

# **Ezi-SERVO<sup>®</sup> II** **Plus-E MINI**

## **Closed Loop Stepping System**

### User Manual

### Communication Function

( Rev.02)



## Table of Contents

Table of Contents .....	2
1. Communication Protocol .....	3
1-1. Communication Functions .....	3
1-1-1. Communication Specifications .....	3
1-1-2. Ethernet IP Address .....	3
1-1-3. Ethernet Protocol .....	4
1-1-4. Receive Frame Data Structure .....	4
1-1-5. Transmit Frame Structure and Communication Error .....	4
1-2. Structure of Frame .....	6
1-2-1. Frame type Data Configuration .....	6
1-2-2. Parameter Lists .....	23
1-2-3. Bit Setup of Control Output Pin .....	25
1-2-4. Bit Setup of Control Input Pin .....	26
1-2-5. Bit Setup of Status Flag .....	27
1-2-6. Position Table Item .....	28
1-3. Program Method .....	29
2. Library for PC Program .....	30
2-1. Library Configuration .....	30
2-2. Communication Status Window .....	32
2-3. Drive Connection Function .....	35
2-4. Parameter Control Function .....	52
2-5. Servo Control Function .....	58
2-6. Control I/O Function .....	62
2-7. Position Control Function .....	73
2-8. Drive Status Control Function .....	84
2-9. Operation Control Function .....	90
2-10. Position Table Control Function .....	115
2-11. Ethernet Address Related Function .....	124
2-12. Other Control Function .....	128
3. Appendix – Network Information Setting with DHCP .....	134
3-1. DHCP Function .....	134
3-2. Network Setting Using DHCP (Plus-E series) .....	134

# 1. Communication Protocol

## 1 - 1 . Communication Functions

Ezi-SERVOII Plus-E MINI can control up to 254(1~254) axis by multidrop link at Ethernet.

### 1 - 1 - 1 . Communication Specifications

Item	Specification
Communication Speed	10/100base-T/TX
Communication Type(Protocol)	TCP/ (Port No. : <b>2001,2002</b> ) UDP (Port No. : <b>3001,3002</b> )
Max Cabling Length	Within 100m
Min Cable length between drive	More than 20cm
Number of Connected Axes	254 axes (No. 01~FE)

- Port No. 2001, 3001 : For GUI
- Port No. 2002, 3002 : For User Library
- You cannot use Port No. 2001, 3001 while using User Library file.

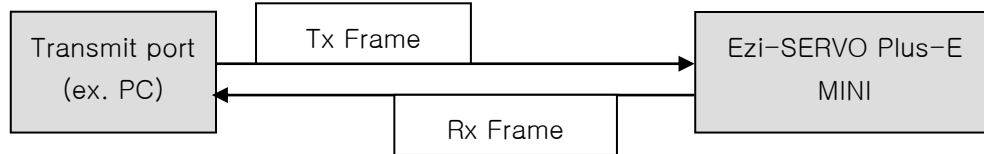
### 1 - 1 - 2 . Ethernet IP Address

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

- When connecting to Ezi-SERVOII Plus-E MINI 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, it cannot be connected.
- If the switch set to 255(FF), IP address is automatically set.  
Because it uses DHCP, IP address is set automatically only when using router.
- When connecting directly from the controller(PC, PLC, etc.) which does not support DHCP, be sure to set the 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 00, 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 : 2bytes
- ② Receiving port number : 2bytes
- ③ Data length : 2bytes, Total length of UDP Header and Frame Data
- ④ Checksum : 2bytes

### 1 - 1 - 4 . Receive Frame Data 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.  
Refer to 「Frametypeand Data configuration」.
- ⑥ Data : The data structure and length of this clause are determined by the Frame type.  
The detailed structure refers to 「Frametypeand Data configuration」 section below.

### 1 - 1 - 5 . Transmit Frame Structure and Communication Error

When any commnad 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	1 byte	0 ~ 252 bytes
					Communication Status	Reply 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 Receive Frame  
(If it does not match the data at the same time of reception, recognize it as an error.)
- ④ Reserved : 1 byte(0x00)
- ⑤ Frame type : Same as Receive Frame  
(If it does not match the data at the time of transmission, recognize it as an error.)
- ⑥ Data : In reply, 1 byte of data indicating communication status(error/normal) is included.  
The simple Execution command has only the communication status data.

The contents of byte indicating communication status are as follows.

Hexa Code	Decimal Code	Description
0x00	0	Communication is normal.
0x80	128	Frame type Error : Receive Frame type command cannot be recognized.
0x81	129	Data error, ROM data read/write error : The received data is out of the specified range.
0x82	130	Response Frame Error : Received Frame is out of this specification.
0x85	133	Running command failure : The user tried to execute new running command in wrong condition as follows. 1) Currently motor is running 2) Currently motor is stopping 3) Servo(Step) is OFF status 4) Try to Z-pulse Origin without external encoder 5) Other wrong motion command
0x86	134	RESET Failure : The user has tried to execute new running in condition as follow. 1) Already RESET status by external input signal
0x87	135	Servo ON fail ① : Try to execute Servo ON during alarm occurring.
0x88	136	Servo ON fail ② : Try to execute Servo ON during emergency stop.
0x89	137	Servo ON fail ③ : 'Servo ON' is set on an external input signal. It is only available to execute Servo ON/OFF by this input signal.
0x90	144	Movement command fail : When the hardware limit was detected, an attempt was made to execute the movement command in the corresponding direction.
0x91	145	Movement command fail : When the software limit was detected, an attempt was made to execute the movement command in the corresponding direction.



**Caution**

- 1) If 'Header' and 'Length' value of response Frame is abnormal, there is no response from the drive.
- 2) If the communication status is displayed to '130', the size of response data is '0' byte.

## 1 - 2 . Structure of Frame

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

(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.

Frame type	Library Name	Description						
0x01 (1)	FAS_ GetboardInfo	<p>Connected slave type and program version information are required.</p> <p>Sending : 0 byte Response : 1~248 bytes</p> <table><tr><td>1 byte</td><td>1 byte</td><td>0~253 bytes</td></tr><tr><td>Communication status</td><td>board type</td><td>ACII string with NULL byte ( strlen() + 1 bytes)</td></tr></table> <p>◆ board type : 95 ( Ezi-SERVOII Plus-E MINI)</p>	1 byte	1 byte	0~253 bytes	Communication status	board type	ACII string with NULL byte ( strlen() + 1 bytes)
1 byte	1 byte	0~253 bytes						
Communication status	board type	ACII string with NULL byte ( strlen() + 1 bytes)						
0x05 (5)	FAS_ GetMotorInfo	<p>Information of motor type connected to board is required.</p> <p>Sending : 0 byte Response : 1~246 bytes</p> <table><tr><td>1 byte</td><td>1 byte</td><td>0~246 bytes</td></tr><tr><td>Communication status</td><td>Motor No. (1~255)</td><td>ACII string with NULL byte ( strlen() + 1 bytes)</td></tr></table>	1 byte	1 byte	0~246 bytes	Communication status	Motor No. (1~255)	ACII string with NULL byte ( strlen() + 1 bytes)
1 byte	1 byte	0~246 bytes						
Communication status	Motor No. (1~255)	ACII string with NULL byte ( strlen() + 1 bytes)						
0x10 (16)	FAS_ SaveAllParameters	<p>Current setting parameters &amp; assign of IO signals are saved in the ROM of the drive. Even though the drive is powered off, saving these must be possible. Values set at 'FAS_SetParameter' &amp; 'FAS_SetIOAssignMap' are saved together.</p> <p>Sending : 0 byte Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table>	1 byte	Communication status				
1 byte								
Communication status								

0x11 (17)	FAS_ GetRomParameter	<p>Specific parameter values in the ROM are read.</p> <p>Sending : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Parameter number (0~32)</td></tr></table> <p>Response : 5 bytes</p> <table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication Status</td><td>Parameter value</td></tr></table> <p>Refer to 「1-2-2. Parameter List」.</p>	1 byte	Parameter number (0~32)	1 byte	4 bytes	Communication Status	Parameter value
1 byte								
Parameter number (0~32)								
1 byte	4 bytes							
Communication Status	Parameter value							
0x12 (18)	FAS_ SetParameter	<p>Specific parameter values are saved to the RAM parameter.</p> <p>Sending : 5 bytes</p> <table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Parameter number (0~32)</td><td>Parameter value</td></tr></table> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication Status</td></tr></table> <p>Refer to 「1-2-2. Parameter List」.</p>	1 byte	4 bytes	Parameter number (0~32)	Parameter value	1 byte	Communication Status
1 byte	4 bytes							
Parameter number (0~32)	Parameter value							
1 byte								
Communication Status								
0x13 (19)	FAS_ GetParameter	<p>Specific parameter values in the RAM are read.</p> <p>Sending : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Parameter number (0~32)</td></tr></table> <p>Response : 5 bytes</p> <table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication Status</td><td>Parameter value</td></tr></table> <p>Refer to 「1-2-2. Parameter List」.</p>	1 byte	Parameter number (0~32)	1 byte	4 bytes	Communication Status	Parameter value
1 byte								
Parameter number (0~32)								
1 byte	4 bytes							
Communication Status	Parameter value							
0x20 (32)	FAS_ SetIOOutput	<p>Output signal level of the control output port is set.</p> <p>Sending : 8 bytes</p> <table><tr><td>4 bytes</td><td>4 bytes</td></tr><tr><td>I/O set mask value</td><td>I/O clear mask value</td></tr></table> <p>When specific bit of the set mask is '1', the relevant output port signal is set to [ON].</p> <p>When specific bit of the clear mask is '1', the relevant output port signal is set to [OFF].</p> <p>For more information, refer to 「1-2-3. Bit setup of Control Output pin」.</p> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication Status</td></tr></table>	4 bytes	4 bytes	I/O set mask value	I/O clear mask value	1 byte	Communication Status
4 bytes	4 bytes							
I/O set mask value	I/O clear mask value							
1 byte								
Communication Status								

0x21 (33)	FAS_ SetIOInput	<p>Input signal level of the control input port is set.</p> <p>Sending : 8 bytes</p> <table><tr><td>4 bytes</td><td>4 bytes</td></tr><tr><td>I/O set mask value</td><td>I/O clear mask value</td></tr></table> <p>When specific bit of the set mask is '1', the relevant input port signal is set to [ON].</p> <p>When specific bit of the clear mask is '1', the relevant input port signal is set to [OFF].</p> <p>For more information, refer to 「1-2-4. Bit setup of Control Input Pin」.</p> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table>	4 bytes	4 bytes	I/O set mask value	I/O clear mask value	1 byte	Communication status
4 bytes	4 bytes							
I/O set mask value	I/O clear mask value							
1 byte								
Communication status								
0x22 (34)	FAS_ GetIOInput	<p>Current input signal status of the control input port is read.</p> <p>Sending : 0 byte</p> <p>Response : 5 bytes</p> <table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication Status</td><td>Input status value</td></tr></table> <p>Relevant bit by each input signal, refer to 「1-2-4. Bit setup of Control Input Pin」.</p>	1 byte	4 bytes	Communication Status	Input status value		
1 byte	4 bytes							
Communication Status	Input status value							
0x23 (35)	FAS_ GetIOOutput	<p>Current output signal status of the control output port is read.</p> <p>Sending : 0 byte</p> <p>Response : 5 bytes</p> <table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication Status</td><td>Output status value</td></tr></table> <p>Relevant bit by each ouput signal, refer to 「1-2-3. Bit setup of Control Output Pin」.</p>	1 byte	4 bytes	Communication Status	Output status value		
1 byte	4 bytes							
Communication Status	Output status value							



0x24 (36)	FAS_ SetIOAssignMap	<p>Allocate control I/O signal to pin of CN1 and set signal level at the same time. To save this setting in ROM memory, execute 'FAS_SaveAllParameters'.</p> <p>Sending : 6 bytes</p> <table><tr><td>1 byte</td><td>4 bytes</td><td>1 byte</td></tr><tr><td>I/O number</td><td>I/O pin masking data</td><td>Setting level</td></tr></table> <p>◆I/O number : '0~5' corresponds to 'Limit+,Limit-,Org,IN1,...,IN3' respectively, and '6,7' corresponds to 'COMP, OUT1' respectively.</p> <p>◆I/O pin masking data : Refer to 「1-2-4. Bit setup of Control Input Pin」.</p> <p>◆Setting level : 0:Active Low, 1:Active High</p> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication Status</td></tr></table>	1 byte	4 bytes	1 byte	I/O number	I/O pin masking data	Setting level	1 byte	Communication Status
1 byte	4 bytes	1 byte								
I/O number	I/O pin masking data	Setting level								
1 byte										
Communication Status										
0x25 (37)	FAS_ GetIOAssignMap	<p>Pin setting status of CN1 port is read.</p> <p>Sending : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>I/O number</td></tr></table> <p>◆I/O number : '0~5' corresponds to 'Limit+, Limit-, Org, IN1, ..., IN3' respectively, and '6,7' correstponds to 'COMP, OUT1' respectively.</p> <p>Response : 6 bytes</p> <table><tr><td>1 byte</td><td>4 bytes</td><td>1 byte</td></tr><tr><td>Communication Status</td><td>IO pin masking status</td><td>Level status</td></tr></table> <p>For more information, refer to '0x24'Frame type.</p>	1 byte	I/O number	1 byte	4 bytes	1 byte	Communication Status	IO pin masking status	Level status
1 byte										
I/O number										
1 byte	4 bytes	1 byte								
Communication Status	IO pin masking status	Level status								
0x26 (38)	FAS_ IOAssignMapReadR OM	<p>Setting status of control I/O signal and level setting value of signal is read from ROM memory area.</p> <p>Sending : 0 byte Response : 2 bytes</p> <table><tr><td>1 byte</td><td>1 byte</td></tr><tr><td>Communication Status</td><td>Command performing status (0: complete, values except 0 : error)</td></tr></table>	1 byte	1 byte	Communication Status	Command performing status (0: complete, values except 0 : error)				
1 byte	1 byte									
Communication Status	Command performing status (0: complete, values except 0 : error)									
0x27 (39)	FAS_ TriggerOutput_Run A	<p>Command for generating control output signal(Compare Out).</p> <p>Sending : 18 bytes</p> <table><tr><td>1 byte</td><td>4 bytes</td><td>4 bytes</td></tr></table>	1 byte	4 bytes	4 bytes					
1 byte	4 bytes	4 bytes								

		<table><tr><td>Output start/stop command (1 : start 0 : stop)</td><td>Output start position [pulse]</td><td>Pulse period [pulse]</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 No. (fix to 0)</td><td>spare</td></tr></table> <p>◆Output start position : First start position value for signal output (-134,217,728 ~134,217,727)</p> <p>◆Pulse period : Setting period of output signal (0 : pulse output only 1 time in pulse start position 1~134,217,727 : pulse output repeatedly depends on setting)</p> <p>◆Pulse width : Setting width of output signal (1 ~1000)</p> <p>Response : 2 bytes</p> <table><tr><td>1 byte</td><td>1 byte</td></tr><tr><td>Communication Status</td><td>Command performing status (0: complete, values except 0: error)</td></tr></table> <p>Enter the value in which the pulse period is calculated by time and the sum of the pulse width is more than 2 [ms]. If it is less than that, it will not work properly.</p>	Output start/stop command (1 : start 0 : stop)	Output start position [pulse]	Pulse period [pulse]	4 bytes	1 byte	4 bytes	Pulse width [msec]	Output pin No. (fix to 0)	spare	1 byte	1 byte	Communication Status	Command performing status (0: complete, values except 0: error)
Output start/stop command (1 : start 0 : stop)	Output start position [pulse]	Pulse period [pulse]													
4 bytes	1 byte	4 bytes													
Pulse width [msec]	Output pin No. (fix to 0)	spare													
1 byte	1 byte														
Communication Status	Command performing status (0: complete, values except 0: error)														
0x28 (40)	FAS_ TriggerOutput_ Status	Command to check whether the current signal(Compare Out) output function is working.  Sending : 0 byte Response : 2 bytes <table><tr><td>1 byte</td><td>1 byte</td></tr><tr><td>Communication Status</td><td>Current status (1: output ON, 0 : output OFF)</td></tr></table>	1 byte	1 byte	Communication Status	Current status (1: output ON, 0 : output OFF)									
1 byte	1 byte														
Communication Status	Current status (1: output ON, 0 : output OFF)														
0x7E (126)	FAS_SetTriggerOutputEx	Setting for generating output at a specific position on the set output. (Available after setting the output signal to User Out) Sending : 245 bytes <table><tr><td>1 byte</td><td>1 byte</td><td>2 bytes</td><td>1byte</td></tr><tr><td>User Out No. (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> <p>◆ Number of output position : 1~60</p> <p>◆ Output position Array : Based on 4bytes, 60 arrays. Even if the number of positions is not 60, the output position array has 60 information inputs.</p>	1 byte	1 byte	2 bytes	1byte	User Out No. (0~8)	Output Start/End command (1: Start 0: End)	Output On time (In ms, 1~65,535)	Output position count	240 bytes	Output position Array(4bytes * 60) Location: -134,217,728~134,217,727			
1 byte	1 byte	2 bytes	1byte												
User Out No. (0~8)	Output Start/End command (1: Start 0: End)	Output On time (In ms, 1~65,535)	Output position count												
240 bytes															
Output position Array(4bytes * 60) Location: -134,217,728~134,217,727															

		<p>◆ Output is automatically closed when the maximum number of outputs is reached.</p> <p>◆ To output, execute the move command after setting.</p> <p>The position of the move command should be greater than the last position if the last position of the output position is positive and less than the last position if the position is negative.</p> <p>Normal output may not be generated depending on the starting position (current position), so it is necessary to set to an appropriate value.</p> <p>Normal output may not be generated depending on drive speed and output On time setting, so it is necessary to set to an appropriate value.</p> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication Status</td></tr></table>	1 byte	Communication Status										
1 byte														
Communication Status														
0x7F (127)	FAS_GetTriggerOutputEx	<p>Command checked from which information and output status set as FAS_SetTriggerOutputEx.</p> <p>Sending : 1byte</p> <table><tr><td>1 byte</td></tr><tr><td>User Out No.</td></tr></table> <p>◆ User Out No. : User Out number from which information can be verified(0~8).</p> <p>Response : 245bytes</p> <table><tr><td>1 byte</td><td>1 byte</td><td>2 bytes</td><td>1byte</td></tr><tr><td>Communication Status</td><td>Output status</td><td>Output On time (In ms, 1~65535)</td><td>Number of output locations (1~60)</td></tr></table> <table><tr><td>240 bytes</td></tr><tr><td>Ouput location Array(4bytes * 60) Location: -134,217,728~134,217,727</td></tr></table> <p>◆ Ouput status : Run/Stop status of corresponding User out number</p> <p>0 : Stop</p> <p>2 : Run</p>	1 byte	User Out No.	1 byte	1 byte	2 bytes	1byte	Communication Status	Output status	Output On time (In ms, 1~65535)	Number of output locations (1~60)	240 bytes	Ouput location Array(4bytes * 60) Location: -134,217,728~134,217,727
1 byte														
User Out No.														
1 byte	1 byte	2 bytes	1byte											
Communication Status	Output status	Output On time (In ms, 1~65535)	Number of output locations (1~60)											
240 bytes														
Ouput location Array(4bytes * 60) Location: -134,217,728~134,217,727														
0x2A (42)	FAS_ServoEnable	<p>Servo ON/OFF status is set.</p> <p>Sending : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>0:OFF, 1:ON</td></tr></table> <p>Response : 1 byte</p>	1 byte	0:OFF, 1:ON										
1 byte														
0:OFF, 1:ON														

		<table><tr><td>1 byte</td></tr><tr><td>Communication Status</td></tr></table>	1 byte	Communication Status		
1 byte						
Communication Status						
0x2B (43)	FAS_ ServoAlarmReset	<p>To reset Alarm status.</p> <p>Sending : 0 byte</p> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication Status</td></tr></table>	1 byte	Communication Status		
1 byte						
Communication Status						
0x2E (46)	FAS_ GetAlarmType	<p>To request current Alarm status and information.</p> <p>Sending : 0 byte</p> <p>Response : 2 bytes</p> <table><tr><td>1 byte</td><td>1 byte</td></tr><tr><td>Communication Status</td><td>Alarm status (0: No alarm, values except 0: Alarm No.)</td></tr></table> <p>◆ Alarm type: No alarm (0)   OverCurrent(1)   OverSpeed(2)   </p>	1 byte	1 byte	Communication Status	Alarm status (0: No alarm, values except 0: Alarm No.)
1 byte	1 byte					
Communication Status	Alarm status (0: No alarm, values except 0: Alarm No.)					

0x33 (51)	FAS_ MoveOriginSingle Axis	<p>To request the motor to return to the origin at the current setting parameter condition.</p> <p>Sending : 0 byte Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication Status</td></tr></table>	1 byte	Communication Status				
1 byte								
Communication Status								
0x34 (52)	FAS_ MoveSingleAxisAbs Pos	<p>To request the motor to move its position as much as the absolute value[pulse].</p> <p>Sending : 8 bytes</p> <table><tr><td>4 bytes</td><td>4 bytes</td></tr><tr><td>Absolute position value</td><td>Running speed[pps]</td></tr></table> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication Status</td></tr></table>	4 bytes	4 bytes	Absolute position value	Running speed[pps]	1 byte	Communication Status
4 bytes	4 bytes							
Absolute position value	Running speed[pps]							
1 byte								
Communication Status								
0x35 (53)	FAS_ MoveSingle AxisIncPos	<p>To request the motor to move its position as much as the incremental value[pulse].</p> <p>Sending : 8 bytes</p> <table><tr><td>4 bytes</td><td>4 bytes</td></tr><tr><td>Incremental position value</td><td>Running speed[pps]</td></tr></table> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication Status</td></tr></table>	4 bytes	4 bytes	Incremental position value	Running speed[pps]	1 byte	Communication Status
4 bytes	4 bytes							
Incremental position value	Running speed[pps]							
1 byte								
Communication Status								
0x36 (54)	FAS_ MoveToLimit	<p>To request the motor to start Limit motion at the current setting parameter condition.</p> <p>Sending : 5 bytes</p> <table><tr><td>4 bytes</td><td>1 byte</td></tr><tr><td>Running speed[pps]</td><td>Running direction ( 0:-Limit 1:+Limit)</td></tr></table> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication Status</td></tr></table>	4 bytes	1 byte	Running speed[pps]	Running direction ( 0:-Limit 1:+Limit)	1 byte	Communication Status
4 bytes	1 byte							
Running speed[pps]	Running direction ( 0:-Limit 1:+Limit)							
1 byte								
Communication Status								

0x37 (55)	FAS_ MoveVelocity	<p>To request the motor to start Jog motion at the current setting parameter condition.</p> <p>Sending : 5 bytes</p> <table><tr><td>4 bytes</td><td>1 byte</td></tr><tr><td>Running speed[pps]</td><td>Running direction ( 0:-Jog 1:+Jog)</td></tr></table> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication Status</td></tr></table>	4 bytes	1 byte	Running speed[pps]	Running direction ( 0:-Jog 1:+Jog)	1 byte	Communication Status
4 bytes	1 byte							
Running speed[pps]	Running direction ( 0:-Jog 1:+Jog)							
1 byte								
Communication Status								
0x38 (56)	FAS_ PositionAbsOverride	<p>To request the motor to change the target absolute position value[pulse] while it is in running.</p> <p>Sending : 4 bytes</p> <table><tr><td>4 bytes</td></tr><tr><td>Changed command position value[pulse]</td></tr></table> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication Status</td></tr></table> <p>◆ Only at constant speed</p>	4 bytes	Changed command position value[pulse]	1 byte	Communication Status		
4 bytes								
Changed command position value[pulse]								
1 byte								
Communication Status								
0x39 (57)	FAS_ PositionIncOverride	<p>To request the motor to change the target incremental position value[pulse] while it is in running.</p> <p>Sending : 4 bytes</p> <table><tr><td>4 bytes</td></tr><tr><td>Changed command position value [pulse]</td></tr></table> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication Status</td></tr></table> <p>◆ Only at constant speed</p>	4 bytes	Changed command position value [pulse]	1 byte	Communication Status		
4 bytes								
Changed command position value [pulse]								
1 byte								
Communication Status								

0x3A (58)	FAS_ VelocityOverride	<p>To request the motor to change the running speed value[pps] while it is in running.</p> <p>Sending : 4 bytes</p> <table><tr><td>4 bytes</td></tr><tr><td>Changed running speed[pps]</td></tr></table> <p>The accel/decel time is assigned to 'Axis Acc Time' and 'Axis Dec Time' value in parameter lists.</p> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication Status</td></tr></table> <p>◆ Only at constant speed.</p> <p>◆ Check the speed range that can be changed according to the command speed (movement command speed in stop state) and use the function. Please refer to the table below.</p> <table><tr><th rowspan="2">Command Speed [pps]</th><th colspan="2">Speed Change Range [pps]</th></tr><tr><th>Min</th><th>Max</th></tr><tr><td>1~983</td><td>1</td><td>4914</td></tr><tr><td>984~1638</td><td>1</td><td>8191</td></tr><tr><td>1639~3276</td><td>1</td><td>16383</td></tr><tr><td>3277~6553</td><td>2</td><td>32766</td></tr><tr><td>6554~16384</td><td>5</td><td>81915</td></tr><tr><td>16385~32768</td><td>10</td><td>163830</td></tr><tr><td>32769~65536</td><td>20</td><td>327660</td></tr><tr><td>65537~163840</td><td>50</td><td>819150</td></tr><tr><td>163840~327680</td><td>100</td><td>1638300</td></tr><tr><td>327681~655360</td><td>200</td><td>3276600</td></tr></table> <p>If set to a value not in the speed change range, the speed changes to the minimum or maximum value within the range.</p>	4 bytes	Changed running speed[pps]	1 byte	Communication Status	Command Speed [pps]	Speed Change Range [pps]		Min	Max	1~983	1	4914	984~1638	1	8191	1639~3276	1	16383	3277~6553	2	32766	6554~16384	5	81915	16385~32768	10	163830	32769~65536	20	327660	65537~163840	50	819150	163840~327680	100	1638300	327681~655360	200	3276600
4 bytes																																									
Changed running speed[pps]																																									
1 byte																																									
Communication Status																																									
Command Speed [pps]	Speed Change Range [pps]																																								
	Min	Max																																							
1~983	1	4914																																							
984~1638	1	8191																																							
1639~3276	1	16383																																							
3277~6553	2	32766																																							
6554~16384	5	81915																																							
16385~32768	10	163830																																							
32769~65536	20	327660																																							
65537~163840	50	819150																																							
163840~327680	100	1638300																																							
327681~655360	200	3276600																																							
0x80 (128)	FAS_ MoveSingleAxisAbs PosEx	<p>To request the motor to move its position as much as the absolute value[pulse] with Custom Accel. / Decel. Time[msec].</p> <p>Sending : 40 bytes</p> <table><tr><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td><td>2 bytes</td></tr><tr><td>Absolute position value</td><td>Running speed[pps]</td><td>Flag option</td><td>Custom Accel. Time (1~9999)</td></tr></table>	4 bytes	4 bytes	4 bytes	2 bytes	Absolute position value	Running speed[pps]	Flag option	Custom Accel. Time (1~9999)																															
4 bytes	4 bytes	4 bytes	2 bytes																																						
Absolute position value	Running speed[pps]	Flag option	Custom Accel. Time (1~9999)																																						

		<table><tr><td>2 bytes</td><td>24 bytes</td></tr><tr><td>Custom Decel. Time (1~9999)</td><td>Reserved</td></tr></table> <p>Flag option : 0x0001 : reserved 0x0002 : Custom Accel. Time is used. 0x0004 : Custom Decel. Time is used.</p> <p>If the bit value is OFF status(0), Accel. / Decel. Time value is used that saved in controller.</p> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication Status</td></tr></table>	2 bytes	24 bytes	Custom Decel. Time (1~9999)	Reserved	1 byte	Communication Status								
2 bytes	24 bytes															
Custom Decel. Time (1~9999)	Reserved															
1 byte																
Communication Status																
0x81 (129)	FAS_ MoveSingle AxisIncPosEx	<p>To request the motor to move its position as much as the incremental value[pulse] with Custom Accel. / Decel. Time[msec].</p> <p>Sending : 40 bytes</p> <table><tr><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td><td>2 bytes</td></tr><tr><td>Incremental position value</td><td>Running speed[pps]</td><td>Flag option</td><td>Custom Accel. Time (1~9999)</td></tr></table> <table><tr><td>2 bytes</td><td>24 bytes</td></tr><tr><td>Custom Decel. Time (1~9999)</td><td>Reserved</td></tr></table> <p>Flag option : 0x0001 : reserved 0x0002 : Custom Accel. Time is used. 0x0004 : Custom Decel. Time is used.</p> <p>If the bit value is OFF status(0), Accel. / Decel. Time value is used that saved in controller.</p> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication Status</td></tr></table>	4 bytes	4 bytes	4 bytes	2 bytes	Incremental position value	Running speed[pps]	Flag option	Custom Accel. Time (1~9999)	2 bytes	24 bytes	Custom Decel. Time (1~9999)	Reserved	1 byte	Communication Status
4 bytes	4 bytes	4 bytes	2 bytes													
Incremental position value	Running speed[pps]	Flag option	Custom Accel. Time (1~9999)													
2 bytes	24 bytes															
Custom Decel. Time (1~9999)	Reserved															
1 byte																
Communication Status																



0x82 (130)	FAS_ MoveVelocityEx	<p>To request the motor to start Jog motion with Accel. / Decel. Time value[msec].</p> <p>Sending : 37 bytes</p> <table><tr><td>4 bytes</td><td>1 byte</td><td>4 bytes</td></tr><tr><td>Running speed[pps]</td><td>Running direction ( 0:-Jog 1:+Jog)</td><td>Flag option</td></tr></table> <table><tr><td>2 bytes</td><td>26 bytes</td></tr><tr><td>Custom Accel./Decel. Time (1~9999)</td><td>Reserved</td></tr></table> <p>Flag option : 0x0001 : reserved 0x0002 : Custom Accel./Decel. Time is used.</p> <p>If the bit value is OFF status(0), Accel. / Decel. Time value is used that saved in controller.</p> <p>Response : 1 byte</p>	4 bytes	1 byte	4 bytes	Running speed[pps]	Running direction ( 0:-Jog 1:+Jog)	Flag option	2 bytes	26 bytes	Custom Accel./Decel. Time (1~9999)	Reserved		
4 bytes	1 byte	4 bytes												
Running speed[pps]	Running direction ( 0:-Jog 1:+Jog)	Flag option												
2 bytes	26 bytes													
Custom Accel./Decel. Time (1~9999)	Reserved													
0x40 (64)	FAS_ GetAxisStatus	<p>To request the Flag value of displaying the running status.</p> <p>Sending : 0 byte Response : 5 bytes</p> <table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication Status</td><td>Status Flag value</td></tr></table> <p>For bit related to each Flag, refer to 「1-2-5. Bit Setup of Status Flag」.</p>	1 byte	4 bytes	Communication Status	Status Flag value								
1 byte	4 bytes													
Communication Status	Status Flag value													
0x41 (65)	FAS_ GetIOAxisStatus	<p>To request the I/O status and the running Flag status. (Frame type 0x22, 0x23, 0x40 are packed.)</p> <p>Sending : 0 byte Response : 13 bytes</p> <table><tr><td>1 byte</td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td></tr><tr><td>Communication Status</td><td>Input status value</td><td>Output status value</td><td>Status Flag value</td></tr></table>	1 byte	4 bytes	4 bytes	4 bytes	Communication Status	Input status value	Output status value	Status Flag value				
1 byte	4 bytes	4 bytes	4 bytes											
Communication Status	Input status value	Output status value	Status Flag value											
0x42 (66)	FAS_ GetMotionStatus	<p>To request the current running progress status and its PT number. (Frame type 0x51, 0x53, 0x54, 0x55 are packed.)</p> <p>Sending : 0 byte Response : 21 bytes</p> <table><tr><td>1 byte</td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td></tr><tr><td>Communication Status</td><td>Command Position value</td><td>Actual Position value</td><td>Position Difference value</td><td>Running speed value</td><td>Current running PT number</td></tr></table>	1 byte	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	Communication Status	Command Position value	Actual Position value	Position Difference value	Running speed value	Current running PT number
1 byte	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes									
Communication Status	Command Position value	Actual Position value	Position Difference value	Running speed value	Current running PT number									

0x43 (67)	FAS_ GetAllStatus	<p>To request all data including the current running stauts. (Frame type 0x41, 0x42 are packed.)</p> <p>Sending : 0 byte Response : 33 bytes</p> <table><tr><td>1 byte</td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td></tr><tr><td>Communication Status</td><td>Input status value</td><td>Oouput 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>	1 byte	4 bytes	4 bytes	4 bytes	Communication Status	Input status value	Oouput status value	Status Flag value	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	Command Position value	Actual Position value	Position Difference value	Running speed value	Current running PT number
1 byte	4 bytes	4 bytes	4 bytes																	
Communication Status	Input status value	Oouput status value	Status Flag value																	
4 bytes	4 bytes	4 bytes	4 bytes	4 bytes																
Command Position value	Actual Position value	Position Difference value	Running speed value	Current running PT number																
0x50 (80)	FAS_ SetCommandPos	<p>The user sets it to the command position value before it starts to operate and then can check how the command position value is changed.</p> <p>Sending : 4 bytes</p> <table><tr><td>4 bytes</td></tr><tr><td>Command position setting count value</td></tr></table> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication Status</td></tr></table>	4 bytes	Command position setting count value	1 byte	Communication Status														
4 bytes																				
Command position setting count value																				
1 byte																				
Communication Status																				
0x51 (81)	FAS_ GetCommandPos	<p>To request the command position value[pulse] being tracked.</p> <p>Sending : 0 byte Response : 5 bytes</p> <table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication Status</td><td>Command position value</td></tr></table>	1 byte	4 bytes	Communication Status	Command position value														
1 byte	4 bytes																			
Communication Status	Command position value																			
0x52 (82)	FAS_ SetActualPos	<p>Ezi-SERVOII Plus-E MINI is the closed loop control drive and so the actual position value is continuously updated while the motor is running. To check the change of the actual position value, user can set the actual position value to the desired value.</p> <p>Sending : 4 bytes</p> <table><tr><td>4 bytes</td></tr><tr><td>Actual position count valule</td></tr></table> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication Status</td></tr></table>	4 bytes	Actual position count valule	1 byte	Communication Status														
4 bytes																				
Actual position count valule																				
1 byte																				
Communication Status																				

0x53 (83)	FAS_ GetActualPos	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>	1 byte	4 bytes	Communication Status	Actual position value
1 byte	4 bytes					
Communication Status	Actual position value					
0x54 (84)	FAS_ GetPosError	To request the difference[pulse] between the command position value and the actual position value.  Sending : 0 byte 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.)	1 byte	4 bytes	Communication Status	Position difference value
1 byte	4 bytes					
Communication Status	Position difference value					
0x55 (85)	FAS_ GetActualVel	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>	1 byte	4 bytes	Communication Status	Speed value
1 byte	4 bytes					
Communication Status	Speed value					
0x56 (86)	FAS_ ClearPosition	Set both the command position value and actual position value to '0'. Sending : 0 byte Response : 1 byte <table><tr><td>1 byte</td></tr><tr><td>Communication Status</td></tr></table>	1 byte	Communication Status		
1 byte						
Communication Status						
0x58 (88)	FAS_ MovePause	To pause or to resume 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>	1 byte	0: pause release, 1: pause start	1 byte	Communication Status
1 byte						
0: pause release, 1: pause start						
1 byte						
Communication Status						

0x60 (96)	FAS_ PosTableReadItem	<p>To read PT values in the RAM of the drive.</p> <p>Sending : 2 bytes</p> <table><tr><td>2 bytes</td></tr><tr><td>PT No. (0~255) to read</td></tr></table> <p>Response : 65 bytes</p> <table><tr><td>1 byte</td><td>64 bytes</td></tr><tr><td>Communication Status</td><td>Relevant PT value</td></tr></table> <p>For items by each PT, refer to 「1-2-6. Position Table Item」.</p>	2 bytes	PT No. (0~255) to read	1 byte	64 bytes	Communication Status	Relevant PT value		
2 bytes										
PT No. (0~255) to read										
1 byte	64 bytes									
Communication Status	Relevant PT value									
0x61 (97)	FAS_ PosTableWriteItem	<p>To save PT values to the RAM of the drive.</p> <p>Sending : 66 bytes</p> <table><tr><td>2 bytes</td><td>64 bytes</td></tr><tr><td>PT No. (0~255)</td><td>Relevant PT value</td></tr></table> <p>For items by each PT, refer to 「1-2-6. Position Table Item」.</p> <p>Response : 2 bytes</p> <table><tr><td>1 byte</td><td>1 byte</td></tr><tr><td>Communication Status</td><td>Command performing status (values except 0 : complete, 0: error)</td></tr></table>	2 bytes	64 bytes	PT No. (0~255)	Relevant PT value	1 byte	1 byte	Communication Status	Command performing status (values except 0 : complete, 0: error)
2 bytes	64 bytes									
PT No. (0~255)	Relevant PT value									
1 byte	1 byte									
Communication Status	Command performing status (values except 0 : complete, 0: error)									
0x62 (98)	FAS_ PosTableReadROM	<p>To read all PT values (256) in the ROM of the drive.</p> <p>Sending : 0 byte</p> <p>Response : 2 bytes</p> <table><tr><td>1 byte</td><td>1 byte</td></tr><tr><td>Communication Status</td><td>Command performing status (0: complete, values except 0: error)</td></tr></table>	1 byte	1 byte	Communication Status	Command performing status (0: complete, values except 0: error)				
1 byte	1 byte									
Communication Status	Command performing status (0: complete, values except 0: error)									
0x63 (99)	FAS_ PosTableWriteROM	<p>To save all PT value(256) to the ROM of the drive.</p> <p>Sending : 0 byte</p> <p>Response : 2 bytes</p> <table><tr><td>1 byte</td><td>1 byte</td></tr><tr><td>Communication Status</td><td>Command performing status (0: complete, values except 0: error)</td></tr></table>	1 byte	1 byte	Communication Status	Command performing status (0: complete, values except 0: error)				
1 byte	1 byte									
Communication Status	Command performing status (0: complete, values except 0: error)									

0x64 (100)	FAS_ PosTableRunItem	<p>To start the position table operation from the designated PT number.</p> <p>Sending : 2 bytes</p> <table><tr><td>2 bytes</td></tr><tr><td>PT No. (0~255)</td></tr></table> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication Status</td></tr></table>	2 bytes	PT No. (0~255)	1 byte	Communication Status														
2 bytes																				
PT No. (0~255)																				
1 byte																				
Communication Status																				
0x6A (106)	FAS_ PosTableReadOneItem	<p>To read specific values of PT items in the RAM of the drive.</p> <p>Sending : 4 bytes</p> <table><tr><td>2 bytes</td><td>2 bytes</td></tr><tr><td>PT No. to read (0~255)</td><td>Offset value of the specific item to read (0~40)</td></tr></table> <p>Refer to 「1-2-6. Position Table Item」 for Offset value.</p> <p>Response : 5 bytes</p> <table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>Communication Status</td><td>Value of the relevant item</td></tr></table>	2 bytes	2 bytes	PT No. to read (0~255)	Offset value of the specific item to read (0~40)	1 byte	4 bytes	Communication Status	Value of the relevant item										
2 bytes	2 bytes																			
PT No. to read (0~255)	Offset value of the specific item to read (0~40)																			
1 byte	4 bytes																			
Communication Status	Value of the relevant item																			
0x6B (107)	FAS_ PosTableWriteOneItem	<p>To save specific values of PT items in the RAM of the drive.</p> <p>Sending : 8 bytes</p> <table><tr><td>2 bytes</td><td>2bytes</td><td>4 bytes</td></tr><tr><td>PT No. To save(0~255)</td><td>Offset value of the specific item to save (0~40)</td><td>Stored value</td></tr></table> <p>Refer to 「1-2-6. Position Table Item」 for Offset value.</p> <p>Response : 2 bytes</p> <table><tr><td>1 byte</td><td>1 byte</td></tr><tr><td>Communication Status</td><td>Command performing status (values except 0: complete, 0: error)</td></tr></table>	2 bytes	2bytes	4 bytes	PT No. To save(0~255)	Offset value of the specific item to save (0~40)	Stored value	1 byte	1 byte	Communication Status	Command performing status (values except 0: complete, 0: error)								
2 bytes	2bytes	4 bytes																		
PT No. To save(0~255)	Offset value of the specific item to save (0~40)	Stored value																		
1 byte	1 byte																			
Communication Status	Command performing status (values except 0: complete, 0: error)																			
0x78 (120)	FAS_ MovePush	<p>To request push motion(To maintain specified motor torque.).</p> <p>Sending : 28 bytes</p> <table><tr><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td><td>2 bytes</td><td>2 bytes</td></tr><tr><td>Normal Start speed</td><td>Normal Move speed</td><td>Normal Position</td><td>Accel time</td><td>Decel time</td></tr></table> <table><tr><td>2 bytes</td><td>4 bytes</td><td>4 bytes</td><td>2 bytes</td></tr><tr><td>Push torque ratio</td><td>Push Move speed</td><td>Push Position</td><td>Push mode</td></tr></table>	4 bytes	4 bytes	4 bytes	2 bytes	2 bytes	Normal Start speed	Normal Move speed	Normal Position	Accel time	Decel time	2 bytes	4 bytes	4 bytes	2 bytes	Push torque ratio	Push Move speed	Push Position	Push mode
4 bytes	4 bytes	4 bytes	2 bytes	2 bytes																
Normal Start speed	Normal Move speed	Normal Position	Accel time	Decel time																
2 bytes	4 bytes	4 bytes	2 bytes																	
Push torque ratio	Push Move speed	Push Position	Push mode																	

		<p>Position move start speed : 1~35,000[pps] Position move move speed : 1~500,000[pps] Position move absolute position value : -134,217,728 ~134,217,727 Position move Acc/Dec time : 1~9,999[ms] Push torque ratio : 20~90[%] Push move speed : 1~33,333[pps] (Max 200[rpm]) (Resolution : 10,000) Push absolute position value : -134,217,728 ~134,217,727 Push mode : 0(stop mode), 1~10,000(non-stop mode)</p> <p>For more information, refer to 「<a href="#">User Manual_Text 8-6. Push Motion Function</a>」.</p> <p>Response : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Communication status</td></tr></table> <p>◆ Be sure to execute Stop(E-Stop) command before the next motion command.</p>	1 byte	Communication status		
1 byte						
Communication status						
0x79 (121)	FAS_ GetPushStatus	<p>To request the current push motion status.</p> <p>Sending : 0 byte Response : 1 bytes</p> <table><tr><td>1 byte</td><td>1 byte</td></tr><tr><td>Communication status</td><td><p>Push motion status (0: normal Servo ON 1: push motioning but the work is not detected 2: work is detected and torque is maintained 3: work is not detected within push motion area. In this case, the Stop (E-Stop) command must be executed before the next motion command.</p></td></tr></table>	1 byte	1 byte	Communication status	<p>Push motion status (0: normal Servo ON 1: push motioning but the work is not detected 2: work is detected and torque is maintained 3: work is not detected within push motion area. In this case, the Stop (E-Stop) command must be executed before the next motion command.</p>
1 byte	1 byte					
Communication status	<p>Push motion status (0: normal Servo ON 1: push motioning but the work is not detected 2: work is detected and torque is maintained 3: work is not detected within push motion area. In this case, the Stop (E-Stop) command must be executed before the next motion command.</p>					

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

## 1 - 2 - 2 . Parameter Lists

No.	Name	Unit	Lower	Upper	Default
0	Pulse Per Revolution		0	8	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]	-2,147,483,648	2,147,483,647	2,147,483,647
10	S/W Limit Minus Value	[pulse]	-2,147,483,648	2,147,483,647	-2,147,483,648
11	S/W Limit Stop Method		0	2	2
12	H/W Limit Stop Method		0	1	0
13	Limit Sensor Logic		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]	-2,147,483,648	2,147,483,647	0
20	Org Position Set	[pulse]	-2,147,483,648	2,147,483,647	0
21	Org Sensor Logic		0	1	0
22	Position Loop Gain		0	63	4
23	Inpos Value		0	63	0
24	Pos Tracking Limit	[pulse]	1	134,217,727	2,500
25	Motion Dir		0	1	0
26	Limit Sensor Dir		0	1	0
27	Org Torque Ratio	[%]	20	90	50
28	Pos. Error Overflow Limit	[pulse]	1	134,217,727	2,500
29	Brake Delay Time <sup>*1</sup>	[msec]	10	5,000	200
30	Run Current	*10[%]	5	15	10
31	Boost Current	*50[%]	0	7	0
32	Stop Current	*10[%]	2	10	5
33	Jog EXT FUNC USE		0	1	0
34	Jog Speed1	[pps]	1	500,000	5,000
35	Jog Speed2	[pps]	1	500,000	5,000

36	Jog Speed3	[pps]	1	500,000	5,000
37	Jog Speed4	[pps]	1	500,000	5,000
38	Jog Speed5	[pps]	1	500,000	5,000
39	Jog Speed6	[pps]	1	500,000	5,000
40	Jog Speed7	[pps]	1	500,000	5,000
41	Use Motion Queue		0	1	0
42	Disconnection Option		0	4	0
43	Communication Timeout	msec	100	60,000	100
44	Motion Profile		0	1	0
45	ORG RET OK OFF OPTION		0	3	0

- Parameter No.45 is available from Firmware [ver.6.1.xx.19].



### 1 - 2 - 3 . Bit Setup of Control Output Pin

This displays the detailed description for '0x20' Frame type.

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

The following table shows bit mask values by each signal.

Signal Name	Relevant bit Position	Signal Name	Relevant bit Position	Signal Name	Relevant bit Position
Compare Out	0x00000001	Origin Search OK	0x00000100	User OUT 1	0x00010000
Inposition	0x00000002	ServoReady	0x00000200	User OUT 2	0x00020000
Alarm	0x00000004	reserved	0x00000400	User OUT 3	0x00040000
Moving	0x00000008	reserved	0x00000800	User OUT 4	0x00080000
Acc/Dec	0x00000010	PT Output0	0x00001000	User OUT 5	0x00100000
ACK	0x00000020	PT Output1	0x00002000	User OUT 6	0x00200000
END	0x00000040	PT Output2	0x00004000	User OUT 7	0x00400000
AlarmBlink	0x00000080	User OUT 0	0x00008000	User OUT 8	0x00800000

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

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

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

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

## 1 - 2 - 4 . Bit Setup of Control Input Pin

It is 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 input without actual input signal.

The following table shows bit mask values by each signal.

Signal Name	Relevant bit Position	Signal Name	Relevant bit Position	Signal Name	Relevant bit Position	Signal Name	Relevant bit Position
Limit+	0x00000001	PT A4	0x00000100	Alarm Reset	0x00010000	JPT input2	0x01000000
Limit-	0x00000002	PT A5/ User IN 6/ Jog0	0x00000200	ServoON	0x00020000	JPT Start	0x02000000
Origin	0x00000004	PT A6/ User IN 7/ Jog1	0x00000400	Pause	0x00040000	User IN 0	0x04000000
Clear Position	0x00000008	PT A7/ User IN 8/ Jog2	0x00000800	Org Search	0x00080000	User IN 1	0x08000000
PT A0	0x00000010	PT Start	0x00001000	Teaching	0x00100000	User IN 2	0x10000000
PT A1	0x00000020	Stop	0x00002000	E-stop	0x00200000	User IN 3	0x20000000
PT A2	0x00000040	Jog+	0x00004000	JPT input0	0x00400000	User IN 4	0x40000000
PT A3	0x00000080	Jog-	0x00008000	JPT input1	0x00800000	User IN 5	0x80000000

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

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

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

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



**Do not mix the bit setup of 'PT A5 ~ PT A7' and 'User IN6 ~ IN8', 'Jog0 ~ Jog2' together on your program.**

## 1 - 2 - 5 . Bit Setup of Status Flag

Refer to 'MOTION\_EziSERVO2\_DEFINE.h' among the include files.

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
FFLAG_ERRPOSOVERFLOW	Position error is higher than 'Pos Error Overflow Limit' parameter after position command.	0X00000080
FFLAG_ERROVERCURRENT	Alarm occurs when an overcurrent occurs in the motor drive element.	0X00000100
FFLAG_ERROVERSPEED	Alarm occurs when the motor speed exceeds 3000 [rpm].	0X00000200
FFLAG_ERRPOSTRACKING	Position error is higher than 'Pos Tracking Limit'parameter during position command run.	0X00000400
FFLAG_ERROVERLOAD	Alarm occurs when a load exceeding the maximum torque of the motor is applied for a distance of more than 5 seconds or more than 10 turns.	0X00000800
FFLAG_ERROVERHEAT	The internal temperature of the drive exceeds 85°C.	0X00001000
FFLAG_ERRBACKEMF	A counter electromotive force of the motor exceeds 70V.	0X00002000
FFLAG_ERRMOTORPOWER	Alarm occurs when it is the motor voltage error.	0X00004000
FFLAG_ERRINPOSITION	Alarm occurs when it is In-position error.	0X00008000
FFLAG_EMGSTOP	If the motor is in an emergency stop.	0X00010000
FFLAG_SLOWSTOP	If the motor is in an emergency stop.	0X00020000
FFLAG_ORIGINRETURNING	During homing operation.	0X00040000
FFLAG_INPOSITION	In-position is complete.	0X00080000
FFLAG_SERVOON	The motor is under Servo On.	0X00100000
FFLAG_ALARMRESET	Alarm Reset is executed.	0X00200000
FFLAG_PTSTOPPED	Position Table operation has been finished.	0X00400000
FFLAG_ORIGINSENSOR	The origin sensor is ON.	0X00800000
FFLAG_ZPULSE	In case of z-pulse type operation during homing operaiton.	0X01000000
FFLAG_ORIGINRETOK	Origin return operation has been finished.	0X02000000
FFLAG_MOTIONDIR	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 in the acceleration section.	0X20000000
FFLAG_MOTIONDECEL	The motor is operating in the deceleration section.	0X40000000
FFLAG_MOTIONCONST	The motor is operating in the constant speed, not acceleration / deceleration sections.	0X80000000

## 1 - 2 - 6 . Position Table Item

Refer to 'motion\_define.h' among the include file.

Name	Name of structure paramater	Number of Byte	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 the setting method by each item, refer to 「[User Manual-Position Table](#)」.

## 1 - 3 . Program Method

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

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

In this case, Library that serviced together (Refer to 「[2. Library for PC Program](#)」) is used.

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

(1) C++

C++ header file(\*.h) and library file(\*.lib or \*.dll) are required. These files are included in ["WWFASTECH\WEzi-MOTION Plus-E V6\WWinclue\WW"](#) and the following contents should be included in the source file for development.

```
#include "WWFASTECH\WEzi-MOTION Plus-E V6\WWinclue\WWFAS_EziMotionPlusE.h"
#include "WWFASTECH\WEzi-MOTION Plus-E V6\WWinclue\WWReturnCodes_Define.h"
#include "WWFASTECH\WEzi-MOTION Plus-E V6\WWinclue\WWMOTION_DEFINE.h"
#include "WWFASTECH\WEzi-MOTION Plus-E V6\WWinclue\WWCOMM_Define.h"
```

Library files are as follows.

1) For 32bit

```
"WWFASTECH\WEzi-MOTION Plus-E V6\WWinclue\WEziMotionPlusE.lib"
"WWFASTECH\WEzi-MOTION Plus-E V6\WWinclue\WEziMotionPlusE.dll"
```

2) For 64bit

```
"WWFASTECH\WEzi-MOTION Plus-E V6\WWinclue_x64\WEziMotionPlusE.lib"
"WWFASTECH\WEzi-MOTION Plus-E V6\WWinclue_x64\WEziMotionPlusE.dll"
```

A sample program source of library is included in the folder below.

```
"WWFASTECH\WEzi-MOTION Plus-E V6\WWinclue\WWEzMotionPlusE_C++_WW"
```

(2) For C#

C# header file and library file are required. These files are included in ["WWFASTECH\WEzi-MOTION Plus-E V6\WWinclue\WW"](#) and the following contents should be included in a source file for development.

```
#include "WWFASTECH\WEzi-MOTION Plus-E V6\WWinclue\WWMOTION_DEFINE_PlusE.cs"
```

Library files are as follows.

1) For 32bit

```
"WWFASTECH\WEzi-MOTION Plus-E V6\WWinclue\WWLIB_EziMOTIONPlusE.cs"
```

2) For 64bit

```
"WWFASTECH\WEzi-MOTION Plus-E V6\WWinclue_x64\WWLIB_EziMOTIONPlusE.cs"
```

A sample program source of library is included in the folder below.

```
"WWFASTECH\WEzi-MOTION Plus-E V6\WWinclue\WWEzMotionPlusE_C#_WW"
```

(3) 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 program using protocol, this service is not provided.

Sort	Name	Return Value	Description
Normal	FMM_OK	0(0x00)	The function has normally performed the command.
Input Error	FMM_NOT_OPEN	1(0x01)	Wrong port number is inputted.
	FMM_INVALID_PORT_NUM	2(0x02)	The port is not connected.
	FMM_INVALID_SLAVE_NUM	3(0x03)	Wrong board number is inputted.
Operation Error	FMM_POSTABLE_ERROR	9(0x09)	An error occurs while read/write to the position table.
Connection Error	FMC_DISCONNECTED	5(0x05)	The relevant board is disconnected.
	FMC_TIMEOUT_ERROR	6(0x06)	Response delay occurs. (100[msec])
	FMC_RECVPACKET_ERROR	8(0x08)	Protocol level error occurs in packet that comes from Drive.

(4) The following table shows return values included commonly in all libraries. The user can check the result (communication status, running status) judged by the drive. Supported for both case of using library(DLL) and programming using protocol.

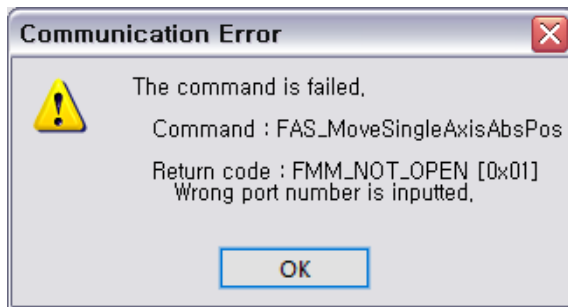
Sort	Name	Return Value	Description
Normal	FMP_OK	0(0x00)	Communication has been normally performed.
Input Error	FMP_FRAMETYPEERROR	128(0x80)	The command that the drive cannot recognize.
	FMP_DATAERROR	129(0x81)	Inputted data is out of the range.
Operation Error	FMP_RUNFAIL	133(0x85)	The motor is already running or not prepared for running. Other wrong motion command.
	FMP_RESETFAIL	134(0x86)	The user cannot execute Alarm Reset 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)	'Servo ON' signal is already assigned to input pin.
Connection Error	FMP_PACKETERROR	130(0x82)	Protocol level error occurs in packed that Drive's received.

For other return values, refer to [1 - 1 - 5 . Transmit Frame Data Structure and Communication Error].

## 2 - 2 . Communication Status Window

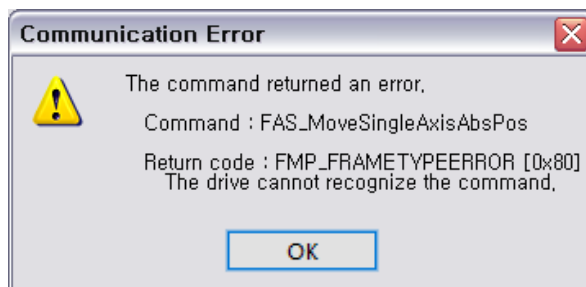
The communication status window can be classified into three types.

(1) Communication Error



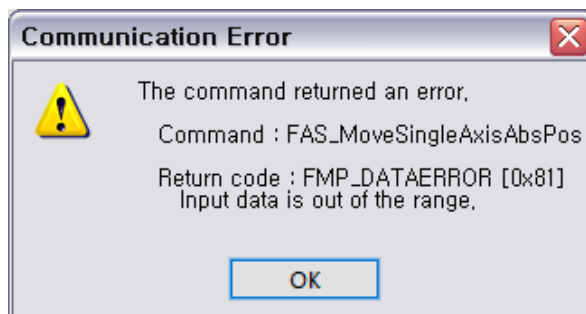
FMM\_NOT\_OPEN,

COM Port is not connected. (This is an error that cannot occur in the GUI.)



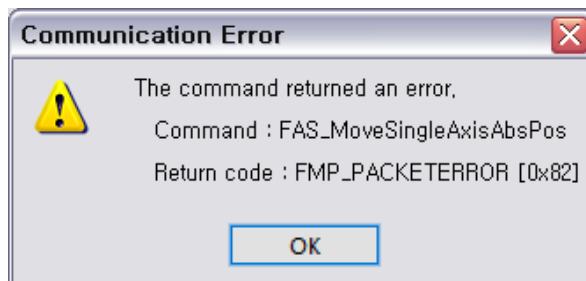
FMP\_FRAMETYPEERROR = 0x80,

Drive cannot recognize the Command or does not support the Command.



FMP\_DATAERROR,

The range of data entered is outside the range supported by Drive.

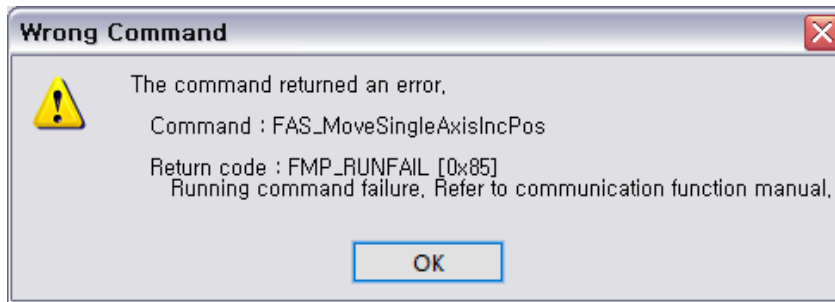


FMP\_PACKETERROR,

The received frame does not meet the standard. (The length of the packet sent to the drive Does not match.)



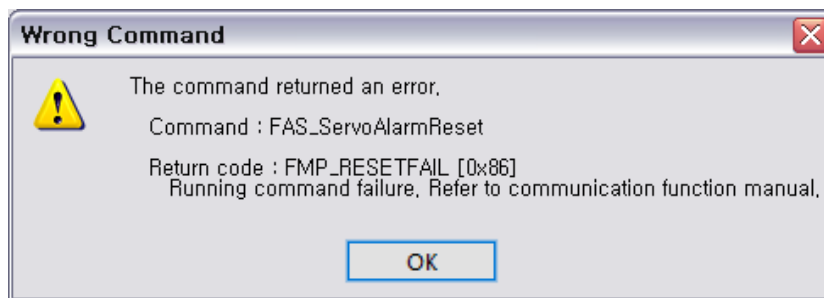
## (2) Wrong Command



FMP\_RUNFAIL

Operation command failure: An attempt was made to execute a new operation under the following conditions.

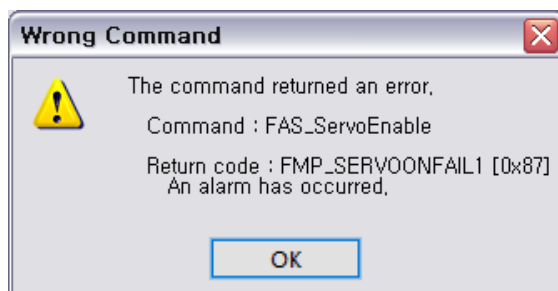
- The motor is already running.
- The motor is under stop command.
- Servo OFF Status.
- Try to Z-pulse Origin without external encoder.
- Other wrong motion command.



FMP\_RESETFAIL,

Attempting to execute Reset under the following conditions.

- Servo ON state
- Already reset by external input signal.



FMP\_SERVOONFAIL1,

An attempt was made to execute the Servo ON command while an alarm occurred.



FMP\_SERVOONFAIL2,

An attempt was made to execute the Servo Off command during an emergency stop.



FMP\_SERVOONFAIL3,

Servo ON Signal is assigned to the external input and cannot be executed.

### (3) Command Execution Error



FMM\_POSTABLE\_ERROR

Failed to execute Position Table related function.

## 2 - 3 . Drive Connection Function

Function Name	Description
<b>FAS_Connect</b>	The drive tries to connect with UDP Protocol. : When it it successfully connected, TRUE will return. Otherwise, FALSE will return.
<b>FAS_ConnectTCP</b>	The drive module tries to connect with TCP Protocol. : When it it successfully connected, TRUE will return. Otherwise, FALSE will return.
<b>FAS_Reconnect</b>	Reconnect with existing IP, Protocol, iBdID.
<b>FAS_AutoReconnect</b>	In case of using TCP, if the response is not within 100[ms] or unintentionally disconnected from TCP, it automatically connects to another port and executes the unsuccessful function again.
<b>FAS_Close</b>	The drive tries to disconnect communication with the drive.
<b>FAS_GetSlaveInfo</b>	The drive reads drive type and program version. : Drive type and version information will be returned.
<b>FAS_GetMotorInfo</b>	Reads information about the type and manufacturer of the motor connected to the drive.
<b>FAS_IsSlaveExist</b>	The drive checks whether there is the relevant drive. : When it exists, TRUE will return. Otherwise, FALSE will return.
<b>FAS_IsBdIDExist</b>	Check whether BdID is used for the IP Address. : When it exists, TRUE will return. Otherwise, FALSE will return.
<b>FAS_IsIPAddressExist</b>	Check whether the IP Address is assigned to the BdID. : When it exists, TRUE will return. Otherwise, FALSE will return.
<b>FAS_EnableLog</b>	Controls output of the Log related to communication error. : When it exists, TRUE will return. Otherwise, FALSE will return.
<b>FAS_SetLogPath</b>	Set the path to save the output Log. : When it exists, TRUE will return. Otherwise, FALSE will return.
<b>FAS_SetLogLevel</b>	Output Log according to the set level. : By default, only Log related to internal communication errors are displayed. (LOG_LEVEL_COMM)
<b>FAS_PrintCustomLog</b>	Output arbitrary Log.

- 1. The following functions are supported by F/W Ver V06.01.020.04 Library Ver 2.0.0.10 or later.

- 1) FAS\_ConnectTCP
- 2) FAS\_Reconnect
- 3) FAS\_SetLogLevel
- 4) FAS\_PrintCustomLog

2. The following function is supported by F/W Ver V06.01.020.05 Library Ver 2.3.0.15 or later.

- 1) FAS\_SetAutoReconnect

## FAS\_Connect

---

FAS\_Connect is a function that connects Ezi-SERVOII Plus-E MINI with UDP Protocol.

Syntax

```
BOOL FAS_Connect(
    BYTE sb1, BYTE sb2, BYTE sb3, BYTE sb4
    int 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) is 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
    int iBdID = 0 // A unique board number of 192.168.0.2
    char lpBuff[256];
    int nBuffSize = 256;
    BYTE nType;
    int nRtn;

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

```

    }

    if (FAS_IsSlaveExist(iBdID) == FALSE)
    {
        // There is no relevant board number.
        // Check the board number of Ezi-SERVOII Plus-E MINI.
        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);

    // Terminate the connection.
    FAS_Close(iBdID);
}

```

See Also

FAS\_Close

## FAS\_ConnectTCP

---

FAS\_ConnectTCP is a function that connects Ezi-SERVOII Plus-E MINI with TCP Protocol.

### Syntax

```
BOOL FAS_ConnectTCP(
    BYTE sb1, BYTE sb2, BYTE sb3, BYTE sb4
    int 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*

It is unique ID of board to connect. The ID(value) is 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
    int iBdID = 0 // A unique board number of 192.168.0.2
    char lpBuff[256];
    int nBuffSize = 256;
    BYTE nType;
    int nRtn;

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

```

    }

    if (FAS_IsSlaveExist(iBdID) == FALSE)
    {
        // There is no relevant board number.
        // Check the board number of Ezi-SERVOII.
        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\_Reconnect

---

This function reconnects the drive without using FAS\_Connect ().

### Syntax

```
void FAS_Reconnect(int iBdID);
```

### Parameters

*iBdID*

Drive ID number to reconnect.

### Remarks

After connecting with FAS\_Connect() function, the connection is terminated or connected again without using FAS\_Connect().

### Example

Refer to FAS\_Connect library.

### See Also

FAS\_Connect



## FAS\_SetAutoReconnect

---

To connect TCP communication automatically.

### Syntax

```
void FAS_SetAutoReconnect(BOOL bSET);
```

### Parameters

*bSET*

Set whether to use the Auto Reconnect function.

### Remarks

If the function used after setting FAS\_SetAutoReconnect to Set is not within 100[ms], or if TCP is unintentionally disconnected, connect using a different port and execute the function that was not answered again.

(Only executed when connected using FAS\_ConnectTCP.)

When connecting to the GUI while using the above functions, be sure to connect the FUI by UDP.

### Example

```
#include "FAS_ EziMOTIONPlusE.h"

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

    // Try to connect.
    if (FAS_ConnectTCP(sb1, sb2, sb3, sb4, iBdID) == FALSE)
    {
        // Connection failed.
        MessageBox(_T("connect fail!"));
        return;
    }
    // Enable Auto Reconnect
    FAS_SetAutoReconnect(SET);
}
```

### See Also

## FAS\_Close

---

Release the connection of the ID.

### Syntax

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

### Parameters

*iBdID*

The ID number of the board iBdID set by FAS\_Connect function.

*pType*

Type number of relevant board.

*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.

FMM\_INVALID\_SLAVE\_NUM : The board of iBdID does not exist.

### Remarks

### Example

Refer to FAS\_Connect library.

### See Also

## FAS\_GetMotorInfo

---

To get the motor information string of the relevant drive.

Syntax

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

Parameters

*iBdID*

The ID number of the board. The iBdID set by the FAS\_Connect function.

*pType*

Type number of the motor.

*lpBuff*

Buffer Pointer to receive motor information string.

*nBuffSize*

The memory allocation size value of lpBuff.

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 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(int iBdID);
```

### Parameters

*iBdID*

The ID number of the board. iBdID set by FAS\_Connect function.

### Return Value

TRUE : Connection status.

FALSE : Disconnection status.

### Remarks

This function is provided only in the library and is not supported for protocol programming.

### Example

Refer to FAS\_Connect library.

### See Also

FAS\_Connect

## FAS\_IsBdIDExist

---

To check that the drive is connected.

### Syntax

```
BOOL FAS_IsBdIDExist(int iBdID, BYTE* sb1, BYTE* sb2, BYTE* sb3, BYTE* sb4 );
```

### Parameters

*iBdID*

The ID number of the board. iBdID set by FAS\_Connect function.

*sb1, sb2, sb3, sb4*

IP Address. ( Ex,192.168.0.10 → sb1:192, sb2:168, sb3:0, sb4:10)

### Return Value

TRUE : Use relevant BdID

FALSE : Not use relevant BdID

### Remarks

This function is provided only in the library and is not supported for protocol programming.

### Example

Refer to FAS\_Connect library.

### See Also

FAS\_Connect

## FAS\_IsIPAddressExist

---

To check that the drive is connected.

### Syntax

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

### Parameters

*sb1, sb2, sb3, sb4*

IP Address. ( Ex,192.168.0.10 → sb1:192, sb2:168, sb3:0, sb4:10)

*iBdID*

The ID number of the board. iBdID set by FAS\_Connect function.

### Return Value

TRUE : Use IP Address

FALSE : Not use IP Address

### Remarks

This function is provided only in the library and is not supported for protocol programming.

### Example

Refer to FAS\_Connect library.

### See Also

FAS\_Connect

## FAS\_EnableLog

---

To control Log output about communication error.

### Syntax

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

### Parameters

*bEnable*

Log output setting.

### Remarks

Controls the Log output which occurs while using Ezi-MOTION Plus-E function in the current process. This setting does not affect the Log output of other processes or programs.

Log starts from FAS\_Connect. When FAS\_Close is used to disconnect the currently connected drive, Log output is terminated. The default setting for the Log output is TRUE.

### Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcDisableLog()
{

    FAS_EnableLog(FALSE);

    // After this, the log is not output.

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

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

    // Terminate the connection.
    FAS_Close(iBdID);
}
```

### See Also

FAS\_SetLogPath



## FAS\_SetLogPath

---

Setup the folder path of Log output files.

### Syntax

```
BOOL FAS_SetLogPath(LPCTSTR IpPath);
```

### Parameters

*IpPath*

Folder path character string of Log output file.

### Return Value

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

### Remarks

This function has to be called before FAS\_Connect library.

If the IpPath 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 the program exist.

### Example

```
#include "FAS_ EziMOTIONPlusE.h"

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

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

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

    // Logs of all functions are displayed in 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

## FAS\_SetLogLevel

---

Set the path to save the output Log.

Syntax

```
BOOL FAS_SetLogLevel(enum LOG_LEVEL level);
```

Parameters

*level*

Log output range setting

Return Value

If a value other than the step setting value is entered, FALSE is returned.

Remarks

LOG\_LEVEL\_COMM : Only Logs related to communication errors are displayed.

LOG\_LEVEL\_PARAM : The parameter setting function Log is additionally output to the above Log output.

LOG\_LEVEL\_MOTION : The motion command function Log is additionally output to the above Log output.

LOG\_LEVEL\_ALL : All Logs that can be outputted are displayed.

Example

```
#include "FAS_ EziMOTIONPlusE.h"

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

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

    FAS_SetLogLevel(LOG_LEVEL_ALL); // Log output range setting

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

        return;
    }

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

See Also

FAS\_EnableLog

## FAS\_PrintCustomLog

---

Set the path to save the output Log.

### Syntax

```
BOOL FAS_PrintCustomLog(  
    int iBdID,  
    enum LOG_LEVEL level,  
    LPCTSTR lpszMsg  
);
```

### Parameters

*iBdID*

ID number of the board. iBdID set in FAS\_Connect function

*level*

Log output range setting

*lpszMsg*

String of Log to be output

### Return Value

If a value other than the step setting value is entered, FALSE is returned.

### Remarks

The Level is equal to the set value(range) of FAS\_SetLogLevel().

Used to output the log at a specific location (function) in the program.

Or, output the log differently from the setting value of FAS\_SetLogLevel () in the program.

### Example

```
#include "FAS_ EziMOTIONPlusE.h"  
  
void funcCustomLog()  
{  
    int iBdID = 0 // A unique board number of 192.168.0.2  
    int level = LOG_LEVEL_PARAM;  
  
    //Communication error and parameter setting function Log output setting  
    FAS_PrintCustomLog (iBdID, level, lpszMsg );  
  
}
```

### See Also

FAS\_SetLogLevel

## 2 - 4 . Parameter Control Function

Function Name	Description
<b>FAS_SaveAllParameters</b>	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.
<b>FAS_SetParameter</b>	The designated parameter is saved to the RAM : Specific parameter is saved.
<b>FAS_GetParameter</b>	The designated parameter is read from the RAM : Specific parameter is read.
<b>FAS_GetROMParameter</b>	The designated parameter is read from the ROM : Specific parameter is read from the ROM.

## FAS\_SaveAllParameters

All parameters and assigned value of In/Out signals are saved in the ROM area.

Syntax

```
Int FAS_SaveAllParameters(  
    int 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 with this function..

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcModifyParameter()
{
    BYTE sb1 = 192, sb2 = 168, sb3=0, sb4=2 // IP :192.168.0.2
    int iBdID = 0;      // A unique board number

    long lParamVal;
    int nRtn;

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

        return;
    }

    // Check Axis Start Speed Parameter value
    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. (Save it to the RAM)

### Syntax

```
int FAS_SetParameter(
    int 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 parameter of specified 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 set the parameter of designated number from the RAM as 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(  
    int 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 of designated number in the RAM area.

### Example

Refer to FAS\_SaveAllParameter library.

### See Also

FAS\_SetParameter



## FAS\_GetROMParameter

---

To call up the parameter saved in the ROM area.

### Syntax

```
int FAS_GetROMParameter(
    int 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.

This function does not change the value in the RAM. To change the value in the RAM, run FAS\_SetParameter.

### Example

Refer to FAS\_SaveAllParameter library.

### See Also

FAS\_SaveAllParameters

## 2 - 5 . Servo Control Function

Function Name	Description
<b>FAS_ServoEnable</b>	Turns ON/OFF the designated drive.
<b>FAS_ServoAlarmReset</b>	Release the alarm of the drive. : Remove the cause of the alarm before doing it.
<b>FAS_GetAlarmType</b>	Check the current alarm occurrence and the type of alarm.

## FAS\_ServoEnable

---

To turn the drive Servo ON/OFF.

### Syntax

```
int FAS_ServoEnable(
    int 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
    int iBdID = 0;      // A unique board number
    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);

        // Servo On when SERVO_ON flag is OFF.
        if (AxisStatus.FFLAG_SERVOON == 0)
        {
            nRtn = FAS_ServoEnable(iBdID, TRUE);
            _ASSERT(nRtn == FMM_OK);
        }

        // If there is an Alarm, execute AlarmReset.
        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(  
    int 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, remove the cause of the alarm first.

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

### Example

Refer to FAS\_ServoEnable library.

### See Also

FAS\_ServoEnable

## 2 - 6 . Control I/O Function

Function Name	Description
<b>FAS_SetIOInput</b>	To set the input signal level of the control input port : Input signal is set to [ON] or [OFF].
<b>FAS_GetIOInput</b>	To read the current input signal status of the control input port : The signal status returns by bit for each input signal.
<b>FAS_SetIOOutput</b>	To set the output signal level of the control input port : Output signal is set to [ON] or [OFF].
<b>FAS_GetIOOutput</b>	To read the current input signal status of the control output port : The signal status returns by bit for each output signal.
<b>FAS_GetIOAssignMap</b>	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.
<b>FAS_SetIOAssignMap</b>	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.
<b>FAS_IOAssignMapReadROM</b>	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(  
    int iBdID,  
    DWORD dwIOSetMask,  
    DWORD dwIOCLRMask  
);
```

Parameters

*iBdID*

The ID number of the board. iBdID set by FAS\_Connect function.

*dwIOSetMask*

Bitmask value of the input to be Set(ON status).

*dwIOCLRMask*

Bitmask value of the input to be Clear(OFF status).

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 not to duplicate bit values of dwIOSetMask and dwIOCLRMask.

Example

```
#include "FAS_ EziMOTIONPlusE.h"  
  
void funcIO()  
{  
    BYTE sb1 = 192, sb2 = 168, sb3=0, sb4=2 // IP :192.168.0.2  
    int iBdID = 0;      // A unique board number  
    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 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 signal 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(
    int 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

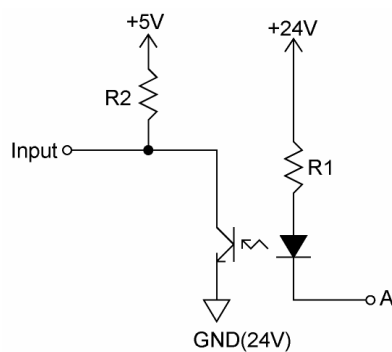
FMM\_OK : Command has been normally performed.

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

FMM\_INVALID\_SLAVE\_NUM : The board of the corresponding iBdID does not exist.

Remarks

There are 6 input pins in Ezi-SERVOII Plus-E MINI. The user can select and program 3 input pins of them. This function can read the input port status by 32bit unit. 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(
    int iBdID,
    DWORD dwIOSetMask,
    DWORD dwIOCLRMask
);
```

Parameters

*iBdID*

The ID number of the board. iBdID set by FAS\_Connect function.

*dwIOSetMask*

Bitmask value of the output to be Set(ON status).

*dwIOCLRMask*

Bitmask value of the output to be Clear(OFF status).

Return Value

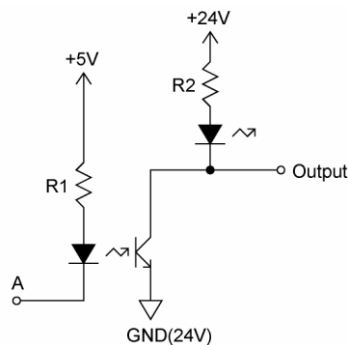
FMM\_OK : Command has been normally performed.

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

FMM\_INVALID\_SLAVE\_NUM : The board of the corresponding iBdID does not exist.

Remarks

There are 2 input pins in Ezi-SERVOII Plus-E MINI. The user can select and program 1 output pin of them.



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

Be careful not to duplicate bit values of dwIOSetMask and dwIOCLRMask.

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(  
    int 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(
    int iBdID,
    BYTE iOPinNo,
    BYTE* nIOLogic,
    BYTE* bLevel
);
```

Parameters

*iBdID*

The ID number of the board. iBdID set by FAS\_Connect function.

*iOPinNo*

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 exist.

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
    int iBdID = 0;      // A unique board number.
    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 0 ~ 11 are possible (Note : 0 ~ 2 are fixed).
nRtn = FAS_SetIOAssignMap(iBdID, iPinNo, SERVO_IN_BITMASK_SERVOON,
LEVEL_LOW_ACTIVE);
_ASSERT(nRtn == FMM_OK);

// Check assigned 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;          // Values 0 ~ 9 are possible (Note : 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(
    int iBdID,
    BYTE iOPinNo,
    BYTE nLogicNo,
    BYTE bLevel
);
```

Parameters

*iBdID*

The ID number of the board. iBdID set by FAS\_Connect function

*iOPinNo*

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 iOPinNo or nIOLogicvalue is out of range.

Remarks

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

Example

Refer to FAS\_GSetIOAssignMap library.

See Also

FAS\_GetIOAssignMap

## FAS\_IOAssignMapReadROM

---

The I/O setting status and signal level values saved in the current ROM area are read.

### Syntax

```
int FAS_PosTableReadROM(  
  
    int 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\_GetIOAssignMap



## 2 - 7 . Position Control Function

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

## FAS\_SetCommandPos

---

To set the Command Position value of Motor.

Syntax

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

Parameters

*iBdID*

The ID number of the board. iBdID set by FAS\_Connect function.

*lCmdPos*

Command 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 user wants.

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcClearPosition()
{

    BYTE sb1 = 192, sb2 = 168, sb3=0, sb4=2 // IP :192.168.0.2
    int iBdID = 0;      // A unique board number
    int nRtn;

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

```
// Initialize Command Position and Actual Position value.  
nRtn = FAS_SetCommandPos(iBdID, 0);  
_ASSERT(nRtn == FMM_OK);  
nRtn = FAS_SetActualPos(iBdID, 0);  
_ASSERT(nRtn == FMM_OK);  
  
// Connection fail.  
FAS_Close(iBdID);  
  
}
```

See Also

FAS\_SetActualPos

## FAS\_SetActualPos

---

To set the Actual Position value of the motor

### Syntax

```
int FAS_SetActualPos(  
    int 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(
    int iBdID,
    long* lCmdPos
);
```

Parameters

*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
    int iBdID = 0;      // A unique board number
    long lValue;
    int nRtn;

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

    // Check the information on the position of Ezi-SERVOII
    nRtn = FAS_GetCommandPos(iBdID, &lValue);
```

```
    _ASSERT(nRtn == FMM_OK);  
    printf("CMDPOS : %d \n", IValue);  
    nRtn = FAS_GetActualPos(iBdID, &IValue);  
    _ASSERT(nRtn == FMM_OK);  
    printf("ACTPOS : %d \n", IValue);  
    nRtn = FAS_GetPosError(iBdID, &IValue);  
    _ASSERT(nRtn == FMM_OK);  
    printf("POSERR : %d \n", IValue);  
    nRtn = FAS_GetActualVel(iBdID, &IValue);  
    _ASSERT(nRtn == FMM_OK);  
    printf("ACTVEL : %d \n", IValue);  
  
    // Connection close  
    FAS_Close(iBdID);  
}
```

See Also

FAS\_GetActualPos

## FAS\_GetActualPos

---

To read the actual position value of the current motor

### Syntax

```
int FAS_GetActualPos(  
    int 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

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

### Example

Refer to FAS\_GetCommandPosition library.

### See Also

FAS\_GetCommandPos

## FAS\_GetPosError

---

To read Position Error value of the motor

### Syntax

```
int FAS_GetPosError(  
    int iBdID,  
    long* IPosErr  
);
```

### Parameters

*iBdID*

The ID number of the board. iBdID set by FAS\_Connect function.

*IPosErr*

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 Actual Velocity value of the motor

### Syntax

```
int FAS_GetActualVel(  
    int 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 Command Position value and Actual Position value of the motor to '0'

### Syntax

```
int FAS_ClearPosition(
    int 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
    int iBdID = 0;      // A unique board number
    int nRtn;

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

    // Initialize Command Position value and Actual Position value 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
<b>FAS_GetIOAxisStatus</b>	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.
<b>FAS_GetMotionStatus</b>	To read the current running progress status and its PT number : The command position value, the actual position value, the speed value will return.
<b>FAS_GetAllStatus</b>	To read all status including the current status at one time : This function is to combine 'FAS_GetIOAxisStatus' function and 'FAS_GetMotionStatus' function.
<b>FAS_GetAxisStatus</b>	To read the running status Flag value of the relevant drive

## FAS\_GetIOAxisStatus

---

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

### Syntax

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

### Parameters

*iBdID*

The ID number of the board. iBdID set by FAS\_Connect function.

*dwInStatus*

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

*dwOutStatus*

Parameter pointer which I/O Output values 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 once

### Syntax

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

### Parameters

*iBdID*

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 board, the motor Axis Status, the motor motion status

Syntax

```
int FAS_GetAllStatus(
    int 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*

Parameter pointer which I/O 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.

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

### Syntax

```
int FAS_GetAxisStatus(  
    int 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 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

## 2 - 9 . Operation Control Function

Function Name	Description
<b>FAS_MoveStop</b>	The motor in running is decelerate and stopped.
<b>FAS_EmergencyStop</b>	The motor in running stops directly without deceleration.
<b>FAS_MoveOriginSingleAxis</b>	The motor starts the origin return.
<b>FAS_MoveSingleAxisAbsPos</b>	The motor moves as much as the given absolute position value.
<b>FAS_MoveSingleAxisIncPos</b>	The motor moves as much as the given incremental position value.
<b>FAS_MoveToLimit</b>	The motor moves up to the position that the Limit sensor is detected.
<b>FAS_MoveVelocity</b>	The motor moves to the given velocity and direction. : This function is available to Jog motion and more.
<b>FAS_PositionAbsOverride</b>	While the motor is running, the target absolute position value[pause] is changed.
<b>FAS_PositionIncOverride</b>	While the motor is running, the target incremental position value[pause] is changed.
<b>FAS_VelocityOverride</b>	While the motor is running, the running velocity value[pps] is changed.
<b>FAS_MoveLinearAbsPos</b>	Linear Interpolation operation is executed as much as absolute value given to two or more drives.
<b>FAS_MoveLinearIncPos</b>	Linear Interpolation operation is executed as much as incremental value given to two or more drives.
<b>FAS_MoveLinearAbsPos2<sup>*1</sup></b>	Improved version of <b>FAS_MoveLinearAbsPos</b> . Acceleration and Deceleration were improved.
<b>FAS_MoveLinearIncPos2<sup>*1</sup></b>	Improved version of <b>FAS_MoveLinearIncPos</b> . Acceleration and Deceleration were improved.
<b>FAS_MoveSingleAxisAbsPosEx</b>	The motor moves as much as the given absolute position value. It is possible to set the acceleration and deceleration times.
<b>FAS_MoveSingleAxisIncPosEx</b>	The motor moves as much as the given incremental position value. It is possible to set the acceleration and deceleration times.
<b>FAS_MoveVelocityEx</b>	The motor moves to the given velocity and directions. : This function is available to Jog motion and more. It is possible to set the acceleration and deceleration times.
<b>FAS_MovePause</b>	In the running state, the operation is paused and the operation resumed in the paused state.

<sup>\*1</sup> These functions are available from Firmware [ver.6.1.xx.18] or higher version.

## FAS\_MoveStop

---

To stop the motor

### Syntax

```
int FAS_MoveStop(  
    int 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 immediately without deceleration

### Syntax

```
int FAS_EmergencyStop(  
    int 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 8-3 Origin Return](#)'.

### Syntax

```
int FAS_MoveOriginSingleAxis(  
    int 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(
    int 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
    int iBdID = 0;      // A unique board number
    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 & (SERVO_IN_LOGIC_STOP | SERVO_IN_LOGIC_PAUSE |
SERVO_IN_LOGIC_ESTOP))
    FAS_SetIOInput(iBdID, 0, SERVO_IN_LOGIC_STOP | SERVO_IN_LOGIC_PAUSE |
SERVO_IN_LOGIC_ESTOP);

// Increase 15000 pulse to the motor.
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'.

```

```

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

// Stand by until motion commnad 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(  
    int 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 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 a command to motor to search the Limit sensor

### Syntax

```
int FAS_MoveToLimit(  
    int iBdID,  
    DWORD IVelocity,  
    int iLimitDir,  
);
```

### Parameters

*iBdID*

The ID number of the board. iBdID set by FAS\_Connect function.

*IVelocity*

Velocity 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(  
    int iBdID,  
    DWORD lVelocity,  
    int iVelDir  
);
```

### 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.

### 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 absolute position move

Syntax

```
int FAS_PositionAbsOverride(
    int 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

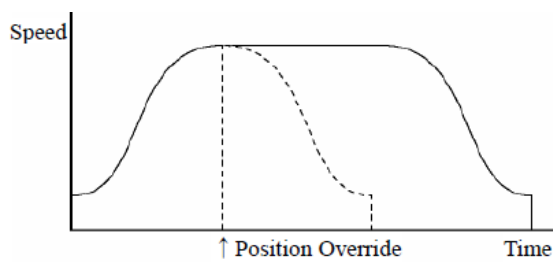
FMM\_OK : Command has been normally performed.

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

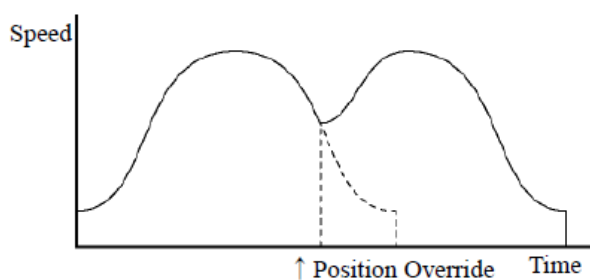
FMM\_INVALID\_SLAVE\_NUM : The board of the corresponding iBdID does not exist.

Remarks

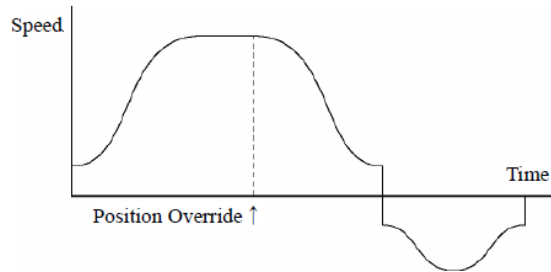
1) If the target position is set to the farther coordinate than the original target position while the motor moves to the accelerated or uniform velocity, the motor moves as the velocity pattern and stops at the target position.



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



- 3) If the changed target position is set to the closer coordinate than the original target position, the motor moves to the changed target position and then stops.



- 4) Can not be used concurrently with the FAS\_PositionAbsOverride library.  
It can not be used concurrently with the FAS\_VelocityOverride library.

#### Example

Refer to FAS\_MoveSingleAxisAbsPos library.

#### See Also

FAS\_PositionIncOverride

## FAS\_PositionIncOverride

---

To change the incremental position value set while the incremental position move

### Syntax

```
int FAS_PositionIncOverride(  
    int iBdID,  
    long IOverridePos  
);
```

### Parameters

*iBdID*

The ID number of the board. iBdID set by FAS\_Connect function.

*IOverridePos*

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.
- 2) Can not be used simultaneously with the FAS\_PositionIncOverride library.  
It can not be used concurrently with the FAS\_VelocityOverride 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(  
    int iBdID,  
    DWORD IVelocity  
);
```

Parameters

*iBdID*

The ID number of the board. iBdID set by FAS\_Connect function.

*IVelocity*

Velocity to be changed

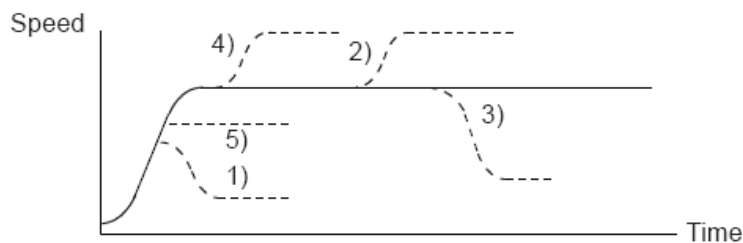
Return Value

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

- Check the speed range that can be changed according to the command speed (movement command speed in stop state) and use the function. Please refer to the table below.

Command Speed [pps]	Speed Change Range[pps]	
	Min.	Max.
1~983	1	4914
984~1638	1	8191
1639~3276	1	16383
3277~6553	2	32766
6554~16384	5	81915
16385~32768	10	163830
32769~65536	20	327660
65537~163840	50	819150
163840~327680	100	1638300
327681~655360	200	3276600

If set to a value not in the speed change range, the speed changes to the minimum or maximum value within the range.

- Can not be used concurrently with the FAS\_PositionIncOverride library.
- Can not be used concurrently with the FAS\_PositionAbsOverride library.

#### Example

Refer to FAS\_MoveSingleAxisAbsPos library.



## FAS\_MoveLinearAbsPos

---

To move the motor to the absolute coordinate

(It is possible to set the acceleration and deceleration times.)

Syntax

```
int FAS_MoveLinearAbsPos(  
    BYTE nNoOfBds,  
    int* iBdID,  
    long* lplAbsPos,  
    DWORD lFeedrate,  
    DWORD wAcceltime  
);
```

Parameters

*nNoOfBds*

Number of drives to execute Linear Motion

*iBdID*

ID array of the drives

*lplAbsPos*

Arrangement of Move Location of the drives

*lFeedrate*

Linear velocity value when moving

*wAcceltime*

Time value of acceleration / deceleration section during movement

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

## FAS\_MoveLinearIncPos

---

To move the motor to the incremental coordinate  
(It is possible to set the acceleration and deceleration times.)

Syntax

```
int FAS_MoveLinearIncPos(
    BYTE nNoOfBds,
    int* iBdID,
    long* lplIncPos,
    DWORD lFeedreat,
    DWORD wAcceltime
);
```

Parameters

*nNoOfBds*

Number of drives to execute Linear Motion

*iBdID*

ID array of the drives

*lplIncPos*

Arrangement of Move Location of the drives

*lFeedrate*

Linear velocity value when moving

*wAcceltime*

Time value of acceleration / deceleration section during movement

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

## FAS\_MoveLinearAbsPos2

---

Improved version of FAS\_MoveLinearAbsPos.

Acceleration and Deceleration were improved.

### Syntax

```
int FAS_MoveLinearAbsPos2(  
    BYTE nNoOfBds,  
    int* iBdID,  
    long* lplAbsPos,  
    DWORD lFeedrate,  
    DWORD wAcceltime  
);
```

### Parameters

*nNoOfBds*

Number of drives to execute linear motion

*iBdID*

ID array of Drives

*lplAbsPos*

Movement position's arrangement of Drives

*lFeedrate*

Linear velocity value during movement

*wAcceltime*

Time value of acceleration / deceleration section during movement

### Return Value

FMM\_OK : The command ran successfully.

FMM\_NOT\_OPEN : The drive is not connected yet.

FMM\_INVALID\_SLAVE\_NUM : The drive of corresponding iBdID does not exist.

### Remarks

## FAS\_MoveLinearIncPos2

---

Improved version of FAS\_MoveLinearIncPos.

Acceleration and Deceleration were improved.

Syntax

```
int FAS_MoveLinearIncPos2(
    BYTE nNoOfBds,
    int* iBdID,
    long* lplIncPos,
    DWORD lFeedrate,
    DWORD wAcceltime
);
```

Parameters

*nNoOfBds*

Number of drives to execute linear motion

*iBdID*

ID array of Drives

*lplAbsPos*

Movement position's arrangement of Drives

*lFeedrate*

Linear velocity value during movement

*wAcceltime*

Time value of acceleration / deceleration section during movement

Return Value

FMM\_OK : The command ran successfully.

FMM\_NOT\_OPEN : The drive is not connected yet.

FMM\_INVALID\_SLAVE\_NUM : The drive of corresponding iBdID does not exist.

Remarks

## FAS\_MoveSingleAxisAbsPosEx

---

To move the motor to the absolute coordinate

(It is possible to set the acceleration and deceleration times.)

Syntax

```
int FAS_MoveSingleAxisAbsPosEx(  
    int 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  
    int iBdID = 0; // A unique board number  
    DWORD dwAxisStatus, dwInput;  
    EZISERVO_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;
}

// Move the motor with a certain acceleration and deceleration 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 motion command is done.
do
{
    Sleep(1);

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

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

// Waiting until motion command 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 move the motor to the incremental coordinate  
(It is possible to set the acceleration and deceleration times.)

Syntax

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

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 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 (  
    int 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  
    int iBdID = 0;          // A unique board number  
    long lVelocity;  
    VELOCITY_OPTION_EX opt = {0};  
    int nRtn;  
  
    // Try to connect
```

```

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

    return;
}

// Move the motor with a certain acceleration and deceleration 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 - 1 0 . Position Table Control Function

Function Name	Description
<b>FAS_PosTableReadItem</b>	To read item values of RAM area of the specific position table
<b>FAS_PosTableWriteItem</b>	To save item values of the specific position table to RAM area
<b>FAS_PosTableWriteROM</b>	To save all of position table values to ROM area : Total 256 PT values are saved.
<b>FAS_PosTableReadROM</b>	To read all of position table values of ROM area : Total 256 PT values are read.
<b>FAS_PosTableRunItem</b>	The motor starts to run from the designated position table in sequence.
<b>FAS_PosTableReadOneItem</b>	Reads the RAM area value of a specific item in a specific position table.
<b>FAS_PosTableWriteOneItem</b>	Saves the RAM area value of a specific item in a specific position table.

## FAS\_PosTableReadItem

---

To read a specific item in the position table

Syntax

```
int FAS_PosTableReadItem(
    int 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
    int iBdID = 0;      // A unique board number
    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, &nodelItem);
    _ASSERT(nRtn == FMM_OK);

    nodelItem.lPosition = 260000; // Change the position value to 260000.
    nodelItem.wBranch = 23;        // Set next command to No.23.
    nodelItem.wContinuous = 1;     // Next command should be connected
                                   // without deceleration.

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

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

    // Save 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_PosTableWriteItem(
    int 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 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 : wItemNo is out of range.

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

### 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 in ROM area

### Syntax

```
int FAS_PosTableWriteROM(  
    int 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 written.

### 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(  
    int 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(  
    int 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

---

To read the value of a specific item in the Position Table

Syntax

```
int FAS_PosTableReadOneItem(
    int 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 the value of a specific item in the Position Table

### Syntax

```
int FAS_PosTableWriteOneItem(  
    int 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 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.

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 - 1 1 . Ethernet Address Related Function

Function Name	Description
<b>FAS_GetEthernetAddr</b>	To read Gateway, Subnet Mask, IP of the product
<b>FAS_SetEthernetAddr</b>	To save Gateway, Subnet Mask, IP to settings and Flashrom Saved value(address) is applied after turing the product off and on again.
<b>FAS_GetMACAddress</b>	To read MAC Address of the product.

## FAS\_GetEthernetAddr

---

To read Gateway, Subnet Mask, IP of the product

### Syntax

```
int FAS_GetEthernetAddr(  
    int iBdID,  
    unsigned long* gateway,  
    unsigned long* subnet,  
    unsigned long* ip  
);
```

### Parameters

*iBdID*

The ID number of the board. iBdID set by FAS\_Connect function

*gateway*

Gateway Value to set

*subnet*

Subnet mask Value to set

*ip*

IP address Value to 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

If Gateway is 192.168.0.1, return value is : 0x0100a8c0

If Subnet Mask is 255.255.255.0, return value is : 0x00ffffff

If IP Address is 192.168.0.2, return value is : 0x0200a8c0

### Example

### See Also

FAS\_SetEthernetAddr

## FAS\_SetEthernetAddr

---

To save and configure Gateway, Subnet Mask, IP Address to the board(flashrom)

### Syntax

```
int FAS_SetEthernetAddr(
    int iBdID,
    unsigned long gateway,
    unsigned long subnet,
    unsigned long ip
);
```

### Parameters

*iBdID*

The ID number of the board. iBdID set by FAS\_Connect function

*gateway*

Gateway Value to set

*subnet*

Subnet mask Value to set

*ip*

IP address Value to 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

If Gateway is set to 192.168.10.1, set value is : 0x010aa8c0

If Subnet Mask is set to 255.255.255.0, set value is : 0x00ffffff

If IP Address is set to 192.168.10.2, set value is : 0x020aa8c0

### Example

### See Also

FAS\_GetEthernetAddr

## FAS\_GetMACAddress

---

To read Gateway, Subnet Mask, IP Address of the board

### Syntax

```
int FAS_GetMACAddress(  
    int iBdID,  
    unsigned long long* MACAddress,  
);
```

### Parameters

*iBdID*

The ID number of the board. iBdID set by FAS\_Connect function

*MACAddress*

MAC Address 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

If MAC Address is 00-04-A3-0E-6A-00, return value is : 0x0000**0004a30e6a00**

### Example

### See Also

FAS\_GetEthernetAddr

## 2 - 1 2 . Other Control Function

Function Name	Description
<b>FAS_TriggerOutput_RunA</b>	Function to generate an output signal at a specific location
<b>FAS_TriggerOutput_Status</b>	Function to check whether output signal(COMP) is generated
<b>FAS_MovePush</b>	Function to move from a specific position holding the power
<b>FAS_GetPushStatus</b>	Function to check the current push motion status



## FAS\_TriggerOutput\_RunA

---

To start/stop the digital output signal (COMP pin) at a specific position during operation by position command.

### Syntax

```
int FAS_TriggerOutput_RunA(  
    int 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 signal output function is working or not

Syntax

```
int FAS_TriggerOutput_Status(  
    int 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\_MovePush

It moves while maintaining a fixed force from a specific position during movement by position command, and stops moving when it touches work during movement, but keeps the force.

### Syntax

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

### Parameters

*iBdID*

The ID number of the board. iBdID set by FAS\_Connect function.

*DWORD dwStartSpd*

Start speed of position command

*DWORD dwMoveSpd*

Operation speed of position command

*long lPosition*

Target absolute position value of position command

*WORD wAccel*

Acceleration time of position command

*WORD wDecel*

Deceleration time of position command

*WORD wPushRate*

Push rate during Push operation

*DWORD dwPushSpd*

Operation speed during Push operation

*long lEndPosition*

Target absolute position value during Push operation

*WORD wPushMode*

Set the mode during Push operation

### 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 iBdlID does not exist.

Remarks

Example

See Also

FAS\_GetPushStatus

## FAS\_GetPushStatus

---

To check the progress of the current push motion

### Syntax

```
int FAS_MovePush(  
int iBdID,  
BYTE* nPushStatus  
);
```

### Parameters

*iBdID*

The ID number of the board. iBdID set by FAS\_Connect function.

*BYTE\* nPushStatus*

Push motion status value to be read.

(Refer to '[1-2-1. Frame Type Data Configuration](#)'.)

### 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\_Move Push

### 3. Appendix – Network Information Setting with DHCP

#### 3 - 1 . DHCP Function

- 1) DHCP(Dynamic Host Configuration Protocol)란?

Standard network protocol used to dynamically configure network information for performing TCP / IP communication, such as IP address.

(Network information : Gateway, Subnet, IP address)

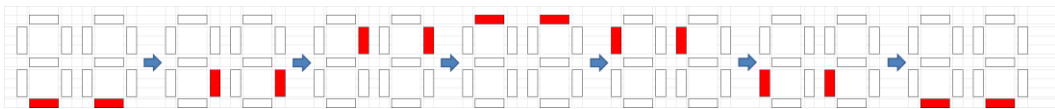
- 1) When not using DHCP

→ If do not use a Gateway, Subnet, IP address set as drive standard, change and save setting using GUI and need to know current network information.

→ When using DHCP, Gateway, Subnet, IP address are automatically set in product. It is necessary to save network information which is automatically set using GUI.

#### 3 - 2 . Network Setting Using DHCP (Plus-E series)

- 1) Set IP setting switch (SW1, SW2) as F,F.
- 2) Connect Ethernet at Ethernet IN Connector
- 3) Power ON
- 4) 7-segment flashes as below



- 5) When the network information is set, the IP address is displayed on the 7-segment (After displaying aaa.bbb.ccc.ddd, displaying Hex.value corresponding to ddd)
- 6) Power OFF after accessing GUI and saving network information (Using Config Slave ID / IP Address)
- 7) Do not overlap the value of IP setting switch(SW1, SW2) with Gateway at 1~254
- 8) Power ON(Setting finished)

● DHCP automatically configures network information, so every time applying power, IP address – ddd in aaa.bbb.ccc.ddd can be changed. Therefore, after setting Network information by DHCP method, 6) must be performed.

● Network information can be set using DHCP only when using a PC or a router with a DHCP server function.



**FASTECH Co., Ltd.**

Rm#1202, 401-dong, Bucheon Techno-Park,  
655, Pyeongcheon-ro, Bucheon-si Gyeonggi-do,  
Republic of Korea (Zip:14502)  
TEL : +82-32-234-6300 FAX : +82-32-234-6302  
E-mail : [fastech@fastech.co.kr](mailto:fastech@fastech.co.kr)  
Homepage : [www.fastech.co.kr](http://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-SERVOII Plus-E MINI is a registered trademark of FASTECH Co.,Ltd. In the national registry.

© Copyright 2020 FASTECH Co.,Ltd. Mar 25, 2021 Rev.02