Ezi-MOTIONLINK® Plus-E

Network Based Motion Controller Plug-in to Servo Drives

User Manual

Communication Function

(**Rev.01**)

FASTECH

Tabls of contents

1 . Communication Protocols

1 - 1 . Communication Functions

Ezi-MOTIONLINK Plus-E can control up to 254 axes (1 ~ 254) using Ethernet communication.

1 - 1 - 1 . Communication Specifications

Item	Specification
Communication Speed	10/100base-T/TX
Communication Type(Protocol)	UDP (Port No. : **3001**)
Max Cabling Length	100m
Min Cable length between drive	More than 20cm
Number of Connected Axes	254axes (No. 01~FE)

1 - 1 - 2 . Ethernet IP address

- Factory Default Value

1) Gateway : 192.168.0.1
2) Subnet Mask : 255.255.255.0
3) IP address : 192.168.0.x (x is set by an external switch)

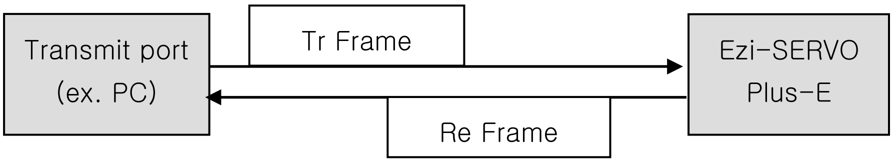
- When connecting to Ezi-MOTIONLINK Plus-E directly from a PC or Ethernet device, be sure to set the network setting according to the above IP address.
  If it is not set or is different, is can not be connected.
- If the switch set to 255(FF), IP address is automatically set.

Because it uses DHCP, IP address set automatically only when using router.

- When connecting directly from the controller(PC, PLC, etc.), be sure to set ther IP address with the switch
- Set the IP address automatically only when the default IP address is not used.
  If the IP set automatically, connect the user program(GUI), save the IP address, turn off the power, and set the last number of the IP with the switch.
- When the IP setting switch set to 0, the IP setting is reset to the above value.

1 - 1 - 3 . Ethernet Protocol

1) Overview of communication FRAME



2) Basic structure of FRAME

UDP Header	*Frame Data*
8bytes	4~254 bytes

The UDP Header contains the following information:

① Transmit port number sending : 2bytes

② Receiving port number : 2bytes

③ Data length : 2bytes, Total length of UDP and Frame Data

④ Checksum : 2bytes

1 - 1 - 4 . Response Frame Structure

The detailed configuration of the receiving frame data is as follows:

Header	Length	Sync No.	Reserved	Frame type	Data
1 byte	1 byte	1 byte	1 byte (0x00)	1 byte	0 ~ 253 bytes.

① Header : 0xAA, Displays that the beginning of Frame

② Length : Length of Data after Length
(Sync No. + Reserved + Frame type + Data)

③ Reserved : 1 byte (Input as "0x00")

④ Sync No. : The Sync number of the packet is used to check whether the command is executed in the drive module
The value should change every time when you send a new command

⑤ Frame type : Specify the command type of the Frame. The types are listed below
See the section 「Frame type and Data configuration」.

⑥ Data : The data structure and length of this clause are determined by the frame type. The detailed structure is
See the 「Frame typeand Data configuration」 section below.

1 - 1 - 5 . Reply Frame Structure and Communication Error

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

Header	Length	Sync No.	Reserved	Frame type	Data	
1 byte	1 byte	1 byte	1 byte (0x00)	1 byte	1byte	0 ~ 252 bytes
					Communication statusCommunication status	ResponseResponse Data

① Header : 0xAA, Displays that the beginning of Frame.

② Length : Length of Data after Length
(Sync No. + Reserved + Frame type + Data)

③ Sync No. : Same as Response Frame
(If it does not match the data at the time of reception, recognize it as an error

condition.)

④ Reserved : 1 byte(0x00)

⑤ Frame type : Same as Response Frame

(If It does not match the data at the time of transmission, recognize it as an error condition.)

⑥ Data : In reply, 1 byte of data indicating communication status(error/normal) is included.
The simple Execution command has only the communication satus data.

The contents of byte indicating communication satus are as follows.

Hexa code	Decimal code	Description
0x00	0	Communication is normal.
0x80	128	Frame type Error : Response Frame type connot be recognized.
0x81	129	Data error, ROM data read/write error. : Data value responded is without the given range.
0x82	130	Response Frame Error : Frame data received is out of this specification.
0x85	133	It is an invalid command such that the motor is already in operation or is not ready for operation.
0x86	134	Alarm Reset command can not be executed in Servo ON state.
0x87	135	An alarm has occurred.
0x88	136	Emergency Stop is in progress.
0x89	137	'Servo ON' is already set for the input signal.

⚠ Caution	**1) If ‘Header’ and ‘Length’ calues of the receiving 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 . Structur of Frame

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

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

● 0xXX of Frame type is value of Hex, the value in ()is Dec.

<table>
<tr><th>Frame type</th><th>Library Name</th><th>Description</th></tr>
<tr><td>0x01
(1)</td><td>FAS_
GetboardInfo</td><td>Connected slave type and program version information are required.

Sending : 0 byte
Response : 1~248 bytes
<table>
<tr><td>1 byte</td><td>1 byte</td><td>0~253 bytes</td></tr>
<tr><td>Communication status</td><td>board</td><td>ACII string with NULL byte
(strlen() + 1 byes)</td></tr>
</table>
◆ board type : 100 : Ezi-MOTIONLINK Plus-E</td></tr>
<tr><td>0x10
(16)</td><td>FAS_
SaveAllParameters</td><td>Current setting parameters & assign of IO signals are saved in the ROM of the drive. Even though the drive is powered off, saving these must be possible. Values set at ‘FAS_SetParameter’ &‘FAS_SetIOAssignMap’ are saved together.

Sending : 0 byte
Response : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>Communication status</td></tr>
</table></td></tr>
<tr><td>0x11
(17)</td><td>FAS_
GetRomParameter</td><td>Specific parameter values in the ROM are read. ParameterParameter value
Sending : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>ParameterParameter number (0~29)</td></tr>
</table>
Response : 5 bytes
<table>
<tr><td>1 byte</td><td>4 bytes</td></tr>
<tr><td>Communication status</td><td>ParameterParameter value</td></tr>
</table>
Refer to 「1-2-2. Parameter List」.</td></tr>
<tr><td>0x12
(18)</td><td>FAS_
SetParameter</td><td>Specific parameter values are saved to the RAM.Parameter
Sending : 5 bytes</td></tr>
</table>

<table>
<tr><td></td><td></td><td>
<table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Parameter number (0~29)</td><td>ParameterParameter value</td></tr></table>
Response : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table>
Refer to 「1-2-2. Parameter List」
</td></tr>
<tr><td>0x13
(19)</td><td>FAS_
GetParameter</td><td>
Specific parameter values in the RAM are read Parameter
Sending : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>Parameter number (0~29)</td></tr></table>
Response : 5 bytes
<table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication status</td><td>ParameterParameter value</td></tr></table>
Refer to 「1-2-2. Parameter List」
</td></tr>
<tr><td>0x20
(32)</td><td>FAS_
SetIOOutput</td><td>
Output signal level of the control output port is set.
Sending : 8 bytes
<table><tr><td>4 bytes</td><td>4 bytes</td></tr><tr><td>I/O set mask value</td><td>I/O clear mask value</td></tr></table>
When specific bit of the set mask is ‘1’, the relevant output port signal is set to [ON].
When specific bit of the clear mask is ‘1’, the relevant output port signal is set to [OFF].
For more information, refer to 「1-2-3.Bit setup of Output Pin」.
Response : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table>
</td></tr>
<tr><td>0x21
(33)</td><td>FAS_
SetIOInput</td><td>
Input signal level of the control input port is set.
Sending : 8 bytes
<table><tr><td>4 bytes</td><td>4 bytes</td></tr><tr><td>I/O set mask value</td><td>I/O clear mask value</td></tr></table>
</td></tr>
</table>

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

Response : 1 byte<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x22
(34)</td><td>FAS_
GetIOInput</td><td>Current input signal status of the control input port is read.

Sending : 0 byte

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

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

<table>
<tr>
<td>0x24
(36)</td>
<td>FAS_
SetIOAssignMap</td>
<td>To assign control I/O signals to the pin of CN1 port and set the signal level. By running 'FAS_SaveAllParameters', you can save the setting value to the ROM.
Sending : 6 bytes
<table>
<tr><td>1 byte</td><td>4 bytes</td><td>1 byte</td></tr>
<tr><td>I/O number</td><td>I/O pin masking data</td><td>Settinglevel</td></tr>
</table>
◆I/O number : '0~3' corresponds to 'Limit+, Limit-, Org,IN1,..., IN9' respectively, '4' corresponds to OUT1' respectively
◆ I/O pin masking data: Refer to 「1-2-4. Bit setup of Input Pin」
.♦ Level Setting: 0: Active Low, 1: Active High
Response : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>Communication status</td></tr>
</table>
</td>
</tr>
<tr>
<td>0x25
(37)</td>
<td>FAS_
GetIOAssignMap</td>
<td>Pin setting status of CN1 port is read
Sending : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>I/O number</td></tr>
</table>
◆I/O number : '0~3' corresponds to 'Limit+, Limit-, Org, IN1,respectively. '4' corresponds to 'OUT1' respectively.
Response : 6 bytes
<table>
<tr><td>1 byte</td><td>4 bytes</td><td>1 byte</td></tr>
<tr><td>Communication status</td><td>IO pin masking status</td><td>Level status</td></tr>
</table>
For more information, refer to '0x24'Frame type.</td>
</tr>
<tr>
<td>0x26
(38)</td>
<td>FAS_
IOAssignMapReadROM</td>
<td>Pin setting status of CN1 port is loaded to RAM from ROM area.
Sending : 0 byte
Response : 2 bytes
<table>
<tr><td>1 byte</td><td>1 byte</td></tr>
<tr><td>Communication status</td><td>Command performing status
(0 : complete, values except 0 : error)</td></tr>
</table>
</td>
</tr>
<tr>
<td>0x27
(39)</td>
<td>FAS_
TriggerOutput_RunA</td>
<td>Start/Stop command for 'Compare Out' signal
(Periodic output)
Sending : 18 bytes
<table>
<tr><td>1 byte</td><td>4 bytes</td><td>4 bytes</td></tr>
<tr><td>Output start/stop
(1:start 0:stop)</td><td>Pulse start position</td><td>Pulse period [pulse]</td></tr>
</table>
</td>
</tr>
</table>

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

Response : 2 bytes
<table>
<tr><td>1 byte</td><td>1 byte</td></tr>
<tr><td>Communication status</td><td>Command performing status
(0 : complete, values except 0: error)</td></tr>
</table>
</td>
</tr>
<tr>
<td>0x28
(40)</td>
<td>FAS_
TriggerOutput_
Status</td>
<td>
Command to check whether the current signal (Compare Out) output function is active or not
Sending : 0 byte
Response : 2 bytes
<table>
<tr><td>1 byte</td><td>1 byte</td></tr>
<tr><td>Communication status</td><td>Status (1:output ON, 0 :output OFF)</td></tr>
</table>
</td>
</tr>
<tr>
<td>0x7E
(126)</td>
<td>FAS_SetTriggerOutputEx</td>
<td>
Setting for generating output at a specific position on the set output
(Available after setting the output signal to User Out)
SendingSending : 245 bytes
<table>
<tr><td>1 byte</td><td>1 byte</td><td>2 bytes</td><td>1byte</td></tr>
<tr><td>User OutNumber
(0~8)</td><td>Output start/End command
(1: Start 0: End)</td><td>Output On Time
(In ms, 1~65,535)</td><td>Output position count</td></tr>
</table>
<table>
<tr><td>240 bytes</td></tr>
<tr><td>Output position Array(4bytes * 60)
Location: -134,217,728~134,217,727</td></tr>
</table>
♦ Number of ouput position : 1~60
♦ Output position With Array : Based on 4byte, 60 arrays even if the number of output positions is not 60, the output position array has 60 information inputs.
♦ Output is automatically close when the number of outputs is reached.
♦ To output, execute the move command after setting.
The position of the move command must be greater if the last position of the number of output positions is positive and less than if it is negative
It is necessary to set the output position to a proper value because the normal output may not be obtained according to the start position (present position).
It is necessary to set the proper value because the normal output may not be
</td>
</tr>
</table>

<table>
<tr><td></td><td></td><td>generated according to the driving speed and output ON time setting.

Response : 1 byte<table><tr><th>1 byte</th></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x7F
(127)</td><td>FAS_GetTriggerOutputEx</td><td>Information that is set with FAS_SetTriggerOutputEx and a command that checks the output status.

SendingSending : 1byte<table><tr><th>1 byte</th></tr><tr><td>User Out number</td></tr></table>♦ User Out number : User out number from which information can be verified(0~8)

Response : 245bytes<table><tr><th>1 byte</th><th>1 byte</th><th>2 bytes</th><th>1byte</th></tr><tr><td>Communication status</td><td>Output status</td><td>Output On time
(ms, 1~65535)</td><td>Number of ouput locations
(1~60)</td></tr></table><table><tr><th>240 bytes</th></tr><tr><td>Output location Array(4bytes * 60)
Location: -134,217,728~134,217,727</td></tr></table>♦ Output status : Run/Stop status of corresponding user out number
0 : Stop
2 : Run</td></tr>
<tr><td>0x2A
(42)</td><td>FAS_
ServoEnable</td><td>Servo ON/OFF status is set.
Sending : 1 byte<table><tr><th>1 byte</th></tr><tr><td>0:OFF, 1:ON</td></tr></table>Response : 1 byte</td></tr>
</table>

		1 byte Communication status
0x2B (43)	FAS_ ServoAlarmReset	Servo alarm status is reset. Sending : 0 byte Response : 1 byte 1 byte Communication status
0x31 (49)	FAS_ MoveStop	To request to stop running the motor Sending : 0 byte Response : 1 byte 1 byte Communication status
0x32 (50)	FAS_ EmergencyStop	To request the running motor to stop emergently Sending : 0 byte Response : 1 byte 1 byte Communication status
0x33 (51)	FAS_ MoveOriginSingleAxis	To request the motor to return to the origin at the current setting parameter condition Sending : 0 byte Response : 1 byte 1 byte Communication status
0x34 (52)	FAS_ MoveSingleAxisAbsPos	To request the motor to move its position as much as the absolute value[pulse] Sending : 8 bytes

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

Sending : 8 bytes
<table>
<tr><td>4 bytes</td><td>4 bytes</td></tr>
<tr><td>Incremental position value</td><td>Running speed [pps]</td></tr>
</table>
Response : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>Communication status</td></tr>
</table>
</td></tr>
<tr><td>0x36
(54)</td><td>FAS_
MoveToLimit</td><td>
To request the motor to start limit motion at the current setting parameter condition

Sending : 5 bytes
<table>
<tr><td>4 bytes</td><td>1 byte</td></tr>
<tr><td>Running speed[pps]</td><td>Running direction (0:-Limit 1:+Limit)</td></tr>
</table>
Response : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>Communication status</td></tr>
</table>
</td></tr>
</table>

<table>
<tr>
<td>0x37
(55)</td>
<td>FAS_
MoveVelocity</td>
<td>To request the motor to start jog motion at the current setting parameter condition

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

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

<table>
<tr>
<td>0x3A
(58)</td>
<td>FAS_
VelocityOverride</td>
<td>
To request the motor to change the running speed value [pps] while it is in running.
Sending : 4 bytes
<table>
<tr><th>4 bytes</th></tr>
<tr><td>Changed Running speed[pps]</td></tr>
</table>
The accel/decel time is assigned to 'Axis Acc Time' and 'Axis Dec Time' value in parameter lists.
Response : 1 byte
<table>
<tr><th>1 byte</th></tr>
<tr><td>Communication status</td></tr>
</table>
♦ Only at constant speed.
</td>
</tr>
<tr>
<td>0x80
(128)</td>
<td>FAS_
MoveSingleAxisAbsPosEx</td>
<td>
Request the motor to move its position as much as the absolute value [pulse] with Custom Accel. / Decel. Time[msec]
Sending : 40 bytes
<table>
<tr><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 ooption :
0x0001 : reserved
0x0002 : Custom Accel. Time is used.
0x0004 : Custom Decel. Time is used.
If the Flag bit is OFF status (0), Accel./Decel. Time value is used that saved in controller.
Response : 1 byte
<table>
<tr><th>1 byte</th></tr>
<tr><td>Communication status</td></tr>
</table>
</td>
</tr>
</table>

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

Sending : 40 bytes
<table>
<tr><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 ooption : 0x0001 : reserved
0x0002 : Custom Accel. Time is used.
0x0004 : Custom Decel. Time is used.
If the Flag bit is OFF status(0), Accel./Decel. Time value is used that saved in controller.

Response : 1 byte
<table>
<tr><th>1 byte</th></tr>
<tr><td>Communication status</td></tr>
</table>
</td>
</tr>
<tr>
<td>0x82
(130)</td>
<td>FAS_
MoveVelocityEx</td>
<td>
Request the motor 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 ooption : 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>
To request the Flag value of displaying the running status

Sending : 0 byte
Response : 5 bytes
</td>
</tr>
</table>

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

Sending : 0 byte
Response : 13 bytes
<table>
<tr><td>1 byte</td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td></tr>
<tr><td>Communication status</td><td>Input status value</td><td>Output status value</td><td>Status flag value</td></tr>
</table>
</td>
</tr>
<tr>
<td>0x42
(66)</td>
<td>FAS_
GetMotionStatus</td>
<td>
To request the current running progress status and its PT number
(Frame type 0x51, 0x53, 0x54, and 0x55 are packed.)

Sending : 0 byte
Response : 21 bytes
<table>
<tr><td>1 byte</td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td></tr>
<tr><td>Communication status</td><td>Command Position value</td><td>Actual Position value</td><td>Position difference value</td><td>Running speed value</td><td>Current running PT number</td></tr>
</table>
</td>
</tr>
<tr>
<td>0x43
(67)</td>
<td>FAS_
GetAllStatus</td>
<td>
To request all data including the current running status
(Frame type 0x41, and 0x42 are packed.)

Sending : 0 byte
Response : 33 bytes
<table>
<tr><td>1 byte</td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td></tr>
<tr><td>Communication status</td><td>Input status value</td><td>Output status value</td><td>Status Flag value</td></tr>
</table>
<table>
<tr><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td></tr>
<tr><td>Command Position value</td><td>Actual Position value</td><td>Position Difference value</td><td>Running speed value</td><td>Current running PT number</td></tr>
</table>
</td>
</tr>
<tr>
<td>0x50
(80)</td>
<td>FAS_
SetCommandPos</td>
<td>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.</td>
</tr>
</table>

<table>
<tr><td></td><td></td><td>Sending : 4 bytes<table><tr><td>4 bytes</td></tr><tr><td>Command position setting count value</td></tr></table>Response : 1 byte<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x51
(81)</td><td>FAS_
GetCommandPos</td><td>To request the command position value [pulse] being tracked.
Sending : 0 byte
Response : 5 bytes<table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication status</td><td>Command position value</td></tr></table></td></tr>
<tr><td>0x52
(82)</td><td>FAS_
SetActualPos</td><td>Ezi-MOTIONLINK Plus-E 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><td>4 bytes</td></tr><tr><td>Actual position count value</td></tr></table>Response : 1 byte<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x53
(83)</td><td>FAS_
GetActualPos</td><td>To request the current actual position value [pulse].
Sending : 0 byte
Response : 5 bytes<table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication status</td><td>Actual position value</td></tr></table></td></tr>
<tr><td>0x54
(84)</td><td>FAS_
GetPosError</td><td>To request the difference [pulse] between the command position value and the actual position value.
Sending : 0 byte</td></tr>
</table>

<table>
<tr><td></td><td></td><td>Response : 5 byte
<table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication status</td><td>Position difference value</td></tr></table>
By this value, the user can check the current running status (how much inposition is tracked).</td></tr>
<tr><td>0x55
(85)</td><td>FAS_
GetActualVel</td><td>To request the current running speed value [pps]
Sending : 0 byte
Response : 5 bytes
<table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication status</td><td>Speed value</td></tr></table></td></tr>
<tr><td>0x56
(86)</td><td>FAS_
ClearPosition</td><td>Set both the command position value and actual position value to '0'.
SendingSending : 0 byte
ResponseResponse : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x58
(88)</td><td>FAS_
MovePause</td><td>To request the pause start and pause end of motor motioning.
Sending : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>0:pause release, 1:pause start</td></tr></table>
Response : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table></td></tr>
<tr><td>0x60
(96)</td><td>FAS_
PosTableReadItem</td><td>To read PT values in the RAM of the drive.
Sending : 2 bytes
<table><tr><td>2 bytes</td></tr><tr><td>Readable PT number (0~255)</td></tr></table>
Response : 65 bytes
<table><tr><td>1 byte</td><td>64 bytes</td></tr><tr><td>Communication status</td><td>Relevant PT values</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td></td><td>For items by each PT,
refer to 「1-2-6. Position Table Item」.</td></tr>
<tr><td>0x61
(97)</td><td>FAS_
PosTableWriteItem</td><td>To save PT values to the RAM of the drive.

Sending : 66 bytes
<table>
<tr><td>2 bytes</td><td>64 bytes</td></tr>
<tr><td>PT 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><td>1 byte</td><td>1 byte</td></tr>
<tr><td>Communication status</td><td>Command performing status
(values except 0 : complete, 0: error)</td></tr>
</table></td></tr>
<tr><td>0x62
(98)</td><td>FAS_
PosTableReadROM</td><td>To read all PT values (256 ea) in the ROM of the drive

Sending : 0 byte

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

Sending : 0 byte

Response : 2 bytes
<table>
<tr><td>1 byte</td><td>1 byte</td></tr>
<tr><td>Communication status</td><td>Command performing status
(0 : complete, values except 0: error)</td></tr>
</table></td></tr>
<tr><td>0x64
(100)</td><td>FAS_
PosTableRunItem</td><td>To start the position table operation from the designated PT number

Sending : 2 bytes
<table>
<tr><td>2 bytes</td></tr>
<tr><td>PT Number
(0~255)</td></tr>
</table>
Response : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>Communication status</td></tr>
</table></td></tr>
</table>

<table>
<tr>
<td>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><td>2 bytes</td><td>2 bytes</td></tr>
<tr><td>PT Number (0~255)</td><td>Offset value 값(0~40)</td></tr>
</table>
Refer to 「1-2-6. Position Table Item」」 for Offset value.

Response : 5 bytes
<table>
<tr><td>1 byte</td><td>4 bytes</td></tr>
<tr><td>Communication status</td><td>Relecant one of PT value</td></tr>
</table>
</td>
</tr>
<tr>
<td>0x6B
(107)</td>
<td>FAS_PosTableWriteOneItem</td>
<td>
To save one of PT calues to the RAM of the drieve.

Sending : 8 bytes
<table>
<tr><td>2 bytes</td><td>2bytes</td><td>4 bytes</td></tr>
<tr><td>PT Number(0~255)</td><td>Offset calue (0~40)</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><td>1 byte</td><td>1 byte</td></tr>
<tr><td>Communication status</td><td>Command performing status
(values except 0 : complete, 0: error)</td></tr>
</table>
</td>
</tr>
</table>

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

1-2-2. Parameter Lists

No	Name	Unit	Lower	Upper	Default
0	Encoder Multiply		0	3	8
1	Axis Max Speed	[pps]	1	2,500,000	500,000
2	Axis Start Speed	[pps]	1	35,000	1
3	Axis Acc Time	[msec]	1	9,999	100
4	Axis Dec Time	[msec]	1	9,999	100
5	Speed Override	[%]	1	500	100
6	Jog Speed	[pps]	1	2,500,000	5,000
7	Jog Start Speed	[pps]	1	35,000	1
8	Jog Acc Dec Time	[msec]	1	9,999	100
9	S/W Limit Plus Value	[pulse]	-134,217,728	134,217,727	134,217,727
10	S/W Limit Minus Value	[pulse]	-134,217,728	134,217,727	-134,217,728
11	S/W Limit Stop Method		0	2	2
12	H/W Limit Stop Method		0	1	0
13	Limit Sensor Logic		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	1
19	Org OffSet	[pulse]	-134,217,728	134,217,727	0
20	Org Position Set	[pulse]	-134,217,728	134,217,727	0
21	Org Sensor Logic		0	1	0
22	Limit Sensor Dir		0	1	0
23	Pulse Type		0	1	1
24	Encoder Dir		0	1	0
25	Motion Dir		0	1	0
26	Servo Alarmreset Logic		0	1	0
27	Servo On Output Logic		0	1	0
28	Servo Alarm Logic		0	1	1
29	Servo Inposition Logic		0	1	0

For details of each parameter, refer to [User's Manual - Chapter 9. Parameters].

1 - 2 - 3 . Bit setup of Output pin

This displays the detailed description for 0x20 Frame type.

This command applies only to 1 signal of "User Output1" among the 24 signal types of control output. The remaining output signals can not be operated by the user, and the output signal is output when the corresponding situation occurs during drive operation.

The following table shows the bit mask values for each signal.

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

【Example 1】 Sending data to turn ON the User Output 0 port.

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

【Example 2】 Sending data to turn OFF the User Output 0 port

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

1 - 2 - 4 . Bit setup of Input pin

This displays the detailed description for 0x21 Frame type.

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

Signal Name	Relevant Bit Position	Signal Name	Relevant Bit Position	Signal Name	Relevant Bit Position	Signal Name	Relevant Bit Position
Limit+	0x00000001	reserved	0x00000100	Alarm Reset	0x00010000	reserved	0x01000000
Limit-	0x00000002	reserved	0x00000200	ServoON	0x00020000	reserved	0x02000000
Origin	0x00000004	reserved	0x00000400	Pause	0x00040000	User IN 0	0x04000000
Clear Position	0x00000008	reserved	0x00000800	Org Search	0x00080000	reserved	0x08000000
reserved	0x00000010	reserved	0x00001000	reserved	0x00100000	reserved	0x10000000
reserved	0x00000020	Stop	0x00002000	E-stop	0x00200000	reserved	0x20000000
reserved	0x00000040	Jog+	0x00004000	reserved	0x00400000	reserved	0x40000000
reserved	0x00000080	Jog-	0x00008000	reserved	0x00800000	reserved	0x80000000

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

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

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

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

1 - 2 - 5 . Bit setup of Status Flag

Refer to 'MOTION_EziSERVO2_DEFINE.h' of include file

Name of Flag Define	Description	Relevant Bit Position
FFLAG_ERRORALL	One or more error occurs.	0X00000001
FFLAG_HWPOSILMT	'+' direction limit sensor turns ON.	0X00000002
FFLAG_HWNEGALMT	'-' direction limit sensor turns ON.	0X00000004
FFLAG_SWPOGILMT	'+' direction program limit is exceeded.	0X00000008
FFLAG_SWNEGALMT	'-' direction program limit is exceeded.	0X00000010
Reserved1		0X00000020
Reserved2		0X00000040
Reserved3		0X00000080
Reserved4		0X00000100
Reserved5		0X00000200
Reserved6		0X00000400
Reserved7		0X00000800
Reserved8		0X00001000
Reserved9		0X00002000
Reserved10		0X00004000
Reserved11		0X00008000
FFLAG_EMGSTOP	The motor is under emergency stop.	0X00010000
FFLAG_SLOWSTOP	The motor is under general stop.	0X00020000
FFLAG_ORIGINRETURNING	The motor is returning to the origin.	0X00040000
FFLAG_INPOSITION	Inposition has been finished.	0X00080000
FFLAG_SERVOON	The motor is under Servo ON.	0X00100000
FFLAG_ALARMRESET	AlarmReset has run.	0X00200000
FFLAG_PTSTOPED	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
FFLAG_MOTIONPAUSE	The motor in running is stopped by Pause command.	0X10000000
FFLAG_MOTIONACCEL	The motor is operating to the acceleration section.	0X20000000
FFLAG_MOTIONDECEL	The motor is operating to the deceleration section.	0X40000000
FFLAG_MOTIONCONST	The motor is operating to the normal speed, not acceleration / deceleration sections.	0X80000000

1 - 2 - 6 . Position Table Item

Refer to 'motion_define.h' of include files.

Name	Name of Structure Parameter	Number of Bytes	Offset value	Unit	Low Limit	Upper Limit
Position	lPosition	4 (signed)	0	[pulse]	-134,217,728	+134,217,727
Low Speed	dwStartSpd	4 (unsigned)	4	[pps]	0	500,000
High Speed	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	10
Wait time	wWaitTime	2 (unsigned)	18	[msec]	0	600,000
Continuous Action	wContinuous	2 (unsigned)	20		0	1
Jump Table No.	wBranch	2 (unsigned)	22		0 10,000	255 10,255
Jump PT 0	wCond_branch0	2 (unsigned)	24		0 10,000	255 10,255
Jump PT 1	wCond_branch1	2 (unsigned)	26		0 10,000	255 10,255
Jump PT 2	wCond_branch2	2 (unsigned)	28		0 10,000	255 10,255
Loop Count	wLoopCount	2 (unsigned)	30		0	100
Loop Jump Table No.	wBranchAfterLoop	2 (unsigned)	32		0 10,000	255 10,255
PT set	wPTSet	2 (unsigned)	34		0	15
Loop Counter Clear	wLoopCountCLR	2 (unsigned)	36		0	255
Check Inposition	bCheckInpos	2 (unsigned)	38		0	1
Compare Position	lTriggerPos	4 (signed)	40	[pulse]	-134,217,728	+134,217,727
Compare Width	wTriggerOnTime	2 (unsigned)	44	[msec]	1	9,999
Push Ratio	wPushRatio	2 (unsigned)	46	[%]	20	90
Push Speed	dwPushSpeed	4 (unsigned)	48	[pps]	0	33,333
Push Position	lPushPosition	4 (signed)	52	[pulse]	-134,217,728	+134,217,727
Push Mode	wPushMode	2 (unsigned)	56		0	10,000
Blank		6 (unsigned)	58	0x00		

For details on how to set each item, refer to the separate manual [Position Table of User's Manual]

In Ezi-MOTIONLINK Plus-E, some functions are limited in the position table function.

(Jump PT0 ~ 2, Loop Jump Table No, Loop Counter Clear, Push related functions are not available.)

1 - 3 . Program Method

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

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

Library that serviced together with Ezi-SERVOⅡ Plus-E have to be used. Refer to 「2. Library for PC Program」

The second method is to send the command character directly without using the library function. Protocol It is necessary to create a low level protocol program like a test program and it is mainly used when a PLC is used as a host controller.

2 . Library for PC Program

2 - 1 . Library configuration

To use this library, C++ header file(*.h) and library file(*.lib or *.dll) are required. These files are included in “₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include₩₩ The following contents should be included in a source file for development.

#include “₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include₩₩FAS_EziMotionPlusE.h”
#include “₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include₩₩ReturnCodes_Define.h”
#include “₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include₩₩MOTION_DEFINE.h”
#include “₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include₩₩COMM_Define.h”

Also, library files are as follows:
“₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include₩₩EziMotionPlusE.lib”
“₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include₩₩EziMotionPlusE.dll”

A sample program source of using library is included in a
“₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩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 usinf of protocol programming, this service not provided.

Div.	Name	Return Value	Description
Normal	FMM_OK	0(0x00)	The function has normally performed the command.
Input Error	FMM_INVALID_SLAVE_NUM	3(0x03)	Wrong slave number is inputted.
Operation Error	FMM_POSTABLE_ERROR	9(0x09)	An error occurs while the motor accesses to the position table.
Connection Error	FMC_DISCONNECTED	5(0x05)	The relevant drive is disconnected.
	FMC_TIMEOUT_ERROR	6(0x06)	Response delay(100 msec) occurs.
	FMC_RECVPACKET_ERROR	8(0x08)	Protocol level error occurs in packet that comes from Ezi-MOTIONLINK

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

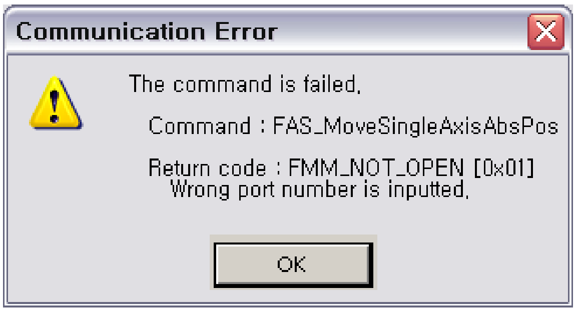
Div.	Name	Return Value	Description
Norma l	FMP_OK	0(0x00)	Communication has been normally performed.
Input Error	FMP_FRAMETYPEERROR	128(0x80)	The drive cannot recognize the command.
	FMP_DATAERROR	129(0x81)	Input data is out of the range.
Operat ion Error	FMP_RUNFAIL	133(0x85)	The motor is already running or not prepared for running. Other wrong motion command.
	FMP_RESETFAIL	134(0x86)	The user cannot execute AlarmReset command while the servo is ON.
	FMP_SERVOONFAIL1	135(0x87)	An alarm has occurred.
	FMP_SERVOONFAIL2	136(0x88)	The motor is under Emergency Stop.
	FMP_SERVOONFAIL3	137(0x89)	'ServoON'signal is already assigned to input pin.
Conne ction Error	FMP_PACKETERROR	130(0x82)	Protocol level error occurs in packet that Ezi-MOTIINLINK received.

2 - 2 . Commmication status window

Above communication status is divided by 3 groups.

.

(1) Communication Error

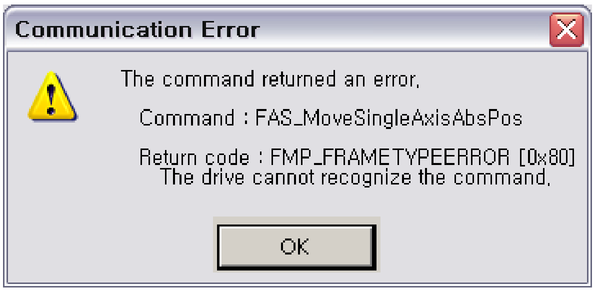

Communication Error

The command is failed.

Command : FAS_MoveSingleAxisAbsPos

Return code : FMM_NOT_OPEN [0x01]
Wrong port number is inputted.

FMM_NOT_OPEN,


Communication Error

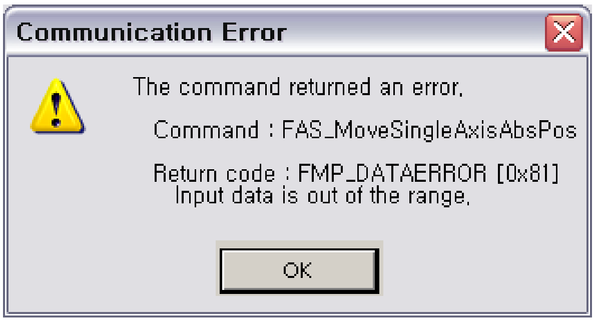
The command returned an error.

Command : FAS_MoveSingleAxisAbsPos

Return code : FMP_FRAMETYPEERROR [0x80]
The drive cannot recognize the command.

FMP_FRAMETYPEERROR = 0x80,

Command not recognized or not supported.


Communication Error

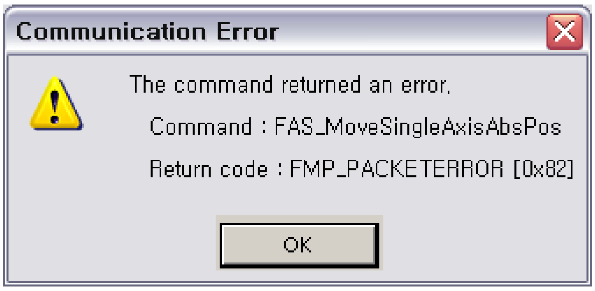
The command returned an error.

Command : FAS_MoveSingleAxisAbsPos

Return code : FMP_DATAERROR [0x81]
Input data is out of the range.

FMP_DATAERROR,

The range of data entered is out of the range supported by Ezi-MOTIONLINK-PE.


Communication Error

The command returned an error.

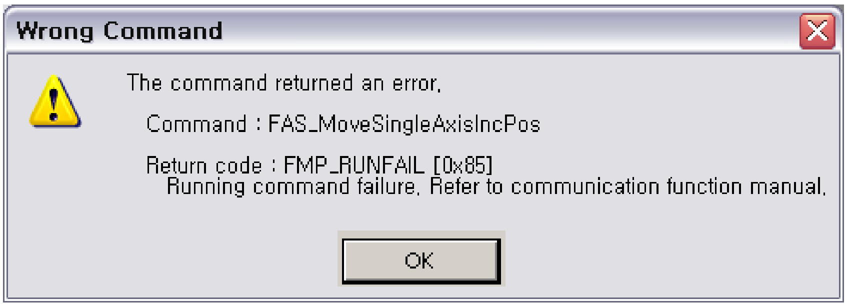
Command : FAS_MoveSingleAxisAbsPos

Return code : FMP_PACKETERROR [0x82]

FMP_PACKETERROR,

The received frame is data that does not conform to the standard. (Packet length sent to zi-MOTIONLINK-PE does not match.)

(2) Wrong Command


Wrong Command

The command returned an error.

Command : FAS_MoveSingleAxisIncPos

Return code : FMP_RUNFAIL [0x85]
Running command failure. Refer to communication function manual.

FMP_RUNFAIL

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

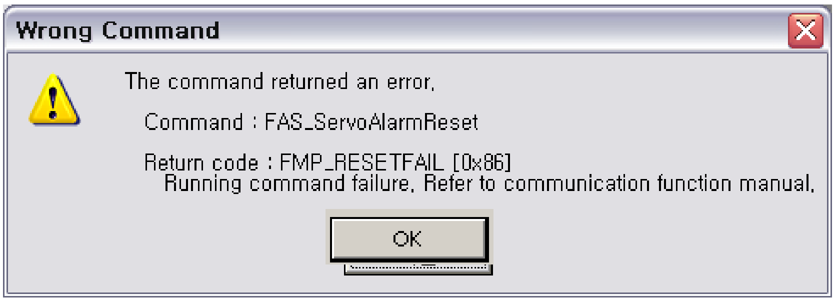
-. The motor is already running

-. The motor is under stop command

-. Servo OFF status

-. Try to Z-pulse Origin without external encoder

-. Other wrong motion command


Wrong Command

The command returned an error.

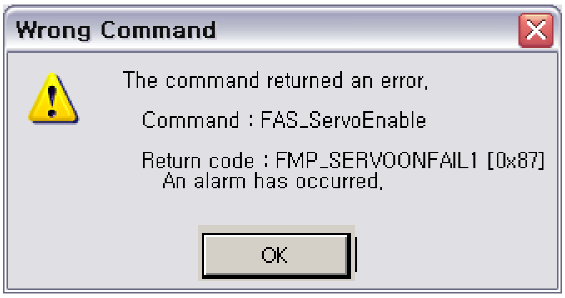
Command : FAS_ServoAlarmReset

Return code : FMP_RESETFAIL [0x86]
Running command failure. Refer to communication function manual.

FMP_RESETFAIL,

It is attempted to perform a reset under the following conditions:

- Servo ON status
- Reset state by external input signal



Wrong Command

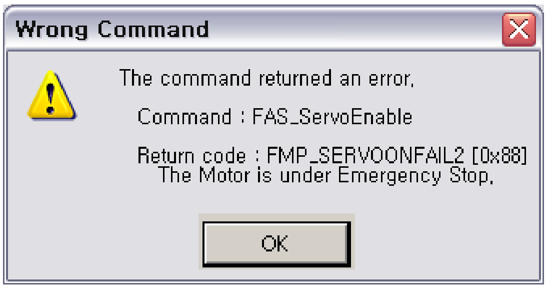
The command returned an error.

Command : FAS_ServoEnable

Return code : FMP_SERVOONFAIL1 [0x87]
An alarm has occurred.

FMP_SERVOONFAIL1,

Wrong 'Servo ON' command during Alarm happens.



Wrong Command

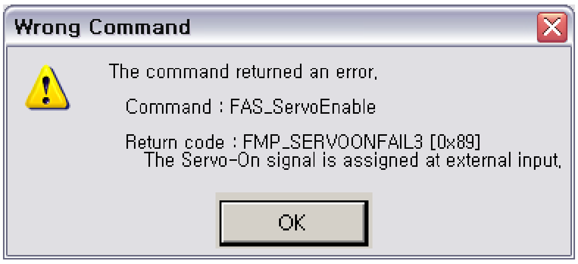
The command returned an error.

Command : FAS_ServoEnable

Return code : FMP_SERVOONFAIL2 [0x88]
The Motor is under Emergency Stop.

FMP_SERVOONFAIL2,

Wrong 'Servo ON' command during E-Stop operation



Wrong Command

The command returned an error.

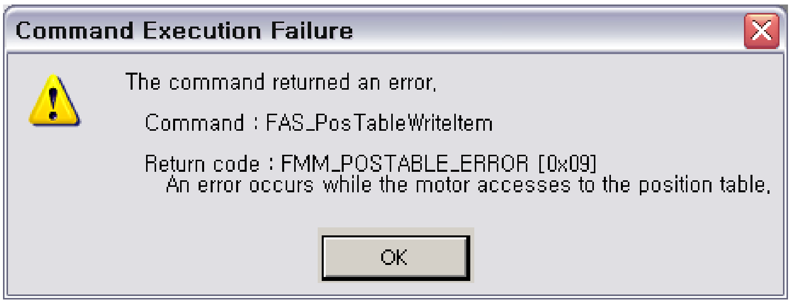
Command : FAS_ServoEnable

Return code : FMP_SERVOONFAIL3 [0x89]
The Servo-On signal is assigned at external input.

FMP_SERVOONFAIL3,

Servo ON signal can not be executed because it is assigned to external input.

(3) Command Execution Error



Command Execution Failure

The command returned an error.

Command : FAS_PosTableWriteItem

Return code : FMM_POSTABLE_ERROR [0x09]
An error occurs while the motor accesses to the position table.

FMM_POSTABLE_ERROR

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

2 - 3 . Drive Link Function

Function Name	Description
FAS_Connect	The drive tries to connect communication with the drive module**:** When it is successfully connected, TRUE will return. Otherwise, FALSE will return.Try to connect
FAS_Close	The drive tries to disconnect communication with the drive module.
FAS_GetboardInfo	The drive reads drive type and program version: Drive type and version information will return.
FAS_IsboardExist	The drive checks whether there is the relevant drive: When it exists, TRUE will return. Otherwise, FALSE will return.
FAS_EnableLog	To select the communication error log function ON/OFF : When it exists, TRUE will return. Otherwise, FALSE will return.
FAS_SetLogPath	To set the saved folder name of error log file : When folder exists, TRUE will return. Otherwise, FALSE will return.

FAS_Connect

FAS_Connect is the function of connecting to Ezi-MOTIONLINK Plus-E.

Syntax

```
BOOL FAS_Connect(
      BYTE sb1, BYTE sb2, BYTE sb3, BYTE sb4
      BYTE iBdID
);
```

Parameters

sb1~4

Enter the IP address of the drive you want to connect to.

ex) 192.168.0.2

sb1 = 192, sb2 = 168, sb3=0, sb4=2

iBdID

Unique ID of board to connect . The ID(value) set by the user.

You can not use the same ID as an IP address.

Return Value

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

Remarks

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcInit()
{
        BYTE sb1 = 192, sb2 = 168, sb3=0, sb4=2 // IP :192.168.0.2
        BYTE iBdID = 0 // The board number of 192.168.0.2
        char lpBuff[256];
        int nBuffSize = 256;
        BYTE nType;
        int nRtn;

        // Try to connectTry to connect
        if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
        {
                // Connection failed.Connection fail
                MessageBox(_T("connect fail!"));
                return;
```

```
    }

    if (FAS_IsSlaveExist(iBdID) == FALSE)
    {
        // There is no relevant board number..
        // Check the board number of Ezi-MOTIONLINK.
        return;
    }

    nRtn = FAS_GetSlaveInfo(iBdID, &nType, lpBuff, nBuffSize);
    if (nRtn != FMM_OK)
    {
        // Command has not been performed properly.
        // Refer to ReturnCodes_Define.h.
    }

    printf("Port : %d (board %d) ₩n", iBdID);
    printf("₩tType : %d ₩n", nType);
    printf("₩tVersion : %d ₩n", lpBuff);

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

See Also

FAS_Close

FAS_Close

To disconnect the serial port being used

Syntax

```
void FAS_Close(
      BYTE iBdID
);
```

Parameters

iBdID

ID number to disconnect.

Remarks

Example

Refer to 'FAS_Connect' library..

See Also

FAS_Connect

FAS_GetSlaveInfo

To get the version information string of the relevant drive

Syntax

```
int FAS_GetboardInfo(
      BYTE iBdID,
      BYTE pType,
      LPSTR lpBuff,
      int nBuffSize
);
```

Parameters

iBdID

The ID number of the board. iBdID set by FAS_Connect function.

pType

Relevant board 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 board has not been connected yet.The board has not been connected yet

FMM_INVALID_SLAVE_NUM : The board of iBdID does not exist.

Remarks

Example

Refer to 'FAS_Connect' library.

See Also

FAS_IsSlaveExist

To check that the drive is connected

Syntax

```
BOOL FAS_IsSlaveExist(
      BYTE iBdID
);
```

Parameters

iBdID

The ID number of the board.   IBdID set by FAS_Connect function .

Return Value

TRUE : The drive is connected.

FALSE : The drive is disconnected.

Remarks

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

Example

Refer to 'FAS_Connect' library.

See Also

FAS_Connect

FAS_EnableLog

To Controls output of communication error related log file.

Syntax

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

Parameters

bEnable

Select output of Log.

Remarks

Select the Log output during Ezi-MOITON Plus-R DLL function used. This setup do not effect th other process or other program.

Log function start from ‘FAS_Connect’ function, the Log output is end when the ‘FAS_Close’ is excuted. The default setting for the Log output is TRUE.

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcDisableLog()
{

        FAS_EnableLog(FALSE);

        // Logs of functions are not output after this.

        BYTE sb1 = 192, sb2 = 168, sb3=0, sb4=2 // IP :192.168.0.2
        BYTE iBdID = 0 // The board number of 192.168.0.2

        // Try to connect.
        if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
        {
                // Connection fail.
                return;
        }

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

See Also

FAS_SetLogPath

FAS_SetLogPath

To setup the folder path of Log output files.

Syntax

BOOL FAS_SetLogPath(LPCTSTR lpPath);

Parameters

lpPath

Folder path Character string of Log output file.

Return Value

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

Remarks

This function have to be called before FAS_Connect library.

If the lpPath value is NULL or the length is 0, the Log path is selected to Ezi-MOTION Plus-E Library folder. The default value for Log path is NULL that the current library and program exist folder.

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcEnableLog()
{
        BYTE sb1 = 192, sb2 = 168, sb3=0, sb4=2 // IP :192.168.0.2
        BYTE iBdID = 0 // The board number of 192.168.0.2

        // Log output.
        FAS_EnableLog(TRUE);   // You do not need to use it.


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

        // Logs of all functions are displayed in the C: ₩ Logs folder.
        // Try to connect.
        if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
        {
                // Connection fail.

                return;
        }

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

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.

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 iBdID
);
```

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function.

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

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

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcModifyParameter()
{
        BYTE sb1 = 192, sb2 = 168, sb3=0, sb4=2 // IP :192.168.0.2
        BYTE iBdID = 0;   //  Unique number of The number of Board

        long lParamVal;
        int nRtn;

        // Try to connect.
        if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
        {
                // Connection fail.

                return;
        }

        // Check Axis Start Speed Parameter.
        nRtn = FAS_GetParameter(iBdID, SERVO_AXISSTARTSPEED, &lParamVal);
        if (nRtn != FMM_OK)
        {
                // The command was not executed normally.
                // Refer to ‘ReturnCodes_Define.h’.
                _ASSERT(FALSE);
        }
        else
        {
                // Parameter value stored in Ezi-SERVOⅡ.
                printf("Parameter [before] : Start Speed = %d ₩n", lParamVal);
        }


        // Change the Start Speed Parameter value to 200 and read the value again.
        nRtn = FAS_SetParameter(iBdID, SERVO_AXISSTARTSPEED, 200);
        _ASSERT(nRtn == FMM_OK);        // If the command was not executed
normally, it stops.

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


        // CHECK THE VALUE STORED IN ROM.
        nRtn = FAS_GetROMParameter(iBdID, SERVO_AXISSTARTSPEED, &lParamVal);
        _ASSERT(nRtn == FMM_OK);        // If the command was not executed
normally, it stops.
        printf("Parameter [ROM] : Start Speed = %d ₩n", lParamVal);


        // Modify the parameter value and save it to ROM.
        nRtn = FAS_SetParameter(iBdID, SERVO_AXISSTARTSPEED, 100);
        _ASSERT(nRtn == FMM_OK);        // If the command was not executed
normally, it stops.

        nRtn = FAS_SaveAllParameters(iBdID);
        _ASSERT(nRtn == FMM_OK);

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

See Also

FAS_GetRomParameter

FAS_SetParameter

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

Syntax

```
int FAS_SetParameter(
      BYTE iBdID,
      BYTE iParamNo,
      long lParamValue
);
```

Parameters

iBdID

The ID number of the board. The iBdID set by the FAS_Connect function.

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 board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

FMM_INVALID_PARAMETER_NUM : The specified parameter of iParam No does not exist.

Remarks

The function operates only for one parameter designated.

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

Example

Refer to ‘FAS_SaveAllParameter’ library.

See Also

FAS_GetParameter

FAS_GetParamater

To call specific parameter values of the board.

Syntax

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

Parameters

iBdID

The ID number of the board. iBdID set by FAS_Connect function.

iParamNo

The number of the parameter to import.

lParamValue

Parameter value.

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

FMM_INVALID_PARAMETER_NUM : The specified parameter of iParamNo does not exist.

Remarks

The function operates only for one parameter designated.

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

Example

Refer to 'FAS_SaveAllParameter' library.

See Also

FAS_SetParameter

FAS_GetROMParameter

To call specific parameter values of the drive

Syntax

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

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

iParamNo

The number of the parameter to import.

lRomParam

Parameter value stored in ROM.

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

FMM_INVALID_PARAMETER_NUM : The specified parameter of iParamNo does not exist.

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_ServoEnable

To turn Servo ON/OFF the drive

Syntax

```
int FAS_ServoEnable(
      BYTE iBdID,
      BOOL bOnOff
);
```

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

bOnOff

Enable or Disable.

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN :   The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

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

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcAxisStatus()
{
        BYTE sb1 = 192, sb2 = 168, sb3=0, sb4=2 // IP :192.168.0.2
        BYTE iBdID = 0;       //   Number of The number of Board
        EZISERVO_AXISSTATUS AxisStatus;
        int nRtn;

        // Try to connect.
        if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
        {
                // Connection fail.
                return;
        }

        nRtn = FAS_GetAxisStatus(iBdID, &(AxisStatus.dwValue));
```

```
		_ASSERT(nRtn == FMM_OK);

		// If SERVO_ON flag turns off, the servo turns on..
		if (AxisStatus.FFLAG_SERVOON == 0)
		{
			nRtn = FAS_ServoEnable(iBdID, 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(iBdID);
			_ASSERT(nRtn == FMM_OK);
		}

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

See Also

FAS_ServoAlarmReset

FAS_ServoAlarmReset

To send AlarmReset command

Syntax

```
int FAS_ServoAlarmReset(
      BYTE iBdID
);
```

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

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 Functiom

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

FAS_SetIOInput

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

Syntax

```
int FAS_SetIOInput(
      BYTE iBdID,
      DWORD dwIOSetMask,
      DWORD dwIOCLRMask
);
```

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

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 board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

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

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcIO()
{
        BYTE sb1 = 192, sb2 = 168, sb3=0, sb4=2 // IP :192.168.0.2
        BYTE iBdID = 0;      //  The number of Board
        DWORD dwInput, dwOutput;
        int nRtn;

        // Try to connect.
        if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
        {
                // Connection fail.

                return;
        }

        // Check I/O input.
        nRtn = FAS_GetIOInput(iBdID, &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(iBdID, SERVO_IN_BITMASK_CLEARPOSITION | SERVO_IN_BITMASK_USERIN1, SERVO_IN_BITMASK_PJOG);
        _ASSERT(nRtn == FMM_OK);

        // Check I/O output.
        nRtn = FAS_GetIOOutput(iBdID, &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(iBdID, 0, SERVO_OUT_BITMASK_USEROUT1 | SERVO_OUT_BITMASK_USEROUT2);
        _ASSERT(nRtn == FMM_OK);

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

See Also

FAS_GetIOInput

FAS_GetIOInput

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

Syntax

```
int FAS_GetIOInput(
      BYTE iBdID,
      DWORD* dwIOInput
);
```

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

dwIOInput

Parameter pointer which input values will be saved.

Return Value

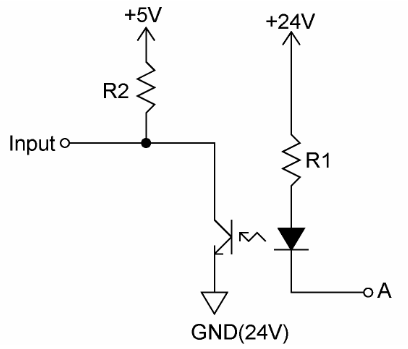
FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

Ezi-MOTIONLINK Plus-E has four inputs, one of which can be selected for customization. This function can read the input port status by 32bit. All of them are insulated by a photocoupler. (Refer to the figure.)



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

Example

Refer to 'FAS_SetIOInput' library..

See Also

FAS_SetIOInput

FAS_SetIOOutput

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

Syntax

```
int FAS_SetIOOutput(
    BYTE iBdID,
    DWORD dwIOSetMask,
    DWORD dwIOCLRMask
);
```

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

dwIOSetMask

Output bitmask value to be set

dwIOCLRMask

Output bitmask value be cleared

Return Value

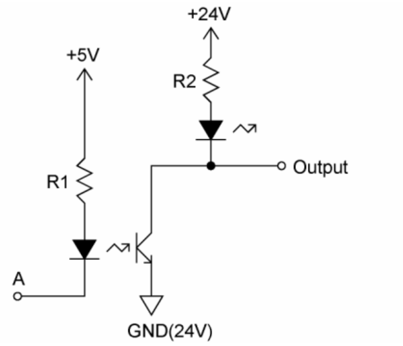
FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

The Ezi-MOTIONLINK Plus-E has two output stages, one of which can be selected for customization.



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

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

Example

Refer to ‘FAS_SetIOInput’ library.

See Also

FAS_GetIOOutput

FAS_GetIOOutput

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

Syntax

```
int FAS_GetIOOutput(
    BYTE iBdID,
    DWORD* dwIOOutput
);
```

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

dwIOInput

Parameter pointer which the output value will be saved...

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

Example

Refer to 'FAS_SetIOInput' library.

See Also

FAS_SetIOOutput

FAS_GetIOAssignMap

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

Syntax

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

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

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 board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not

Remarks

For nIOLogic, refer to 'Motion_define.h'.

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcIOAssign()
{

        BYTE sb1 = 192, sb2 = 168, sb3=0, sb4=2 // IP :192.168.0.2
        BYTE iBdID = 0;      //  The number of Board
        BYTE iPinNo;
        DWORD dwLogicMask;
        BYTE bLevel;
        BYTE i;
        int nRtn;
```

```
    // Try to connect.
    if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
    {
        // Connection fail.
        return;
    }

    // Check assigned information of input pin.
    for (i=0; i</*Input Pin Count*/12; i++)
    {
        nRtn = FAS_GetIOAssignMap(iBdID, 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;         // Values from 0 to 11 are possible (Note: 0 to 2 are fixed).
    nRtn = FAS_SetIOAssignMap(iBdID, 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(iBdID, 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(iBdID, 12/*Input Pin Count*/ + iPinNo,
SERVO_OUT_BITMASK_ALARM, LEVEL_HIGH_ACTIVE);
```

```
ASSERT(nRtn == FMM_OK);

// Connection close.
FAS_Close(iBdID);
```

See Also

FAS_SetIOAssignMap

FAS_SetIOAssignMap

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

Syntax

```
int FAS_SetIOAssignMap(
      BYTE iBdID,
      BYTE iIOPinNo,
      BYTE nLogicNo,
      BYTE bLevel
);
```

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

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 board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

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

Remarks

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

Example

Refer to 'FAS_GSetIOAssignMap' library

See Also

FAS_GetIOAssignMap

FAS_IOAssignMapReadROM

To load the status of CN1 assignment being saved in ROM area

Syntax

```
int FAS_PosTableReadROM(

    BYTE iBdID
);
```

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

FMC_POSTABLE_ERROR : An error occurred while reading Position Table.

Remarks

Example

See Also

FAS_ GetIOAssignMap

2 - 7 . Position Control Function

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

FAS_SetCommandPos

To set the command position value to the motor

Syntax

```
int FAS_SetCommandPos(
    BYTE iBdID,
    long lCmdPos
);
```

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

lCmdPos

Commnad position value to be set.

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

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_ EziMOTIONPlusE.h"

void funcClearPosition()
{

        BYTE sb1 = 192, sb2 = 168, sb3=0, sb4=2 // IP :192.168.0.2
        BYTE iBdID = 0;     //  The number of Board
        int nRtn;

        // Try to connect.
        if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
        {
                // Connection fail.
                return;
        }
        // Initialize Command Position and Actual Position values to 0.
        nRtn = FAS_SetCommandPos(iBdID, 0);
        _ASSERT(nRtn == FMM_OK);
        nRtn = FAS_SetActualPos(iBdID, 0);
        _ASSERT(nRtn == FMM_OK);

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

See Also

FAS_SetActualPos

FAS_SetActualPos

To set the actual position value to the motor

Syntax

```
int FAS_SetActualPos(
      BYTE iBdID,
      long lActPos
);
```

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

lActPos

Actual position value to be set.

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN :   The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

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

Example

Refer to ‘FAS_GetActualPos’ library.

See Also

FAS_SetCommandPos

FAS_GetCommandPos

To read the command position of the current motor

Syntax

```
int FAS_GetCommandPos(
      BYTE iBdID,
      long* lCmdPos
);
```

Parameters

i iBdID

The ID number of the board. IBdID set by FAS_Connect function

lCmdPos

Parameter pointer that command position value will be saved

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN :   The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

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

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcDisplayStatus()
{

        BYTE sb1 = 192, sb2 = 168, sb3=0, sb4=2 // IP :192.168.0.2
        BYTE iBdID = 0;       // The number of Board
        long lValue;
        int nRtn;

        // 연결합니다.
        if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
        {
                // Connection fail.
                return;
        }

        // Check the information on the position of Ezi-MOTIONLINK PlusE.
```

```
		nRtn = FAS_GetCommandPos(iBdID, &lValue);
		_ASSERT(nRtn == FMM_OK);
		printf("CMDPOS : %d ₩n", lValue);
		nRtn = FAS_GetActualPos(iBdID, &lValue);
		_ASSERT(nRtn == FMM_OK);
		printf("ACTPOS : %d ₩n", lValue);
		nRtn = FAS_GetPosError(iBdID, &lValue);
		_ASSERT(nRtn == FMM_OK);
		printf("POSERR : %d ₩n", lValue);
		nRtn = FAS_GetActualVel(iBdID, &lValue);
		_ASSERT(nRtn == FMM_OK);
		printf("ACTVEL : %d ₩n", lValue);

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

See Also

FAS_GetActualPos

FAS_GetActualPos

To read the actual position value of the motor

Syntax

```
int FAS_GetActualPos(
    BYTE iBdID,
    long* lActPos
);
```

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

lActPos

Parameter pointer which the actual position value will be saved.

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN :   The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

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

Example

Refer to 'FAS_GetCOmmandPosition' library.

See Also

FAS_GetCommandPos

FAS_GetPosError

To read the position error of the motor

Syntax

```
int FAS_GetPosError(
      BYTE iBdID,
      long* lPosErr
);
```

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

lPosErr

Parameter pointer which the position error value will be saved..

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

Example

Refer to 'FAS_GetCOmmandPosition' library.

See Also

FAS_GetCommandPos,

FAS_GetActualPos

FAS_GetActualVel

To read the actual velocity of the motor

Syntax

```
int FAS_GetActualVel(
      BYTE iBdID,
      long* lActVel
);
```

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

lActVel

Parameter pointer which the actual velocity value will be saved

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

Example

Refer to 'FAS_GetCOmmandPosition' library

See Also

FAS_ClearPosition

To set the command position value and actual value to ‘0’

Syntax

```
int FAS_ClearPosition(
    BYTE iBdID
);
```

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

The position value is set by the user.

It is mainly used at system initialization.

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcClearPosition()
{

        BYTE sb1 = 192, sb2 = 168, sb3=0, sb4=2 // IP :192.168.0.2
        BYTE iBdID = 0;     //  The number of Board
        int nRtn;

        // Try to connect.
        if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
        {
                // Connection fail.
                return;
        }

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

        // Connection close.
```

```
		FAS_Close(iBdID);
	}
```

See Also

FAS_SetActualPos, FAS_SetCommandPos

2 - 8 . Drive Status Control Function

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

FAS_GetIOAxisStatus

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

Syntax

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

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

dwInStatus

I/O Parameter pointer which input values will be saved.

dwOutStatus

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

dwAxisStatus

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

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

Example

Refer to ‘FAS_MoveSingleAxisAbsPos’ library.

See Also

FAS_GetMotionStatus

To read the motion status of current motor at one time

Syntax

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

Parameters

BdID

The ID number of the board. IBdID set by FAS_Connect function

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 board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_GetAllStatus

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

Syntax

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

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

dwInStatus

I/O Parameter pointer which input values will be saved.

dwOutStatus

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

dwAxisStatus

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

lCmdPos

Parameter pointer which the command position value will be saved

lActPos

Parameter pointer which the actual position value will be saved

lPosErr

Parameter pointer which the position error value will be saved.

lActVel

Parameter pointer which the actual velocity value will be saved

wPosItemNo

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

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist

Remarks

Example

Refer to ‘FAS_MoveSingleAxisAbsPos’ library.

See Also

FAS_GetAxisStatus

FAS_GetMotionStatus

FAS_GetAxisStatus

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

Syntax

```
int FAS_GetAxisStatus(
      BYTE iBdID,
      DWORD* dwAxisStatus
);
```

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

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 board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library..

See Also

2 - 9 . Operation Control Function

Function Name	Description
FAS_MoveStop	The motor in running is decelerate and stopped.
FAS_EmergencyStop	The motor in running stops directly without deceleration
FAS_MoveOriginSingleAxis	The motor starts the origin return.
FAS_MoveSingleAxisAbsPos	The motor moves as much as the given absolute position value.
FAS_MoveSingleAxisIncPos	The motor moves as much as the given incremental position value.
FAS_MoveToLimit	The motor moves up to the position that the limit sensor is detected.
FAS_MoveVelocity	The motor moves to the given velocity and direction: This function is available to Jog motion.
FAS_PositionAbsOverride	While the motor is running, the target absolute position value [pulse] is changed.
FAS_PositionIncOverride	While the motor is running, the target incremental position value [pulse] is changed.
FAS_VelocityOverride	While the motor is running, the running velocity value [pulse] is changed. (**Caution** : 'Changed running speed' must be over 30[pps])
FAS_MoveSingleAxisAbsPosEx	The motor moves as much as the given absolute position value with custom accel/decel time value.
FAS_MoveSingleAxisIncPosEx	The motor moves as much as the given incremental position value with custom accel/decel time value.
FAS_MoveVelocityEx	The motor moves to the given velocity and direction: This function is available to Jog motion with custom accel/decel time value.
FAS_MovePause	The motor starts pause in runing or the motor starts again in pause status.

FAS_MoveStop

To stop the motor.

Syntax

```
int FAS_MoveStop(
      BYTE iBdID,
      );
```

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_EmergencyStop

To stop the motor without deceleration

Syntax

```
int FAS_EmergencyStop(
      BYTE iBdID,
      );
```

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

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

Example

Refer to ‘FAS_MoveSingleAxisAbsPos’ library.

See Also

FAS_MoveOriginSingleAxis

To search the origin of system. For more information, refer to 'User Manual_Text 9.3 Origin Return'

Syntax

```
int FAS_MoveOriginSingleAxis(
      BYTE iBdID,
      );
```

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library..

See Also

FAS_MoveSingleAxisAbsPos

To move the motor to the absolute coordinate

Syntax

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

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

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 board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcMove()
{

        BYTE sb1 = 192, sb2 = 168, sb3=0, sb4=2 // IP :192.168.0.2
        BYTE iBdID = 0;      //   The number of Board
        DWORD dwAxisStatus, dwInput;
        EZISERVO_AXISSTATUS stAxisStatus;
        long lAbsPos, lIncPos, lVelocity;
        int nRtn;

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

```
	{
		// Connection fail.

		return;
	}

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

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

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

	if (dwInput & (MOTIONLINK2_IN_LOGIC_STOP | MOTIONLINK2_IN_LOGIC_PAUSE |
MOTIONLINK2_IN_LOGIC_ESTOP))
		FAS_SetIOInput(iBdID, 0, MOTIONLINK2_IN_LOGIC_STOP |
MOTIONLINK2_IN_LOGIC_PAUSE | MOTIONLINK2_IN_LOGIC_ESTOP);

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

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

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

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

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

		// Stand by until motion command is completely finished.
		do
		{
			Sleep(1);
			nRtn = FAS_GetAxisStatus(iBdID, &dwAxisStatus);
			_ASSERT(nRtn == FMM_OK);
			stAxisStatus.dwValue = dwAxisStatus;
		}
		while (stAxisStatus.FFLAG_MOTIONING);

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

See Also

FAS_MoveSingleAxisIncPos

To move the motor to the incremental coordinate value

Syntax

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

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

lIncPos

Incremental coordinate of position to move

lVelocity

Velocity when the motor movesVelocity when the motor moves

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

Example

Refer to ‘FAS_MoveSingleAxisAbsPos’ library.

See Also

FAS_MoveToLimit

To give the motor a command to search the limit sensor

Syntax

```
int FAS_MoveToLimit(
      BYTE iBdID,
      DWORD lVelocity,
      int iLimitDir,
      );
```

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

lVelocity

Velocioty when the motor movesVelocity when the motor moves

iLimitDir

Limit direction which the motor moves

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

Example

Refer to ‘FAS_MoveSingleAxisAbsPos’ library.

See Also

FAS_MoveVelocity

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

Syntax

```
int FAS_MoveVelocity(
      BYTE iBdID,
      DWORD lVelocity,
      int iVelDir
);
```

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

lVelocity

Velocity when the motor movesVelocity when the motor moves

iVelDir

Direction which the motor moves

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

Example

Refer to ‘FAS_MoveSingleAxisAbsPos’ library.

See Also

FAS_PositionAbsOverride

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

Syntax

```
int FAS_PositionAbsOverride(
      BYTE iBdID,
      long lOverridePos
);
```

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

lOverridePos

Absolute coordinate position value to be changed..

Return Value

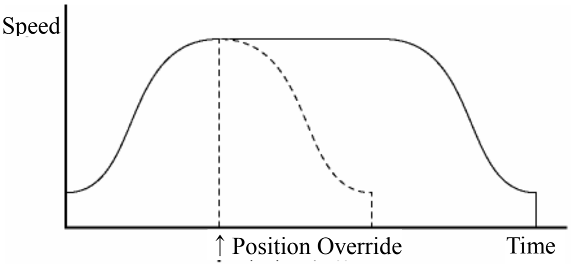
FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The board has not been connected yet.

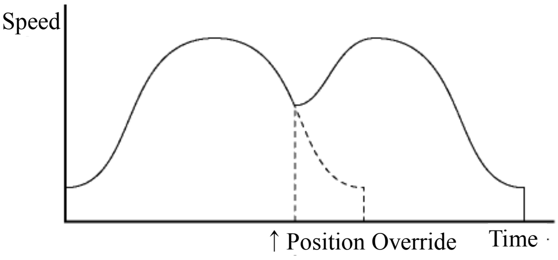
FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

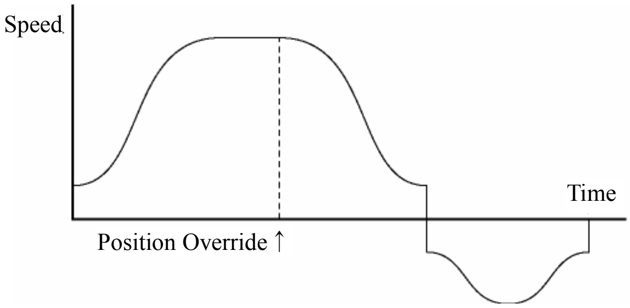
1) If the target position is set to the farther coordinate than the original target position while the motor moves to the accelerated or uniform velocity, the motor moves to the velocity pattern until then and stops the target position.



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



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



Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_PositionIncOverride

FAS_PositionIncOverride

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

Syntax

```
int FAS_PositionIncOverride(
      BYTE iBdID,
      long lOverridePos
);
```

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

lOverridePos

Incremental coordinate position value to be changed

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

1) Refer to 'FAS_PositionAbsOverride' library.

Example

Refer to 'FAS_MoveSingleAxisAbsPos' library.

See Also

FAS_PositionAbsOverride

FAS_VelocityOverride

To change the velocity set while the motor moves.

Syntax

```
int FAS_VelocityOverride(
      BYTE iBdID,
      DWORD lVelocity
);
```

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

lVelocity

Velocity to be changed in [pps].

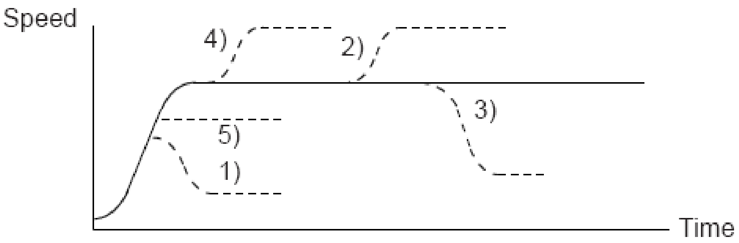
Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks



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

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

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

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

Example

Refer to ‘FAS_MoveSingleAxisAbsPos’ library.

See Also

FAS_MoveSingleAxisAbsPosEx

To moves the motor to a specific absolute coordinate value. (Operation acceleration and deceleration time can be specified)

Syntax

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

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

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 board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

Refer to MOTION_OPTION_EX struct.

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcMoveEx()
{

        BYTE sb1 = 192, sb2 = 168, sb3=0, sb4=2 // IP :192.168.0.2
        BYTE iBdID = 0;      // The number of Board
        DWORD dwAxisStatus, dwInput;
        MOTIONLINK2_AXISSTATUS stAxisStatus;
        long lAbsPos, lIncPos, lVelocity;
        MOTION_OPTION_EX opt = {0};
```

```
int nRtn;

// Try to connect.
if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
{
        // Connection fail.

        return;
}

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

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

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

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

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

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

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

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

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

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

See Also

FAS_MoveSingleAxisIncPosEx

To moves the motor to a specific relative coordinate value. (Operation acceleration and deceleration time can be specified)

Syntax

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

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

lIncPos

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 board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

Example

See Also

FAS_MoveVelocityEx

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

Syntax

```
int FAS_ MoveVelocityEx (
      BYTE iBdID,
      DWORD lVelocity,
      int iVelDir,
      VELOCITY_OPTION_EX* lpExOption
);
```

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

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 board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

Refer to VELOCITY_OPTION_EX struct.

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcMoveVelocityEx()
{

        BYTE sb1 = 192, sb2 = 168, sb3=0, sb4=2 // IP :192.168.0.2
        BYTE iBdID = 0;       // The number of Board
        long lVelocity;
        VELOCITY_OPTION_EX opt = {0};
        int nRtn;
```

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

                        return;
                }

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

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

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

                Sleep(5000);
                FAS_MoveStop(iBdID);
        }
```

See Also

2 - 10 . Position Table Control Function

Function Name	Description
FAS_PosTableReadItem	To read items of RAM area in the specific all items of position table
FAS_PosTableWriteItem	To save specific all items of position table items to RAM area
FAS_PosTableWriteROM	To save all of position table values to ROM area : Total 256 PT values are saved.
FAS_PosTableReadROM	To read position table values in ROM area : Total 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

FAS_PosTableReadItem

To read a specific item in the position table

Syntax

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

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

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 board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

FMM_INVALID_PARAMETER_NUM : wItemNo is out of range

Remarks

Example

```
#include "FAS_EziMOTIONPlusE.h"

void funcPosTable()
{

        BYTE sb1 = 192, sb2 = 168, sb3=0, sb4=2 // IP :192.168.0.2
        BYTE iBdID = 0;      //   The number of Board

        WORD wItemNo;
        ITEM_NODE nodeItem;
        int nRtn;

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

```
                {
                        // Connection fail.

                        return;
                }

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

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

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

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

                // Save currently edited position table data in the ROM.
                nRtn = FAS_PosTableWriteROM(iBdID);
                _ASSERT(nRtn == FMM_OK);

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

See Also

FAS_PosTableWriteItem

FAS_PosTableWriteItem

To edit specific items in the position table

Syntax

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

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

wItemNo

Item number to be edited

lpItem

Item structure pointer to be editedItem number to be edited

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

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

FMM_INVALID_PARAMETER_NUM: wItemNo is out of range

Remarks

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

Example

See Also

FAS_PosTableWriteROM

To save all current position table items to ROM area

Syntax

```
int FAS_PosTableWriteROM(
      BYTE iBdID
);
```

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

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

Remarks

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

Example

See Also

FAS_PosTableReadROM

FAS_PosTableReadROM

To read position table items being saved in ROM area

Syntax

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

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

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

Remarks

Example

See Also

FAS_PosTableWriteROM

FAS_PosTableRunItem

To perform command from a specific item in the position table

Syntax

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

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

wItemNo

Item number to start motion.

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

FMM_INVALID_PARAMETER_NUM : wItemNo is out of range.

Remarks

Example

See Also

FAS_GetAllStatus

FAS_MoveStop

FAS_EmergencyStop

FAS_PosTableReadOneItem

Positin Table 의 특정 Item 의 특정 항목의 값을 읽어옵니다.

Syntax

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

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

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 board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

FMM_INVALID_PARAMETER_NUM : wItemNo is out of range.

Remarks

Example

See Also

FAS_PosTableReadItem

FAS_PosTableWriteOneItem

FAS_PosTableWriteOneItem

To edit one item in the specific position table

Syntax

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

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

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 board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

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

FMM_INVALID_PARAMETER_NUM : wItemNo is out of range.

Remarks

Example

See Also

FAS_PosTableWriteItem

FAS_PosTableReadOneItem

2-11. Other Control Function

Function Name	Description
FAS_TriggerOutput_RunA	A function for generating an output signal at a specific position (periodic position)
FAS_TriggerOutput_Status	A function for generating an output signal at a specific position (periodic position)
FAS_SetTriggerOutputEx	A function for generating an output signal at a specific position (Up to 60 different locations)
FAS_SetTriggerOutputEx	Checking the setting of "FAS_SetTriggerOutputEx" and checking whether it is executed or not

- **After setting FAS_TriggerOutput_RunA and FAS_SetTriggerOutputEx, output signal is generated when moving using operation control function.**

FAS_TriggerOutput_RunA

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

Syntax

```
        int FAS_TriggerOutput_RunA(
        BYTE iBdID,
        BOOL bStartTrigger,
        long lStartPos,
        DWORD dwPeriod,
        DWORD dwPulseTime,
);
```

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

bStartTrigger

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

long lStartPos

Output start position [pulse]

DWORD dwPeriod

Period of output signal [pulse]

DWORD dwPulseTime

Width of output signal [msec]

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

Example

See Also

FAS_TriggerOutput_Status

FAS_ TriggerOutput_Status

To check if the trigger output is working or not.

Syntax

```
        int FAS_TrggerOutput_Status(
        BYTE iBdID,
        BYTE* bTriggerStatus
);
```

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

bTriggerStatus

Current status of signal output.

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN :   The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

Example

See Also

FAS_ TriggerOutput_RunA

FAS_SetTriggerOutputEx

This command is used to generate the output at a specific position on the set output.

Syntax

```
        int FAS_SetTriggerOutPutEx(
        BYTE iBdID,
        BYTE nOutputNo,
        BYTE bRun,
        WORD wOnTime,
        BYTE nTriggerCount,
        long* arrTriggerPosition,
);
```

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

nOutputNo

User Out No.

bRun

Output start / end

wOnTime

Output On Time [msec]

nTriggerCount

Number of output positions (up to 60)

arrTriggerPosition

Output location Array

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

Example

```
#include "FAS_EziMOTIONPlusE.h"

void funcPosTable()
{

        BYTE sb1 = 192, sb2 = 168, sb3=0, sb4=2 // IP :192.168.0.2
        BYTE iBdID = 0;      // The number of Board
        BYTE bRun = 1 ;    // Run
        BYTE nOutputNo =0; // User Out0
        WORD wOnTime = 100; //On time 100[ms]
        BYTE nTriggerCount = 30; //Trigger outpur number : 30

        int nRtn;
        long arrTriggerPosition[60] = {0}; // Trigger Position array initialization

        for (int i = 0; i< nTriggerCount ; i++) // Trigger Position array
        {
            arrTriggerPosition[i] = 10000*I
        }


        // Try to connect
        if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
        {
                // connection failled
                return;
        }

        nRtn = FAS_SetTriggerOuputEx(iBdID, nOutputNo, bRun, wOnTime, nTriggerCount,
                arrTriggerPosition);
        _ASSERT(nRtn == FMM_OK);

                FAS_Close(iBdID);
}
```

See Also

FAS_GetTriggerOutputEx

FAS_GetTriggerOutputEx

FAS_SetTriggerOutputEx 으로 설정된 정보 및 출력 상태를 확인하는 명령입니다

Syntax

```
    int FAS_TriggerOutput_RunA(
    BYTE iBdID,
    BYTE* nOutputNo,
    BYTE* bRun,
    WORD* wOnTime,
    BYTE* nTriggerCount,
    long* arrTriggerPosition,
);
```

Parameters

iBdID

The ID number of the board. IBdID set by FAS_Connect function

nOutputNo

Variable pointer to store user out number

bRun

Variable pointer to store output start / end value

wOnTime

Variable pointer to store the output on time value

nTriggerCount

Variable pointer to store output position count value

arrTriggerPosition

Variable pointer to store output position array value

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The board has not been connected yet.

FMM_INVALID_SLAVE_NUM : The board of the corresponding iBdID does not exist.

Remarks

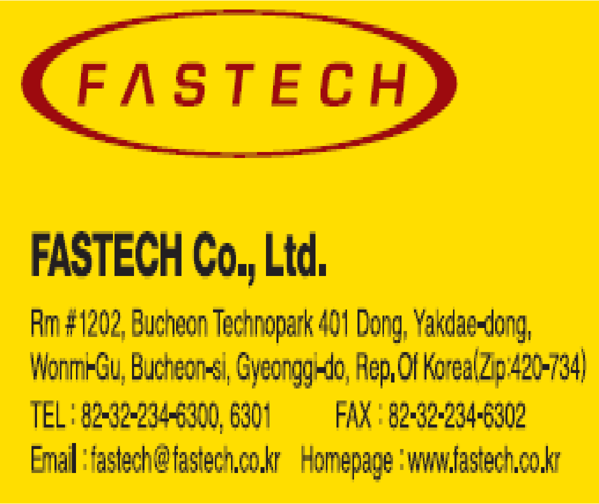
Example

```
#include "FAS_EziMOTIONPlusE.h"

void funcPosTable()
{

        BYTE sb1 = 192, sb2 = 168, sb3=0, sb4=2 // IP :192.168.0.2
        BYTE iBdID = 0;      //  The number of Board
```

```
	BYTE bRun = 0 ;	// Run
	BYTE nOutputNo =0; // User Out
	WORD wOnTime = 0; //On time
	BYTE nTriggerCount = 0; //Trigger ourput munber

	int nRtn;
	long arrTriggerPosition[60] = {0}; // Trigger Position array initialization

	// Try to connect
	if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
	{
		// Connecton failled.
		return;
	}

	nRtn = FAS_GetTriggerOuputEx(iBdID, nOutputNo, &bRun, &wOnTime,
&nTriggerCount,   arrTriggerPosition);
	_ASSERT(nRtn == FMM_OK);

		FAS_Close(iBdID);
}
```

See Also

FAS_SetTriggerOutputEx


FASTECH

FASTECH Co., Ltd.

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

- It is prohibited to unauthorized or reproduced in whole or in part described in the User's Guide
- If you need a user manual to the loss or damage, etc., please contact us or your nearest distributor.
- User manual are subject to change without notice to improve the product or quantitative changes in specifications and user's manual.
- Ezi-MOTIONLINK Plus-E is registered trademark of FASTECH Co., Ltd in the national registration

ⓒ Copyright 2017 FASTECH Co.,Ltd. Jan 31, 2017 Rev.01

www.fastech.co.kr