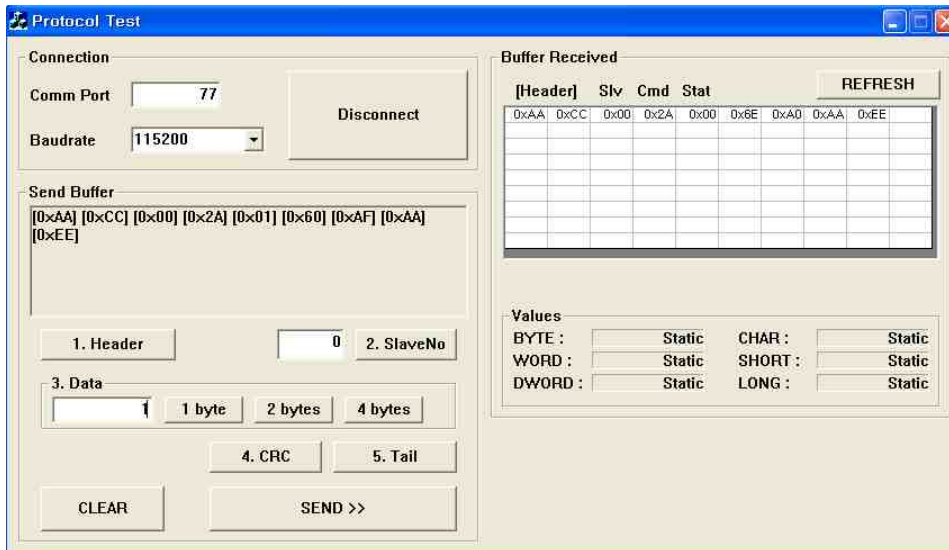
FAS\_Move Push

### 3. Protocol for PLC Program

Next window is open 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



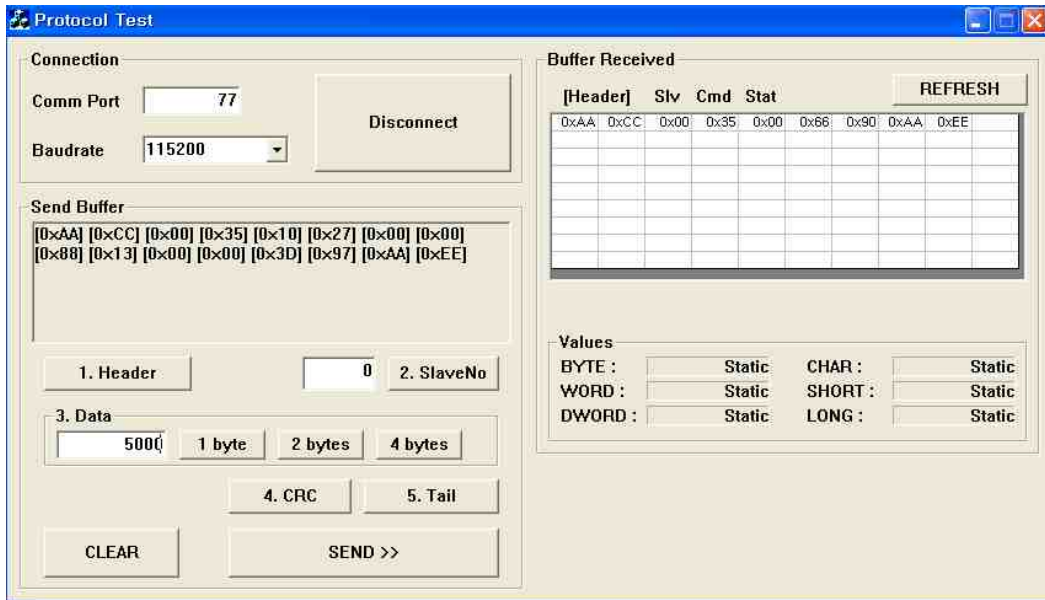
The header and tail information is needed for protocol programming. Additionally Frame Data (Slave ID, Frame type, Data and CRC) is also needed in every protocol with header and tail.

- 1) Insert 'Comm Port' number and click 'Connect' button.
- 2) Header : Click 'Header' and you can see '[0XAA][0xCC]' on 'Send Buffer' window.
- 3) Slave ID : Insert your slave number (above example is '0') and click 'SlaveNo'.
- 4) Frame type : Insert 'Frame type'.  
You can find next table information in '1-2-1. Frame Type and Data Configuration' on UserManual(EziSERVO ALL)\_Communication Function.

Frame type	DLL Library name	Data		
42 (0x2A)	FAS_ServoEnable	Setting the Servo ON/OFF status. Sending : 1 byte <table border="1" style="margin-left: 20px;"> <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				

- 5) Data : To make Servo ON status, the data is '1'. Insert '1' in  area and click '1 byte'.
- 6) CRC : Click 'CRC' and the 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 ALL.  
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 ALL on 'Buffer Received' window.

## (2) Motion command



- 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 : After sending command you can check the motor rotation and if click 'Send' more the motor will rotate one more time.

## (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(EziSERVO ALL)\_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. Mar 31, 2012 Rev.01.01.04