

Ezi-SERVO[®] Plus-R

Closed Loop Stepping System
with Network based Motion Controller

User Manual

Communication Function



www.fastech.co.kr

- Table of Contents -

1. Communication Protocols.....	5
1-1. Communication Functions	5
1-1-1. Communication Specifications	5
1-1-2. RS-485 Communication Protocol	5
1-1-3. CRC Calculation Example.....	6
1-1-4. Response Frame Structure and Communication Error.....	8
1-2. Structure of Frame type	9
1-2-1. Frame type and Data Configuration.....	9
1-2-2. Parameter Lists	19
1-2-3. Bit setup of Output pin.....	20
1-2-4. Bit setup of Input pin.....	20
1-2-5. Bit setup of Status Flag.....	21
1-2-6. Position Table Item.....	22
1-2-7. Information of Motors.....	22
1-3. Program Method.....	23
2. Library for PC Program.....	24
2-1. Library Configuration	24
2-2. Drive Link Function.....	25
FAS_Connect.....	26
FAS_Close.....	28
FAS_GetSlaveInfo.....	29
FAS_GetMotor Info.....	30
FAS_IsSlaveExist.....	31
2-3. Parameter Control Function.....	32
FAS_SaveAllParameters.....	33


FAS_SetParameter	35
FAS_GetParameter	36
FAS_GetROMParameter	37
2-4. Servo Control Function	38
FAS_ServoEnable.....	39
FAS_ServoAlarmReset.....	41
2-5. Control I/O Function.....	42
FAS_SetIOInput.....	43
FAS_GetIOInput.....	45
FAS_SetIOOutput.....	46
FAS_GetIOOutput.....	47
FAS_GetIOAssignMap.....	48
FAS_SetIOAssignMap.....	50
FAS_IOAssignMapReadROM.....	51
2-6. Position Control Function.....	52
FAS_SetCommandPos.....	53
FAS_SetActualPos.....	54
FAS_GetCommandPos.....	55
FAS_GetActualPos.....	57
FAS_GetPosError.....	58
FAS_GetActualVel.....	59
FAS_ClearPosition.....	60
2-7. Drive Status Control Function.....	61
FAS_GetIOAxisStatus.....	62
FAS_GetMotionStatus.....	63
FAS_GetAllStatus.....	64
FAS_GetAxisStatus.....	65
2-8. Running Control Function	66
FAS_IsMotioning.....	67

FAS_MoveStop.....	68
FAS_EmergencyStop.....	69
FAS_MoveOriginSingleAxis.....	70
FAS_MoveSingleAxisAbsPos.....	71
FAS_MoveSingleAxisIncPos.....	73
FAS_MoveToLimit.....	74
FAS_MoveVelocity.....	75
FAS_PositionAbsOverride.....	76
FAS_PositionIncOverride.....	78
FAS_VelocityOverride.....	79
FAS_AllMoveStop.....	80
FAS_AllEmergencyStop.....	81
FAS_AllMoveOriginSingleAxis.....	82
FAS_AllMoveSingleAxisAbsPos.....	83
FAS_AllMoveSingleAxisIncPos.....	84
2-9. Position Table Control Function.....	85
FAS_PosTableReadItem.....	86
FAS_PosTableWriteItem.....	88
FAS_PosTableWriteROM.....	89
FAS_PosTableReadROM.....	90
FAS_PosTableRunItem.....	91
3. Protocol for PLC Program.....	92

1. Communication Protocols

1-1. Communication Functions

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

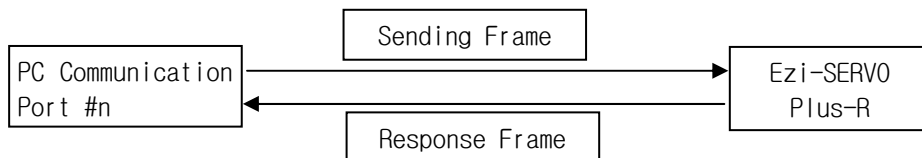
 Caution	<p>Pay attention that when Windows goes into standby or power-save mode, serial communication is basically disconnected. When the system is recovered from standby mode, it should be connected again with serial communication. This is also applicable to the library provided.</p>
--	---

1-1-1. Communication Specifications

Specification	RS-485
Communication Type	Synchronous
	Half-duplex
Baud Rate [bps]	9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600
Data Type	8bit ASCII Code, HEX
Parity	No
Stop Bit	1bit
CRC Check	Yes
Max Cabling Length (Converter ↔ Drive)	30 m
Min Cable length between drive	More than 60 cm
Number of Connected Axes	16 axes (No. 0~F)

1-1-2. RS-485 Communication Protocol

1) Overview of communication FRAME



2) Basic structure of Frame

Header	<i>Frame Data</i>	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


```

unsigned short CalcCRC(unsigned char *pDataBuffer, unsigned int DataLen)
{
    unsigned char    CRChi = 0xFF ; /* high byte of CRC initialized */
    unsigned char    CRCLo = 0xFF ; /* low byte of CRC initialized */
    unsigned int     Index ;      /* will index into CRC lookup table */

    while (DataLen--)            /* Pass through Data buffer */
    {
        Index = CRChi ^ *pDataBuffer++ ; /* calculate the CRC */
        CRChi = CRCLo ^ auchCRChi[Index] ;
        CRCLo = auchCRCLo[Index] ;
    }
    return (CRChi << 8 | CRCLo) ;
}

```

1-1-4. Response Frame Structure and Communication Error


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

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

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

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

Code	Description
0	Communication is normal.
128	Frame Type Error : Responded Frame type cannot be recognized.
129	Data error, ROM data read/write error : Data value responded is without the given range.
130	Received Frame Error : Frame data received is out of this specification.
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
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
135	Servo ON Failure ① : While an alarm occurs, the user has tried to execute Servo ON command.
136	Servo ON Failure ② : While Emergency Stop occurs, the user has tried to execute Servo ON command.

	<p>1) If 'Header' and 'Slave ID' values in the sending Frame are abnormal, there is no response from the drive.</p> <p>2) If the communication status is displayed to '130', the size of response data is '0' byte.</p>
---	---

1-2. Structure of Frame type

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</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 「1-1-7.Information of Motors」</p>	1 byte	1 bytes	0~246 bytes	Communication status	Motor type (1~255)	ACII string with NULL byte (strlen() + 1 bytes)
1 byte	1 bytes	0~246 bytes						
Communication status	Motor type (1~255)	ACII string with NULL byte (strlen() + 1 bytes)						
0x10 (16)	FAS_ SaveAllParameters	<p>Current setting parameters & 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' & '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 (0~29)</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 (0~29)	1 byte	4 bytes	Communication status	Parameter value
1 byte								
Parameter number (0~29)								
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="547 248 1157 327"> <tr> <td>1 byte</td> <td>4 bytes</td> </tr> <tr> <td>Parameter number (0~29)</td> <td>Parameter value</td> </tr> </table> <p>Response : 1 byte</p> <table border="1" data-bbox="547 405 861 483"> <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 (0~29)	Parameter value	1 byte	Communication status
1 byte	4 bytes							
Parameter number (0~29)	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="547 669 833 786"> <tr> <td>1 byte</td> </tr> <tr> <td>Parameter number (0~29)</td> </tr> </table> <p>Response : 5 bytes</p> <table border="1" data-bbox="547 842 1050 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 (0~29)	1 byte	4 bytes	Communication status	Parameter value
1 byte								
Parameter number (0~29)								
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="547 1099 1069 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="547 1429 841 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="547 1630 1035 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="547 1960 829 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="547 286 1093 344"> <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="547 573 1054 658"> <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="547 925 1222 983"> <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"> ◆ I/O number: '0~11' corresponds to 'Limit+, Limit-, Org, IN1, ..., IN9' respectively, and '12~22' corresponds to 'COMP, OUT1, ..., OUT9' respectively. ◆ I/O pin masking data: Refer to 「1-2-4. Bit setup of Input Pin」. ◆ Level Setting: 1:Active Low, 0:Active High <p>Response : 1 byte</p> <table border="1" data-bbox="547 1223 831 1281"> <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="547 1386 799 1464"> <tr> <td>1 byte</td> </tr> <tr> <td>I/O number</td> </tr> </table> <ul style="list-style-type: none"> ◆ I/O number: '0~11' corresponds to 'Limit+, Limit-, Org, IN1, ..., IN9' respectively, and '12~22' corresponds to 'COMP, OUT1, ..., OUT9' respectively. <p>Response : 6 bytes</p> <table border="1" data-bbox="547 1668 1323 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="547 315 1323 423"> <tr> <td data-bbox="547 315 823 353">1 byte</td> <td data-bbox="823 315 1323 353">1 byte</td> </tr> <tr> <td data-bbox="547 353 823 423">Communication status</td> <td data-bbox="823 353 1323 423">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)					
0x2A (42)	FAS_ ServoEnable	<p>Servo ON/OFF status is set.</p> <p>Sending : 1 byte</p> <table border="1" data-bbox="547 591 799 669"> <tr> <td data-bbox="547 591 799 629">1 byte</td> </tr> <tr> <td data-bbox="547 629 799 669">0:OFF, 1:ON</td> </tr> </table> <p>Response : 1 byte</p> <table border="1" data-bbox="547 732 823 810"> <tr> <td data-bbox="547 732 823 770">1 byte</td> </tr> <tr> <td data-bbox="547 770 823 810">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="547 956 834 1034"> <tr> <td data-bbox="547 956 834 994">1 byte</td> </tr> <tr> <td data-bbox="547 994 834 1034">Communication status</td> </tr> </table>	1 byte	Communication status		
1 byte						
Communication status						
0x30 (48)	FAS_ IsMotioning	<p>To check that the motor is running. It is irrelevant to inposition status.</p> <p>Sending : 0 byte</p> <p>Response : 2 bytes</p> <table border="1" data-bbox="547 1245 1074 1323"> <tr> <td data-bbox="547 1245 834 1283">1 byte</td> <td data-bbox="834 1245 1074 1283">1 byte</td> </tr> <tr> <td data-bbox="547 1283 834 1323">Communication status</td> <td data-bbox="834 1283 1074 1323">0: Stop, 1: Running</td> </tr> </table>	1 byte	1 byte	Communication status	0: Stop, 1: Running
1 byte	1 byte					
Communication status	0: Stop, 1: Running					
0x31 (49)	FAS_ MoveStop	<p>To request to stop running the motor</p> <p>Sending : 0 byte</p> <p>Response : 1 byte</p> <table border="1" data-bbox="547 1556 844 1635"> <tr> <td data-bbox="547 1556 844 1594">1 byte</td> </tr> <tr> <td data-bbox="547 1594 844 1635">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</p> <p>Response : 1 byte</p> <table border="1" data-bbox="547 1843 849 1921"> <tr> <td data-bbox="547 1843 849 1881">1 byte</td> </tr> <tr> <td data-bbox="547 1881 849 1921">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="547 322 828 389"> <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="547 546 1102 613"> <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="547 689 831 757"> <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="547 920 1102 1016"> <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 1055 831 1122"> <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 1263 1297 1330"> <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 1368 831 1435"> <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 1576 1307 1644"> <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 1682 853 1749"> <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 1890 1037 1957"> <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 1995 841 2063"> <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="547 302 1051 380"> <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="547 416 847 495"> <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="547 607 944 685"> <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="547 721 860 799"> <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	<p>To request stop for all motor that connected in same port.</p> <p>Sending : 0 byte (Slave number must be '99')</p> <p>Response : no response</p>				
0x3C (60)	FAS_ AllEmergencyStop	<p>To request emergency stop for all motor that connected in same port.</p> <p>Sending : 0 byte (Slave number must be '99')</p> <p>Response : no response</p>				
0x3D (61)	FAS_All MoveOriginSingleAxis	<p>To request return to the origin at the current setting parameter condition for all motors that connected in same port.</p> <p>Sending : 0 byte (Slave number must be '99')</p> <p>Response : no response</p>				
0x3E (62)	FAS_All SingleAxisAbsPos	<p>To request move its position as much as the absolute value[pulse] for all motors that connected in same port.</p> <p>Sending : 8 bytes (Slave number must be '99')</p> <table border="1" data-bbox="547 1682 1102 1751"> <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 : no response</p>	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	<p>To request move its position as much as the incremental value[pulse] for all motors that connected in same port.</p> <p>Sending : 8 bytes (Slave number must be '99')</p> <table border="1" data-bbox="547 275 1174 342"> <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 : no response</p>	4 bytes	4 bytes	incremental position value	Running speed [pps]														
4 bytes	4 bytes																			
incremental position value	Running speed [pps]																			
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="547 656 1050 734"> <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 1041 1347 1155"> <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="547 1393 1409 1547"> <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="547 1736 1308 1836"> <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="547 1863 1351 1995"> <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 Plus-R 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="547 405 1003 479"> <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="547 517 828 591"> <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</p> <p>Response : 5 bytes</p> <table border="1" data-bbox="547 736 1109 810"> <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 Plus-R 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="547 1068 1003 1142"> <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="547 1180 863 1254"> <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="547 1404 1139 1478"> <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="547 1718 1163 1792"> <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 Response : 5 bytes</p> <table border="1" data-bbox="549 304 1050 383"> <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 Plus-R 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 Response : 1 byte</p> <table border="1" data-bbox="549 689 863 768"> <tr> <td>1 byte</td> </tr> <tr> <td>Communication status</td> </tr> </table>	1 byte	Communication status						
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="549 965 928 1043"> <tr> <td>2 bytes</td> </tr> <tr> <td>Readable PT number (0~255)</td> </tr> </table> <p>Response : 65 bytes</p> <table border="1" data-bbox="549 1077 1118 1155"> <tr> <td>1 byte</td> <td>64 bytes</td> </tr> <tr> <td>Communication status</td> <td>Relevant PT values</td> </tr> </table> <p>For items by each PT, refer to 「1-2-6. Position Table Item」.</p>	2 bytes	Readable PT number (0~255)	1 byte	64 bytes	Communication status	Relevant PT values		
2 bytes										
Readable PT number (0~255)										
1 byte	64 bytes									
Communication status	Relevant PT values									
0x61 (97)	FAS_ PosTableWriteItem	<p>To save PT values to the RAM of the drive.</p> <p>Sending : 66 bytes</p> <table border="1" data-bbox="549 1317 1050 1395"> <tr> <td>2 bytes</td> <td>64 bytes</td> </tr> <tr> <td>PT number (0~255)</td> <td>Relevant PT value</td> </tr> </table> <p>For items by each PT, refer to 「1-2-6. Position Table Item」.</p> <p>Response : 2 bytes</p> <table border="1" data-bbox="549 1469 1326 1563"> <tr> <td>1 byte</td> <td>1 byte</td> </tr> <tr> <td>Communication status</td> <td>Command performing status (values except 0 : complete, 0: error)</td> </tr> </table>	2 bytes	64 bytes	PT number (0~255)	Relevant PT value	1 byte	1 byte	Communication status	Command performing status (values except 0 : complete, 0: error)
2 bytes	64 bytes									
PT number (0~255)	Relevant PT value									
1 byte	1 byte									
Communication status	Command performing status (values except 0 : complete, 0: error)									
0x62 (98)	FAS_ PosTableReadROM	<p>To read all PT values (256 ea) in the ROM of the drive</p> <p>Sending : 0 byte Response : 2 bytes</p> <table border="1" data-bbox="549 1753 1326 1848"> <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(256 ea) to the ROM of the drive.</p> <p>Sending : 0 byte Response : 2 bytes</p> <table border="1" data-bbox="549 304 1323 398"> <tr> <td data-bbox="549 304 836 338">1 byte</td> <td data-bbox="836 304 1323 338">1 byte</td> </tr> <tr> <td data-bbox="549 338 836 398">Communication status</td> <td data-bbox="836 338 1323 398">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="549 517 799 595"> <tr> <td data-bbox="549 517 799 551">2 bytes</td> </tr> <tr> <td data-bbox="549 551 799 595">PT Number (0~255)</td> </tr> </table> <p>Response : 1 byte</p> <table border="1" data-bbox="549 629 836 707"> <tr> <td data-bbox="549 629 836 663">1 byte</td> </tr> <tr> <td data-bbox="549 663 836 707">Communication status</td> </tr> </table>	2 bytes	PT Number (0~255)	1 byte	Communication status
2 bytes						
PT Number (0~255)						
1 byte						
Communication status						

* Frame Type '0x65' ~ '0x69' 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]	0	+134,217,727	2,500
28	Motion Dir		0	1	0
29	Limit Sensor Dir		0	1	0

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.

Name of Flag Define	Contents	Relevant Bit Position
FFLAG_ERRORALL	One or more error occurs.	0X00000001
FFLAG_HWPOSITLMT	'+' 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	Inposition error is larger than 'Pos Tracking Limit' .	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_ERRREVPWR	A counter electromotive force of the motor exceeds 70V.	0X00002000
FFLAG_ERRMOTORPOWER	The motor voltage is abnormal.	0X00004000
FFLAG_ERRINPOSITION	The motor is out of inposition.	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_PTSTOPPED	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	Unit	Low Limit	Upper Limit
Position	lPosition	4 (signed)	[pulse]	-134217728	+134217728
Low Speed	dwStartSpd	4 (unsigned)	[pps]	0	500000
High Speed	dwMoveSpd	4 (unsigned)	[pps]	0	500000
Accel. Time	wAccelRate	2 (unsigned)	[msec]	1	9999
Decel. Time	wDecelRate	2 (unsigned)	[msec]	1	9999
Command	wCommand	2 (unsigned)		0	9
Wait time	wWaitTime	2 (unsigned)	[msec]	0	600000
Continuous Action	wContinuous	2 (unsigned)		0	1
Jump Table No.	wBranch	2 (unsigned)		0 10000	255 10255
Jump PT 0	wCond_branch0	2 (unsigned)		0 10000	255 10255
Jump PT 1	wCond_branch1	2 (unsigned)		0 10000	255 10255
Jump PT 2	wCond_branch2	2 (unsigned)		0 10000	255 10255
Loop Count	wLoopCount	2 (unsigned)		0	100
Loop Jump Table No.	wBranchAfterLoop	2 (unsigned)		0 10000	255 10255
PT set	wPTSet	2 (unsigned)		0	15
Loop Counter Clear	wLoopCountCLR	2 (unsigned)		0	255
Check Inposition	bCheckInpos	2 (unsigned)		0	1
Blank		24 (unsigned)		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 Plus-R.

The first is normally used method that using Visual C++ language under window system of PC. Library that serviced together with Ezi-SERVO Plus-R 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 Plus-R. The user have to prepare low level protocol programming like a hyper terminal. This method is normally used for PLC system. For excise the protocol programming 'Protocol Test.exe' GUI program is serviced together. Refer to [「3. Protocol for PLC Program」](#).

2. Library for PC Program

2-1. Library Configuration

To use this library, C++ header file(*.h) and library file(*.lib or *.dll) are required. These files 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	Description
Normal	FMM_OK	The function has normally performed the command.
Input Error	FMM_NOT_OPEN	Wrong port number is inputted.
	FMM_INVALID_SLAVE_NUM	Wrong slave number is inputted.
	FMM_INVALID_PARAMETER_NUM	Wrong parameter number or wrong function factor is inputted.
Operation Error	FMM_POSTABLE_ERROR	An error occurs while the motor accesses to the position table.
Connection Error	FMC_DISCONNECTED	The relevant drive is disconnected.
	FMC_TIMEOUT_ERROR	Response delay(100 msec) occurs.
	FMC_CRCFAILED_ERROR	Checksum error occurs.
	FMC_RECVPACKET_ERROR	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	Description
Normal	FMP_OK	Communication has been normally performed.
Input Error	FMP_FRAMETYPEERROR	The drive cannot recognize the command.
	FMP_DATAERROR	Input data is out of the range.
Operation Error	FMP_BUSYMOTOR	The motor is running. Please wait until it stops running.
	FMP_RESETFAIL	The user cannot execute AlarmReset command while the servo is ON.
	FMP_SERVOONFAIL1	An alarm has occurred.
	FMP_SERVOONFAIL2	The motor is under Emergency Stop.
Connection Error	FMP_PACKETERROR	Protocol level error occurs in packet that Drive's received.

2-2. 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 Plus-R.

Syntax

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

Parameters

nPortNo

Select a serial port to be connected.

dwBaud

Input the Baudrate of the serial port.

Return Value

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

Remarks

Example

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

```
        // Disconnected.  
        FAS_Close();  
    }
```

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

2-3. 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 lParamVal;  
    int nRtn;  
  
    // Connected.  
    if (FAS_Connect(nPortNo, 115200) == FALSE)  
    {  
        // Connection fails.  
        // The port is not connected or the baud rate may be wrong.  
        return;  
    }  
  
    // Check Axis Start Speed Parameter.  
    nRtn = FAS_GetParameter(nPortNo, iSlaveNo, AXIS_AXISSTARTSPEED, &lParamVal);  
    if (nRtn != FMM_OK)  
    {  
        // Command has not been performed normally.  
        // Refer to ReturnCodes_Define.h.  
        _ASSERT(FALSE);  
    }  
    else  
    {  
        // Parameter value being saved to Ezi-SERVO.  
        printf("Parameter [before] : Start Speed = %d Wn", lParamVal);  
    }  
}
```

```

// Change Start Speed Parameter into 200 and then read it again.
nRtn = FAS_SetParameter(nPortNo, iSlaveNo, AXIS_AXISSTARTSPEED, 200);
_ASSERT(nRtn == FMM_OK); // If command is not performed normally, the motor will
stop.

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

// Check the value saved to the ROM.
nRtn = FAS_GetROMParameter(nPortNo, iSlaveNo, AXIS_AXISSTARTSPEED, 200);
_ASSERT(nRtn == FMM_OK); // If command is not performed normally, the motor will
stop.
printf("Parameter [ROM] : Start Speed = %d Wn", IParamVal);

// Edit the parameter value and then save it to the ROM.
nRtn = FAS_SetParameter(nPortNo, iSlaveNo, AXIS_AXISSTARTSPEED, 100);
_ASSERT(nRtn == FMM_OK); // If command is not performed normally, the motor will
stop.

nRtn = FAS_SaveAllParameters(nPortNo, iSlaveNo);
_ASSERT(nRtn == FMM_OK);

// Disconnected.
FAS_Close();
}

```

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 IParamValue  
);
```

Parameters

nPortNo
Port number of relevant drive

iSlaveNo
Slave number of relevant drive

iParamNo
Parameter number to be edited

IParamValue
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

2-4. 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_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)  
    DWORD dwAxisStatus;  
    int nRtn;  
  
    // Connected.  
    if (FAS_Connect(nPortNo, 115200) == FALSE)  
    {  
        // Connection fails.  
        // The port is not connected or the baud rate may be wrong.  
        return;  
    }  
  
    nRtn = FAS_GetAxisStatus(nPortNo, iSlaveNo, &dwAxisStatus);  
    _ASSERT(nRtn == FMM_OK);  
  
    // If SERVO_ON flag turns off, the servo turns on.  
    if ((dwAxisStatus & FFLAG_SERVOON) == 0)  
    {  
        nRtn = FAS_ServoEnable(nPortNo, iSlaveNo, TRUE);  
        _ASSERT(nRtn == FMM_OK);  
    }  
  
    // If there is an alarm, AlarmReset runs.  
    if (dwAxisStatus & (FFLAG_ERRORALL | FFLAG_ERROVERCURRENT | FFLAG_ERROVERLOAD))  
    {  
        nRtn = FAS_ServoAlarmReset(nPortNo, iSlaveNo);  
        _ASSERT(nRtn == FMM_OK);  
    }  
}
```

```
        // Disconnected.  
        FAS_Close();  
    }
```

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

2-5. 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 9 variable signals returns by bit to the Input and Output port.
FAS_SetI0AssignMap	To assign the control I/O signal to CN1 port pin and also set the signal level : Setting for each 9 variable signals is assigned to the Input and Output port.
FAS_I0AssignMapReadROM	To load the pin setting status of CN1 port from ROM area to RAM area.

FAS_SetIOInput

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

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

dwIOSetMask

Input bitmask value to be set

dwIOCLRMask

Input bitmask value to be cleared

Return Value

FMM_OK : Command has been 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 dwIOSet, dwIOClr, dwInput, dwOutput;  
    int nRtn;  
  
    // Connected.  
    if (FAS_Connect(nPortNo, 115200) == FALSE)  
    {  
        // Connection fails.  
        // The port is not connected or the baud rate may be wrong.  
        return;  
    }  
  
    // Check I/O Input.  
    nRtn = FAS_GetIOInput(nPortNo, iSlaveNo, &dwInput);  
    _ASSERT(nRtn == FMM_OK);  
    if (dwInput & IN_BITMASK_LIMITP)  
    {  
        // Limit+ input is On.  
    }  
}
```

```

    if (dwInput & IN_BITMASK_USERIN0)
    {
        // User Input 0 is On.
    }

    // Turn on Clear Position and User Input 1 and then turn off Jog+ Input.
    nRtn = FAS_SetIOInput(nPortNo, iSlaveNo, IN_BITMASK_CLEARPOSITION |
IN_BITMASK_USERIN1, IN_BITMASK_PJOG);
    _ASSERT(nRtn == FMM_OK);

    // Check I/O Output.
    nRtn = FAS_GetIOOutput(nPortNo, iSlaveNo, &dwOutput);
    _ASSERT(nRtn == FMM_OK);
    if (dwOutput & OUT_BITMASK_USEROUT0)
    {
        // User output 0 is On.
    }

    // Turn off User Output 1 and 2 signals.
    nRtn = FAS_SetIOOutput(nPortNo, iSlaveNo, 0, OUT_BITMASK_USEROUT1 |
OUT_BITMASK_USEROUT2);
    _ASSERT(nRtn == FMM_OK);

    // Disconnected.
    FAS_Close();
}

```

See Also

FAS_GetIOInput

FAS_GetIOInput

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

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

dwIOInput

Parameter pointer which input values will be saved

Return Value

FMM_OK : Command has been normally performed.

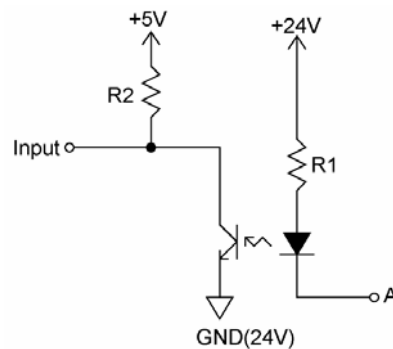
FMM_NOT_OPEN : The drive has not been connected yet.

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 12 input pins in EziSERVO PlusR. The user can select and use 9 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 '1-1-5. Frame Type and Data Configuration' .

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

dwIOSetMask

Output bitmask value to be set

dwIOCLRMask

Output bitmask value be cleared

Return Value

FMM_OK : Command has been normally performed.

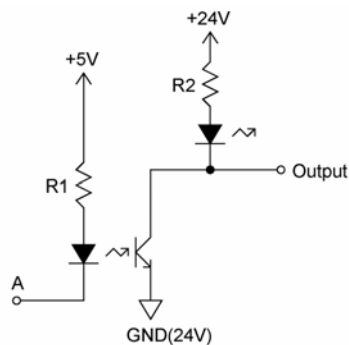
FMM_NOT_OPEN : The drive has not been connected yet.

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 10 input pins in EziSERVO PlusR. The user can select and use 9 output pins of them.



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

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

Example

Refer to FAS_SetIOInput.

See Also

FAS_GetIOOutput

FAS_GetIOOutput

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

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

dwIOInput

Parameter pointer 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 '1-1-5. Frame Type and Data Configuration' .

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, nLogicNo, bLevel;  
    BYTE i;  
    int nRtn;  
  
    // Connected.  
    if (FAS_Connect(nPortNo, 115200) == FALSE)  
    {  
        // Connection fails.  
        // The port is not connected or the baud rate may be wrong.  
        return;  
    }  
  
    // Check input pin assign information.  
    for (i=0; i<IN_PIN_CNT; i++)  
    {  
        nRtn = FAS_GetIOAssignMap(nPortNo, iSlaveNo, i, &nLogicNo, &bLevel);  
    }  
}
```

```

        _ASSERT(nRtn == FMM_OK);

        if (nLogicNo < IN_LOGIC_CNT)
            printf("Input Pin %d : Logic No %d (%s)Wn", i, nLogicNo, ((bLevel
== LEVEL_LOW_ACTIVE) ? "Low Active" : "High Active"));
        else
            printf("Input Pin %d : Not assignedWn", i);
    }

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

    // Check output pin assign information.
    for (i=0; i<OUT_PIN_CNT; i++)
    {
        nRtn = FAS_GetIOAssignMap(nPortNo, iSlaveNo, IN_PIN_CNT + i, &nLogicNo,
&bLevel);
        _ASSERT(nRtn == FMM_OK);

        if (nLogicNo < OUT_LOGIC_CNT)
            printf("Output Pin %d : Logic No %d (%s)Wn", i, nLogicNo, ((bLevel
== LEVEL_LOW_ACTIVE) ? "Low Active" : "High Active"));
        else
            printf("Output Pin %d : Not assignedWn", i);
    }

    // Assign ALARM Logic (High Active) to output pin 9.
    iPinNo = 9;          // 0 ~ 9 vaue is available. (Caution : 0 is fixed to COMPOUT.).
    nRtn = FAS_SetIOAssignMap(nPortNo, iSlaveNo, IN_PIN_CNT + iPinNo, OUT_LOGIC_ALARM,
LEVEL_HIGH_ACTIVE);
    _ASSERT(nRtn == FMM_OK);

    // Disconnected.
    FAS_Close();
}

```

See Also

FAS_SetIOAssignMap

FAS_SetIOAssignMap

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

Syntax

```
int FAS_SetIOAssignMap(  
    BYTE nPortNo,  
    BYTE iSlaveNo,  
    BYTE iIOPinNo,  
    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

2-6. 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;  
  
    // Connected.  
    if (FAS_Connect(nPortNo, 115200) == FALSE)  
    {  
        // Connection fails.  
        // The port is not connected or the baud rate may be wrong.  
        return;  
    }  
    // Initialize Command Position and Actual Position values  
    nRtn = FAS_SetCommandPos(nPortNo, iSlaveNo, 0);  
    _ASSERT(nRtn == FMM_OK);  
    nRtn = FAS_SetActualPos(nPortNo, iSlaveNo, 0);  
    _ASSERT(nRtn == FMM_OK);  
  
    // Disconnected.  
    FAS_Close();  
}
```

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;  
  
    // Connected.  
    if (FAS_Connect(nPortNo, 115200) == FALSE)  
    {  
        // Connection fails.  
        // The port is not connected or the baud rate may be wrong.  
        return;  
    }  
  
    // Check position information of Ezi-SERVO.  
    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);  
  
        // Disconnected.  
        FAS_Close();  
    }  
}
```

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;  
  
    // Connected.  
    if (FAS_Connect(nPortNo, 115200) == FALSE)  
    {  
        // Connection fails.  
        // The port is not connected or the baud rate may be wrong.  
        return;  
    }  
    // Initialize Command Position and Actual Position values to '0'  
    nRtn = FAS_ClearPosition(nPortNo, iSlaveNo);  
    _ASSERT(nRtn == FMM_OK);  
    // Disconnected.  
    FAS_Close();  
}
```

See Also

FAS_SetActualPos

2-7. 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 '1-1-5. Frame Type and Data Configuration' .

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

dwAxisStatus

Parameter pointer 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

2-8. Running Control Function

Function Name	Description
FAS_IsMotioning	To check that the motor is running : When it is running, TRUE will return. When it stops, FALSE will return.
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_IsMotioning

To check that the motor is running

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

Return Value

TRUE : The motor is running.

FALSE : The motor stops.

Remarks

This function checks motioning, motion accel, motion decel, motion continuous out of Flat bits in FAS_GetAxisStatus function, and returns the result.

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_GetAxisStatus

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;  
    long lAbsPos, lIncPos, lVelocity;  
    int nRtn;  
  
    // Connected.  
    if (FAS_Connect(nPortNo, 115200) == FALSE)  
    {  
        // Connection fails.  
        // The port is not connected or the baud rate may be wrong.  
        return;  
    }  
  
    // Check error and Servo On status.  
    nRtn = FAS_GetAxisStatus(nPortNo, iSlaveNo, &dwAxisStatus);  
    _ASSERT(nRtn == FMM_OK);  
  
    if (dwAxisStatus & FFLAG_ERRORALL)  
        FAS_ServoAlarmReset(nPortNo, iSlaveNo);  
    if ((dwAxisStatus & FFLAG_SERVOON) == FALSE)  
        FAS_ServoEnable(nPortNo, iSlaveNo, TRUE);  
}
```

```

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

if (dwInput & (IN_LOGIC_STOP | IN_LOGIC_PAUSE | IN_LOGIC_PSTOP | IN_LOGIC_NSTOP |
IN_LOGIC_ESTOP))
    FAS_SetIOInput(nPortNo, iSlaveNo, 0, IN_LOGIC_STOP | IN_LOGIC_PAUSE |
IN_LOGIC_PSTOP | IN_LOGIC_NSTOP | 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.
while (FAS_IsMotioning(nPortNo, iSlaveNo))
    Sleep(1);

// 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.
while (FAS_IsMotioning(nPortNo, iSlaveNo))
    Sleep(1);

// Disconnected.
FAS_Close();
}

```

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.

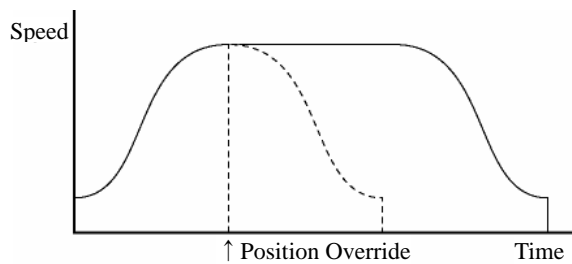
FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

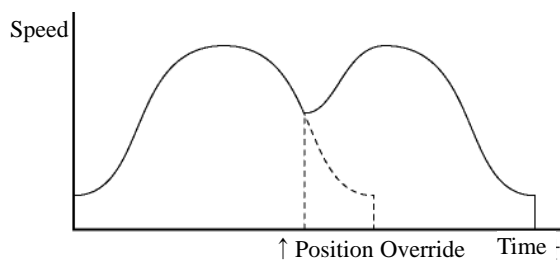
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.

Remarks

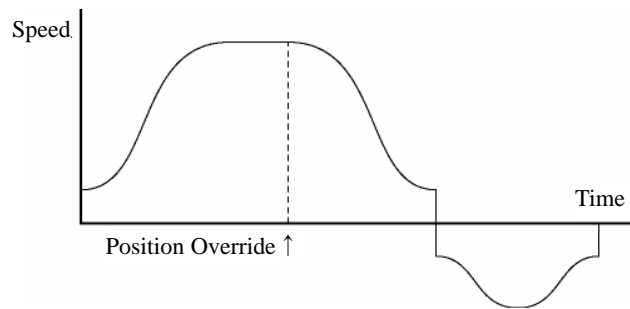
- 1) If the target position is set to the farther coordinate than the original target position while the motor moves to the accelerated or uniform velocity, the motor moves to the velocity pattern until then and stops the target position.



- 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

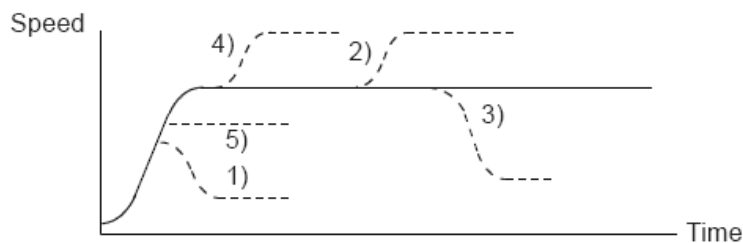
FMM_OK : Command has been normally performed.

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_AIIMoveStop

To stop the motor that connected in same port.

Syntax

```
int FAS_AIIMoveStop(  
    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_AIEmergencyStop

To stop the motor that connected in same port without deceleration

Syntax

```
int FAS_AIEmergencyStop(  
    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_AIIMoveOriginSingleAxis

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_AIIMoveOriginSingleAxis(  
    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

2-9. Position Table Control Function

Function Name	Description
FAS_PosTableReadItem	To read items of RAM area in the specific position table
FAS_PosTableWriteItem	To save specific position table 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 256 PT values are read.
FAS_PosTableRunItem	The motor starts to run from the designated position table in sequence.

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

```
void funcMove()  
{  
    BYTE nPortNo = 1; // COMM Port Number  
    BYTE iSlaveNo = 0; // Slave No (0 ~ 15)  
    WORD wItemNo;  
    ITEM_NODE nodelItem;  
    int nRtn;  
  
    // Connected.  
    if (FAS_Connect(nPortNo, 115200) == FALSE)  
    {  
        // Connection fails.  
        // The port is not connected or the baud rate may be wrong.  
        return;  
    }  
  
    // Read No.20 Position Table vaule 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 into 260000.  
    nodelItem.wBranch = 23; // Set next command to 23.  
    nodelItem.wContinuous = 1; // Next command should be connected without  
    deceleration.
```

```
nRtn = FAS_PosTableWriteItem(nPortNo, iSlaveNo, wItemNo, nodelItem);
_ASSERT(nRtn == FMM_OK);

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

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

// Disconnected.
FAS_Close();
}
```

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

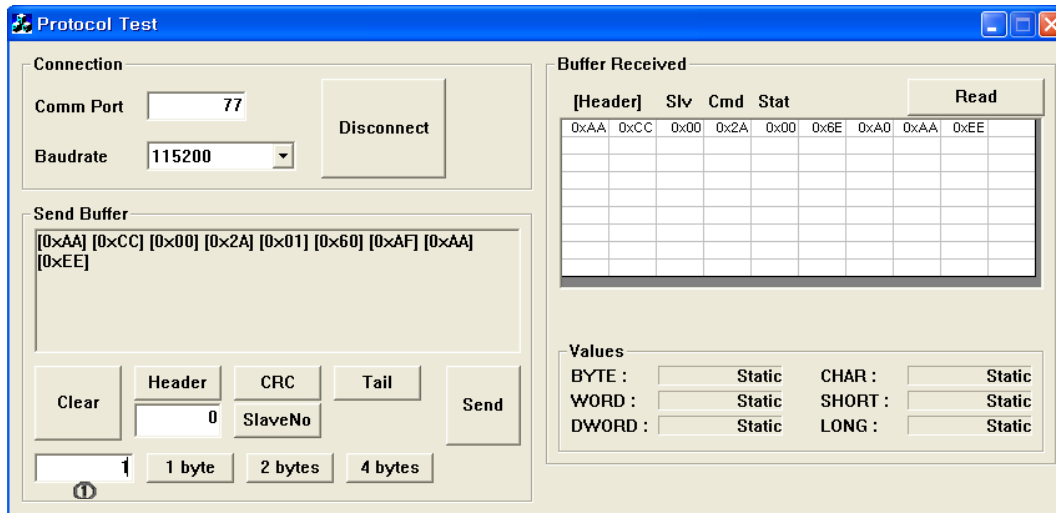
FAS_EmergencyStop

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



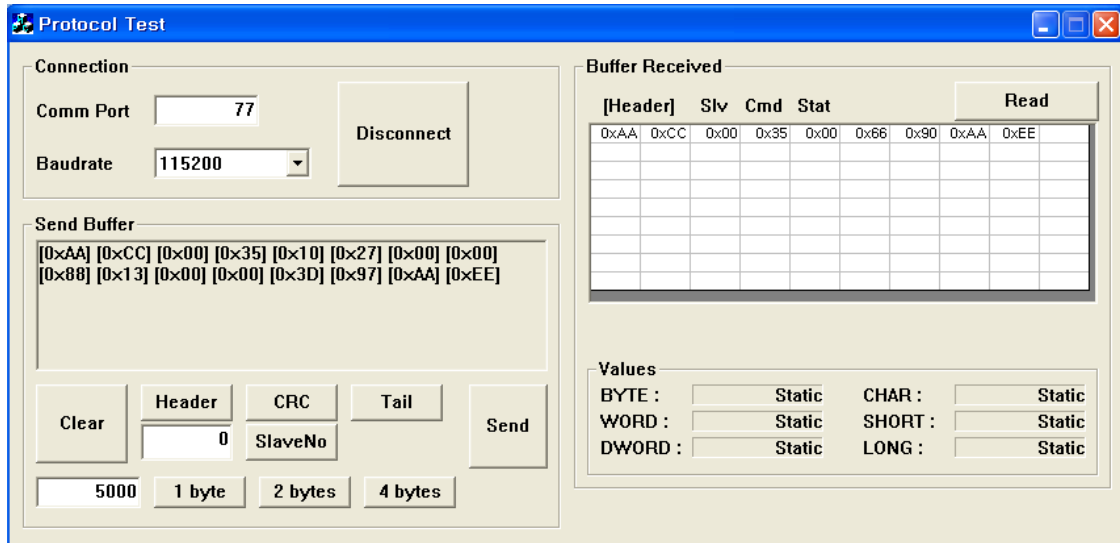
The header and tail information is needed for protocol programming. Additionally Frame Data (Slave ID, Frame type, Data and CRC) is also needed in every protocol with header and tail.

- 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 PlusR)_Communication Function.

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

- 5) Data : To make Servo ON status, the data is '1'. Insert '1' in  area and click .
- 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 Plus-R.
You can check the motor torque and LED flash for Servo ON status.
- 9) After sending command you can check the answering informations from Ezi-SERVO Plus-R 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 PlusR)_Communication Function.



FASTECH Co., Ltd.

Rm #1403, Bucheon Technopark 401 Dong,
Yakdae-dong, Wonmi-Gu, Bucheon-si,
Gyeonggi-do, Rep. Of Korea (Zip:420-734)
TEL : 82-32-234-6300
FAX : 82-32-234-6302
Email : fastech@empal.com
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. May 15, 2009 rev.04.01.02