DINGS
Precision Motion Specialist

VER 1.0

Technical Manual

DS-CL28-SA / DS-CL42-SA

DINGS
Precision Motion Specialist

Table of Contents

Click to return to table of contents

1. Communication Protocols 5

1.1 Communication Functions..... 5

1.1.1 Communication Specifications..... 6

1.1.2 RS-485 Communication Protocol..... 6

1.1.3 CRC Calculation Example..... 8

1.1.4 Response Frame Format and Communication Error..... 11

1.2 Definition of Frame Type..... 13

1.2.1 Frame Type of DS-CL28/42-SA..... 13

1.2.2 Frame Type and Data Configuration..... 15

1.2.3 Parameter Lists..... 27

1.2.4 Definition of Input Pin..... 28

1.2.5 Definition of Status Flag..... 29

1.2.6 Position Table Item..... 30

1.2.7 Information of Motors..... 31

1.3 Programming..... 31

2. Library for PC Program..... 32

2.1 Library Configuration..... 32

2.2 Communication Status window..... 34

2.3 Drive Link Function..... 40

2.3.1 FAS_Connect..... 41

2.3.2 FAS_Close..... 43

2.3.3 FAS_GetSlaveInfo..... 44

2.3.4 FAS_GetMotorInfo..... 45

2.3.5 FAS_IsSlaveExist..... 46

2.3.6 FAS_EnableLog..... 47

2.3.7 FAS_SetLogPath..... 48

2.4 Parameter Control Function.....50
2.4.1 FAS_SaveAllParameters.....51
2.4.2 FAS_SetParameter.....53
2.4.3 FAS_GetParameter.....54
2.4.4 FAS_GetROMParameter.....55
2.5 Servo Control Function.....56
2.5.1 FAS_ServoEnable.....57
2.5.2 FAS_ServoAlarmReset.....59
2.6 Control I/O Function.....60
2.6.1 FAS_SetIOInput.....61
2.6.2 FAS_GetIOInput.....63
2.6.3 FAS_GetIOAssignMap.....64
2.6.4 FAS_SetIOAssignMap.....67
2.6.5 FAS_IOAssignMapReadROM.....68
2.7 Position Control Function.....69
2.7.1 FAS_SetCommandPos.....70
2.7.2 FAS_SetActualPos.....72
2.7.3 FAS_GetCommandPos.....73
2.7.4 FAS_GetActualPos.....75
2.7.5 FAS_GetPosError.....76
2.7.6 FAS_GetActualVel.....77
2.7.7 FAS_ClearPosition.....78
2.8 Drive Status Control Function.....80
2.8.1 FAS_GetIOAxisStatus.....81
2.8.2 FAS_GetMotionStatus.....82
2.8.3 FAS_GetAllStatus.....83
2.8.4 FAS_GetAxisStatus.....85

2.9 Running Control Function.....86
2.9.1 FAS_MoveStop.....87
2.9.2 FAS_EmergencyStop.....88
2.9.3 FAS_MoveOriginSingleAxis.....89
2.9.4 FAS_MoveSingleAxisAbsPos.....90
2.9.5 FAS_MoveSingleAxisIncPos.....93
2.9.6 FAS_MoveToLimit.....94
2.9.7 FAS_MoveVelocity.....95
2.9.8 FAS_AllMoveStop.....96
2.9.9 FAS_AllEmergencyStop.....97
2.9.10 FAS_AllMoveOriginSingleAxis.....98
2.9.11 FAS_AllMoveSingleAxisAbsPos.....99
2.9.12 FAS_AllMoveSingleAxisIncPos.....100
2.9.13 FAS_MoveSingleAxisAbsPosEx.....101
2.9.14 FAS_MoveSingleAxisIncPosEx.....104
2.9.15 FAS_MoveVelocityEx.....105
2.10 Position Table Control Function.....107
2.10.1 FAS_PosTableReadItem.....108
2.10.2 FAS_PosTableWriteItem.....110
2.10.3 FAS_PosTableWriteROM.....111
2.10.4 FAS_PosTableReadROM.....112
2.10.5 FAS_PosTableRunItem.....113
2.10.6 FAS_PosTableReadOneItem.....114
2.10.7 FAS_PosTableWriteOneItem.....115
3. Protocol for PLC Program.....116
3.1 Servo On/Off Command.....116
3.2 Motion Command.....118
3.3 PLC Programming.....118

1. Communication Protocols

1.1 Communication Functions

DS-CL28/42-SA can control up to 16 axis by multidrop link at RS-485

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

1.1.1 Communication Specifications

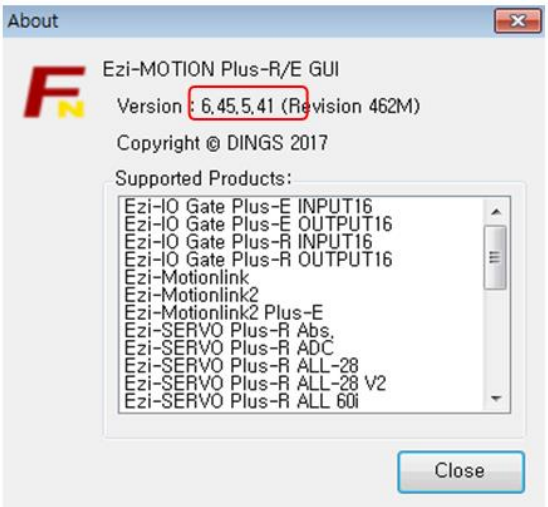
Specification	RS-485
Communication Type	Asynchronous
	Half-duplex
Baud Rate [bps]	115,200
Data Type	8bit Binary
Parity	No
Stop Bit	1bit
CRC Check	Yes
Max Cabling Length (Converter ↔ Drive)	30 m
Min Cable length between drive	More than 60 cm
Number of Connected Axis	16 axis (No. 0~F)

1.1.2 RS-485 Communication Protocol (Ver 6)

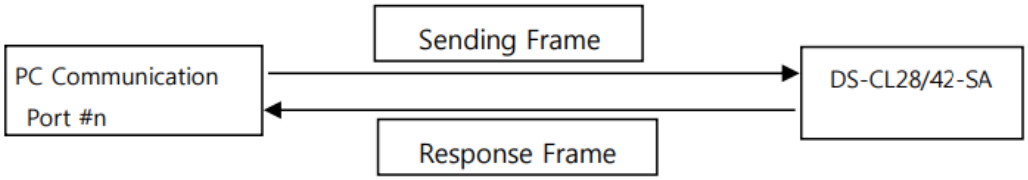
There are 2 kinds of program version for DS-CL28/42-SA. This manual support for Version 6 level. (DS-CL28/42-SA support version 6 only.)

After running the user program(GUI), You can check the version with 'About Ezi-MOTION Plus-R GUI' menu in 'Help' menu.

※ Version 6.45.5.41 and later versions of "Ezi-MOTION Plus-R GUI" supports DS-CL28/42-SA.



(1) Overview of Communication Frame



(2) Format of Frame

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

① 0xAA : Delimited byte

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

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

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

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

Detailed Frame Data is configured as follows:

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

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

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

③ Data : Data format and length is determined by frame type. For more information, refer to「Frame Type and Data Configuration」section.

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

Notes **Byte stuffing : add a byte' to frame data section to distinguish 'Header' and 'Tail' frame data**

1.1.3 CRC Calculation Example

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

(1) '0xA001'of CRC16

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

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

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

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

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

        /* Initialize CRC */
        wCrc = 0xffff;

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

           /* Loop through all 8 data bits */

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

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

1.1.4 Response Frame Format and Communication Error(ver 6)

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

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

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

② Frame type : Same to sending Frame.
(When this is not same to sending data, it should be recognized as the error status.)

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

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

Hexa Code	Decimal Code	Description
0x00	0	Communication is normal.
0x80	128	Frame Type Error : Responded Frame type cannot be recognized.
0x81	129	Data error, ROM data read/write error : Data value responded is without the given range.
0x82	130	Received Frame Error : Frame data received is out of this specification.
0x85	133	Running Command Failure : The user has tried to execute new running commands in wrong condition as follows. 1) currently motor is running 2) currently motor is stopping 3) currently Servo is OFF status 4) try to Z-pulse Origin without encoder 5) other wrong motion command
0x86	134	RESET Failure : The user has tried to execute new running commands in wrong condition as follows. 1) While the servo is ON 2) Already RESET in ON by external input signal
0x87	135	Servo ON Failure ① : While an alarm occurs, the user has tried to execute Servo ON command.

0x88	136	Servo ON Failure ② : While Emergency Stop occurs, the user has tried to execute Servo ON command.
0x89	137	Servo ON Failure ③ : 'Servo ON' signal is assigned to input pin already. Servo ON/OFF Can execute by external input signal only.
0x8A	138	Servo OFF Failure : While the Servo ON process is working, the user has tried to execute Servo OFF command.
0xAA	170	CRC Error : Frame data received is out of CRC format. In this case, DLL Library of sending side automatically try to send one more time.

Warning

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

1.2 Definition of Frame Type (Ver 6)

1.2.1 Frame Type of DS-CL28/42-SA

Frame type	Library name	Support	Contents
0x01	FAS_GetSlaveInfo	O	Requests the slave type and program version info.
0x05	FAS_GetMotorInfo	O	Requests the motor info.
0x10	FAS_SaveAllParameters	O	Saves all parameters to ROM area.
0x11	FAS_GetRomParameter	O	Reads the specified parameter in ROM area.
0x12	FAS_SetParameter	O	Writes the specified parameter in RAM area.
0x13	FAS_GetParameter	O	Read the specified parameter in RAM area.
0x20	FAS_SetIOOutput	X	Sets output port level.
0x21	FAS_SetIOInput	O	Sets input port level.
0x22	FAS_GetIOInput	O	Reads input port level.
0x23	FAS_GetIOOutput	X	Reads output port level.
0x24	FAS_SetIOAssignMap	O	Assigns the specified function to IO port
0x25	FAS_GetIOAssignMap	O	Reads the IO port assignment.
0x26	FAS_IOAssignMapReadROM	O	Loads the IO port assignments from ROM to RAM area.
0x27	FAS_TriggerOutput_RunA	X	Sets the condition of “Compare Out” function.
0x28	FAS_TriggerOutput_Status	X	Inquires the status of compare out.
0x2A	FAS_ServoEnable	O	Sets servo-on/off
0x2B	FAS_ServoAlarmReset	O	Resets alarm state.
0x2E	FAS_GetAlarmType	O	Reads alarm info.
0x31	FAS_MoveStop	O	Stops motor.
0x32	FAS_EmergencyStop	O	Stops motor immediately.
0x33	FAS_MoveOriginSingleAxis	O	Starts to search the origin.
0x34	FAS_MoveSingleAxisAbsPos	O	Moves to the specified absolute position.
0x35	FAS_MoveSingleAxisIncPos	O	Moves to the specified relative position.
0x36	FAS_MoveToLimit	O	Moves continuously until the limit sensor input.
0x37	FAS_MoveVelocity	O	Start Jog movement.
0x38	FAS_PositionAbsOverride	X	Modifies the destination with the absolute position during the motioning.
0x39	FAS_PositionIncOverride	X	Modifies the destination with the relative position during the motioning.
0x3A	FAS_VelocityOverride	X	Modifies the velocity during the motioning.
0x3B	FAS_AllMoveStop	O	Stops all drives connected.
0x3C	FAS_AllEmergencyStop	O	Stops all drives connected immediately.
0x3D	FAS_AllMoveOriginSingleAxis	O	Starts all drives connected to search the origin.
0x3E	FAS_AllSingleAxisAbsPos	O	Starts all drives connected to move to the specified absolute position.
0x3F	FAS_AllSingleAxisIncPos	O	Starts all drives connected to move to the specified relative position.

0x80	FAS_MoveSingleAxisAbsPosEx	O	Starts to move to the specified absolute position with the custom Accel/Decel times.
0x81	FAS_MoveSingleAxisIncPosEx	O	Starts to move to the specified relative position with custom Accel/Decel times.
0x82	FAS_MoveVelocityEx	O	Start to jog movement with custom Accel/Decel times.
	FAS_MoveLinearAbsPos	X	Starts linear Interpolation movement to the absolute position.
	FAS_MoveLinearIncPos	X	Starts linear Interpolation movement to the relative position.
0x40	FAS_GetAxisStatus	O	Reads the running status flag of drive.
0x41	FAS_GetIOAxisStatus	O	Reads the I/O status and the status flag of drive.
0x42	FAS_GetMotionStatus	O	Reads the current motion status.
0x43	FAS_GetAllStatus	O	Reads all status including the current I/O status
0x50	FAS_SetCommandPos	O	Writes the command position.
0x51	FAS_GetCommandPos	O	Reads the command position.
0x52	FAS_SetActualPos	O	Write the actual position.
0x53	FAS_GetActualPos	O	Reads the actual position.
0x54	FAS_GetPosError	O	Reads the positon error.
0x55	FAS_GetActualVel	O	Reads the actual velocity.
0x56	FAS_ClearPosition	O	Clears command & actual position.
0x58	FAS_MovePause	X	Pauses for motioning or Runs again
0x60	FAS_PosTableReadItem	O1)	Reads PT item in RAM area.
0x61	FAS_PosTableWriteItem	O1)	Writes PT item in RAM area.
0x62	FAS_PosTableReadROM	O1)	Loads PT item from ROM to RAM area.
0x63	FAS_PosTableWriteROM	O1)	Saves PT item to ROM area.
0x64	FAS_PosTableRunItem	O1)	Starts to run PT table from the specified PT number.
0x6A	FAS_PosTableReadOneItem	O1)	Reads the sub item of the specified PT in RAM area.
0x6B	FAS_PosTableWriteOneItem	O1)	Writes the sub item of the specified PT in RAM area.
0x78	FAS_MovePush	X	Start the push motion with the specified condition.
0x79	FAS_GetPushStatus	X	Reads the push motion status.

*1) V06.01.30.22 and later versions(DS-CL28/42-SA) of firmware supports this functions.

1.2.2 Frame Type and Data Contiguration

(1) The follow table displays the content and configuration of data by frame type

<table>
<tr><th>Frame Type</th><th>Function Name</th><th>Description</th></tr>
<tr><td>0x01 (1)</td><td>FAS_
GetSlaveInfo</td><td>Requests the connected slave type and program version information.
Sending : 0 byte
Response : 1~248 bytes
<table>
<tr><th>1 byte</th><th>1 byte</th><th>0~246 bytes</th></tr>
<tr><td>Communication status</td><td>Slave type</td><td>ACII string
with NULL byte
(strlen() + 1 byte)</td></tr>
</table>
◆ Slave type : 108 : DS-CL28-SA
◆ Slave type : 109 : DS-CL42-SA</td></tr>
<tr><td>0x05 (5)</td><td>FAS_
GetMotorInfo</td><td>Requests connected motor type and motor manufacturer information.
Sending : 0 byte
Response : 1~248 bytes
<table>
<tr><th>1 byte</th><th>1 byte</th><th>0~246 bytes</th></tr>
<tr><td>Communication status</td><td>Motor type
(1~255)</td><td>ACII string
with NULL byte
(strlen() + 1 byte)</td></tr>
</table>
◆ Motor type : refer to 「1-2-5.Information of Motors」</td></tr>
<tr><td>0x10 (16)</td><td>FAS_
SaveAllParameters</td><td>Saves current parameters set & assignments of IO port are saved in the ROM of the drive. After executing this command, the parameters are retained regardless of power status.
Values set by 'FAS_SetParameter' &'FAS_SetIOAssignMap' are saved together.
Sending : 0 byte
Response : 1 byte
<table>
<tr><th>1 byte</th></tr>
<tr><td>Communication status</td></tr>
</table></td></tr>
<tr><td>0x11 (17)</td><td>FAS_
GetRomParameter</td><td>Reads the specified parameter in the ROM.
Sending : 1 byte
<table>
<tr><th>1 byte</th></tr>
<tr><td>Parameter number (0~37)</td></tr>
</table>
Response : 5 bytes
<table>
<tr><th>1 byte</th><th>4 bytes</th></tr>
<tr><td>Communication status</td><td>Parameter value</td></tr>
</table>
Refer to 「1-2-2.Parameter List」</td></tr>
</table>

<table>
<tr><td>0x12 (18)</td><td>FAS_
SetParameter</td><td>Writes the specified parameter to the RAM in the drive.

Sending : 5 bytes
<table><tr><th>1 byte</th><th>4 bytes</th></tr><tr><td>Parameter number (0~37)</td><td>Parameter value</td></tr></table>
Response : 1 byte
<table><tr><th>1 byte</th></tr><tr><td>Communication status</td></tr></table>Refer to「1-2-2.Parameter List」</td></tr>
<tr><td>0x13 (19)</td><td>FAS_
GetParameter</td><td>Reads the specific parameter in the RAM.

Sending : 1 byte
<table><tr><th>1 byte</th></tr><tr><td>Parameter number (0~37)</td></tr></table>
Response : 5 bytes
<table><tr><th>1 byte</th><th>4 bytes</th></tr><tr><td>Communication status</td><td>Parameter value</td></tr></table>Refer to「1-2-2.Parameter List」</td></tr>
<tr><td>0x21 (33)</td><td>FAS_
SetIOInput</td><td>Forces to set or clear the specified bit to input port.

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

Response : 1 byte
<table><tr><th>1 byte</th></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x22 (34)</td><td>FAS_
GetIOInput</td><td>Reads the current status of the control input port.

Sending : 0 byte
Response : 5 bytes
<table><tr><th>1 byte</th><th>4 bytes</th></tr><tr><td>Communication status</td><td>Input status value</td></tr></table>Relevant bit by each input signal, refer to「1-2-3. Bit setup of Input Pin」.</td></tr>
</table>

<table>
<tr>
<td>0x24 (36)</td>
<td>FAS_
SetIOAssignMap</td>
<td>
Assigns control or IO function to the port and set the signal level.
You can save the settings to the ROM with 'FAS_SaveAllParameters'

Sending : 6 bytes
<table>
<tr><th>1 byte</th><th>4 bytes</th><th>1 byte</th></tr>
<tr><td>I/O number</td><td>I/O pin masking data</td><td>Setting level</td></tr>
</table>

◆I/O number: '0~3' corresponds to 'Limit+, Limit-, Org, IN1' respectively (DS-CL28-SA)
'0~6' corresponds to 'IN1, IN2, IN3, IN4, IN5, IN6, IN7' respectively (DS-CL42-SA)
◆I/O pin masking data: Refer to 「1-2-3. Bit setup of Input Pin」.
◆Setting level: 0:Active Low, 1:Active High
◆Before changing the IO assignment, It should be to set the IO masking data to '0'.
If you want to change the IO bit setting while the bit is used, Pin masking data should be reset to '0' before changing.

Response : 1 byte
<table>
<tr><th>1 byte</th></tr>
<tr><td>Communication status</td></tr>
</table>
</td>
</tr>
<tr>
<td>0x25 (37)</td>
<td>FAS_
GetIOAssignMap</td>
<td>
Reads the setting of IO port from RAM area.

Sending : 1 byte
<table>
<tr><th>1 byte</th></tr>
<tr><td>I/O number</td></tr>
</table>

◆I/O number: '0~3' corresponds to 'Limit+, Limit-, Org, IN1' respectively (DS-CL28-SA)
'0~6' corresponds to 'IN1, IN2, IN3, IN4, IN5, IN6, IN7' respectively (DS-CL42-SA)

Response : 6 bytes
<table>
<tr><th>1 byte</th><th>4 bytes</th><th>1 byte</th></tr>
<tr><td>Communication status</td><td>IO pin masking status</td><td>Level status</td></tr>
</table>
For more information, refer to '0x24'Frame type.
</td>
</tr>
<tr>
<td>0x26 (38)</td>
<td>FAS_
IOAssignMap
ReadROM</td>
<td>
Loads the setting of IO port from ROM area to RAM.

Sending : 0 byte
Response : 2 bytes
<table>
<tr><th>1 byte</th><th>1 byte</th></tr>
<tr><td>Communication status</td><td>Execution status
(0 : complete, values except 0: error)</td></tr>
</table>
</td>
</tr>
</table>

0x2A (42)	FAS_ ServoEnable	Enables/Disables the servo. Sending : 1 byte **1 byte**: 0:OFF, 1:ON Response : 1 byte **1 byte**: Communication status
0x2B (43)	FAS_ ServoAlarmReset	Forces to reset alarm status. Sending : 0 byte Response : 1 byte **1 byte**: Communication status
0x2E (46)	FAS_ ServoAlarmtype	Reads the current alarm type Sending: 0 byte Response: 2 bytes **1 byte**: Communication status; **1 byte**: Alarm type (0~15) ◆ Alarm type: No alarm (0) OverCurrent(1) OverSpeed(2) Position Tracking(3) OverLoad(4) OverTemperature(5) BackEMF(6) MotorConnect(7) EncoderConnect(8) Inposition(10) ROMdevice(12) Position Overflow(15)
0x31 (49)	FAS_ MoveStop	Requests the drive to stop the running motor Sending : 0 byte Response : 1 byte **1 byte**: Communication status
0x32 (50)	FAS_ EmergencyStop	Requests the drive to stop the running motor emergently Sending : 0 byte Response : 1 byte **1 byte**: Communication status
0x33 (51)	FAS_ MoveOrigin SingleAxis	Requests the drive to return to the origin at the current setting parameter condition Sending : 0 byte Response : 1 byte **1 byte**: Communication status

0x34 (52)	FAS_ MoveSingleAxis AbsPos	Requests the drive to move to the specified position.(Absolute position) Sending : 8 bytes **4 bytes** | **4 bytes** Absolute position value | Running speed [pps] Response : 1 byte **1 byte** Communication status
0x35 (53)	FAS_ MoveSingleAxis IncPos	Requests the drive to move the specified amount[pulse]. (Incremental Position) Sending : 8 bytes **4 bytes** | **4 bytes** Incremental position value | Running speed [pps] Response : 1 byte **1 byte** Communication status
0x36 (54)	FAS_ MoveToLimit	Requests the drive to start limit motion at the current setting parameter condition Sending : 5 bytes **4 bytes** | **1 byte** Running speed [pps] | Running direction (0: -Limit 1: +Limit) Response : 1 byte **1 byte** Communication status
0x37 (55)	FAS_ MoveVelocity	Requests the drive to start jog motion at the current setting parameter condition Sending : 5 bytes **4 bytes** | **1 byte** Running speed [pps] | Running direction (0: -Jog 1: +Jog) Response : 1 byte **1 byte** Communication status
0x3B (59)	FAS_ AllMoveStop	Requests all the drives connected to stop. Sending : 0 byte (Slave number must be '99') Response : no response

<table>
<tr><td>0x3C
(60)</td><td>FAS_
AllEmergencyStop</td><td>Requests all the drive connected to do emergency stop

Sending : 0 byte
(Slave number must be ‘99’)
Response : no response</td></tr>
<tr><td>0x3D
(61)</td><td>FAS_All
MoveOriginSingl
eAxis</td><td>Requests all the drive connected to return to the origin at the current setting parameter condition.

Sending : 0 byte
(Slave number must be ‘99’)
Response : no response</td></tr>
<tr><td>0x3E
(62)</td><td>FAS_All
SingleAxisAbsPos</td><td>Requests all the drive connected to move to specified position.

Sending : 8 bytes (Slave number must be ‘99’)
<table><tr><th>4 bytes</th><th>4 bytes</th></tr><tr><td>Absolute position value</td><td>Running speed [pps]</td></tr></table>
Response : no response</td></tr>
<tr><td>0x3F
(63)</td><td>FAS_All
SingleAxisIncPos</td><td>Requests all the drive connected to move the specified amount[pulse] relatively.

Sending : 8 bytes (Slave number must be ‘99’)
<table><tr><th>4 bytes</th><th>4 bytes</th></tr><tr><td>incremental position value</td><td>Running speed [pps]</td></tr></table>
Response : no response</td></tr>
<tr><td>0x80
(128)</td><td>FAS_
MoveSingleAxis
AbsPosEx</td><td>Requests the drive to move the specified position as much as the absolute value [pulse] with Custom Accel. / Decel. Time [msec]

Sending: 40 bytes
<table><tr><th>4 bytes</th><th>4 bytes</th><th>4 bytes</th><th>2 bytes</th></tr><tr><td>Absolute position value</td><td>Running speed [pps]</td><td>Flag option</td><td>Custom Accel. Time (1~9999)</td></tr></table>
<table><tr><th>2 bytes</th><th>24 bytes</th></tr><tr><td>Custom Decel. Time (1~9999)</td><td>Reserved</td></tr></table>
Flag option : 0x0001 : reserved
0x0002 : Custom Accel. Time is used.
0x0004 : Custom Decel. Time is used.
If the Flag bit is OFF(0), Accel./Decel. Time value is used that saved in controller.

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

<table>
<tr>
<td>0x81
(129)</td>
<td>FAS_
MoveSingleAxis
IncPosEx</td>
<td>Requests the drive to move the specified amount [pulse] with the Custom Accel. / Decel. Time [msec]

Sending: 40 bytes
<table>
<tr><th>4 bytes</th><th>4 bytes</th><th>4 bytes</th><th>2 bytes</th></tr>
<tr><td>incremental position value</td><td>Running speed [pps]</td><td>Flag option</td><td>Custom Accel. Time (1~9999)</td></tr>
</table>
<table>
<tr><th>2 bytes</th><th>24 bytes</th></tr>
<tr><td>Custom Decel. Time (1~9999)</td><td>Reserved</td></tr>
</table>
Flag option : 0x0001 : reserved
0x0002 : Custom Accel. Time is used.
0x0004 : Custom Decel. Time is used.
If the Flag bit is OFF(0), Accel./Decel. Time value is used that saved in controller.

Response: 1 byte</td>
</tr>
<tr>
<td>0x82
(130)</td>
<td>FAS_
MoveVelocityEx</td>
<td>Requests the drive to start jog motion at the current setting parameter condition with custom Accel/Decel time value [msec].

Sending: 37 bytes
<table>
<tr><th>4 bytes</th><th>1 byte</th><th>4 bytes</th></tr>
<tr><td>Running speed [pps]</td><td>Running direction (0: -Jog 1: +Jog)</td><td>Flag option</td></tr>
</table>
<table>
<tr><th>2 bytes</th><th>26 bytes</th></tr>
<tr><td>Custom Accel./Decel. Time (1~9999)</td><td>Reserved</td></tr>
</table>
Flag option : 0x0001 : reserved
0x0002 : Custom Accel./Decel. Time is used.
If the Flag bit is OFF status(0), Accel./Decel. Time value is used that saved in controller.

Response : 1 byte</td>
</tr>
<tr>
<td>0x40
(64)</td>
<td>FAS_
GetAxisStatus</td>
<td>Reads the status flags of drive.

Sending : 0 byte
Response : 5 bytes
<table>
<tr><th>1 byte</th><th>4 bytes</th></tr>
<tr><td>Communication status</td><td>Status flag value</td></tr>
</table>
For bit related to each Flag, refer to 「1-2-5. Bit setup of Status Flag」.</td>
</tr>
</table>

0x41 (65) — FAS_GetIOAxisStatus

Reads the I/O status and the status flag of drive.
(Frame type 0x22 + 0x23)

Sending : 0 byte
Response : 13 bytes

1 byte	4 bytes	4 bytes	4 bytes
Communication status	Input status value	Reserved	Status flag value

0x42 (66) — FAS_GetMotionStatus

Reads the current motion status.
(Frame type 0x51, 0x53, 0x54, and 0x55 are packed.)

Sending : 0 byte
Response : 21 bytes

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)]

4 bytes	1 byte
Running speed [pps]	Running direction (0: -Limit 1: +Limit)

1) V06.01.30.22 and later F/W version supports this field. Previous version is filled with 0.

0x43 (67) — FAS_GetAllStatus

Reads all status including the current motion status
(Frame type 0x41 + 0x42)

Sending : 0 byte
Response : 33 bytes

1 byte	4 bytes	4 bytes	4 bytes
Communication status	Input status value	Reserved	Status flag value

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)]

1) V06.01.30.22 and later F/W version supports this field. Previous version is filled with 0.

<table>
<tr>
<td>0x50
(80)</td>
<td>FAS_
SetCommand
Pos</td>
<td>DS-CL28/42-SA 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.

Sending : 4 bytes
<table><tr><th>4 bytes</th></tr><tr><td>Command position setting count value</td></tr></table>
Response : 1 byte
<table><tr><th>1 byte</th></tr><tr><td>Communication status</td></tr></table></td>
</tr>
<tr>
<td>0x51
(81)</td>
<td>FAS_
GetCommand
Pos</td>
<td>Reads the command position value[pulse] being tracked.

Sending : 0 byte
Response : 5 bytes
<table><tr><th>1 byte</th><th>4 bytes</th></tr><tr><td>Communication status</td><td>Command position value</td></tr></table></td>
</tr>
<tr>
<td>0x52
(82)</td>
<td>FAS_
SetActualPos</td>
<td>DS-CL28/42-SA 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.

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

Sending : 0 byte
Response : 5 bytes
<table><tr><th>1 byte</th><th>4 bytes</th></tr><tr><td>Communication status</td><td>Actual position value</td></tr></table></td>
</tr>
</table>

<table>
<tr>
<td>0x54
(84)</td>
<td>FAS_
GetPosError</td>
<td>Reads the difference[pulse] between the command position value and the actual position value.

Sending : 0 byte
Response : 5 bytes
<table>
<tr><th>1 byte</th><th>4 bytes</th></tr>
<tr><td>Communication status</td><td>Position difference value</td></tr>
</table>
By this value, the user can check the current running status (how much inposition is tracked).</td>
</tr>
<tr>
<td>0x55
(85)</td>
<td>FAS_
GetActualVel</td>
<td>Reads the current running speed value [pps]

Sending : 0 byte
Response : 5 bytes
<table>
<tr><th>1 byte</th><th>4 bytes</th></tr>
<tr><td>Communication status</td><td>Speed value</td></tr>
</table>
</td>
</tr>
<tr>
<td>0x56
(86)</td>
<td>FAS_
ClearPosition</td>
<td>DS-CL28/42-SA 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.

Sending : 0 byte
Response : 1 byte
<table>
<tr><th>1 byte</th></tr>
<tr><td>Communication status</td></tr>
</table>
</td>
</tr>
<tr>
<td>0x60
(96)</td>
<td>FAS_
PosTableRead
Item</td>
<td>To read PT values in the RAM of the drive.

Sending : 2 bytes
<table>
<tr><th>2 bytes</th></tr>
<tr><td>Readable PT number (0~255)</td></tr>
</table>
Response : 65 bytes
<table>
<tr><th>1 byte</th><th>64 bytes</th></tr>
<tr><td>Communication status</td><td>Relevant PT values</td></tr>
</table>
For items by each PT, refer to 「1-2-6. Position Table Item」.
* Only V06.01.30.22 and later F/W version supports this command.</td>
</tr>
</table>

<table>
<tr>
<td>0x61
(97)</td>
<td>FAS_
PosTableWrite
Item</td>
<td>To save PT values to the RAM of the drive.

Sending : 66 bytes
<table>
<tr><th>2 bytes</th><th>64 bytes</th></tr>
<tr><td>PT number (0~255)</td><td>Relevant PT value</td></tr>
</table>
For items by each PT, refer to 「1-2-6. Position Table Item」.

Response : 2 bytes
<table>
<tr><th>1 byte</th><th>1 byte</th></tr>
<tr><td>Communication status</td><td>Command performing status
(values except 0 : complete, 0: error)</td></tr>
</table>
* Only V06.01.30.22 and later F/W version supports this command.</td>
</tr>
<tr>
<td>0x62
(98)</td>
<td>FAS_
PosTableRead
ROM</td>
<td>To read all PT values (256 ea) in the ROM of the drive

Sending : 0 byte
Response : 2 bytes
<table>
<tr><th>1 byte</th><th>1 byte</th></tr>
<tr><td>Communication status</td><td>Command performing status
(0 : complete, values except 0: error)</td></tr>
</table>
* Only V06.01.30.22 and later F/W version supports this command.</td>
</tr>
<tr>
<td>0x63
(99)</td>
<td>FAS_
PosTableWrite
ROM</td>
<td>To save all PT value(256 ea) to the ROM of the drive.

Sending : 0 byte
Response : 2 bytes
<table>
<tr><th>1 byte</th><th>1 byte</th></tr>
<tr><td>Communication status</td><td>Command performing status
(0 : complete, values except 0: error)</td></tr>
</table>
* Only V06.01.30.22 and later F/W version supports this command.</td>
</tr>
<tr>
<td>0x64
(100)</td>
<td>FAS_
PosTableRunItem</td>
<td>To start the position table operation from the designated PT number

Sending : 2 bytes
<table>
<tr><th>2 bytes</th></tr>
<tr><td>Readable PT number (0~255)</td></tr>
</table>
Response : 1 byte
<table>
<tr><th>1 byte</th></tr>
<tr><td>Communication status</td></tr>
</table>
* Only V06.01.30.22 and later F/W version supports this command.</td>
</tr>
</table>

<table>
<tr>
<td>0x6A
(106)</td>
<td>FAS_
PosTableReadOneItem</td>
<td>To read one of PT values in the RAM of the drive.

Sending: 4 bytes
<table><tr><th>2 bytes</th><th>2 bytes</th></tr><tr><td>PT number (0~255)</td><td>Offset value(0~56)</td></tr></table>Refer to 「1-2-6. Position Table Item」for Offset value

Response : 5 bytes
<table><tr><th>1 byte</th><th>4 bytes</th></tr><tr><td>Communication status</td><td>Relevant one of PT value</td></tr></table>* Only V06.01.30.22 and later F/W version supports this command.</td>
</tr>
<tr>
<td>0x6B
(107)</td>
<td>FAS_
PosTableWriteOneItem</td>
<td>To save one of PT values to the RAM of the drive.

Sending: 8 bytes
<table><tr><th>2 bytes</th><th>2 bytes</th><th>4 bytes</th></tr><tr><td>PT Number (0~255)</td><td>Offset value (0~56)</td><td>Relevant one of PT value</td></tr></table>Refer to 「1-2-6. Position Table Item」for Offset value

Response: 2 bytes
<table><tr><th>1 byte</th><th>1 byte</th></tr><tr><td>Communication status</td><td>Command performing status
(values except 0 : complete, 0: error)</td></tr></table>* Only V06.01.30.22 and later F/W version supports this command.</td>
</tr>
</table>

1.2.3 Parameter Lists

No	Item	Unit	Lower Limit	Higher Limit	Default
0	Pulse per Revolution		0	9*1 8*2	9*1 8*2
1	Axis Max Speed	[pps]	1	800,000*1 500,000*2	800,000*1 500,000*2
2	Axis Start Speed	[pps]	1	35,000	1
3	Axis Acc Time	[msec]	1	9,999	100
4	Axis Dec Time	[msec]	1	9,999	100
5	Reserved	[%]	100	100	100
6	Jog Speed	[pps]	1	800,000*1 500,000*2	5,000
7	Jog Start Speed	[pps]	1	35,000	1
8	Jog Acc Dec Time	[msec]	1	9,999	100
9	S/W Limit Plus Value	[pulse]	-2,147,483,648	2,147,483,647	2,147,483,647
10	S/W Limit Minus Value	[pulse]	-2,147,483,648	2,147,483,647	-2,147,483,648
11	S/W Limit Stop Method		0	2	2
12	H/W Limit Stop Method		0	1	0
13	Limit Sensor Logic[1] / Reserved		0	1	0
14	Org Speed	[pps]	1	500,000	5,000
15	Org Search Speed	[pps]	1	50,000	1,000
16	Org Acc Dec Time	[msec]	1	9,999	50
17	Org Method		0	7	0
18	Org Dir		0	1	0
19	Org Offset	[pulse]	-2,147,483,648	2,147,483,647	0
20	Org Position Set	[pulse]	-2,147,483,648	2,147,483,647	0
21	Org Sensor Logic[1] / Reserved		0	1	0
22	Position Loop Gain		0	63	4
23	Inpos Value		0	63	0
24	Pos Tracking Limit	[pulse]	1	2,147,483,647	4,000*1 2,500*2
25	Motion Dir		0	1	0
26	Limit Sensor Dir[1] / Reserved		0	1	0
27	Org Torque Ratio	[%]	60*1 20*2	100*1 90*2	60*1 20*2
28	Pos. Error Overflow Limit	[pulse]	1	2,147,483,647	4,000*1 2,500*2
29	Reserved	[msec]	0	0	0
30	Run Current	*10[%]	5	15	10
31	Boost Current	*50[%]	0	7	0

Notes

1) **This parameter is not used in V06.01.30.22 and later version (DS-CL28/42-SA). It can be set by using IO Settings.**

*1 : DS-CL28-SA

*2 : DS-CL42-SA

1.2.4 Definition 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	Relavant Bit Position	Signal Name	Relavant Bit Position	Signal Name	Relavant Bit Position	Signal Name	Relavant Bit Position
Limit+	0x00000001	PT A4[1)]	0x00000100	AlarmReset	0x00010000	JPT input 2[1)]	0x01000000
Limit-	0x00000002	PT A5[1)]	0x00000200	ServoON	0x00020000	JPT Start[1)]	0x02000000
Origin	0x00000004	PT A6[1)]	0x00000400	Reserved	0x00040000	User IN 0	0x04000000
Clear Position	0x00000008	PT A7[1)]	0x00000800	Org Search	0x00080000	User IN 1	0x08000000
PT A0[1)]	0x00000010	PT Start[1)]	0x00001000	Reserved	0x00100000	User IN 2	0x10000000
PT A1[1)]	0x00000020	Stop	0x00002000	E-stop	0x00200000	User IN 3	0x20000000
PT A2[1)]	0x00000040	Jog+	0x00004000	JPT input 0[1)]	0x00400000	User IN 4	0x40000000
PT A3[1)]	0x00000080	Jog-	0x00008000	JPT input 1[1)]	0x00800000	User IN 5	0x80000000

Notes

1) V06.01.30.22 and later F/W version supports this signal.

Example

1. Sending data to turn ON the Jog+ port

4 bytes (I/O set mask value)	4 bytes (I/O clear mask value)
0x00004000	0x00000000

Example

2. Sending data to turn ON the Jog+ port

4 bytes (I/O set mask value)	4 bytes (I/O clear mask value)
0x00000000	0x00004000

1.2.5 Definition of Status Flag

Refer to 'motion_define.h' of include files

Name of Flag Define	Contents	Relevant Bit Position
FFLAG_ERRORALL	One or more error occurs.	0X00000001
FFLAG_HWPOSILMT	'+' direction limit sensor turns ON.	0X00000002
FFLAG_HWNEGALMT	'-' direction limit sensor turns ON.	0X00000004
FFLAG_SWPOGILMT	'+' direction program limit is exceeded.	0X00000008
FFLAG_SWNEGALMT	'-' direction program limit is exceeded.	0X00000010
Reserved1		0X00000020
Reserved2		0X00000040
FFLAG_ERRPOSOVERFLOW	Position error is higher than 'Pos Error Overflow Limit' parameter after position command	0X00000080
FFLAG_ERROVERCURRENT	The motor driving device is under over-current.	0X00000100
FFLAG_ERROVERSPEED	The motor speed exceeded 3300[rpm].	0X00000200
FFLAG_ERRPOSTRACKING	Position error is higher than 'Pos Tracking Limit' parameter during position command run.	0X00000400
FFLAG_ERROVERLOAD	Load exceeding the max torque of the motor is loaded more than 5 seconds.	0X00000800
FFLAG_ERROVERHEAT	The internal temperature of the drive exceeds 85°C.	0X00001000
FFLAG_ERRBACKEMF	A counter electromotive force of the motor exceeds 48V.	0X00002000
Reserved	Always '0'	0X00004000
FFLAG_ERRINPOSITION	After operation is finished, a position error occurs for more than 3 seconds.	0X00008000
FFLAG_EMGSTOP	The motor is under emergency stop.	0X00010000
FFLAG_SLOWSTOP	The motor is under general stop.	0X00020000
FFLAG_ORIGINRETURNING	The motor is returning to the origin.	0X00040000
FFLAG_INPOSITION	Inposition has been finished.	0X00080000
FFLAG_SERVOON	The motor is under Servo ON.	0X00100000
FFLAG_ALARMRESET	AlarmReset has run.	0X00200000
FFLAG_PTSTOPED [1)]	Position Table operation has been finished.	0X00400000
FFLAG_ORIGINSENSOR	The origin sensor is ON.	0X00800000
FFLAG_ZPULSE	The motor is in the z-pulse position of encoder.	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
Reserved	Always '0'	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

Notes **1) Only V06.01.30.22 and later F/W version supports this flag.**

1.2.6 Position Tanle Item

(V06.01.30.22 and later F/W version supports PT function)

Refer to 'motion_define.h' of include files

Name	Name of Structure Parameter	Number of Bytes	Offset value	Unit	Low Limit *1	Upper Limit *1
Position	lPosition	4 (signed)	0	[pulse]	-2,147,483,648	2,147,483,647
Low Speed	dwStartSpd	4 (unsigned)	4	[pps]	0	35,000
HighSpeed	dwMoveSpd	4 (unsigned)	8	[pps]	0	500,000
Accel. Time	wAccelRate	2 (unsigned)	12	[msec]	1	9,999
Decel. Time	wDecelRate	2 (unsigned)	14	[msec]	1	9,999
Command	wCommand	2 (unsigned)	16		0	9
Wait time	wWaitTime	2 (unsigned)	18	[msec]	0	60,000
Blank		2 (unsigned)				
Jump Table No.	wBranch	2 (unsigned)	22		0 10000	255 10255
Jump PT 0	wCond_branch0	2 (unsigned)	24		0 10000	255 10255
Jump PT 1	wCond_branch1	2 (unsigned)	26		0 10000	255 10255
Jump PT 2	wCond_branch2	2 (unsigned)	28		0 10000	255 10255
Loop Count	wLoopCount	2 (unsigned)	30		0	100
Loop Jump Table No.	wBranchAfterLoop	2 (unsigned)	32		0 10000	255 10255
PT set	wPTSet	2 (unsigned)	34		0	15
Loop Counter Clear	wLoopCountCLR	2 (unsigned)	36		0	255
Check Inposition	bCheckInpos	2 (unsigned)	38		0	1
Compare Position	lTriggerPos	4 (signed)	40	[pulse]	-2,147,483,648	2,147,483,647
Compare Width	wTriggerOnTime	2 (unsigned)	44	[msec]	1	9,999
Push Ratio	wPushRatio	2 (unsigned)	46	[%]	20	90
Push Speed	dwPushSpeed	4 (unsigned)	48	[pps]	0	100,000
Push Position	lPushPosition	4 (signed)	52	[pulse]	-2,147,483,648	2,147,483,647
Push Mode	wPushMode	2 (unsigned)	56		0	1
Blank		6 (unsigned)	58		0x00	

1.2.7 Information of Motors

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

Ex **Example**

28L : Motor Flange size is 28mm and Long size
Second field is the motor maker information.

Display	Maker
DIN	Dings

1.3 Programming

There are 2 method of programming for driving DS-CL28/42-SA The first is normally used method that using Visual C++ language under window system of PC. Library that serviced together with DS-CL28/42-SA have to be used. Refer to **「2. Library for PC Program」**

The second method can be accomplished by sending command characters directly to DS-CL28/42-SA. The user have to prepare low level protocol programming like 'Protocol Test' program. This method is normally used for PLC system. For excise the protocol programming,'ProtocolTest_PlusR.exe' GUI program is serviced together. Refer to **「3. Protocol for PLC Program」.**

2. Library for PC Program (Ver 6)

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 “₩₩DINGS₩₩Ezi-MOTION Plus-RnE₩₩include₩₩” The following contents should be included in a source file for development.

#include “₩₩DINGS₩₩Ezi-MOTION Plus-RnE₩₩include₩₩FAS_EziMotionPlusR.h”

#include “₩₩DINGS₩₩Ezi-MOTION Plus-RnE₩₩include₩₩COMM_Define.h”

#include “₩₩DINGS₩₩Ezi-MOTION Plus-RnE₩₩include₩₩MOTION_DEFINE.h”

#include “₩₩DINGS₩₩Ezi-MOTION Plus-RnE₩₩include₩₩ReturnCodes_Define.h”

Also, library files are as follows:

- For 32bit Windows

“₩₩DINGS₩₩Ezi-MOTION Plus-RnE₩₩include₩₩EziMotionPlusR.lib”

“₩₩DINGS₩₩Ezi-MOTION Plus-RnE₩₩include₩₩EziMotionPlusR.dll”

- For 64bit Windows

“₩₩DINGS₩₩Ezi-MOTION Plus-RnE₩₩include₩₩x64₩₩EziMotionPlusRx64.lib”

“₩₩DINGS₩₩Ezi-MOTION Plus-RnE₩₩include₩₩x64₩₩EziMotionPlusRx64.dll”

A sample program source of using library is included in a “₩₩DINGS₩₩Ezi-MOTION Plus-RnE ₩₩Examples₩₩” folder.

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

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

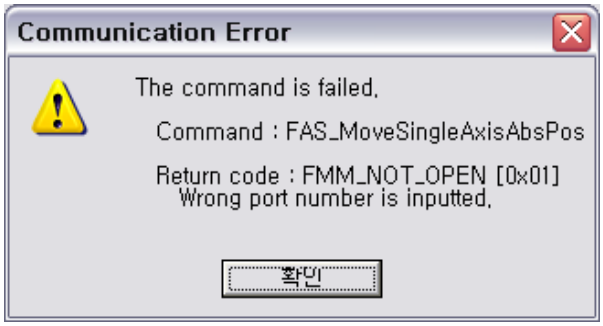
(2) The following table shows return values included commonly in all libraries. The user can check - 31 - Library for PC Program (Ver6) the result (communication status, running status) judged by the drive. When the user develops programs by using protocols without libraries (DLL), they are available as well.

Item	Definition	Returned Value	Description
Normal	FMP_OK	0	Communication has been normally performed.
Input Error	FMP_FRAMETYPEERROR	128	The drive cannot recognize the command.
	FMP_DATAERROR	129	Input data is out of the range.
Operation Error	FMP_ RUNFAIL	133	The motor is already running or not prepared for running. Other wrong motion command.
	FMP_RESETFAIL	134	The user cannot execute AlarmReset command while the servo is ON.
	FMP_SERVOONFAIL1	135	An alarm has occurred.
	FMP_SERVOONFAIL2	136	The motor is under Emergency Stop.
	FMP_SERVOONFAIL3	137	'ServoON'signal is already assigned to input pin.
	FMP_SERVOOFFFAIL	138	The motor is under Servo ON process.
Connection Error	FMP_PACKETERROR	130	Protocol level error occurs in packet that Drive's received.
	FMP_PACKETCRCERROR	170	CRC value is not correct in packet that Drive's received.

2.2 Communication Status Window

Above return value can be categorized into 3 groups.

(1) Communication Error

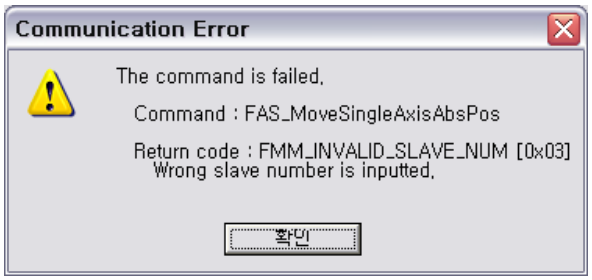


FMM_NOT_OPEN,

COM Port is not connected.

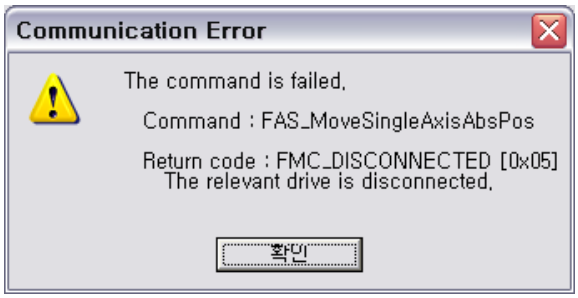
FMM_INVALID_PORT_NUM,

COM Port number is not exist. Please check the ‘Device Manager’ window in Window OS.



FMM_INVALID_SLAVE_NUM,

Slave number is not exist. Please check the ID value of the drive.



FMC_DISCONNECTED = 5,

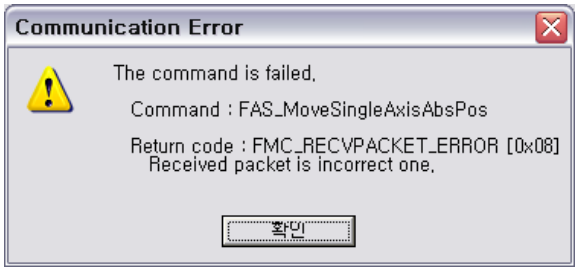
COM Port is disconnect during communication. Please check the communication cable Or Power of the drive.

FMC_TIMEOUT_ERROR,

There is no response from the drive.

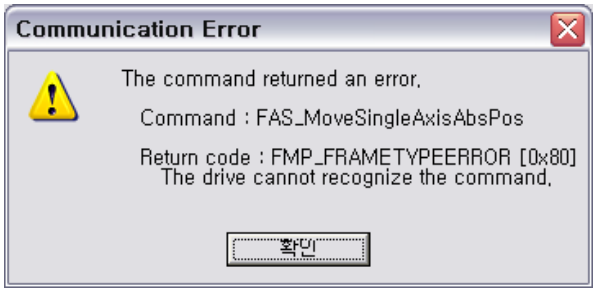
FMC_CRCFAILED_ERROR

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



FMC_RECVPACKET_ERROR,

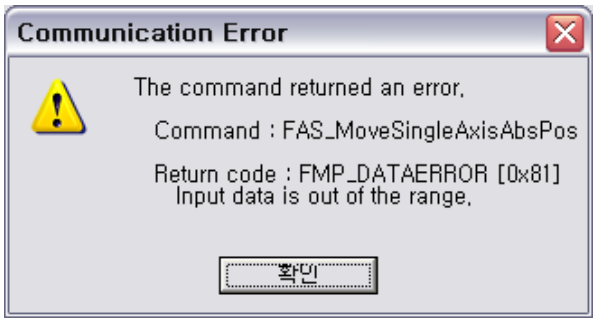
The length of received packet is not correct. Please check the possibility of noise on communication cable.



FMP_FRAMETYPEERROR = 0x80,

Drive cannot recognize the command.

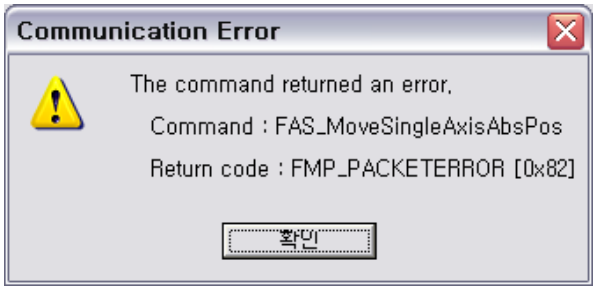
Please check if the command that you sent is correct.



FMP_DATAERROR,

The value of the sent data is out of range.

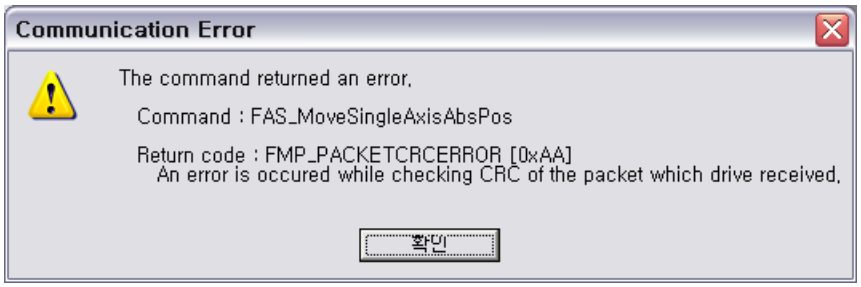
Please check if the value is within the proper range.



FMP_PACKETERROR,

The length of received packet on drive is not correct.

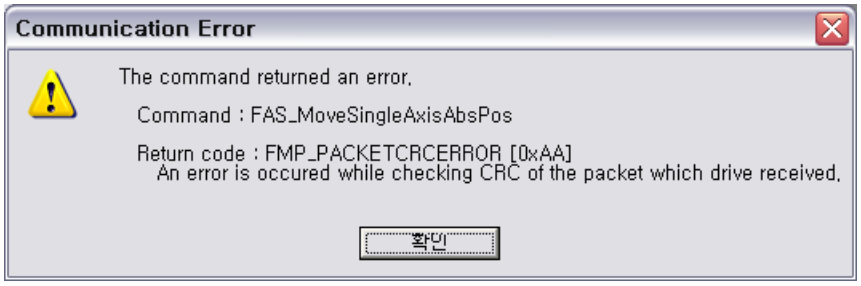
Please check the possibility of noise on communication cable.



FMP_PACKETCRCERROR = 0xAA,

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

(2) Wrong Command



FMP_RUNFAIL = 0x85,

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

-. The motor is already running

-. The motor is under stop command

-. Servo OFF status

-. other wrong motion command

FMP_RESETFAIL,

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

-. Servo ON status

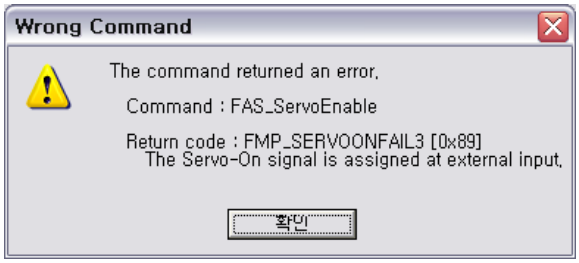
-. Already ‘Reset’ status by external input signal.

FMP_SERVOONFAIL1,

Wrong ‘Servo ON’ command during Alarm happens.

FMP_SERVOONFAIL2,

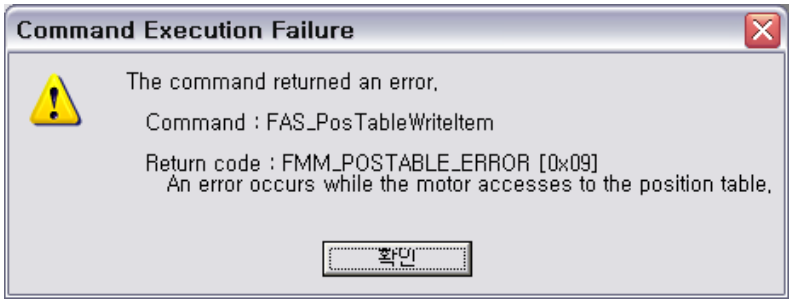
Wrong ‘Servo ON’ command during E-Stop happens.



FMP_SERVOONFAIL3,

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

(3) Command Execution Error



FMM_POSTABLE_ERROR,

The execution of DLL library for ‘Position Table’ is failed.

2.3 Drive Link Function

Function Name	Description
FAS_Connect	The drive tries to make connection with the drive module: When connected successfully, TRUE will be returned. Otherwise, FALSE will be returned.
FAS_Close	The drive tries to disconnect communication with the drive module.
FAS_GetSlaveInfo	The drive reads drive type and program version: Drive type and version information will be returned.
FAS_GetMotorInfo	The drive reads motor type and maker: Motor type and manufacturer information will be returned.
FAS_IsSlaveExist	The drive checks whether there is the relevant drive: When it exists, TRUE will be returned. Otherwise, FALSE will be returned.
FAS_EnableLog	To select the communication error log function ON/OFF : When it exists, TRUE will be returned. Otherwise, FALSE will be returned.
FAS_SetLogPath	To set the saved folder name of error log file : When folder exists, TRUE will be returned. Otherwise, FALSE will be returned.

2.3.1 FAS_Connect

FAS_Connect makes connection with DS-CL28/42-SA

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 connected successfully, TRUE will be returned. Otherwise, FALSE will be returned.

Remarks

Example

```
#include "FAS_EziMOTIONPlusR.h"

void funcInit()
{
        BYTE nPortNo = 1;  // COMM Port Number
        DWORD dwBaudrate = 115200;          // Baudrate. (Be variable by setting)
        BYTE iSlaveNo = 0;  // Slave No (0 ~ 15)
        char lpBuff[256];
        int nBuffSize = 256;
        BYTE nType;
        int nRtn;

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

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

```
                // There is no relevant slave number.
                // Check the slave number of DS-CL28/42-SA
                return;
        }

        nRtn = FAS_GetSlaveInfo(nPortNo, iSlaveNo, &nType, lpBuff, nBuffSize);
        if (nRtn != FMM_OK)

        {
                // Command has not been performed properly.
                // Refer to ReturnCodes_Define.h.
        }

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

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

See Also

FAS_Close

2.3.2 FAS_Close

Disconnects 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

2.3.3 FAS_GetSlaveInfo

Gets the version info. 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

2.3.4 FAS_GetMotorInfo

Gets 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

2.3.5 FAS_IsSlaveExist

Checks if 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 by the library only and it cannot be applicable to the protocol program mode.

Example

Refer to 'FAS_Connect' library.

See Also

FAS_Connect

2.3.6 FAS_EnableLog

Enables/Disables the log function of communication error.

Syntax

```
void FAS_EnableLog(BOOL bEnable);
```

Parameters

bEnable

Select output of Log.

Remarks

Enables/Disables the Log output during Ezi-MOITON Plus-R DLL function used. This setup
does not affect the log function of other process or other program.
'FAS_Connect'starts the log function and 'FAS_Close'ends it

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcDisableLog()
{
        BYTE nPortNo = 1;

        FAS_EnableLog(FALSE);



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

        // Connection close..
        FAS_Close(nPortNo);
}
```

See Also

FAS_SetLogPath

2.3.7 FAS_SetLogPath

Setup the path of Log output files.

Syntax

```
BOOL FAS_SetLogPath(LPCTSTR lpPath);
```

Parameters

lpPath

Folder path Character string of Log output file.

Return Value

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

Remarks

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

Example

```
#include "FAS_ EziMOTIONPlusR.h"

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

        // Log output.
        FAS_EnableLog(TRUE);

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

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

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

```
        }

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

See Also

FAS_EnableLog

2.4 Parameter Control Function

Function Name	Description
FAS_SaveAllParameters	Current parameters are saved to the ROM: Even after the 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.

2.4.1 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;

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

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

	// Check Axis Start Speed Parameter.
	nRtn = FAS_GetParameter(nPortNo, iSlaveNo, SERVO_AXISSTARTSPEED, &lParamVal);
	if (nRtn != FMM_OK)
	{
		// Command has not been performed properly.
		// Refer to ReturnCodes_Define.h.
		_ASSERT(FALSE);
	}
	else
	{
		// Parameter value saved in DS-CL28/42-SA.
		printf("Parameter [before] : Start Speed = %d ₩n", lParamVal);
	}
	// Change Axis Start Speed parameter as 200 then read it again.
	nRtn = FAS_SetParameter(nPortNo, iSlaveNo, SERVO_AXISSTARTSPEED, 200);
	_ASSERT(nRtn == FMM_OK);  // You have to check if the command didn't execute
correctly.

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


	// Check the value saved in the ROM.
	nRtn = FAS_GetROMParameter(nPortNo, iSlaveNo, SERVO_AXISSTARTSPEED, &lParamVal);
	_ASSERT(nRtn == FMM_OK);  // You have to check if the command didn't execute
correctly.
	printf("Parameter [ROM] : Start Speed = %d ₩n", lParamVal);
	// Edit the parameter value then save it in the ROM.
	nRtn = FAS_SetParameter(nPortNo, iSlaveNo, SERVO_AXISSTARTSPEED, 100);
	_ASSERT(nRtn == FMM_OK);  // You have to check if the command didn't execute
correctly.
	nRtn = FAS_SaveAllParameters(nPortNo, iSlaveNo);
	_ASSERT(nRtn == FMM_OK);
	// Disconnect.
	FAS_Close(nPortNo);
}
```

See Also

FAS_GetRomParameter

2.4.2 FAS_SetParameter

Writes the relevant parameter value to the RAM in the drive.

Syntax

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

Parameters

nPortNo

Port number of relevant drive

iSlaveNo

Slave number of relevant drive

iParamNo

Parameter number to be edited

lParamValue

Parameter value to be edited

Return Value

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

Remarks

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

Example

Refer to 'FAS_SaveAllParameter' library.

See Also

FAS_GetParameter

2.4.3 FAS_GetParameter

Reads specific parameter values of the drive

Syntax

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

Parameters

nPortNo

Port number of relevant drive

iSlaveNo

Slave number of relevant drive

iParamNo

Parameter number to be imported

lParamValue

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

2.4.4 FAS_GetRomParameter

Reads parameters saved in the ROM

Syntax

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

Parameters

nPortNo

Port number of relevant drive

iSlaveNo

Slave number of relevant drive

iParamNo

Parameter number to be imported

lRomParam

Parameter values saved in the ROM

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

FMM_INVALID_PARAMETER_NUM : There is no parameter of designated iParamNo.

Remarks

To call parameter values saved in the ROM

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

Example

Refer to 'FAS_SaveAllParameter' library.

See Also

FAS_SaveAllParameters

2.5 Servo Control Function

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

2.5.1 FAS_ServoEnable

To turn ON/OFF the drive servo

Syntax

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

Parameters

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

Return Value

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

Remarks

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

Example

```
#include "FAS_ EziMOTIONPlusR.h"

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

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

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

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

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

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

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

See Also

FAS_ServoAlarmReset

2.5.2 FAS_ServoAlarmReset

Sends AlarmReset command

Syntax

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

Parameters

nPortNo

Port number of relevant drive

iSlaveNo

Slave number of relevant drive

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

Remarks

Before sending this command, troubleshoot the alarm cause.

For alarm cause, refer to 'User Manual_Text'.

Example

Refer to 'FAS_ServoEnable' library

See Also

FAS_ServoEnable

2.6 Control I/O Function

Function Name	Description
FAS_SetIOInput	Sets the input signal level of the control input port : Input signal is set to [ON] or [OFF].
FAS_GetIOInput	Reads the current input signal status of the control input port : The signal status returns by bit for each input signal.
FAS_GetIOAssignMap	Reads the pin setting status of the IO port : The setting status for each variable signals returns by bit to the Input port.
FAS_SetIOAssignMap	Assigns the control I/O signal to IO port pin and also set the signal level : Setting for each 9 variable signals is assigned to the IO port.
FAS_IOAssignMapReadROM	Loads the pin setting status of IO port from ROM area to RAM area.

2.6.1 FAS_SetIOInput

Sets I/O input. For more information, refer to '1-2. Definition of Frame Type'.

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

dwIOSetMask

Input bitmask value to be set

dwIOCLRMask

Input bitmask value to be cleared

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

Remarks

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

Example

```
#include "FAS_ EziMOTIONPlusR.h"

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

        // Try to connect
```

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

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

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

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

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

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

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

See Also

FAS_GetIOInput

2.6.2 FAS_GetIOInput

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

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

dwIOInput

Parameter pointer which input values will be saved

Return Value

FMM_OK : Command has been normally performed.

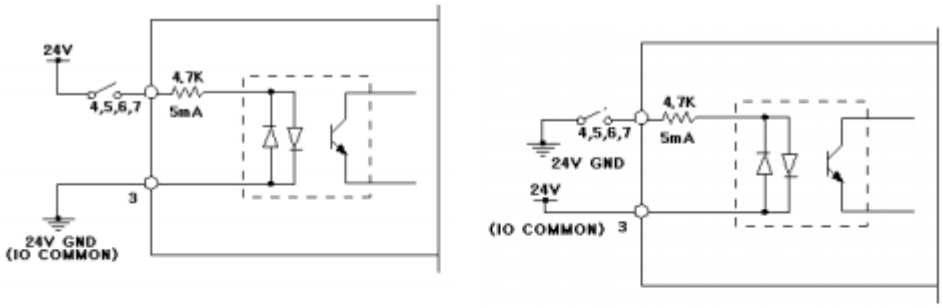
FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

Remarks

This function can read the input port status by 32bit. All of them are insulated by a photocoupler. (Refer to the figure.)



When the LED of photocoupler is turned off, the input is recognized to High.

Example

Refer to 'FAS_SetIOInput' library.

See Also

FAS_SetIOInput

2.6.3 FAS_GetIOAssignMap

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

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

iIOPinNo

I/O pin number to be read

nIOLogic

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

bLevel

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

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

Remarks

For nIOLogic, refer to 'Motion_define.h'.

Example

```
#include "FAS_ EziMOTIONPlusR.h"

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

```
	DWORD dwLogicMask;
	BYTE bLevel;
	BYTE i;
	int nRtn;

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

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

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

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

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

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

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

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

See Also

FAS_SetIOAssignMap

2.6.4 FAS_SetIOAssignMap

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

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

iIOPinNo

I/O Pin number to be read

nIOLogic

Logic value to be assigned to the relevant pin

bLevel

Active Level value of the relevant logic

Return Value

FMM_OK : Command has been normally performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.
FMM_INVALID_PARAMETER_NUM : Designated iIOPinNo or nIOLogic value is out of range.

Remarks

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

Example

Refer to 'FAS_GSetIOAssignMap' library

See Also

FAS_GetIOAssignMap

2.6.5 FAS_IOAssignMapReadROM

Loads the settings of IO assignment in ROM area

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

Remarks

Example

See Also

FAS_ GetIOAssignMap

2.7 Position Control Function

Function Name	Description
FAS_SetCommandPos	Sets the command position value to the specified value
FAS_SetActualPos	Sets the actual position value to the specified value
FAS_GetCommandPos	Reads the command position value
FAS_GetActualPos	Reads the actual position value
FAS_GetPosError	Reads the difference between the actual position value and the command position value
FAS_GetActualVel	Reads the actual running speed value while the motor is moving
FAS_ClearPosition	Sets the command position and actual position value to '0'

2.7.1 FAS_SetCommandPos

Sets 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

Commnad position value to be set.

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

Remarks

The user sets the position command (pulse output counter) value.
This function is generally used when the user sets the current position to coordinates that he wants.

Example

```
#include "FAS_ EziMOTIONPlusR.h"

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

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

```
            // Connection failed.
            // The port is not connected or the baudrate may be wrong.
            return;
        }
        // Initialize Command Position and Actual Position values to 0.
        nRtn = FAS_SetCommandPos(nPortNo, iSlaveNo, 0);
        _ASSERT(nRtn == FMM_OK);
        nRtn = FAS_SetActualPos(nPortNo, iSlaveNo, 0);
        _ASSERT(nRtn == FMM_OK);

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

See Also

FAS_SetActualPos

2.7.2 FAS_SetActualPos

Sets 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

2.7.3 FAS_GetCommandPos

Reads the command position of the current motor

Syntax

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

Parameters

nPortNo

Port number of relevant drive

iSlaveNo

Slave number of relevant drive

lCmdPos

Parameter pointer that command position value will be saved

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

Remarks

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

Example

```
#include "FAS_ EziMOTIONPlusR.h"

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

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

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

        // Check position information of DS-CL28/42-SA.
        nRtn = FAS_GetCommandPos(nPortNo, iSlaveNo, &lValue);
        _ASSERT(nRtn == FMM_OK);
        printf("CMDPOS : %d ₩n", lValue);
        nRtn = FAS_GetActualPos(nPortNo, iSlaveNo, &lValue);
        _ASSERT(nRtn == FMM_OK);
        printf("ACTPOS : %d ₩n", lValue);
        nRtn = FAS_GetPosError(nPortNo, iSlaveNo, &lValue);
        _ASSERT(nRtn == FMM_OK);
        printf("POSERR : %d ₩n", lValue);
        nRtn = FAS_GetActualVel(nPortNo, iSlaveNo, &lValue);
        _ASSERT(nRtn == FMM_OK);
        printf("ACTVEL : %d ₩n", lValue);

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

See Also

FAS_GetActualPos

2.7.4 FAS_GetActualPos

Reads 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

2.7.5 FAS_GetPosError

Reads the position error of the drive

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

2.7.6 FAS_GetActualVel

Reads 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

2.7.7 FAS_ClearPosition

Sets the command position value and actual value to '0'

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

Remarks

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

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

Example

```
#include "FAS_ EziMOTIONPlusR.h"

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

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

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

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

See Also

FAS_SetActualPos

2.8 Drive Status Control Function

Function Name	Description
FAS_GetIOAxisStatus	Reads control I/O status, running status Flag value : The current input status value and the running status Flag value will return.
FAS_GetMotionStatus	Reads the current running progress status: The command position value, the actual position value, the speed value will return.
FAS_GetAllStatus	Reads 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	Reads the running status Flag value of the relevant drive

2.8.1 FAS_GetIOAxisStatus

To read I/O Input 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 reserved(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

2.8.2 FAS_GetMotionStatus

Reads 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

2.8.3 FAS_GetAllStatus

Reads I/O Input 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 reserved(I/O output) value will be saved.

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

lCmdPos
Parameter pointer which the command position value will be saved

lActPos
Parameter pointer which the actual position value will be saved

lPosErr
Parameter pointer which the position error value will be saved

lActVel
Parameter pointer which the actual velocity value will be saved

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

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The 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

2.8.4 FAS_GetAxisStatus

Reads the motor Axis Status value. For status Flag, refer to '1-2. Definition of Frame Type'.

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

dwAxisStatus

Parameter pointer which the axis status value of the relevant motor

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

2.9 Running Control Function

Function Name	Description
FAS_MoveStop	The motor in running is decelerate and stopped.
FAS_EmergencyStop	The motor in running stops directly without deceleration
FAS_MoveOriginSingleAxis	The motor starts the origin return.
FAS_MoveSingleAxisAbsPos	The motor moves as much as the given absolute position value.
FAS_MoveSingleAxisIncPos	The motor moves as much as the given incremental position value.
FAS_MoveToLimit	The motor moves up to the position that the limit sensor is detected.
FAS_MoveVelocity	The motor moves to the given velocity and direction: This function is available to Jog motion.
FAS_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_MoveSingleAxisAbsPosEx	The motor moves as much as the given absolute position value with custom accel/decel time value.
FAS_MoveSingleAxisIncPosEx	The motor moves as much as the given incremental position value with custom accel/decel time value.
FAS_MoveVelocityEx	The motor moves to the given velocity and direction: This function is available to Jog motion with custom accel/decel time value.

2.9.1 FAS_MoveStop

Stops 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

2.9.2 FAS_EmergencyStop

Stops the motor immediately

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

2.9.3 FAS_MoveOriginSingleAxis

Searchs the origin of system. For more information, refer to 'User Manual_Text 9.2 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

2.9.4 FAS_MoveSingleAxisAbsPos

Moves the motor to the absolute coordinate

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

lAbsPos

Absolute coordinate of position to move

lVelocity

Velocity when the motor moves

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

Remarks

Example

```
#include "FAS_ EziMOTIONPlusR.h"

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

```
long lAbsPos, lIncPos, lVelocity;
int nRtn;

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

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

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

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

if (dwInput & (SERVO_IN_BITMASK_STOP | SERVO_IN_ BITMASK _PAUSE | SERVO_IN_
BITMASK _ESTOP))
        FAS_SetIOInput(nPortNo, iSlaveNo, 0, SERVO_IN_ BITMASK _STOP | SERVO_IN_
BITMASK _PAUSE | SERVO_IN_ BITMASK _ESTOP);

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

// Stand by until motion command is completely finished.
do
```

```
{
        Sleep(1);

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

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

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

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

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

See Also

2.9.5 FAS_MoveSingleAxisIncPos

Moves 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

2.9.6 FAS_MoveToLimit

Sends 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

Velocioty 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

2.9.7 FAS_MoveVelocity

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

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

lVelocity

Velocity when the motor moves

iVelDir

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

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

2.9.8 FAS_AllMoveStop

Stops the motor that connected in same port.

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

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

Return Value

No response

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

2.9.9 FAS_AllEmergencyStop

Stops the motor that connected in same port without deceleration

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

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

Return Value

No response

Remarks

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

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

2.9.10 FAS_AllMoveOriginSingleAxis

Searchs the origin of system for all motor that is connected in same port. For more information, refer to 'User Manual_Text 9.2 Origin Return'.

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

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

Return Value

No response

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

2.9.11 FAS_AllMoveSingleAxisAbsPos

Moves the motor that connected in same port to the absolute coordinate

Syntax

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

Parameters

nPortNo

Port number of relevant 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

2.9.12 FAS_AllMoveSingleAxisIncPos

Moves the motor that connected in same port to the incremental coordinate value

Syntax

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

Parameters

nPortNo

Port number of relevant 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

2.9.13 FAS_MoveSingleAxisAbsPosEx

Moves the motor to the absolute coordinate

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

lAbsPos

Absolute coordinate of position to move

lVelocity

Velocity when the motor moves

lpExOption

Custom option.

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

Remarks

Refer to MOTION_OPTION_EX struct.

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcMoveEx()
{
        BYTE nPortNo = 1;  // COMM Port Number
        BYTE iSlaveNo = 0;  // Slave No (0 ~ 15)
```

```
DWORD dwAxisStatus, dwInput;
EZISERVO_AXISSTATUS stAxisStatus;
long lAbsPos, lIncPos, lVelocity;
MOTION_OPTION_EX opt = {0};
int nRtn;

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

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

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

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

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

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


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

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

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

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

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

See Also

2.9.14 FAS_MoveSingleAxisIncPosEx

Moves the motor to the Incremental coordinate

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

lIncPos

Incremental coordinate of position to move

lVelocity

Velocity when the motor moves

lpExOption

Custom option.

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

Remarks

Example

See Also

2.9.15 FAS_MoveVelocityEx

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

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

lVelocity

Velocity when the motor moves

iVelDir

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

lpExOption

Custom option.

Return Value

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

Remarks

Refer to VELOCITY_OPTION_EX struct.

Example

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

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

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

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

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

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

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

See Also

2.10 Position Table Control Function

(V06.01.30.22 and later F/W version supports PT function)

Function Name	Description
FAS_PosTableReadItem	To read items of RAM area in the specific all items of position table
FAS_PosTableWriteItem	To save specific all items of position table items to RAM area
FAS_PosTableWriteROM	To save all of position table values to ROM area : Total 256 PT values are aved.
FAS_PosTableReadROM	To read position table values in ROM area : Total 256 PT values are read.
FAS_PosTableRunItem	The motor starts to run from the designated position table in sequence.
FAS_PosTableReadOneItem	To read items of RAM area in the specific one item of position table
FAS_PosTableWriteOneItem	To save specific one item of position table items to RAM area

2.10.1 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

V06.01.30.22 and later F/W version supports PT function

Example

```
#include "FAS_EziMOTIONPlusR.h"

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

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

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

		nodeItem.lPosition = 260000;  // Change the position value to 260000.
		nodeItem.wBranch = 23;                  // Set next command to 23.
		nodeItem.wContinuous = 1;               // Next command should be connected
	without deceleration.

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

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

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

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

See Also

FAS_PosTableWriteItem

2.10.2 FAS_PosTableWriteItem

To edit specific items in the position table

Syntax

```
int FAS_PosIableWriteItem(
    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.

V06.01.30.22 and later F/W version supports PT function

Example

See Also

FAS_PosTableReadItem

2.10.3 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.
V06.01.30.22 and later F/W version supports PT function

Example

See Also

FAS_PosTableReadROM

2.10.4 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

V06.01.30.22 and later F/W version supports PT function

Example

See Also

FAS_PosTableWriteROM

2.10.5 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

V06.01.30.22 and later F/W version supports PT function

Example

See Also

FAS_GetAllStatus

FAS_MoveStop

FAS_EmergencyStop

2.10.6 FAS_PosTableReadOneItem

To read a one item in the specific position table

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

wItemNo

Item number to be read

wOffset

offset value which will be read in PT items. (Refer to '1-2-6. Position Table Item')

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

FMM_INVALID_PARAMETER_NUM : wItemNo is out of range.

Remarks

V06.01.30.22 and later F/W version supports PT function

Example

See Also

FAS_PosTableWriteOneItem

2.10.7 FAS_PosTableWriteOneItem

To edit one item in the specific position table

Syntax

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

Parameters

nPortNo

Port number of relevant drive.

iSlaveNo

Slave number of relevant drive.

wItemNo

Item number to be edited

wOffset

offset value which will be save in PT items. (Refer to '1-2-6. Position Table Item')

Return Value

FMM_OK : Command has been normally performed.
FMM_NOT_OPEN : The drive has not been connected yet.
FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.
FMM_INVALID_SLAVE_NUM : There is no drive of iSlaveNo in the relevant port.
FMC_POSTABLE_ERROR : An error occurs while position table is being written.
FMM_INVALID_PARAMETER_NUM : wItemNo is out of range.

Remarks

V06.01.30.22 and later F/W version supports PT function

Example

See Also

FAS_PosTableReadOneItem

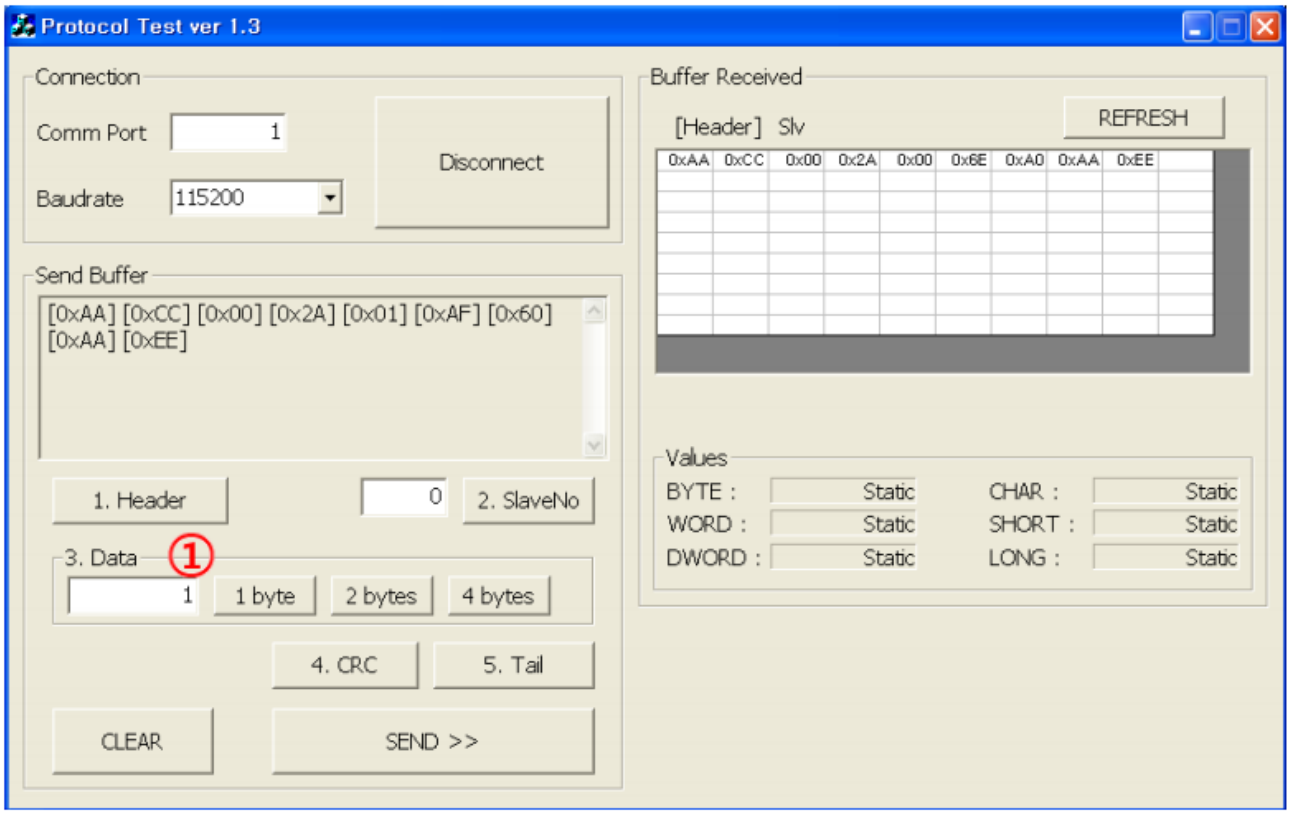
3. Protocol for PLC Program

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.

3.1 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 ‘Slave No’.

(4) Frame type : Insert ‘Frame type’.

You can find next table information in ‘1.2.2 Frame Type and Data Configuration’ on Technical Manual(DS-CL28/42-SA)_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

Insert ‘42’ in area① and click ‘1 byte’ because the size of Frame Type is 1 byte.

(5) Data : To make Servo ON status, the data is ‘1’. Insert ‘1’ in area① and click 1 byte’.

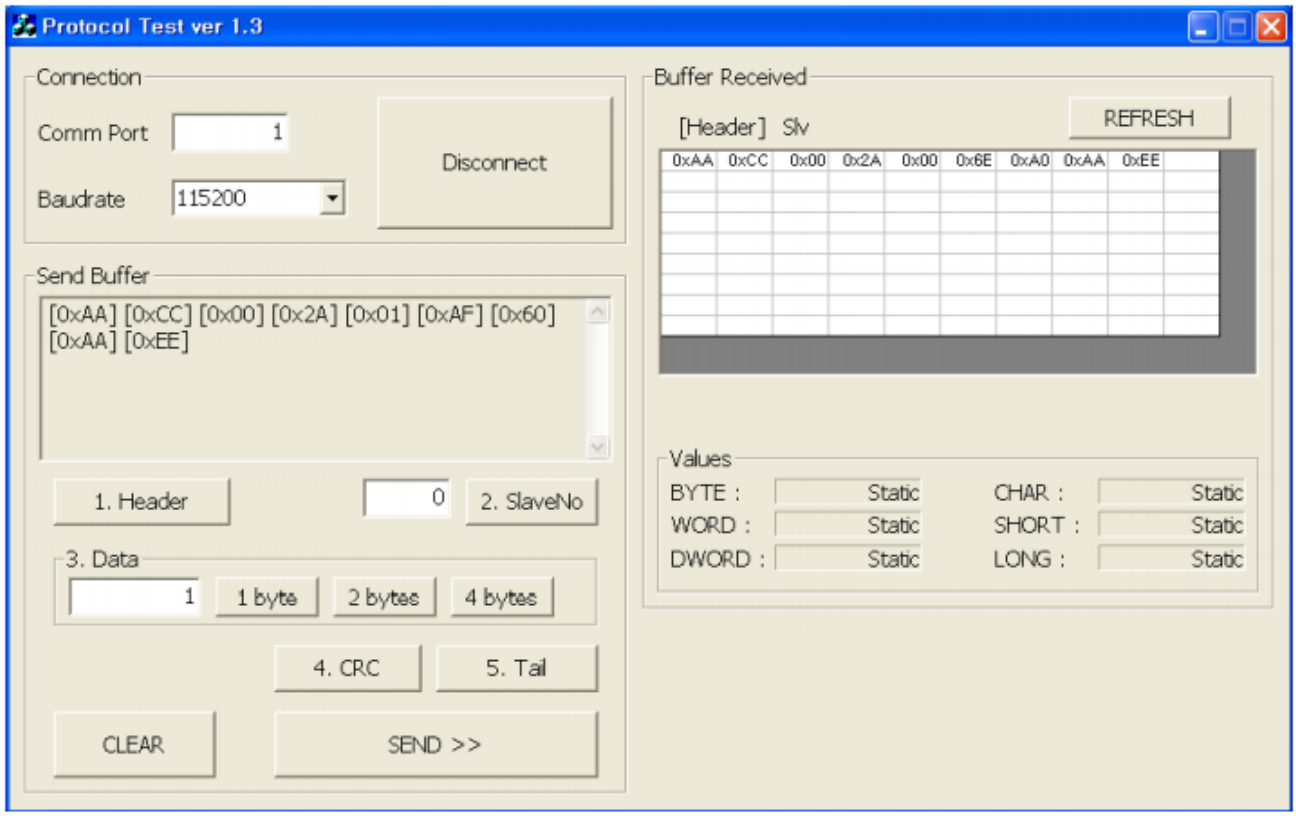
(6) CRC : Click ‘CRC’ and the calculated result value(2 bytes) is displayed on ‘Send Buffer’ window.

(7) Tail : click ‘Tail’ and you can see ‘[0XAA][0xEE]’ on ‘Send Buffer’ window.

(8) Finally click ‘Send’ button to send command characters to DS-CL28/42-SA. You can check the motor torque and LED flash for Servo ON status.

(9) After sending command you can check the answering informations from DS-CL28/42-SA on ‘Buffer Received’ window.

3.2 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 ‘4bytes’.

(5) Data(Running speed) : insert‘5000’and click ‘4 bytes’.

(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.

3.3. 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 Technical Manual (DS-CL28/42-SA)_Communication Function.

International Customer

Person in Charge :

Daniel Jang

daniel@dingsmotion.com

Building 1#, 355 Longjin Road, Changzhou Economic Development Zone, Jiangsu, China

+86-519-85177826, 85177827

North America Customer

Person in Charge :

Nicolas Ha

sales@dingsmotionusa.com

335 Cochrane Circle Morgan Hill, CA 95037

+1-408-612-4970

China Customer

Person in Charge :

Sweet Shi

info@dingsmotion.com

Building 1#, 355 Longjin Road, Changzhou Economic Development Zone, Jiangsu, China

+86-519-85177826, 85177827

DINGS
Precision Motion Specialist

+86-0519-8517 7825

+86-0519-8517 7807

Building 1#, 355 Longjin Road, Changzhou Economic Development Zone, Jiangsu, China

www.dingsmotion.com

It is prohibited to copyright or replication of the part or whole of user manual without permission.