Ezi-SERVO® II Plus-E ALL

Closed Loop Stepping System

사용자 매뉴얼

통신 기능편

(Rev.02)

FASTECH

목 차

1. 통신 프로토콜

1 - 1 . 통신 기능

Ezi-SERVOII Plus-E ALL 은 Ehternet 통신을 이용하여 최대 254 축(1~254)까지 제어가 가능합니다.

1 - 1 - 1 . 통신 사양

항 목	사 양
통신 속도	10/100base-T/TX
통신 방식(Protocol)	TCP (Port No. : **2001,2002**)
	UDP (Port No. : **3001,3002**)
최대 배선 길이	100m 이내
드라이브간 최소 배선 길이	20cm 이상
접속 축수	254 축 (No. 01~FE)

- Port No. 2001, 3001 : GUI 용
- Port No. 2002, 3002 : 사용자 Library 용
- 제공되는 사용자 Library 파일을 사용 시에 Port No. 2001, 3001 은 사용할 수 없습니다.

1 - 1 - 2 . Ethernet IP 주소

1) Subnet Mask : 255.255.255.0
2) Gateway : 192.168.0.1
3) IP address : 192.168.0.x (x 는 외부 스위치에 의해 설정)

- PC 또는 Ethernet 장치에서 직접 Ezi-SERVOII Plus-E ALL 에 연결할 경우 반드시 Network 설정을 위의 IP 주소에 준하여 설정하여 주십시오.
  설정이 안되었거나 상이할 경우 연결할 수 없습니다.
- 스위치를 255(FF)로 설정하면 IP Address 는 자동으로 설정됩니다.
  DHCP 를 사용하기 때문에 공유기를 사용할 경우에만 IP Address 가 자동으로 설정됩니다.
- DHCP 서버 기능이 없는 제어기(PC, PLC 등)에서 직접 연결할 경우에는 반드시 스위치로 IP Address 를 설정하십시오.
- 기본 IP Address 를 사용하지 않을 경우에만 IP Address 를 자동으로 설정하십시오.
  자동으로 IP 가 설정 되면 사용자 프로그램(GUI)를 접속하여 IP Address 를 저장한 후에 전원을 차단하고 스위치로 IP 의 마지막 번호를 설정하십시오
- IP 설정 스위치를 "00"으로 하면 IP 설정이 상기 값으로 초기화됩니다.

1 - 1 - 3 . Ethernet 프로토콜

1) 통신 FRAME 의 개요

2) FRAME 의 기본 구조

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

UDP Header 에는 다음의 정보가 포함됩니다.

① 송신측 포트 번호 : 2 bytes
② 수신측 포트 번호 : 2 bytes
③ 데이터 길이 : 2 bytes, UDP Header 와 Frame Data 의 총 길이
④ 체크섬 : 2 bytes

1 - 1 - 4 . 수신 Frame Data 구조

수신 ***Frame Data*** 항의 세부 구성은 다음과 같습니다.

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, Frame 의 시작을 표시
② Length : Length 이후의 Data 의 총길이
(Sync No. + Reserved + Frame type + Data)
③ Reserved : 1 byte ("0x00" 로 입력)
④ Sync No. : 해당 packet 의 일련번호로서 명령이 드라이브 모듈에서 실행되었는지의 여부를 확인하기 위함입니다. 새로운 명령을 보낼 때마다 이 값은 변경되어야 합니다.
⑤ Frame type : 해당 Frame 의 명령어 종류를 지정합니다. 그 종류는 아래의 「1-2. Frame type 과 Data 구성」절을 참조하십시오.
⑥ Data : 이 항의 데이터 구조 및 길이 등은 Frame type 에 따라 정해집니다. 그 자세한 구조는 아래의 「1-2. Frame type 과 Data 구성」절을 참조하십시오.

1 - 1 - 5 . 답신 Frame Data 구조와 통신 에러

송신측 명령에 대한 답신측의 Frame 기본 구조는 동일합니다. 다만 아래의 ***Frame Data*** 항에서의 차이점은 '통신 상태'가 추가되어 다음과 같습니다.

Header	Length	Sync No.	Reserved	Frame type	Data	
1 byte	1 byte	1 byte	1 byte (0x00)	1 byte	1 byte	0 ~ 252 bytes
					통신 상태	답신 Data

① Header : 0xAA, Frame 의 시작을 표시함.
② Length : Length 이후의 Data 의 총길이
(Sync No. + Reserved + Frame type + Data)
③ Sync No. : 답신 Frame 과 동일

(수신시의 데이터와 일치하지 않으면 에러 상태로 인식하십시오.)

④ Reserved : 1 byte(0x00)

⑤ Frame type : 수신 Frame 과 동일
(송신시의 데이터와 일치하지 않으면 에러 상태로 인식하십시오.)

⑥ Data : 답신 때에는 통신 상태(에러 / 정상)를 나타내는 1 byte 의 Data 가 포함됩니다.
단순 실행 명령에는 통신 상태 Data 만 있습니다.

통신 상태를 나타내는 byte 의 값에 대한 내용은 아래의 표와 같습니다.

Hexa 값	Decimal 값	내 용
0x00	0	통신이 정상 상태입니다.
0x80	128	Frame type 에러 : 수신한 Frame type 명령을 인식할 수 없습니다.
0x81	129	데이터 에러, ROM 데이터 읽기, 쓰기 에러 : 수신한 데이터의 값이 정해진 범위 외의 데이터입니다.
0x82	130	수신 Frame 에러 : 수신된 Frame 이 규격에 맞지 않는 데이터입니다.
0x85	133	운전 명령 실패 : 다음과 같은 상태에서 새로운 운전을 실행하려고 했습니다. 1) 현재 모터가 운전 중 2) 정지 명령 중 3) Servo OFF 상태 4) 외부 엔코더 없이 Z-pulse Origin 을 시도 5) 기타 잘못된 운전 명령
0x86	134	Reset 실패 : 다음과 같은 상태에서 Reset 을 실행하려고 했습니다. 1) Servo ON 상태 2) 외부 입력 신호에 의해 이미 Reset 상태임
0x87	135	Servo ON 실패 ① : 알람 발생중에 Servo ON 명령을 실행하려고 했습니다.
0x88	136	Servo ON 실패 ② : 비상 정지중에 Servo ON 명령을 실행하려고 했습니다.
0x89	137	Servo ON 실패 ③ : 외부 입력 신호에 'Servo ON'이 설정되었습니다. 이 입력 신호로만 Servo ON/OFF 를 실행할 수 있습니다.

⚠ 주의	**1) 수신 Frame 의 'Header'와 'Length'값이 정상이 아니면, 드라이브 측에서 답신을 하지 않습니다.** **2) 통신 상태 '130'번의 에러가 발생한 경우에는 답신 데이터의 크기는 0 byte 입니다.**

1 - 2 . Frame 의 구성

1 - 2 - 1 . Frame type 별 송수신 내용

(1) 다음 표는 Frame type 값에 따른 Data 항의 내용과 구성에 대한 설명입니다.

● Frame type 의 0xXX 는 Hex. 값이며 ()안의 값은 Dec.입니다.

<table>
<tr><th>Frame type</th><th>라이브러리 명</th><th>내용</th></tr>
<tr><td>0x01
(1)</td><td>FAS_
GetboardInfo</td><td>연결된 slave 의 종류와 프로그램 Version 정보를 요청.

송신 : 0 byte
답신 : 1~248 bytes

<table><tr><td>1 byte</td><td>1 byte</td><td>0~253 bytes</td></tr><tr><td>통신 상태</td><td>board 종류</td><td>ACII string with NULL byte
(strlen() + 1 byte)</td></tr></table>
◆ board 종류 : 91 (Ezi-SERVOII Plus-E ALL)</td></tr>
<tr><td>0x05
(5)</td><td>FAS_
GetMotorInfo</td><td>board 에 연결된 모터의 종류에 대한 정보를 요청.

송신 : 0 byte
답신 : 1~246 bytes

<table><tr><td>1 byte</td><td>1 byte</td><td>0~246 bytes</td></tr><tr><td>통신 상태</td><td>Motor 번호
(1~255)</td><td>ACII string with NULL byte
(strlen() + 1 byte)</td></tr></table>
◆ Motor 종류 : 「1-2-7. 모터 종류」의 표를 참조.</td></tr>
<tr><td>0x10
(16)</td><td>FAS_
SaveAllParameters</td><td>현재 설정된 파라미터 값과 입출력 신호의 할당값들을 드라이브의 ROM 메모리에 저장함. 이는 드라이브 전원을 OFF 해도 해당 값들이 저장되어 있도록 합니다.
따라서 'FAS_SetParameter'와 'FAS_SetIOAssignMap'에서 설정된 값들이 함께 저장됨.

송신 : 0 byte
답신 : 1 byte

<table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table></td></tr>
</table>

<table>
<tr>
<td>0x11
(17)</td>
<td>FAS_
GetRomParameter</td>
<td>ROM 메모리의 특정 파라미터 값을 읽어들임.

송신 : 1 byte

<table>
<tr><td>1 byte</td></tr>
<tr><td>파라미터 번호 (0~32)</td></tr>
</table>

답신 : 5 bytes

<table>
<tr><td>1 byte</td><td>4 bytes</td></tr>
<tr><td>통신 상태</td><td>파라미터 값</td></tr>
</table>

「1-2-2. 파라미터 목록표」를 참조.</td>
</tr>
<tr>
<td>0x12
(18)</td>
<td>FAS_
SetParameter</td>
<td>특정 파라미터 값을 드라이브의 RAM 메모리에 저장함.

송신 : 5 bytes

<table>
<tr><td>1 byte</td><td>4 bytes</td></tr>
<tr><td>파라미터 번호 (0~32)</td><td>파라미터 값</td></tr>
</table>

답신 : 1 byte

<table>
<tr><td>1 byte</td></tr>
<tr><td>통신 상태</td></tr>
</table>

「1-2-2. 파라미터 목록표」를 참조.</td>
</tr>
<tr>
<td>0x13
(19)</td>
<td>FAS_
GetParameter</td>
<td>RAM 메모리의 특정 파라미터 값을 읽어들임.

송신 : 1 byte

<table>
<tr><td>1 byte</td></tr>
<tr><td>파라미터 번호 (0~32)</td></tr>
</table>

답신 : 5 bytes

<table>
<tr><td>1 byte</td><td>4 bytes</td></tr>
<tr><td>통신 상태</td><td>파라미터 값</td></tr>
</table>

「1-2-2. 파라미터 목록표」를 참조.</td>
</tr>
<tr>
<td>0x20
(32)</td>
<td>FAS_
SetIOOutput</td>
<td>제어 출력단의 출력 신호 레벨을 설정함.

송신 : 8 bytes
<table>
<tr><td>4 bytes</td><td>4 bytes</td></tr>
<tr><td>I/O set mask 값</td><td>I/O clear mask 값</td></tr>
</table>

Set mask 의 특정 bit 값이 1 이면 해당 출력단 신호는 [ON]이 된다.
Clear mask 의 특정 bit 값이 1 이면 해당 출력단 신호는 [OFF]가 된다.
자세한 사항은「1-2-3. 제어 출력핀의 bit 설정」항을 참조.

답신 : 1 byte

<table>
<tr><td>1 byte</td></tr>
<tr><td>통신 상태</td></tr>
</table>
</td>
</tr>
</table>

<table>
<tr>
<td>0x21
(33)</td>
<td>FAS_
SetIOInput</td>
<td>제어 입력단의 입력 신호 레벨을 설정함.

송신 : 8 bytes
<table>
<tr><td>4 bytes</td><td>4 bytes</td></tr>
<tr><td>I/O set mask 값</td><td>I/O clear mask 값</td></tr>
</table>

Set mask 의 특정 bit 값이 1 이면 해당 입력단 신호는 [ON]이 된다.
Clear mask 의 특정 bit 값이 1 이면 해당 입력단 신호는 [OFF]가 된다.
자세한 사항은 「1-2-4. 제어 입력핀의 bit 설정」항을 참조.

답신 : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>통신 상태</td></tr>
</table>
</td>
</tr>
<tr>
<td>0x22
(34)</td>
<td>FAS_
GetIOInput</td>
<td>제어 입력단의 현재 입력 신호 상태를 읽어들임.

송신 : 0 byte

답신 : 5 bytes
<table>
<tr><td>1 byte</td><td>4 bytes</td></tr>
<tr><td>통신 상태</td><td>입력 상태 값</td></tr>
</table>

각 입력 신호별 해당 bit 는 「1-2-4. 제어 입력핀의 bit 설정」항을 참조.</td>
</tr>
<tr>
<td>0x23
(35)</td>
<td>FAS_
GetIOOutput</td>
<td>제어 출력단의 현재 출력 신호 상태를 읽어들임.

송신 : 0 byte

답신 : 5 bytes
<table>
<tr><td>1 byte</td><td>4 bytes</td></tr>
<tr><td>통신 상태</td><td>출력 상태 값</td></tr>
</table>

각 출력 신호별 해당 bit 는 「1-2-3. 제어 출력핀의 bit 설정」항을 참조.</td>
</tr>
<tr>
<td>0x24
(36)</td>
<td>FAS_
SetIOAssignMap</td>
<td>제어 입출력 신호를 CN1 단자대의 pin 에 할당하며 동시에 신호 level 을 설정해 줍니다. 이 설정값을 ROM 메모리에 저장하기 위해서는 'FAS_SaveAllParameters'를 실행함.

송신 : 6 bytes
<table>
<tr><td>1 byte</td><td>4 bytes</td><td>1 byte</td></tr>
<tr><td>I/O number</td><td>I/O pin masking data</td><td>설정 level</td></tr>
</table>
◆I/O number : '0~5'은 각각 'Limit+,Limit-,Org,IN1,...,IN3'에 해당됨.
'6,7'는 각각 'COMP, OUT1'에 해당됨.
◆I/O pin masking data : 「1-2-4. 제어 입력핀의 bit 설정」항을 참조.
◆설정 level : 0:Active Low, 1:Active High
답신 : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>통신 상태</td></tr>
</table>
</td>
</tr>
</table>

<table>
<tr>
<td>0x25
(37)</td>
<td>FAS_
GetIOAssignMap</td>
<td>CN1 단자대의 pin 의 설정 상태를 읽어들임.

송신 : 1 byte

<table>
<tr><td>1 byte</td></tr>
<tr><td>I/O number</td></tr>
</table>

◆I/O number : '0~5'은 각각 'Limit+, Limit-, Org, IN1, ..., IN3'에 해당됨.
'6,7'는 각각 'COMP, OUT1'에 해당됨.

답신 : 6 bytes
<table>
<tr><td>1 byte</td><td>4 bytes</td><td>1 byte</td></tr>
<tr><td>통신 상태</td><td>IO pin masking 상태</td><td>Level 상태</td></tr>
</table>

자세한 사항은 위의 '0x24'Frame type 항을 참조하십시오.</td>
</tr>
<tr>
<td>0x26
(38)</td>
<td>FAS_
IOAssignMapReadROM</td>
<td>제어 입출력 신호의 설정 상태와 신호의 레벨 설정값을 ROM 메모리에서 읽어들임.

송신 : 0 byte
답신 : 2 bytes
<table>
<tr><td>1 byte</td><td>1 byte</td></tr>
<tr><td>통신 상태</td><td>명령 수행 여부 (0: 완료, 0 이외의 값: 에러)</td></tr>
</table>
</td>
</tr>
<tr>
<td>0x27
(39)</td>
<td>FAS_
TriggerOutput_RunA</td>
<td>제어 출력 신호(Compare Out)를 발생시키기 위한 명령.

송신 : 18 bytes
<table>
<tr><td>1 byte</td><td>4 bytes</td><td>4 bytes</td></tr>
<tr><td>출력 시작/종료 명령
(1:시작 0:종료)</td><td>출력 시작 위치
[pulse]</td><td>Pulse 주기
[pulse]</td></tr>
</table>

<table>
<tr><td>4 bytes</td><td>1 byte</td><td>4 bytes</td></tr>
<tr><td>Pulse 폭
[msec]</td><td>출력 pin 번호
(0 으로 고정됨)</td><td>여분</td></tr>
</table>
◆출력 시작 위치 : 신호를 출력하기 위한 최초의 시작 위치값
(-134,217,728 ~134,217,727)
◆Pulse 주기 : 출력되는 신호의 주기 설정함
(0 : 펄스가 시작 위치에서 1 회만 출력됨
1~134,217,727 : 펄스 주기에 따라 반복 출력됨)
◆Pulse 폭 : 출력되는 신호의 폭을 설정함 (1 ~1000)
답신 : 2 bytes
<table>
<tr><td>1 byte</td><td>1 byte</td></tr>
<tr><td>통신 상태</td><td>명령 수행 여부 (0: 완료, 0 이외의 값: 에러)</td></tr>
</table>

● Pulse 주기는 시간으로 계산하여 Pulse 폭과 합한 값이 2[ms]이상이 되는 값을 입력하십시오. 그 이하일 경우 정상적인 동작이 되지 않습니다.</td>
</tr>
</table>

<table>
<tr>
<td>0x28
(40)</td>
<td>FAS_
TriggerOutput_
Status</td>
<td>현재 신호(Compare Out)출력 기능이 작동중인지 여부를 확인하는 명령.

송신 : 0 byte
답신 : 2 bytes

<table>
<tr><td>1 byte</td><td>1 byte</td></tr>
<tr><td>통신 상태</td><td>현재 상태 (1: 출력 중, 0 : 종료)</td></tr>
</table>
</td>
</tr>
<tr>
<td>0x7E
(126)</td>
<td>FAS_SetTriggerOutputEx</td>
<td>설정된 출력에 특정 위치에서 출력을 발생시키기 위한 설정
(출력 신호를 User Out 으로 설정한 후에 사용 가능)
송신 : 245 bytes
<table>
<tr><td>1 byte</td><td>1 byte</td><td>2 bytes</td><td>1 byte</td></tr>
<tr><td>User Out 번호
(0~8)</td><td>출력 시작/종료 명령
(1: 시작 0: 종료)</td><td>출력 On 시간
(ms 단위, 1~65,535)</td><td>출력위치
개수</td></tr>
</table>
<table>
<tr><td>240 bytes</td></tr>
<tr><td>출력 위치 Array(4 bytes * 60)
위치: -134,217,728~134,217,727</td></tr>
</table>
◆ 출력 위치 개수 : 1~60
◆ 출력 위치 Array : 4 bytes 의 위치로 60 개 Array
출력 위치 개수가 60 개가 아니더라도 출력 위치 Array 에는 60 개의 정보 입력
◆ 출력 위치 개수만큼 출력이 되면 자동으로 종료.
◆ 출력하기 위해서는 설정 후 이동 명령을 실행.
이동 명령의 위치가 반드시 출력 위치 개수의 마지막 위치가 양수이면 보다 커야하고 음수이면 보다는 작아야함
시작 위치(현재 위치)에 따라 정상적인 출력이 안되는 경우가 발생되므로 적정한 값으로 출력 위치 설정이 필요함.
구동 속도와 출력 On 시간 설정에 따라서 정상적인 출력이 안되는 경우가 발생되므로 적정한 값으로 설정이 필요함.

답신 : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>통신 상태</td></tr>
</table>
</td>
</tr>
<tr>
<td>0x7F
(127)</td>
<td>FAS_GetTriggerOutputEx</td>
<td>FAS_SetTriggerOutputEx 로 설정된 정보 및 출력 상태를 확인하는 명령

송신 : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>User Out 번호</td></tr>
</table>
◆ User Out 번호 : 정보를 확인할 User Out 번호(0~8)</td>
</tr>
</table>

<table>
<tr><td></td><td></td><td>수신 : 245 bytes
<table>
<tr><td>1 byte</td><td>1 byte</td><td>2 bytes</td><td>1 byte</td></tr>
<tr><td>통신 상태</td><td>출력 상태</td><td>출력 On 시간
(ms 단위, 1~65535)</td><td>출력 위치의 개수
(1~60)</td></tr>
</table>
<table>
<tr><td>240 bytes</td></tr>
<tr><td>출력 위치 Array(4 bytes * 60)
위치: -134,217,728~134,217,727</td></tr>
</table>
◆ 출력 상태 : 해당 User Out 번호의 Run/Stop 상태
0 : Stop
2 : Run</td></tr>
<tr><td>0x2A
(42)</td><td>FAS_
ServoEnable</td><td>Servo ON/OFF 상태를 설정함
송신 : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>0 : OFF, 1 : ON</td></tr>
</table>
답신 : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>통신 상태</td></tr>
</table></td></tr>
<tr><td>0x2B
(43)</td><td>FAS_
ServoAlarmReset</td><td>Servo Alarm 상태를 reset 시킴.
송신 : 0 byte
답신 : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>통신 상태</td></tr>
</table></td></tr>
<tr><td>0x2E
(46)</td><td>FAS_
GetAlarmType</td><td>현재 Alarm 상태 및 정보를 요청.
송신 : 0 byte
답신 : 2 bytes
<table>
<tr><td>1 byte</td><td>1 byte</td></tr>
<tr><td>통신 상태</td><td>Alarm 상태 (0: No alarm, 0 이외의 값: Alarm 번호)</td></tr>
</table>
◆ Alarm type: No alarm (0) OverCurrent(1) OverSpeed(2)
PosTracking(3) OverLoad(4) OverTemperature(5)
BackEMF(6) MotorConnect(7) EncoderConnect(8)
Inposition(10) ROMdevice(12) Position Overflow(15)</td></tr>
</table>

<table>
<tr><td>0x31
(49)</td><td>FAS_
MoveStop</td><td>현재 운전중인 모터의 정지를 요청.

송신 : 0 byte
답신 : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table></td></tr>
<tr><td>0x32
(50)</td><td>FAS_
EmergencyStop</td><td>현재 운전중인 모터의 비상 정지를 요청.

송신 : 0 byte
답신 : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table></td></tr>
<tr><td>0x33
(51)</td><td>FAS_
MoveOriginSingle
Axis</td><td>현재 설정된 파리미터의 조건으로 원점 복귀 운전 시작을 요청.

송신 : 0 byte
답신 : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table></td></tr>
<tr><td>0x34
(52)</td><td>FAS_
MoveSingleAxisAbs
Pos</td><td>절대값[pulse] 위치만큼의 이동 운전 시작을 요청.

송신 : 8 bytes
<table><tr><td>4 bytes</td><td>4 bytes</td></tr><tr><td>절대 위치값</td><td>운전 속도[pps]</td></tr></table>답신 : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table></td></tr>
<tr><td>0x35
(53)</td><td>FAS_
MoveSingle
AxisIncPos</td><td>상대값[pulse] 위치만큼의 이동 운전 시작을 요청.

송신 : 8 bytes
<table><tr><td>4 bytes</td><td>4 bytes</td></tr><tr><td>상대 위치값</td><td>운전속도[pps]</td></tr></table>답신 : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table></td></tr>
<tr><td>0x36
(54)</td><td>FAS_
MoveToLimit</td><td>현재 설정된 파리미터의 조건으로 Limit 운전 시작을 요청.

송신 : 5 bytes</td></tr>
</table>

<table>
<tr><td></td><td></td><td>
<table>
<tr><td>4 bytes</td><td>1 byte</td></tr>
<tr><td>운전 속도[pps]</td><td>운전 방향 (0: -Limit 1: +Limit)</td></tr>
</table>

답신 : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>통신 상태</td></tr>
</table>
</td></tr>
<tr><td>0x37
(55)</td><td>FAS_
MoveVelocity</td><td>현재 설정된 파리미터의 조건으로 Jog 운전 시작을 요청.

송신 : 5 bytes
<table>
<tr><td>4 bytes</td><td>1 byte</td></tr>
<tr><td>운전 속도[pps]</td><td>운전 방향 (0: -Jog 1: +Jog)</td></tr>
</table>

답신 : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>통신 상태</td></tr>
</table>
</td></tr>
<tr><td>0x38
(56)</td><td>FAS_
PositionAbsOverrid
e</td><td>운전중인 상태에서 목표 절대 위치값[pulse]의 변경을 요청.

송신 : 4 bytes
<table>
<tr><td>4 bytes</td></tr>
<tr><td>변경된 목표 위치값[pulse]</td></tr>
</table>

답신 : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>통신 상태</td></tr>
</table>
◆ 등속 구간에서만 가능</td></tr>
<tr><td>0x39
(57)</td><td>FAS_
PositionIncOverride</td><td>운전중인 상태에서 목표 상대 위치값[pulse]의 변경을 요청.
송신 : 4 bytes
<table>
<tr><td>4 bytes</td></tr>
<tr><td>변경된 목표 위치값[pulse]</td></tr>
</table>

답신 : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>통신 상태</td></tr>
</table>
◆ 등속 구간에서만 가능</td></tr>
</table>

<table>
<tr>
<td>0x3A
(58)</td>
<td>FAS_
VelocityOverride</td>
<td>
운전중인 상태에서 운전 속도값[pps]의 변경을 요청.

송신 : 4 bytes
<table>
<tr><td>4 bytes</td></tr>
<tr><td>변경된 운전 속도[pps]</td></tr>
</table>

속도 변경 시 적용되는 가감속값은 파라미터의 'Axis Acc Time'과
'Axis Dec Time'값으로 지정된다.

답신 : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>통신 상태</td></tr>
</table>

◆ 등속 구간에서만 가능
◆ 지령 속도(정지상태에서의 이동 지령 속도)에 따라 변경 가능한 속도 범위를 확인하시고 함수를 사용하시기 바랍니다. 아래 표의 내용을 참고 하십시오.
<table>
<tr><td rowspan="2">지령 속도
[pps]</td><td colspan="2">속도 변경 범위[pps]</td></tr>
<tr><td>최소</td><td>최대</td></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>163841~327680</td><td>100</td><td>1638300</td></tr>
<tr><td>327681~655360</td><td>200</td><td>3276600</td></tr>
</table>
속도 변경 범위에 들어 있지 않은 값으로 설정할 경우 범위 내에서의 최소값 또는 최대 값의 속도로 변경됩니다.
</td>
</tr>
<tr>
<td>0x80
(128)</td>
<td>FAS_
MoveSingleAxisAbs
PosEx</td>
<td>
가감속값[msec]을 포함한 절대값[pulse] 위치만큼의 이동 운전 시작을 요청.

송신 : 40 bytes
<table>
<tr><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td><td>2 bytes</td></tr>
<tr><td>절대 위치값</td><td>운전 속도[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>

Flag ooption :
</td>
</tr>
</table>

<table>
<tr><td></td><td></td><td>0x0001 : reserved
0x0002 : Custom Accel. Time 값을 적용함.
0x0004 : Custom Decel. Time 값을 적용함.

해당 bit 값이 OFF 상태(0)인 경우에는 컨트롤러 내부에 지정된 값이 적용됩니다.

답신 : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table></td></tr>
<tr><td>0x81
(129)</td><td>FAS_
MoveSingle
AxisIncPosEx</td><td>가감속값[msec]을 포함한 상대값[pulse] 위치만큼의 이동 운전 시작을 요청.

송신 : 40 bytes
<table><tr><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td><td>2 bytes</td></tr><tr><td>상대 위치값</td><td>운전 속도[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>Flag option : 0x0001 : Reserved
0x0002 : Custom Accel. Time 값을 적용함.
0x0004 : Custom Decel. Time 값을 적용함.
해당 bit 값이 OFF 상태(0)인 경우에는 컨트롤러 내부에 저장된 값이 적용됩니다.

답신 : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table></td></tr>
<tr><td>0x82
(130)</td><td>FAS_
MoveVelocityEx</td><td>가감속값[msec]을 포함한 Jog 운전 시작을 요청.

송신 : 37 bytes
<table><tr><td>4 bytes</td><td>1 byte</td><td>4 bytes</td></tr><tr><td>운전 속도[pps]</td><td>운전 방향 (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>Flag option : 0x0001 : Reserved
0x0002 : Custom Accel./Decel. Time 값을 적용함.

해당 bit 값이 OFF 상태(0)인 경우에는 컨트롤러 내부에 지정된 값이 적용됩니다.
답신 : 1 byte</td></tr>
</table>

<table>
<tr>
<td>0x40
(64)</td>
<td>FAS_
GetAxisStatus</td>
<td>운전 상태를 표시해주는 Flag 값을 요청.

송신 : 0 byte
답신 : 5 bytes
<table>
<tr><td>1 byte</td><td>4 bytes</td></tr>
<tr><td>통신 상태</td><td>상태 Flag 값</td></tr>
</table>
각 Flag 별 해당 bit 는 「1-2-5. 상태 Flag 의 bit 설정」의 표를 참조.</td>
</tr>
<tr>
<td>0x41
(65)</td>
<td>FAS_
GetIOAxisStatus</td>
<td>제어 입출력 상태와 운전 Flag 상태를 요청.
(Frame type 0x22, 0x23, 0x40 을 묶어 놓은 것)

송신 : 0 byte
답신 : 13 bytes
<table>
<tr><td>1 byte</td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td></tr>
<tr><td>통신 상태</td><td>입력 상태 값</td><td>출력 상태 값</td><td>상태 Flag 값</td></tr>
</table>
</td>
</tr>
<tr>
<td>0x42
(66)</td>
<td>FAS_
GetMotionStatus</td>
<td>현재 운전 진행 상황 및 운전중인 PT 번호를 요청.
(Frame type 0x51, 0x53, 0x54, 0x55 를 묶어 놓은 것임)

송신 : 0 byte
답신 : 21 byte
<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>통신
상태</td><td>Command
Position 값</td><td>Actual
Position 값</td><td>Position
차이 값</td><td>운전
속도값</td><td>현재 운전중인
PT 번호</td></tr>
</table>
</td>
</tr>
<tr>
<td>0x43
(67)</td>
<td>FAS_
GetAllStatus</td>
<td>현재의 운전 상태를 모두 포함하여 요청. (Frame type 0x41, 0x42 를 묶어 놓은 것)

송신 : 0 byte
답신 : 33 bytes
<table>
<tr><td>1 byte</td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td></tr>
<tr><td>통신 상태</td><td>입력 상태 값</td><td>출력 상태 값</td><td>상태 Flag 값</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 값</td><td>Actual
Position 값</td><td>Position
차이 값</td><td>운전
속도값</td><td>현재 운전중인
PT 번호</td></tr>
</table>
</td>
</tr>
<tr>
<td>0x50
(80)</td>
<td>FAS_
SetCommandPos</td>
<td>목표 위치값(Command position)을 운전 시작 전에 설정하면, 추후 목표 위치값의 변화를 확인할 수 있다.

송신 : 4 bytes
<table>
<tr><td>4 bytes</td></tr>
<tr><td>목표 위치 설정 count 값</td></tr>
</table>

답신 : 1 byte</td>
</tr>
</table>

<table>
<tr><td></td><td></td><td><table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table></td></tr>
<tr><td>0x51
(81)</td><td>FAS_
GetCommandPos</td><td>현재 추종 중인 목표 위치(Command position) 값[pulse]을 요청.

송신 : 0 byte
답신 : 5 bytes
<table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>통신 상태</td><td>목표 위치값</td></tr></table></td></tr>
<tr><td>0x52
(82)</td><td>FAS_
SetActualPos</td><td>Ezi-SERVOⅡ Plus-E ALL 은 폐루프 제어 방식이므로, 운전중 실제 위치값이 계속 갱신된다. 이 실제 위치값(Actual position)을 운전 시작전에 이 값으로 설정하면 추후 실제 위치값의 변화를 확인할 수 있다.

송신 : 4 bytes
<table><tr><td>4 bytes</td></tr><tr><td>실제 위치 count 값</td></tr></table>
답신 : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table></td></tr>
<tr><td>0x53
(83)</td><td>FAS_
GetActualPos</td><td>현재의 실제 위치(Actual position) 값[pulse]을 요청.
송신 : 0 byte

답신 : 5 bytes
<table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>통신 상태</td><td>실제 위치값</td></tr></table></td></tr>
<tr><td>0x54
(84)</td><td>FAS_
GetPosError</td><td>현재의 목표 위치(Command position)값과 실제 위치(Actual position)값의 차이 값[pulse]을 요청.

송신 : 0 byte

답신 : 5 byte
<table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>통신 상태</td><td>위치 차이 값</td></tr></table>이 값으로 현재의 운전 상태(Inposition 추종 정도)를 확인할 수 있음.</td></tr>
<tr><td>0x55
(85)</td><td>FAS_
GetActualVel</td><td>현재의 운전 속도값[pps]을 요청.

송신 : 0 byte
답신 : 5 bytes
<table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>통신 상태</td><td>속도값</td></tr></table></td></tr>
</table>

<table>
<tr>
<td>0x56
(86)</td>
<td>FAS_
ClearPosition</td>
<td>목표 위치값(Command position)과 실제 위치값(Actual position)을 모두 '0'으로 설정.
송신 : 0 byte
답신 : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table></td>
</tr>
<tr>
<td>0x58
(88)</td>
<td>FAS_
MovePause</td>
<td>현재의 운전 상태를 일시 정지 및 일시 정지 해제로 요청

송신 : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>0: 일시 정지 해제, 1: 일시 정지</td></tr></table>
답신 : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table></td>
</tr>
<tr>
<td>0x60
(96)</td>
<td>FAS_
PosTableReadItem</td>
<td>드라이브의 RAM 메모리의 PT 항목 값들을 읽어들임.

송신 : 2 bytes
<table><tr><td>2 bytes</td></tr><tr><td>읽어들일 PT 번호(0~255)</td></tr></table>
답신 : 65 bytes
<table><tr><td>1 byte</td><td>64 bytes</td></tr><tr><td>통신 상태</td><td>해당 PT 의 항목 값</td></tr></table>
각 PT 별 항목 내용은 「1-2-6. 포지션 테이블 항목」참조.</td>
</tr>
<tr>
<td>0x61
(97)</td>
<td>FAS_
PosTableWriteItem</td>
<td>드라이브의 RAM 메모리에 PT 항목 값들을 저장함.

송신 : 66 bytes
<table><tr><td>2 bytes</td><td>64 bytes</td></tr><tr><td>PT 번호(0~255)</td><td>해당 PT 의 항목 값</td></tr></table>
각 PT 별 항목 내용은 「1-2-6. 포지션 테이블 항목」참조.

답신 : 2 bytes
<table><tr><td>1 byte</td><td>1 byte</td></tr><tr><td>통신 상태</td><td>명령 수행 여부 (0 이외의 값: 완료, 0: 에러)</td></tr></table></td>
</tr>
</table>

<table>
<tr><td>0x62
(98)</td><td>FAS_
PosTableReadROM</td><td>드라이브의 ROM 메모리의 모든(256 개) PT 항목 값을 읽어들임.

송신 : 0 byte

답신 : 2 bytes
<table><tr><td>1 byte</td><td>1 byte</td></tr><tr><td>통신 상태</td><td>명령 수행 여부 (0: 완료, 0 이외의 값: 에러)</td></tr></table></td></tr>
<tr><td>0x63
(99)</td><td>FAS_
PosTableWriteROM</td><td>드라이브의 ROM 메모리에 모든(256 개) PT 항목 값들을 저장함.

송신 : 0 byte

답신 : 2 bytes
<table><tr><td>1 byte</td><td>1 byte</td></tr><tr><td>통신 상태</td><td>명령 수행 여부 (0: 완료 0 이외의 값: 에러)</td></tr></table></td></tr>
<tr><td>0x64
(100)</td><td>FAS_
PosTableRunItem</td><td>지정된 PT 번호에서부터 포지션 테이블 운전을 시작함.

송신 : 2 bytes
<table><tr><td>2 bytes</td></tr><tr><td>PT 번호(0~255)</td></tr></table>
답신 : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table></td></tr>
<tr><td>0x6A
(106)</td><td>FAS_
PosTableReadOneItem</td><td>드라이브의 RAM 메모리에 PT 항목 중 특정 값을 읽어들임.

송신 : 4 bytes
<table><tr><td>2 bytes</td><td>2 bytes</td></tr><tr><td>읽어들일 PT 번호(0~255)</td><td>읽어들일 특정 항목의 Offset 값(0~40)</td></tr></table>특정 항목 Offset 값은 「1-2-6. 포지션 테이블 항목」참조.

답신 : 5 bytes
<table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>통신 상태</td><td>해당 항목의 값</td></tr></table></td></tr>
<tr><td>0x6B
(107)</td><td>FAS_
PosTableWriteOneItem</td><td>드라이브의 RAM 메모리의 PT 항목 중 특정 값을 저장함.

송신 : 8 bytes
<table><tr><td>2 bytes</td><td>2 bytes</td><td>4 bytes</td></tr><tr><td>저장 할
PT 번호(0~255)</td><td>저장 할 특정 항목의
Offset 값(0~40)</td><td>저장값</td></tr></table>특정 항목 Offset 값은 「1-2-6. 포지션 테이블 항목」참조.

답신 : 2 bytes</td></tr>
</table>

<table>
<tr><td></td><td></td><td>
<table>
<tr><td>1 byte</td><td>1 byte</td></tr>
<tr><td>통신 상태</td><td>명령 수행 여부 (0 이외의 값: 완료, 0: 에러)</td></tr>
</table>
</td></tr>
<tr><td>0x78
(120)</td><td>FAS_
MovePush</td><td>
정해진 힘을 유지하기 위한 push motion 운전 시작을 요청.

송신 : 28 bytes
<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>위치 이동
시작 속도값</td><td>위치 이동
운전 속도값</td><td>위치 이동
절대 위치값</td><td>위치 이동
가속 시간</td><td>위치 이동
감속 시간</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 이동
torque 비율</td><td>push 이동
운전 속도값</td><td>push 이동
절대 위치값</td><td>Push mode</td></tr>
</table>
위치 이동 시작 속도값 : 1~35,000[pps]
위치 이동 운전 속도값 : 1~500,000[pps]
위치 이동 절대 위치값 : -134,217,728~134,217,727
위치 이동 가속 시간, 위치 이동 감속 시간 : 1~9,999[ms]
push 이동 torque 비율 : 20~90[%]
push 이동 운전 속도값 : 1~33,333[pps] (최대 200[rpm]) (분해능 : 10,000)
push 이동 절대 위치값 : -134,217,728~134,217,727
push mode : 0(stop mode), 1~10,000(non-stop mode)

자세한 사항은「사용자 매뉴얼 본문편 8-6. Push Motion 기능」참조.

답신 : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>통신 상태</td></tr>
</table>
◆ 다음 모션 명령 전에 반드시 Stop(E-Stop)명령을 실행
</td></tr>
<tr><td>0x79
(121)</td><td>FAS_
GetPushStatus</td><td>
현재의 push motion 운전 상태의 확인을 요청.

송신 : 0 byte
답신 : 1 byte
<table>
<tr><td>1 byte</td><td>1 byte</td></tr>
<tr><td>통신 상태</td><td>Push motion 상태
0: 일반 position 위치 이동 대기 상태
1: push motion 중이며, work 는 접촉하지 않은 상태
2: work 에 접촉되었고, 힘이 유지되고 있는 상태
3: push motion 이 완료되었지만 work 를 접촉하지 못한 상태. 이 경우에는 다음 모션 명령 전에 반드시 Stop(E-Stop) 명령을 실행.</td></tr>
</table>
</td></tr>
</table>

- Frame Type '0x65 ~ 0x69', '0x90 ~ 0x92'는 내부 사용 목적으로 할당되어 있습니다.

1 - 2 - 2 . 파라미터 목록표

번호	이름	단위	하한	상한	출하치
0	Pulse Per Revolution		0	8	8
1	Axis Max Speed	[pps]	1	500,000	500,000
2	Axis Start Speed	[pps]	1	35,000	1
3	Axis Acc Time	[msec]	1	9,999	100
4	Axis Dec Time	[msec]	1	9,999	100
5	Speed Override	[%]	1	500	100
6	Jog Speed	[pps]	1	500,000	5,000
7	Jog Start Speed	[pps]	1	500,000	1
8	Jog Acc Dec Time	[msec]	1	9,999	100
9	S/W Limit Plus Value	[pulse]	-134,217,728	134,217,727	134,217,727
10	S/W Limit Minus Value	[pulse]	-134,217,728	134,217,727	-134,217,728
11	S/W Limit Stop Method		0	2	2
12	H/W Limit Stop Method		0	1	0
13	Limit Sensor Logic		0	1	0
14	Org Speed	[pps]	1	500,000	5,000
15	Org Search Speed	[pps]	1	50,000	1,000
16	Org Acc Dec Time	[msec]	1	9,999	50
17	Org Method		0	7	0
18	Org Dir		0	1	1
19	Org OffSet	[pulse]	-134,217,728	134,217,727	0
20	Org Position Set	[pulse]	-134,217,728	134,217,727	0
21	Org Sensor Logic		0	1	0
22	Position Loop Gain		0	127	4
23	Inpos Value		0	63	0
24	Pos Tracking Limit	[pulse]	1	134,217,727	5,000
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	5,000
29*1	Brake Delay Time	[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

*1 86[mm] 모터용 드라이브에는 해당 파라미터는 사용되지 않습니다.

- 44 번의 파라미터는 Firmware [ver.6.1.20.18]부터 사용 됩니다.

1 - 2 - 3 . 제어 출력핀의 bit 설정

'0x20' 번 Frame type 에 대한 세부 설명입니다.
이 명령은 제어 출력단 24 개 신호 종류 중 「User Output0」~「User Output8」까지 9 개의 신호에 대해서만 적용됩니다. 나머지 15 개의 출력 신호는 사용자가 임의로 조작할 수 없으며, 드라이브 운전중 해당 상황이 발생했을 때 출력 신호를 내보냅니다.
다음표는 각 신호별 bit mask 값을 표시합니다.

신호명	해당 bit 위치	신호명	해당 bit 위치	신호명	해당 bit 위치
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	Brake	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

【예 1】User Output 5 의 단자대를 ON 시키고자 할 경우의 송신 데이터입니다.

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

【예 2】User Output 5 의 단자대를 OFF 시키고자 할 경우의 송신 데이터입니다.

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

1 - 2 - 4 . 제어 입력핀의 bit 설정

'0x21' 번 Frame type 에 대한 세부 설명입니다.

이 명령은 제어 입력단 32 개의 신호에 대해 적용됩니다. 실제 입력 신호가 없는 상태에서 신호가 입력된 것처럼 시험용으로 사용할 수 있습니다.

다음 표는 각 신호별 bit mask 값을 표시합니다.

신호명	해당 bit 위치	신호명	해당 bit 위치	신호명	해당 bit 위치	신호명	해당 bit 위치
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

【예 1】Pause 의 단자대를 ON 시키고자 할 경우의 송신 데이터입니다.

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

【예 2】Pause 의 단자대를 OFF 시키고자 할 경우의 송신 데이터입니다.

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

⚠ 주의	**'PT A5 ~ PT A7'의 기능과 'User IN6 ~IN8', 'Jog0~Jog2 의 기능을 동시에 사용할 수 없습니다.**

1 - 2 - 5 . 상태 Flag 의 bit 설정

include file 중 'MOTION_EziSERVO2_DEFINE.h' 참조

Flag define 명	내용	해당 bit 위치
FFLAG_ERRORALL	여러 에러 중 하나 이상의 에러가 발생한 경우.	0X00000001
FFLAG_HWPOSILMT	+방향 Limit 센서가 ON 이 된 경우.	0X00000002
FFLAG_HWNEGALMT	-방향 Limit 센서가 ON 이 된 경우.	0X00000004
FFLAG_SWPOGILMT	+방향 프로그램 Limit 를 초과한 경우.	0X00000008
FFLAG_SWNEGALMT	-방향 프로그램 Limit 를 초과한 경우.	0X00000010
Reserved1		0X00000020
Reserved2		0X00000040
FFLAG_ERRPOSOVERFLOW	위치 명령 완료후 위치 오차가 'Pos Error Overflow Limit' 값보다 크게 발생한 경우.	0X00000080
FFLAG_ERROVERCURRENT	모터 구동 소자에 과전류 이상이 발생한 경우 Alarm 발생.	0X00000100
FFLAG_ERROVERSPEED	모터의 속도가 3000[rpm] 초과한 경우 Alarm 발생.	0X00000200
FFLAG_ERRPOSTRACKING	위치 명령 중 위치 오차가 'Pos Tracking Limit'값보다 크게 발생한 경우.	0X00000400
FFLAG_ERROVERLOAD	모터의 최대 토크를 초과하는 부하가 5 초 이상 또는 10 바퀴 이상의 거리에 가해졌을 경우 Alarm 발생.	0X00000800
FFLAG_ERROVERHEAT	드라이브의 내부 온도가 85°C 를 초과하는 경우 Alarm 발생.	0X00001000
FFLAG_ERRBACKEMF	모터의 역기전력 전압이 70V 를 초과하는 경우 Alarm 발생.	0X00002000
FFLAG_ERRMOTORPOWER	모터 전압 이상일 경우 Alarm 발생.	0X00004000
FFLAG_ERRINPOSITION	Inposition 이상일 경우 Alarm 발생.	0X00008000
FFLAG_EMGSTOP	모터가 비상 정지 상태일 경우.	0X00010000
FFLAG_SLOWSTOP	모터가 일반 정지 상태일 경우.	0X00020000
FFLAG_ORIGINRETURNING	원점 복귀 운전 중 일 경우.	0X00040000
FFLAG_INPOSITION	Inposition 이 완료된 상태일 경우.	0X00080000
FFLAG_SERVOON	모터가 Servo ON 상태일 경우.	0X00100000
FFLAG_ALARMRESET	Alarm Reset 명령이 실시된 상태일 경우.	0X00200000
FFLAG_PTSTOPED	포지션 테이블 운전이 종료된 상태일 경우.	0X00400000
FFLAG_ORIGINSENSOR	원점 센서가 ON 되어 있는 상태일 경우.	0X00800000
FFLAG_ZPULSE	원점 복귀 운전 중 z-pulse 방식의 동작인 경우.	0X01000000
FFLAG_ORIGINRETOK	원점 복귀 운전이 완료 된 상황일 경우.	0X02000000
FFLAG_MOTIONDIR	모터의 운전 방향 (+방향: OFF, -방향: ON)	0X04000000
FFLAG_MOTIONING	모터가 현재 운전중일 경우.	0X08000000
FFLAG_MOTIONPAUSE	운전중 Pause 명령으로 정지 상태일 경우.	0X10000000
FFLAG_MOTIONACCEL	가속 구간의 운전중일 경우.	0X20000000
FFLAG_MOTIONDECEL	감속 구간의 운전중일 경우.	0X40000000
FFLAG_MOTIONCONST	가/감속 구간이 아닌 정속도 운전중인 상태일 경우.	0X80000000

1 - 2 - 6 . 포지션 테이블 항목

include file 중 'motion_define.h' 참조

명칭	구조체의 변수 명칭	Byte 수	Offset 위치	단위	하한	상한
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		

항목별 설정 방법에 대한 사항은 별책「사용자 매뉴얼 포지션 테이블편」을 참조하십시오.

1 - 3 . 프로그램의 종류

Ezi-SERVOII Plus-E ALL 을 사용하기 위한 프로그램 방법은 두 가지가 있습니다.

첫 번째는 일반적으로 사용하는 방법으로서 PC 의 window system 에서 Visual C++ 언어를 사용하는 방법입니다.
이 때에는 제품과 함께 제공된 라이브러리 (「2. PC 프로그램용 라이브러리」를 참조)를 사용합니다.

두 번째는 라이브러리 함수를 사용하지 않고 사용자가 직접 명령어 (command character)를 전송하는 방법입니다. Protocol Test 프로그램과 같이 low level 의 protocol program 을 사용자가 직접 작성해야 하며, 주로 상위 제어기로 PLC 등을 사용할 경우에 적용됩니다.

2. PC 프로그램용 라이브러리

2 - 1 . 라이브러리의 구성

(1) C++용

C++ header file(*.h)와 library file(*.lib or *.dll) 이 필요합니다. 이 파일들은 "₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include₩₩에 있으며 개발용 source file 에 다음 내용을 포함 시키십시오.

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

또한 라이브러리 파일은 다음과 같습니다.

1) 32bit 용

"₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include₩₩EziMotionPlusE.lib"
"₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include₩₩EziMotionPlusE.dll"

2) 64bit 용

"₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include_x64₩₩EziMotionPlusE.lib"
"₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include_x64₩₩EziMotionPlusE.dll"

이 라이브러리를 사용한 sample program source 등이 "₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩Examples₩₩C++₩₩"폴더에 포함되어 있습니다.

(2) C#용

C# header file 와 library file 이 필요합니다. 이 파일들은 "₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩의 하위폴더에 있으며 개발용 source file 에 다음 내용을 포함 시키십시오.

#include "₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include₩₩MOTION_DEFINE_PlusE.cs"

또한 라이브러리 파일은 다음과 같습니다.

1) 32bit 용

"₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include₩₩LIB_EziMOTIONPlusE.cs"

2) 64bit 용

"₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include_x64₩₩LIB_EziMOTIONPlusE.cs"

이 라이브러리를 사용한 sample program source 등이 "₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩Examples₩₩C#₩₩"폴더에 포함되어 있습니다.

(3) 다음의 표는 각 라이브러리 함수 사용 시 리턴되는 값들에 대한 설명입니다. 이 리턴 값들은 라이브러리(DLL) 함수에서만 확인할 수 있고 프로토콜을 사용한 프로그램 방식에서는 지원되지 않습니다.

구분	명칭	리턴값	내용
정상	FMM_OK	0(0x00)	함수가 정상적으로 명령을 수행하였습니다.
입력 에러	FMM_NOT_OPEN	1(0x01)	잘못된 Port 번호를 입력하였습니다.
	FMM_INVALID_PORT_NUM	2(0x02)	연결되지 않은 Port 번호입니다.
	FMM_INVALID_SLAVE_NUM	3(0x03)	잘못된 Board 번호를 입력하였습니다.
운전 에러	FMM_POSTABLE_ERROR	9(0x09)	포지션 테이블 읽기/쓰기 중 에러가 발생 하였습니다.
연결 에러	FMC_DISCONNECTED	5(0x05)	해당 Board 가 연결 해제 되었습니다.
	FMC_TIMEOUT_ERROR	6(0x06)	정해진 시간(100msec)동안 응답이 없습니다.
	FMC_RECVPACKET_ERROR	8(0x08)	드라이브로부터 받은 패킷에서 프로토콜 레벨의 에러가 발생하였습니다.

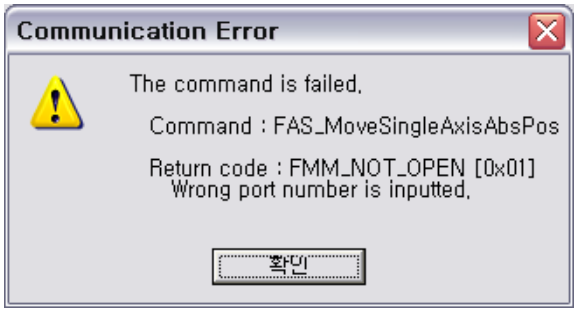
(4) 다음의 표는 모든 라이브러리에 공통적으로 포함되는 리턴값으로서 드라이브에서 판단한 결과(통신 상태, 운전 상태)를 확인할 수 있는 기능입니다. 라이브러리(DLL)를 사용하는 경우와 프로토콜을 사용한 프로그래밍 모두에 지원 됩니다.

구분	명칭	리턴값	내용
정상	FMP_OK	0(0x00)	통신이 정상적으로 수행되었습니다.
입력 에러	FMP_FRAMETYPEERROR	128(0x80)	Board 가 인식하지 못하는 명령어 입니다.
	FMP_DATAERROR	129(0x81)	입력한 데이터가 범위를 벗어났습니다.
운전 에러	FMP_RUNFAIL	133(0x85)	모터가 이미 운전중이거나 운전할 수 있는 준비가 되어 있지 않는 등 잘못된 명령입니다.
	FMP_RESETFAIL	134(0x86)	Servo ON 상태에서 Alarm Reset 명령을 실행할 수 없습니다.
	FMP_SERVOONFAIL1	135(0x87)	Alarm 이 발생한 상태입니다.
	FMP_SERVOONFAIL2	136(0x88)	Emergency Stop 중 입니다.
	FMP_SERVOONFAIL3	137(0x89)	입력 신호에 'Servo ON'이 이미 설정되었습니다.
연결 에러	FMP_PACKETERROR	130(0x82)	드라이브가 받은 패킷에서 프로토콜 레벨의 에러가 발생하였습니다.

2 - 2 . 통신 상태 표시용 창

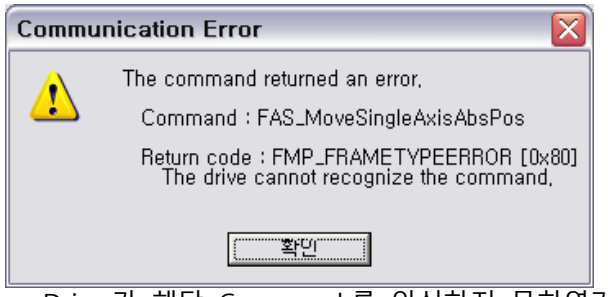
위의 통신 상태는 크게 3 가지로 분류할 수 있다.

(1) Communication Error



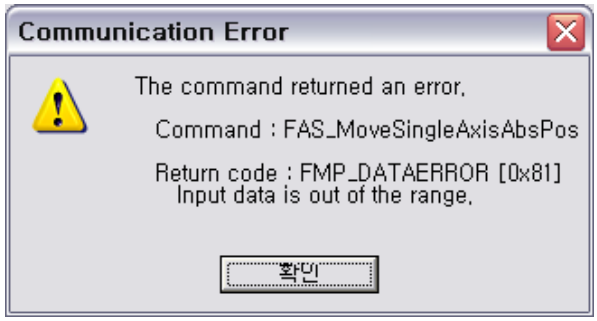
FMM_NOT_OPEN,

COM Port 가 연결되지 않았습니다. (GUI 에서는 발생할 수 없는 에러 입니다.)



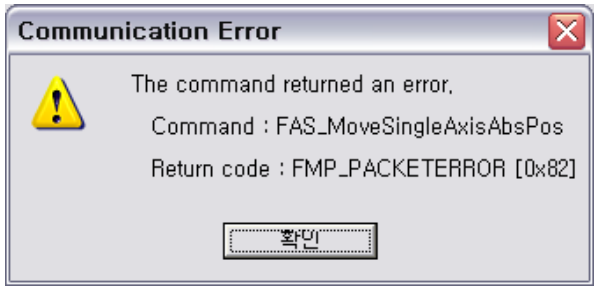
FMP_FRAMETYPEERROR = 0x80,

Drive 가 해당 Command 를 인식하지 못하였거나, 지원하지 않는 Command 입니다.



FMP_DATAERROR,

입력된 데이터의 범위가 Drive 가 지원하는 범위를 벗어났습니다.



FMP_PACKETERROR,

수신된 Frame 이 규격에 맞지 않는 데이터입니다. (Drive 로 보낸 Packet 의 길이가 일치하지 않습니다.)

(2) Wrong Command

FMP_RUNFAIL

운전 명령 실패: 다음과 같은 상태에서 새로운 운전을 실행하려고 했습니다.

- 현재 모터가 운전 중
- 정지 명령 중
- Servo OFF 상태
- 외부 엔코더 없이 Z-pulse Origin 을 시도
- 기타 잘못된 운전 명령

FMP_RESETFAIL,

다음과 같은 상태에서 Reset 을 실행하려고 했습니다.

- Servo ON 상태
- 외부 입력 신호에 의해 이미 Reset 상태.

FMP_SERVOONFAIL1,

알람 발생 중에 Servo ON 명령을 실행하려고 했습니다.

FMP_SERVOONFAIL2,

비상 정지 중에 Servo Off 명령을 실행하려고 했습니다.

FMP_SERVOONFAIL3,

Servo ON Signal 이 외부 Input 에 할당되어 있어 실행할 수 없습니다.

(3) Command Execution Error

FMM_POSTABLE_ERROR

,

Position Table 관련 함수의 실행 실패.

2 - 3 . 드라이브 연결함수

함수명	내용
FAS_Connect	드라이브와 UDP Protocol 로 연결을 시도합니다. : 성공적으로 접속했다면 TRUE 를, 접속에 실패를 했다면 FALSE 를 리턴합니다.
FAS_ConnectTCP	드라이브 모듈과 TCP Protocol 로 연결을 시도합니다. : 성공적으로 접속했다면 TRUE 를, 접속에 실패를 했다면 FALSE 를 리턴합니다.
FAS_Reconnect	기존의 IP, Protocol, iBdID 로 다시 연결합니다.
FAS_AutoReconnect	TCP 를 사용할 경우 응답이 100[ms]이내에 없거나, 의도하지 않게 TCP 가 disconnect 가 되었을 경우, 자동으로 다른 port 로 connect 하고 응답받지 못한 함수를 다시 실행합니다.
FAS_Close	드라이브와 통신 해지를 시도합니다.
FAS_GetSlaveInfo	드라이브의 종류와 프로그램 Version 을 읽어들입니다. : 드라이브 종류와 Version 정보를 리턴합니다.
FAS_GetMotorInfo	드라이브에 연결된 모터의 종류와 제조사에 대한 정보를 읽어들입니다.
FAS_IsSlaveExist	해당 드라이브의 존재 여부를 확인합니다. : 존재하면 TRUE 를, 접속에 실패를 했다면 FALSE 를 리턴합니다.
FAS_IsBdIDExist	해당 IP Address 에 BdID 의 사용 여부를 확인합니다. : 존재하면 TRUE 를, 접속에 실패를 했다면 FALSE 를 리턴합니다.
FAS_IsIPAddressExist	해당 BdID 에 IP Address assign 여부를 확인합니다. : 존재하면 TRUE 를, 접속에 실패를 했다면 FALSE 를 리턴합니다.
FAS_EnableLog	통신 오류 관련 Log 의 출력을 제어합니다. : 존재하면 TRUE 를, 접속에 실패를 했다면 FALSE 를 리턴합니다.
FAS_SetLogPath	출력될 Log 가 저장될 경로를 설정합니다. : 존재하면 TRUE 를, 접속에 실패를 했다면 FALSE 를 리턴합니다.
FAS_SetLogLevel	Log 를 설정된 Level 에 따라 출력합니다. : 기본으로 내부 통신 오류 관련 Log 만 출력하게 설정되어 있습니다. (LOG_LEVEL_COMM)
FAS_PrintCustomLog	임의의 Log 를 출력 합니다.

- **1. 다음의 함수는 F/W Ver V06.01.020.04 Library Ver 2.0.0.10 이상에서 지원합니다**
  1) **FAS_ConnectTCP**
  2) **FAS_Reconnect**
  3) **FAS_SetLogLevel**
  4) **FAS_PrintCustomLog**

2. **다음의 함수는 F/W Ver V06.01.020.05 Library Ver 2.3.0.15 이상에서 지원합니다**

1) FAS_SetAutoReconnect

FAS_Connect

FAS_Connect 함수는 Ezi-SERVOⅡ Plus-E ALL 에 UDP Protocol 으로 접속하는 함수입니다.

Syntax

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

Parameters

sb1~4

접속하려는 드라이브의 IP 주소를 입력합니다.

ex) 192.168.0.2 일경우

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

iBdID

접속하는 Board 의 고유 ID 입니다. 사용자가 설정하는 ID(값)입니다.

IP 주소와 같이 동일한 ID 사용이 불가합니다.

Return Value

성공적으로 접속했다면 TRUE 를, 접속에 실패를 했다면 FALSE 를 리턴합니다.

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 // 192.168.0.2 의 Board 고유 번호
        char lpBuff[256];
        int nBuffSize = 256;
        BYTE nType;
        int nRtn;

        // 연결합니다.
        if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
        {
                // 연결이 실패하였습니다.
                MessageBox(_T("연결 실패!"));
                return;
```

```
        }

        if (FAS_IsSlaveExist(iBdID) == FALSE)
        {
                // 해당 board 번호는 존재하지 않습니다.
                // Ezi-SERVOII의 board 번호를 확인하십시오.
                return;
        }

        nRtn = FAS_GetSlaveInfo(iBdID, &nType, lpBuff, nBuffSize);
        if (nRtn != FMM_OK)
        {
                // 명령이 정상적으로 수행되지 않았습니다.
                // ReturnCodes_Define.h 를 참조하십시오.
        }

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

        // 연결을 종료합니다.
        FAS_Close(iBdID);
}
```

See Also

FAS_Close

FAS_ConnectTCP

FAS_Connect 함수는 Ezi-SERVOⅡ Plus-E ALL 에 TCP Protocol 으로 접속하는 함수입니다.

Syntax

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

Parameters

sb1~4

접속하려는 드라이브의 IP 주소를 입력합니다.

ex) 192.168.0.2 일경우

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

iBdID

접속하는 Board 의 고유 ID 입니다. 사용자가 설정하는 ID(값)입니다.

IP 주소와 같이 동일한 ID 사용이 불가합니다.

Return Value

성공적으로 접속했다면 TRUE 를, 접속에 실패를 했다면 FALSE 를 리턴합니다.

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 // 192.168.0.2 의 Board 고유 번호
        char lpBuff[256];
        int nBuffSize = 256;
        BYTE nType;
        int nRtn;

        // 연결합니다.
        if (FAS_ConnectTCP(sb1, sb2, sb3, sb4, iBdID) == FALSE)
        {
                // 연결이 실패하였습니다.
                MessageBox(_T("연결 실패!"));
                return;
```

```
        }

        if (FAS_IsSlaveExist(iBdID) == FALSE)
        {
                // 해당 board 번호는 존재하지 않습니다.
                // Ezi-SERVOII의 board 번호를 확인하십시오.
                return;
        }

        nRtn = FAS_GetSlaveInfo(iBdID, &nType, lpBuff, nBuffSize);
        if (nRtn != FMM_OK)
        {
                // 명령이 정상적으로 수행되지 않았습니다.
                // ReturnCodes_Define.h 를 참조하십시오.
        }

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

        // 연결을 종료합니다.
        FAS_Close(iBdID);
}
```

See Also

FAS_Close

FAS_Reconnect

FAS_Connect()를 사용하지 않고 재연결하는 함수입니다.

Syntax

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

Parameters

iBdID

다시 연결할 드라이브 ID 번호.

Remarks

FAS_Connect()함수로 연결된 후에 연결을 종료 시키거나 종료 되었을 때 다시 FAS_Connect()를 사용하지 않고 통신 연결하는 명령입니다.

Example

FAS_Connect 라이브러리 참조.

See Also

FAS_Connect

FAS_SetAutoReconnect

TCP 의 통신을 자동으로 connect 합니다.

Syntax

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

Parameters

bSET

Auto Reconnect 기능을 사용 여부를 설정합니다.

Remarks

FAS_SetAutoReconnect 를 Set 으로 설정한 후에 사용한 함수에서 응답이 100[ms] 내에 없거나, 의도하지 않게 TCP 가 disconnect 가 되었을 경우, 다른 port 를 이용하여 connect 하고 응답받지 못한 함수를 다시 실행합니다.
(FAS_ConnectTCP 를 이용하여 접속한 경우에만 정상적으로 실행됩니다.)

- 위의 기능을 사용하면서 GUI 를 접속할 경우에는 반드시 GUI 는 UDP 로 접속하십시오.

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcInit()
{
        BYTE sb1 = 192, sb2 = 168, sb3=0, sb4=2 // IP :192.168.0.2
        int iBdID = 0 // 192.168.0.2 의 Board 고유 번호
        char lpBuff[256];
        int nBuffSize = 256;
        BYTE nType;
        int nRtn;

        // 연결합니다.
        if (FAS_ConnectTCP(sb1, sb2, sb3, sb4, iBdID) == FALSE)
        {
                // 연결이 실패하였습니다.
                MessageBox(_T("연결 실패!"));
                return;
        }
        // Auto Reconnect 를 사용으로 설정합니다
        FAS_SetAutoReconnect(SET);
}
```

See Also

FAS_Close

해당 ID 의 통신 연결을 해제 합니다.

Syntax

```
void FAS_Close(int iBdID);
```

Parameters

iBdID

연결을 해제할 드라이브 ID 번호.

Remarks

Example

FAS_Connect 라이브러리 참조.

See Also

FAS_Connect

FAS_GetSlaveInfo

해당 Board 의 Version 정보 문자열을 받아옵니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

pType

해당 Board 의 Type 번호.

lpBuff

Version 정보 문자열을 입력받을 Buffer Pointer.

nBuffSize

lpBuff 의 메모리 할당 크기 값.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 Board 는 존재하지 않습니다.

Remarks

Example

FAS_Connect 라이브러리 참조.

See Also

FAS_GetMotorInfo

해당 Board 에 연결된 모터의 정보 문자열을 받아옵니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID.

pType

해당 Motor 의 Type 번호.

lpBuff

Motor 정보 문자열을 입력받을 Buffer Pointer.

nBuffSize

lpBuff 의 메모리 할당 크기 값.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.

FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 Board 는 존재하지 않습니다.

Remarks

Example

FAS_Connect 라이브러리 참조.

See Also

FAS_IsSlaveExist

현재 드라이브가 연결 상태인지를 확인합니다.

Syntax

```
BOOL FAS_IsSlaveExist(int iBdID);
```

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

Return Value

TRUE : 연결 상태.
FALSE : 해지 상태.

Remarks

이 함수는 라이브러리에서만 제공되며 프로토콜 프로그램 방식에는 지원되지 않습니다.

Example

FAS_Connect 라이브러리 참조.

See Also

FAS_Connect

FAS_IsBdIDExist

현재 드라이브가 연결 상태인지를 확인합니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

sb1, sb2, sb3, sb4

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

Return Value

TRUE : 해당 BdID 사용

FALSE : 해당 BdID 사용하지 않음

Remarks

이 함수는 라이브러리에서만 제공되며 프로토콜 프로그램 방식에는 지원되지 않습니다.

Example

FAS_Connect 라이브러리 참조.

See Also

FAS_Connect

FAS_IsIPAddressExist

현재 드라이브가 연결 상태인지를 확인합니다.

Syntax

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

Parameters

sb1, sb2, sb3, sb4

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

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

Return Value

TRUE : 해당 IP Address 사용.

FALSE : 해당 IP Address 사용하지 않음.

Remarks

이 함수는 라이브러리에서만 제공되며 프로토콜 프로그램 방식에는 지원되지 않습니다.

Example

FAS_Connect 라이브러리 참조.

See Also

FAS_Connect

FAS_EnableLog

통신 오류 관련 Log 의 출력을 제어합니다.

Syntax

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

Parameters

bEnable

Log 를 출력할지 설정합니다.

Remarks

현재 Process 에서 Ezi-MOTION Plus-E 함수를 사용하는 중에 발생하는 Log 의 출력을 제어합니다. 이 설정은 다른 Process 혹은 다른 프로그램의 Log 출력에 영향을 끼치지 않습니다.
Log 는 FAS_Connect 부터 발생하며, FAS_Close 를 하여 현재 연결된 드라이브의 접속을 해제하면 Log 의 출력은 종료됩니다. Log 출력의 기본 설정 값은 TRUE 입니다.

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcDisableLog()
{
        FAS_EnableLog(FALSE);

        // 이 후 함수들의 Log 는 출력되지 않습니다.

        BYTE sb1 = 192, sb2 = 168, sb3=0, sb4=2 // IP :192.168.0.2
        int iBdID = 0 // 192.168.0.2 의 Board 고유 번호

        // 연결을 시도합니다.
        if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
        {
                // 연결이 실패하였습니다.
                return;
        }

        // 연결을 종료합니다.
        FAS_Close(iBdID);
}
```

See Also

FAS_SetLogPath

FAS_SetLogPath

출력될 Log 가 저장될 경로를 설정합니다.

Syntax

BOOL FAS_SetLogPath(LPCTSTR lpPath);

Parameters

lpPath

Log 가 저장될 절대 경로 문자열.

Return Value

입력된 경로가 존재하지 않거나 접근할 수 없을 경우 FALSE 를 리턴합니다.

Remarks

이 함수는 FAS_Connect 함수를 사용하기 전에 호출되어야 합니다.
lpPath 값이 NULL 이거나 길이가 0 인 문자열을 입력할 경우 Log 경로는 현재 Ezi-MOTION Plus-E Library 를 사용하는 프로그램이 존재하는 폴더로 설정됩니다.
Log 경로의 기본값은 NULL 로서 현재 프로그램이 존재하는 폴더입니다.

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcEnableLog()
{
        BYTE sb1 = 192, sb2 = 168, sb3=0, sb4=2 // IP :192.168.0.2
        int iBdID = 0 // 192.168.0.2 의 Board 고유 번호

        // Log 를 출력합니다.
        FAS_EnableLog(TRUE);   // 사용하지 않아도 됩니다.

        if (!FAS_SetLogPath(_T("C:₩₩Logs₩₩"))) // C:₩Logs 폴더가 존재하여야 합니다.
        {
                // Log 경로가 존재하지 않습니다.
                Return;
        }

        // 모든 함수들의 Log 는 C:₩Logs 폴더에 출력됩니다.

        // 연결을 시도합니다.
        if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
        {
                // 연결이 실패하였습니다.

                return;
        }

        // 연결을 종료합니다.
        FAS_Close(iBdID);
}
```

See Also

FAS_EnableLog

FAS_SetLogLevel

출력될 Log 가 저장될 경로를 설정합니다.

Syntax

BOOL FAS_SetLogLevel(enum LOG_LEVEL level);

Parameters

level

Log 출력의 범위 설정

Return Value

단계 설정값 이외의 값을 입력시에 FALSE 를 리턴합니다.

Remarks

LOG_LEVEL_COMM : 통신 오류 관련 Log 만 출력합니다.

LOG_LEVEL_PARAM : 위의 Log 출력에 Parameter 설정 함수 Log 가 추가로 출력 됩니다.

LOG_LEVEL_MOTION : 위의 Log 출력에 Motion 명령 함수 Log 가 추가로 출력 됩니다.

LOG_LEVEL_ALL : 출력될수 있는 모든 Log 가 출력됩니다.

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcEnableLog()
{
        BYTE sb1 = 192, sb2 = 168, sb3=0, sb4=2 // IP :192.168.0.2
        int iBdID = 0 // 192.168.0.2 의 Board 고유 번호

        // Log 를 출력합니다.
        FAS_EnableLog(TRUE);   // 사용하지 않아도 됩니다.

        FAS_SetLogLevel(LOG_LEVEL_ALL); // Log 출력 범위를 설정합니다.

        // 연결을 시도합니다.
        if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
        {
                // 연결이 실패하였습니다.

                return;
        }

        // 연결을 종료합니다.
        FAS_Close(iBdID);
}
```

See Also

FAS_EnableLog

FAS_PrintCustomLog

출력될 Log 가 저장될 경로를 설정합니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

level

Log 출력의 범위 설정

lpszMsg

출력될 Log 의 문자열

Return Value

단계 설정값 이외의 값을 입력시에 FALSE 를 리턴합니다.

Remarks

Level 은 FAS_SetLogLevel()의 설정 값(범위)와 동일
사용자 프로그램에서 프로그램 내의 특정 위치(함수)에서 Log 를 출력할 때 사용
또는, 프로그램 내에서 FAS_SetLogLevel()의 설정값과 다르게 log 출력할 때 사용

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcCustomLog()
{
        int iBdID = 0 // 192.168.0.2 의 Board 고유 번호
        int level = LOG_LEVEL_PARAM;

        //통신 오류와 파라미터 설정 함수 Log 출력 설정
        FAS_PrintCustomLog (iBdID, level, lpszMsg );

}
```

See Also

FAS_ SetLogLevel

2 - 4 . 파라미터 제어 함수

함수명	내용
FAS_SaveAllParameters	현재 상태의 Parameter 들을 ROM 에 저장합니다 : 운전 속도, 가감속 시간, 원점 복귀 관련 등의 파라미터를 드라이브 전원 OFF 후에도 보존되도록 저장합니다.
FAS_SetParameter	지정한 Parameter 값을 RAM 에 저장합니다 : 특정 파라미터 값을 저장합니다.
FAS_GetParameter	지정한 Parameter 값을 RAM 에서 읽어들입니다 : 특정 파라미터 값을 읽어 들입니다.
FAS_GetROMParameter	지정한 Parameter 값을 ROM 에서 읽어들입니다 : ROM 의 특정 파라미터 값을 읽어 들입니다.

FAS_SaveAllParameters

현재까지 수정된 모든 Parameter 값과 입출력 신호의 할당 값들을 ROM 에 저장합니다.

Syntax

```
Int FAS_SaveAllParameters(
    int iBdID
);
```

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : Board 를 연결하기 전입니다.

FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

현재의 파라미터 값들 외에 'FAS_SetIOAssignMap' 라이브러리로 설정된 값도 함께 ROM 메모리에 저장됩니다.

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcModifyParameter()
{
        BYTE sb1 = 192, sb2 = 168, sb3=0, sb4=2 // IP :192.168.0.2
        int iBdID = 0;          //  Board 의 고유 번호

        long lParamVal;
        int nRtn;

        // 연결합니다.
        if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
        {
                // 연결이 실패하였습니다.

                return;
        }

        // Axis Start Speed Parameter 값을 확인합니다.
        nRtn = FAS_GetParameter(iBdID, SERVO_AXISSTARTSPEED, &lParamVal);
        if (nRtn != FMM_OK)
        {
                // 명령이 정상적으로 수행되지 않았습니다.
                // ReturnCodes_Define.h 를 참조하십시오.
                _ASSERT(FALSE);
        }
        else
        {
                // 현재 Ezi-SERVOⅡ에 저장되어 있는 Parameter 값 입니다.
```

```
            printf("Parameter [before] : Start Speed = %d ₩n", lParamVal);
        }


        // Start Speed Parameter 값을 200 으로 변경한 후 다시 값을 읽어봅니다.
        nRtn = FAS_SetParameter(iBdID, SERVO_AXISSTARTSPEED, 200);
        _ASSERT(nRtn == FMM_OK);   // 명령이 정상적으로 수행되지 않았다면 멈춥니다.

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


        // ROM 에 저장되어 있는 값을 확인합니다.
        nRtn = FAS_GetROMParameter(iBdID, SERVO_AXISSTARTSPEED, &lParamVal);
        _ASSERT(nRtn == FMM_OK);   // 명령이 정상적으로 수행되지 않았다면 멈춥니다.
        printf("Parameter [ROM] : Start Speed = %d ₩n", lParamVal);


        // Parameter 값을 수정 한 후 ROM 에 저장합니다.
        nRtn = FAS_SetParameter(iBdID, SERVO_AXISSTARTSPEED, 100);
        _ASSERT(nRtn == FMM_OK);   // 명령이 정상적으로 수행되지 않았다면 멈춥니다.

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

        // 연결을 종료합니다.
        FAS_Close(iBdID);
}
```

See Also

FAS_GetRomParameter

FAS_SetParameter

해당 Parameter 값을 수정(RAM 에 저장)합니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID .

iParamNo

수정 하려는 Parameter 의 번호.

lParamValue

수정하려는 Parameter 값.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.

FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 Board 는 존재하지 않습니다.
FMM_INVALID_PARAMETER_NUM : 지정한 iParamNo 의 Parameter 는 존재하지 않습니다.

Remarks

지정된 하나의 파라미터에 대해서만 적용합니다.
드라이브상의 파라미터는 2개의 메모리에 저장하고 있습니다. 즉 ROM에는 전원 오프 시 영구히 저장하는 파라미터이고, 전원 온 시 ROM 의 파라미터를 DSP 의 RAM 상으로 복사하여 사용합니다. 사용자가 파라미터 변경 시 ROM 의 파라미터가 변경되는 것이 아니고, RAM 상의 파라미터가 변경됩니다. 이 함수는 RAM 에서 지정된 번호의 파라미터를 해당 값으로 설정합니다.

Example

FAS_SaveAllParameter 라이브러리 참조

See Also

FAS_GetParameter

FAS_GetParamater

Board 의 특정 Parameter 값을 불러옵니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID .

iParamNo

가져오려는 Parameter 의 번호.

lParamValue

Parameter 값.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.
FMM_INVALID_PARAMETER_NUM : 지정한 iParamNo 의 Parameter 는 존재하지 않습니다.

Remarks

지정된 하나의 파라미터에 대해서만 적용합니다.
드라이브상의 파라미터는 2개의 메모리에 저장되어 있습니다. 즉 ROM에는 전원 오프 시 영구히 저장하는 파라미터이고, 전원 온 시 ROM 의 파라미터를 DSP 의 RAM 상으로 복사하여 사용합니다. 사용자가 파라미터 변경 시 ROM 의 파라미터가 변경되는 것이 아니고, RAM 상의 파라미터가 변경됩니다. 이 함수는 RAM 에 지정된 번호의 파라미터를 읽어들입니다.

Example

FAS_SaveAllParameter 라이브러리 참조

See Also

FAS_SetParameter

FAS_GetROMParameter

ROM 에 저장되어 있는 Parameter 를 불러옵니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

iParamNo

가져오려는 Parameter 의 번호.

lRomParam

ROM 에 저장되어 있던 Parameter 값.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.

FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.
FMM_INVALID_PARAMETER_NUM : 지정한 iParamNo 의 Parameter 는 존재하지 않습니다.

Remarks

ROM 에 저장되어 있던 파라미터 값을 불러옵니다.
이 함수의 실행으로 RAM 에 있는 값이 바뀌지는 않습니다. 이를 위해서는 FAS_SetParameter 를 실행하십시오.

Example

FAS_SaveAllParameter 라이브러리 참조.

See Also

FAS_SaveAllParameters

2 - 5 . 서보 제어 함수

함수명	내용
FAS_ServoEnable	지정한 드라이브의 Servo 상태를 ON/OFF 시킵니다.
FAS_ServoAlarmReset	알람이 발생한 드라이브의 알람을 해제시킵니다 : 알람이 발생한 원인을 제거한 후 실시하십시오.
FAS_GetAlarmType	현재 알람의 발생 여부 및 알람의 종류를 확인합니다.

FAS_ServoEnable

드라이브를 Servo ON/OFF 합니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

bOnOff

Enable 혹은 Disable.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.

FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

Enable 후 Axis Status 의 Servo ON flag 가 'ON'이 되는데는 일정 시간이 소요됩니다.

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcAxisStatus()
{
        BYTE sb1 = 192, sb2 = 168, sb3=0, sb4=2 // IP :192.168.0.2
        int iBdID = 0;          //  Board 의 고유 번호
        EZISERVO_AXISSTATUS AxisStatus;
        int nRtn;

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

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

```
        _ASSERT(nRtn == FMM_OK);

        // SERVO_ON flag 가 OFF 되어 있으면 Servo On 을 합니다.
        if (AxisStatus.FFLAG_SERVOON == 0)
        {
                nRtn = FAS_ServoEnable(iBdID, TRUE);
                _ASSERT(nRtn == FMM_OK);
        }

        // Alarm 이 있으면 AlarmReset 을 합니다.
        if (AxisStatus.FFLAG_ERRORALL || AxisStatus.FFLAG_ERROVERCURRENT ||
AxisStatus.FFLAG_ERROVERLOAD)
        {
                nRtn = FAS_ServoAlarmReset(iBdID);
                _ASSERT(nRtn == FMM_OK);
        }

        // 연결을 종료합니다.
        FAS_Close(iBdID);
}
```

See Also

FAS_ServoAlarmReset

FAS_ServoAlarmReset

AlarmReset 명령을 보냅니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

이 명령을 보내기 전에 알람이 발생한 원인을 먼저 제거하십시오.
알람의 원인에 대해서는 '사용자 매뉴얼_본문편'을 참조하십시오.

Example

FAS_ServoEnable 라이브러리 참조

See Also

FAS_ServoEnable

2 - 6 . 제어 입출력 함수

함수명	내용
FAS_SetIOInput	제어 입력단의 입력 신호 레벨을 설정합니다 : 입력 신호를 [ON] 또는 [OFF] 상태로 만들어 줍니다.
FAS_GetIOInput	제어 입력단의 현재 입력 신호 상태를 읽어 들입니다 : 각 입력 신호에 대해 bit 단위로 리턴합니다.
FAS_SetIOOutput	제어 입력단의 출력 신호 레벨을 설정합니다 : 출력 신호를 [ON] 또는 [OFF] 상태로 만들어 줍니다.
FAS_GetIOOutput	제어 출력단의 현재 입력 신호 상태를 읽어 들입니다 : 각 출력 신호에 대해 bit 단위로 리턴합니다.
FAS_GetIOAssignMap	CN1 단자대 pin 의 설정 상태를 읽어들입니다 : 입력 및 출력단 가변 설정이 가능한 신호의 설정 상태 값등을 bit 단위로 리턴합니다.
FAS_SetIOAssignMap	제어 입출력 신호를 CN1 단자대의 pin 에 할당함과 동시에 신호 level 을 설정합니다 : 입력 및 출력단 가변 설정이 가능한 신호에 설정 합니다.
FAS_IOAssignMapReadROM	제어 입력 및 출력단의 설정 상태와 신호 레벨 상태 값을 ROM 으로부터 RAM 으로 읽어 들입니다.

FAS_SetIOInput

I/O Input 을 설정합니다. 자세한 설정 방법은 '1-2. Frame 의 구성'을 참조하십시오.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

dwIOSetMask

Set(ON 상태)할 Input 의 bitmask 값.

dwIOCLRMask

Clear(OFF 상태)할 Input 의 bitmask 값.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

dwIOSetMask 와 dwIOCLRMask 의 bit 값이 중복되지 않도록 주의하십시오.

Example

```
#include "FAS_ EziMOTIONPlusE.h"

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

        // 연결합니다.
        if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
        {
                // 연결이 실패하였습니다.

                return;
```

```
	}

	// I/O Input 을 확인합니다.
	nRtn = FAS_GetIOInput(iBdID, &dwInput);
	_ASSERT(nRtn == FMM_OK);
	if (dwInput & SERVO_IN_BITMASK_LIMITP)
	{
		// Limit+ 입력이 ON 되어 있습니다.
	}

	if (dwInput & SERVO_IN_BITMASK_USERIN0)
	{
		// User Input 0 이 ON 되어 있습니다.
	}

	// Clear Position 과 User Input 1 를 ON 하고 Jog+ 입력을 OFF 합니다.
	nRtn = FAS_SetIOInput(iBdID, SERVO_IN_BITMASK_CLEARPOSITION |
SERVO_IN_BITMASK_USERIN1, SERVO_IN_BITMASK_PJOG);
	_ASSERT(nRtn == FMM_OK);

	// I/O Output 을 확인합니다.
	nRtn = FAS_GetIOOutput(iBdID, &dwOutput);
	_ASSERT(nRtn == FMM_OK);
	if (dwOutput & SERVO_OUT_BITMASK_USEROUT0)
	{
		// User Output 0 신호가 ON 되어 있습니다.
	}

	// User Output 1 과 2 신호를 OFF 합니다.
	nRtn = FAS_SetIOOutput(iBdID, 0, SERVO_OUT_BITMASK_USEROUT1 |
SERVO_OUT_BITMASK_USEROUT2);
	_ASSERT(nRtn == FMM_OK);

	// 연결을 종료합니다.
	FAS_Close(iBdID);
}
```

See Also

FAS_GetIOInput

FAS_GetIOInput

I/O Input 값을 읽어옵니다. 자세한 설정 방법은 '1-2. Frame 의 구성'을 참조하십시오.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

dwIOInput

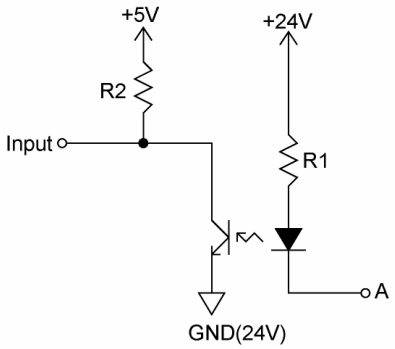
Input 값이 저장될 변수 포인터.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

Ezi-SERVOII Plus-E ALL 에는 총 6 개의 입력단이 있으며 이 중에서 3 개의 입력단을 사용자 지정으로 선택하여 사용할 수 있습니다. 이 함수는 입력 포트의 상태를 32bit 단위로 읽어 들이기 위한 함수이며, 모두 포토커플러 절연이 되어 있습니다. (그림 참조)



외부 입력으로부터 A 점의 전압이 24V 이면 Input 은 5V(High)로 인식됩니다.

Example

FAS_SetIOInput 라이브러리를 참조.

See Also

FAS_SetIOInput

FAS_SetIOOutput

I/O Output 값을 설정합니다. 자세한 설정 방법은 '1-2. Frame 의 구성'을 참조하십시오.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

dwIOSetMask

Set(ON 상태)할 Output 의 bitmask 값.

dwIOCLRMask

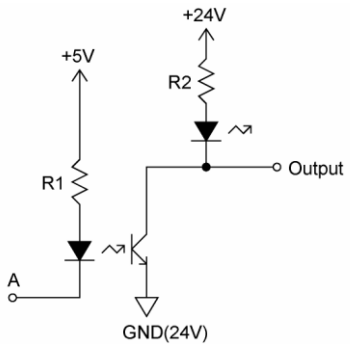
Clear(OFF 상태)할 Output 의 bitmask 값.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

Ezi-SERVOⅡ Plus-E ALL 에는 총 2 개의 출력단이 있으며, 이 중에서 1 개의 출력단을 사용자 지정으로 선택하여 사용할 수 있습니다.



출력 데이터가 1 이면 A 단자에는 0V, 0 이면 +5V 가 걸림.

dwIOSetMask 와 dwIOCLRMask 의 bit 값이 중복되지 않도록 주의하십시오

Example

FAS_SetIOInput 라이브러리를 참조.

See Also

FAS_GetIOOutput

FAS_GetIOOutput

I/O Output 값을 읽어옵니다. 자세한 설정 방법은 '1-2. Frame 의 구성'을 참조하십시오.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

dwIOInput

Output 값이 저장될 변수 포인터.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

Example

FAS_SetIOInput 라이브러리를 참조.

See Also

FAS_SetIOOutput

FAS_GetIOAssignMap

I/O Assign Map 을 읽어옵니다. 자세한 설정 방법은 '1-2. Frame 의 구성'을 참조하십시오.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

iIOPinNo

읽어올 I/O Pin 번호.

nIOLogic

해당 Pin 에 할당된 Logic 값이 저장될 변수 포인터.

bLevel

해당 Logic 의 Active Level 값이 저장될 변수 포인터.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

nIOLogic 에 대해서는 'Motion_define.h'를 참조하십시오.

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcIOAssign()
{

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

```
	// 연결합니다.
	if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
	{
		// 연결이 실패하였습니다.
		return;
	}

	// Input pin 할당 정보를 확인합니다.
	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);
	}

	// Input pin 3 에 SERVOON Logic (Low Active)을 할당함.
	iPinNo = 3;		// 0 ~ 11 의 값이 가능함 (주의 : 0 ~ 2 는 고정되어 있음).
	nRtn = FAS_SetIOAssignMap(iBdID, iPinNo, SERVO_IN_BITMASK_SERVOON,
LEVEL_LOW_ACTIVE);
	_ASSERT(nRtn == FMM_OK);

	// Output pin 할당 정보를 확인합니다.
	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);
	}

	// Output pin 9 에 ALARM Logic (High Active)을 할당합니다.
	iPinNo = 9;		// 0 ~ 9 의 값이 가능함 (주의 : 0 은 COMPOUT 으로 고정되어
있음).
	nRtn = FAS_SetIOAssignMap(iBdID, 12/*Input Pin Count*/ + iPinNo,
```

```
        SERVO_OUT_BITMASK_ALARM, LEVEL_HIGH_ACTIVE);
                _ASSERT(nRtn == FMM_OK);

                // 연결을 종료합니다.
                FAS_Close(iBdID);
        }
```

See Also

FAS_SetIOAssignMap

FAS_SetIOAssignMap

I/O Assign Map 을 설정합니다. 자세한 설정 방법은 '1-2. Frame 의 구성'을 참조하십시오.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

iIOPinNo

읽어올 I/O Pin 번호.

nIOLogic

해당 Pin 에 할당할 Logic 값.

bLevel

해당 Logic 의 Active Level 값.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.
FMM_INVALID_PARAMETER_NUM : 지정한 iIOPinNo 혹은 nIOLogic 값이 범위를 벗어났습니다.

Remarks

현재의 설정 값을 ROM 메모리에 저장하기 위해서는 'FAS_SaveAllParameters' 라이브러리를 실행 합니다.

Example

FAS_GSetIOAssignMap 라이브러리 참조.

See Also

FAS_GetIOAssignMap

FAS_IOAssignMapReadROM

현재 ROM 에 저장된 입출력 설정 상태 및 신호 레벨 값들을 읽어들입니다.

Syntax

```
int FAS_PosTableReadROM(

    int iBdID
);
```

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.
FMC_POSTABLE_ERROR : Position Table 을 읽는 도중에 에러가 발생하였습니다.

Remarks

Example

See Also

FAS_ GetIOAssignMap

2 - 7 . 위치 제어 함수

함수명	내용
FAS_SetCommandPos	목표(Command) 위치값을 임의의 값으로 설정합니다.
FAS_SetActualPos	실제(Actual) 위치값을 임의의 값으로 설정합니다.
FAS_GetCommandPos	현재 목표(Command) 위치값을 읽어들입니다.
FAS_GetActualPos	실제(Actual) 위치값을 읽어들입니다.
FAS_GetPosError	현재 실제(Actual) 위치값과 목표 위치(Command)값의 차이를 읽어들입니다.
FAS_GetActualVel	현재 이동중인 운전의 실제 운전 속도값을 읽어들입니다.
FAS_ClearPosition	목표(Command) 위치값과 실제(Actual) 위치값을 '0'으로 설정합니다.

FAS_SetCommandPos

Motor 의 Command Position 값을 설정합니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

lCmdPos

설정할 Command Position 값.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

위치 지령 (펄스 출력 카운터) 값을 사용자가 설정합니다.
주로 현재의 위치를 사용자가 원하는 좌표 값으로 설정 시 사용됩니다

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcClearPosition()
{
        BYTE sb1 = 192, sb2 = 168, sb3=0, sb4=2 // IP :192.168.0.2
        int iBdID = 0;          //  Board 의 고유 번호
        int nRtn;

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

        // Command Position, Actual Position 값을 초기화합니다.
```

```
        nRtn = FAS_SetCommandPos(iBdID, 0);
        _ASSERT(nRtn == FMM_OK);
        nRtn = FAS_SetActualPos(iBdID, 0);
        _ASSERT(nRtn == FMM_OK);

        // 연결을 종료합니다.
        FAS_Close(iBdID);
}
```

See Also

FAS_SetActualPos

FAS_SetActualPos

Motor 의 Actual Position 값을 설정합니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

lActPos

설정할 Actual Position 값.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

엔코더 피드백 카운터 값을 사용자가 원하는 값으로 설정합니다.

Example

FAS_GetActualPos 라이브러리 참조.

See Also

FAS_SetCommandPos

FAS_GetCommandPos

현재 Motor 의 Command Position 값을 읽어옵니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

lCmdPos

Command Position 값이 저장될 변수 포인터.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

위치 지령 (펄스 출력 카운터) 값을 읽어옵니다.

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcDisplayStatus()
{

        BYTE sb1 = 192, sb2 = 168, sb3=0, sb4=2 // IP :192.168.0.2
        int iBdID = 0;          //  Board 의 고유 번호
        long lValue;
        int nRtn;

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

        // Ezi-SERVOⅡ의 Position 에 관한 정보를 확인합니다.
        nRtn = FAS_GetCommandPos(iBdID, &lValue);
```

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

        // 연결을 종료합니다.
        FAS_Close(iBdID);
}
```

See Also

FAS_GetActualPos

FAS_GetActualPos

현재 Motor 의 Actual Position 값을 읽어옵니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

lActPos

Actual Position 값이 저장될 변수 포인터.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

주로 위치 결정 완료 후 실제의 위치를 확인 시 사용합니다.

Example

FAS_GetCommandPosition 라이브러리 참조.

See Also

FAS_GetCommandPos

FAS_GetPosError

Motor 의 Position Error 값을 읽어옵니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

lPosErr

Position Error 값이 저장될 변수 포인터.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

Example

FAS_GetCommandPosition 라이브러리 참조.

See Also

FAS_GetCommandPos,
FAS_GetActualPos

FAS_GetActualVel

Motor 의 Actual Velocity 값을 읽어옵니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

lActVel

Actual Velocity 값이 저장될 변수 포인터.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

Example

FAS_GetCommandPosition 라이브러리 참조.

See Also

FAS_ClearPosition

Motor 의 Command Position 값과 Actual Position 값을 '0'으로 설정합니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

위치 값을 사용자가 설정합니다.
주로 시스템 초기화 시에 사용됩니다.

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcClearPosition()
{
        BYTE sb1 = 192, sb2 = 168, sb3=0, sb4=2 // IP :192.168.0.2
        int iBdID = 0;          //  Board 의 고유 번호
        int nRtn;

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

        // Command Position, Actual Position 값을 0 으로 초기화합니다.
        nRtn = FAS_ClearPosition(iBdID);
        _ASSERT(nRtn == FMM_OK);

        // 연결을 종료합니다.
```

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

See Also

FAS_SetActualPos, FAS_SetCommandPos

2 - 8 . 드라이브 상태 제어 함수

함수명	내용
FAS_GetIOAxisStatus	제어 입출력 상태와 운전 상태 Flag 값을 읽어들입니다. : 현재의 입력 상태 값, 출력 설정 상태 값 및 운전상태 Flag 값을 리턴합니다.
FAS_GetMotionStatus	현재 운전 진행 상황 및 운전중인 PT 번호를 읽어들입니다. : Command position, Actual position, 속도값 등을 리턴합니다.
FAS_GetAllStatus	현재 상태를 모두 포함하여 한꺼번에 읽어들입니다. : 'FAS_GetIOAxisStatus'함수와 'FAS_ GetMotionStatus' 함수를 결합한 것입니다.
FAS_GetAxisStatus	해당 드라이브의 운전 상태 flag 값을 읽어들입니다.

FAS_GetIOAxisStatus

해당 Board 의 I/O Input, Output 값과 Motor Axis Status 값을 모두 읽어옵니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

dwInStatus

I/O Input 값이 저장될 변수 포인터.

dwOutStatus

I/O Output 값이 저장될 변수 포인터.

dwAxisStatus

해당 Motor 의 Axis Status 값이 저장될 변수 포인터.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

Example

FAS_MoveSingleAxisAbsPos 라이브러리를 참조.

See Also

FAS_GetMotionStatus

현재 Motor 의 Motion Status 들을 한번에 읽어옵니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

lCmdPos

Command Position 값이 저장될 변수 포인터.

lActPos

Actual Position 값이 저장될 변수 포인터.

lPosErr

Position Error 값이 저장될 변수 포인터.

lActVel

Actual Velocity 값이 저장될 변수 포인터.

wPosItemNo

Position Table 의 현재 실행 Item Number 값이 저장될 변수 포인터.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : Board 를 연결하기 전입니다.

FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

Example

FAS_MoveSingleAxisAbsPos 라이브러리를 참조

See Also

FAS_GetAllStatus

해당 Board 의 I/O Input, Output 값과 Motor Axis Status, Motor 의 Motion Status 값들을 모두 읽어옵니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

dwInStatus

I/O Input 값이 저장될 변수 포인터.

dwOutStatus

I/O Output 값이 저장될 변수 포인터.

dwAxisStatus

해당 Motor 의 Axis Status 값이 저장될 변수 포인터.

lCmdPos

Command Position 값이 저장될 변수 포인터.

lActPos

Actual Position 값이 저장될 변수 포인터.

lPosErr

Position Error 값이 저장될 변수 포인터.

lActVel

Actual Velocity 값이 저장될 변수 포인터.

wPosItemNo

Position Table 의 현재 실행 Item Number 값이 저장될 변수 포인터.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : Board 를 연결하기 전입니다.

FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

Example

FAS_MoveSingleAxisAbsPos 라이브러리를 참조.

See Also

FAS_GetAxisStatus
FAS_GetMotionStatus

FAS_GetAxisStatus

Motor의 Axis Status 값을 읽어옵니다. Status Flag 값에 대한 설명은 '1-2. Frame의 구성'을 참조하십시오.

Syntax

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

Parameters

iBdID

해당 Board의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

dwAxisStatus

해당 Motor의 Axis Status 값이 저장될 변수 포인터.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID의 board는 존재하지 않습니다.

Remarks

Example

FAS_MoveSingleAxisAbsPos 라이브러리를 참조.

See Also

2 - 9 . 운전 제어 함수

함수명	내용
FAS_MoveStop	운전중인 모터를 감속하면서 정지시킵니다.
FAS_EmergencyStop	운전중인 모터를 감속없이 즉시 정지시킵니다.
FAS_MoveOriginSingleAxis	원점 복귀 운전을 시작합니다.
FAS_MoveSingleAxisAbsPos	주어진 절대(Absolute) 위치값 만큼 운전을 실시합니다.
FAS_MoveSingleAxisIncPos	주어진 상대(Incremental) 위치값 만큼 운전을 실시합니다.
FAS_MoveToLimit	Limit 센서가 감지되는 위치까지 운전을 실시합니다.
FAS_MoveVelocity	주어진 속도와 방향으로 운전을 시작합니다 : Jog 운전 등에 사용됩니다.
FAS_PositionAbsOverride	운전중인 상태에서 목표 절대 위치값 [pulse]을 변경 합니다.
FAS_PositionIncOverride	운전중인 상태에서 목표 상대 위치값 [pulse]을 변경 합니다.
FAS_VelocityOverride	운전중인 상태에서 운전 속도값[pps]을 변경 합니다.
FAS_MoveLinearAbsPos	2 개의 이상의 Drive 에 주어진 절대(Absolute) 위치값 만큼 Linear Interpolation 운전을 실시 합니다
FAS_MoveLinearIncPos	2 개의 이상의 Drive 에 주어진 상대(Incremental) 위치값 만큼 Linear Interpolation 운전을 실시 합니다
FAS_MoveLinearAbsPos2[*1]	**FAS_MoveLinearAbsPos** 의 가감속이 개선된 함수입니다**.**
FAS_ MoveLinearIncPos2[*1]	**FAS_MoveLinearIncPos** 의 가감속이 개선된 함수입니다**.**
FAS_MoveSingleAxisAbsPosEx	주어진 절대(Absolute) 위치값 만큼 운전을 실시합니다. 가속 및 감속 시간을 설정할 수 있습니다.
FAS_MoveSingleAxisIncPosEx	주어진 상대(Incremental) 위치값 만큼 운전을 실시합니다. 가속 및 감속 시간을 설정할 수 있습니다.
FAS_MoveVelocityEx	주어진 속도와 방향으로 운전을 시작합니다 : Jog 운전등에 사용됩니다. 가속 및 감속 시간을 설정할 수 있습니다.
FAS_MovePause	운전중인 상태에서 운전의 일시 정지 및 일시 정지 상태에서의 운전 재개합니다.

[*1] 함수는 Firmware [ver.6.1.20.18]부터 사용 됩니다.

FAS_MoveStop

Motor 를 정지시킵니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

Example

FAS_MoveSingleAxisAbsPos 라이브러리를 참조.

See Also

FAS_EmergencyStop

Motor 를 급정지 시킵니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

이 함수는 감속 단계가 없으므로 기계에 가해지는 충격에 주의하십시오.

Example

FAS_MoveSingleAxisAbsPos 라이브러리를 참조.

See Also

FAS_MoveOriginSingleAxis

시스템의 원점(Origin)을 찾습니다.

자세한 사항은 '사용자 매뉴얼_본문편의 8.3 원점 복귀'항을 참조하십시오.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : Board 를 연결하기 전입니다.

FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

Example

FAS_MoveSingleAxisAbsPos 라이브러리를 참조.

See Also

FAS_MoveSingleAxisAbsPos

Motor 를 절대 좌표 값으로 이동시킵니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

lAbsPos

이동할 위치의 절대 좌표 값.

lVelocity

이동 시 속도 값.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

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;          //  Board 의 고유 번호
        DWORD dwAxisStatus, dwInput;
        EZISERVO_AXISSTATUS stAxisStatus;
        long lAbsPos, lIncPos, lVelocity;
        int nRtn;

        // 연결합니다.
        if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
```

```
{
        // 연결이 실패하였습니다.

        return;
}

// Error 와 Servo On 상태를 체크합니다.
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);

// Input 상태를 확인합니다.
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);

// Motor 를 15000 pulse 증가시킵니다.
lIncPos = 15000;
lVelocity = 30000;
nRtn = FAS_MoveSingleAxisIncPos(iBdID, lIncPos, lVelocity);
_ASSERT(nRtn == FMM_OK);

// Motion 명령이 완전히 끝날 때 까지 대기.
do
{
        Sleep(1);

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

// Motor 를 0 으로 옮깁니다.
```

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

        // Motion 명령이 완전히 끝날 때 까지 대기.
        do
        {
                Sleep(1);
                nRtn = FAS_GetAxisStatus(iBdID, &dwAxisStatus);
                _ASSERT(nRtn == FMM_OK);
                stAxisStatus.dwValue = dwAxisStatus;
        }
        while (stAxisStatus.FFLAG_MOTIONING);

        // 연결을 종료합니다.
        FAS_Close(iBdID);
}
```

See Also

FAS_MoveSingleAxisIncPos

Motor 를 상대 좌표 값으로 이동시킵니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

lIncPos

이동할 위치의 상대 좌표 값.

lVelocity

이동 시 속도 값.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

Example

FAS_MoveSingleAxisAbsPos 라이브러리를 참조.

See Also

FAS_MoveToLimit

Motor 에게 해당 Limit 를 찾도록 명령합니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

lVelocity

이동 시 속도 값.

iLimitDir

찾아갈 Limit 의 방향.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

Example

FAS_MoveSingleAxisAbsPos 라이브러리를 참조.

See Also

FAS_MoveVelocity

Motor 를 해당 방향, 해당 속도로 이동시킵니다. Jog 운전시 사용됩니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

lVelocity

이동 시 속도 값.

iVelDir

이동 할 방향.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

Example

FAS_MoveSingleAxisAbsPos 라이브러리를 참조.

See Also

FAS_PositionAbsOverride

Motor 의 절대 위치 이동 중 설정되었던 절대 위치 값을 변경합니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

lOverridePos

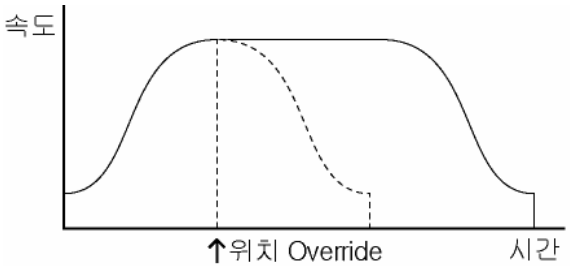
변경할 절대 좌표 위치값.

Return Value

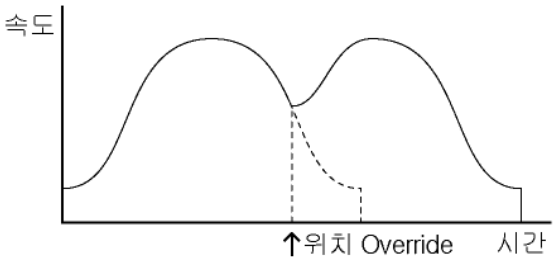
FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

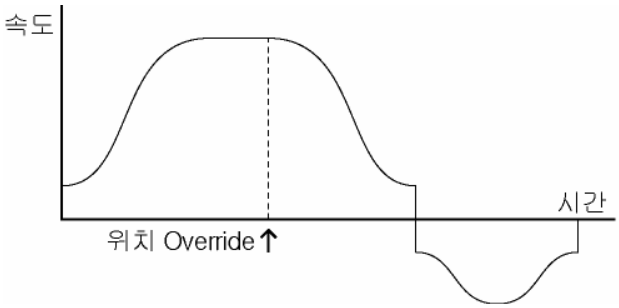
1) 가속 혹은 등속 중에 목표 위치를 원래의 목표 위치보다 먼 좌표로 설정하면 속도 패턴은 그때까지의 속도 패턴으로 동작하여 변경한 목표 위치에서 정지합니다.



2) 감속 중에 목표 위치를 변경하면 속도 패턴은 다시 등속 속도까지 가속한 후 변경한 목표 위치에서 정지합니다.



3) 변경한 목표 위치가 원래의 목표 위치 보다 가까운 곳에 설정된 경우에는 변경된 목표 위치로 이동한 후 정지합니다.



4) FAS_PositionAbsOverride 라이브러리와 동시에 사용할 수 없습니다.
FAS_VelocityOverride 라이브러리와 동시에 사용할 수 없습니다.

Example

FAS_MoveSingleAxisAbsPos 라이브러리를 참조.

See Also

FAS_PositionIncOverride

FAS_PositionIncOverride

Motor 의 상대위치 이동 중 설정되었던 상대 위치 값을 변경합니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

lOverridePos

변경할 상대 좌표 위치 값.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

1) FAS_PositionAbsOverride 라이브러리를 참조.
2) FAS_PositionIncOverride 라이브러리와 동시에 사용할 수 없습니다.
   FAS_VelocityOverride 라이브러리와 동시에 사용할 수 없습니다.

Example

FAS_MoveSingleAxisAbsPos 라이브러리를 참조.

See Also

FAS_PositionAbsOverride

FAS_VelocityOverride

Motor 의 이동 중 설정되었던 속도 값을 변경합니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

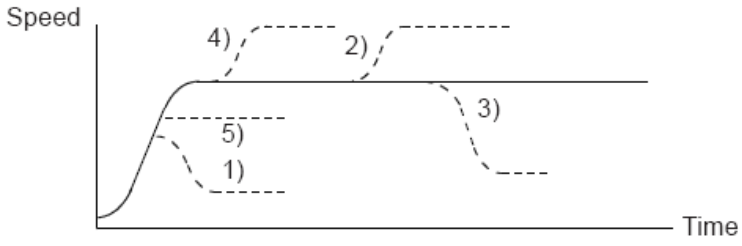
lVelocity

변경할 속도 값.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks



1) ((change speed) < (speed before change)) 인 경우 새로운 속도 패턴을 이용한 가감속을 통하여 change speed 까지 도달합니다.
5) ((change speed) ≥ (speed before change)) 인 경우 속도 패턴이 없는 가감속을 통하여 change speed 까지 도달합니다.
4) speed before change 까지는 속도 패턴의 변경없이 도달한 후, change speed 까지는 새로운 속도 패턴 특성으로 도달합니다.
2), 3) 가감속이 완료된 후에는 change speed 의 속도 패턴 특성에 맞추어서 change speed 까지 도달합니다.

- 지령 속도(정지상태에서의 이동 지령 속도)에 따라 변경 가능한 속도 범위를 확인하시고 함수를 사용하시기 바랍니다. 아래 표의 내용을 참고 하십시오.

지령 속도 [pps]	속도 변경 범위[pps]	
	최소	최대
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
163841~327680	100	1638300
327681~655360	200	3276600

속도 변경 범위에 들어 있지 않은 값으로 설정할 경우 범위 내에서의 최소값 또는 최대 값의 속도로 변경됩니다.

- FAS_PositionIncOverride 라이브러리와 동시에 사용할 수 없습니다.
- FAS_PositionAbsOverride 라이브러리와 동시에 사용할 수 없습니다.

Example

FAS_MoveSingleAxisAbsPos 라이브러리를 참조.

See Also

FAS_MoveLinearAbsPos

2 개의 이상의 Drive 에 주어진 절대(Absolute) 위치값 만큼 Linear Interpolation 운전을 실시 합니다

Syntax

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

Parameters

nNoOfBds

Linear Motion 을 실행하고자 하는 Drive 의 수

iBdID

해당 Drive 들의 ID 배열

lplAbsPos

해당 Drive 들의 이동위치 배열

lFeedrate

이동 시 선속도 값.

wAcceltime

이동 시 가감속 구간의 시간 값

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : Board 를 연결하기 전입니다.

FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

FAS_ MoveLinearIncPos

2 개의 이상의 Drive 에 주어진 상대(Incremental) 위치값 만큼 Linear Interpolation 운전을 실시 합니다

Syntax

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

Parameters

nNoOfBds

Linear Motion 을 실행하고자 하는 Drive 의 수

iBdID

해당 Drive 들의 ID 배열

lplIncPos

해당 Drive 들의 이동 거리 배열

lFeedrate

이동 시 선속도 값.

wAcceltime

이동 시 가감속 구간의 시간 값

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : Board 를 연결하기 전입니다.

FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

FAS_MoveLinearAbsPos2

FAS_MoveLinearAbsPos 의 가감속이 개선된 함수입니다.

Syntax

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

Parameters

nNoOfBds

Linear Motion 을 실행하고자 하는 Drive 의 수

iBdID

해당 Drive 들의 ID 배열

lplAbsPos

해당 Drive 들의 이동위치 배열

lFeedrate

이동 시 선속도 값.

wAcceltime

이동 시 가감속 구간의 시간 값

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : Board 를 연결하기 전입니다.

FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

FAS_ MoveLinearIncPos2

FAS_MoveLinearIncPos 의 가감속이 개선된 함수입니다.

Syntax

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

Parameters

nNoOfBds

Linear Motion 을 실행하고자 하는 Drive 의 수

iBdID

해당 Drive 들의 ID 배열

lplIncPos

해당 Drive 들의 이동 거리 배열

lFeedrate

이동 시 선속도 값.

wAcceltime

이동 시 가감속 구간의 시간 값

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : Board 를 연결하기 전입니다.

FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

FAS_MoveSingleAxisAbsPosEx

Motor 를 특정 절대 좌표 값으로 이동시킵니다. (운전 가속 및 감속 시간 지정 가능)

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

lAbsPos

이동할 위치의 상대좌표 값

lVelocity

이동 시 속도 값.

lpExOption

Custom option.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

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

```
// 연결합니다.
if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
{
	// 연결이 실패하였습니다.

	return;
}

// 특정 가감속 시간으로 모터를 움직입니다. : 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);

// Motion 명령이 완전히 끝날 때 까지 대기.
do
{
	Sleep(1);

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

// 0 의 위치로 이동시킵니다.
lAbsPos = 0;
lVelocity = 20000;
nRtn = FAS_MoveSingleAxisAbsPos(iBdID, lAbsPos, lVelocity);
_ASSERT(nRtn == FMM_OK);

// Motion 명령이 완전히 끝날 때까지 대기.
do
{
	Sleep(1);

	nRtn = FAS_GetAxisStatus(iBdID, &dwAxisStatus);
```

```
            _ASSERT(nRtn == FMM_OK);
            stAxisStatus.dwValue = dwAxisStatus;
        }
        while (stAxisStatus.FFLAG_MOTIONING);

        // 연결을 종료합니다.
        FAS_Close(iBdID);
    }
```

See Also

FAS_MoveSingleAxisIncPosEx

Motor 를 특정 상대 좌표 값으로 이동시킵니다. (운전 가속 및 감속 시간 지정 가능)

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

lIncPos

이동할 위치의 상대좌표 값.

lVelocity

이동 시 속도 값.

lpExOption

Custom option.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : Board 를 연결하기 전입니다.

FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

Example

See Also

FAS_MoveVelocityEx

Motor 를 해당 방향, 해당 속도, 해당 가감속도로 이동시킵니다. Jog 운전시 사용됩니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

lVelocity

이동 시 속도 값.

iVelDir

이동 할 방향 (0: -Jog, 1: +Jog)

lpExOption

Custom option.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

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;          //  Board 의 고유 번호
        long lVelocity;
        VELOCITY_OPTION_EX opt = {0};
        int nRtn;

        // 연결합니다.
        if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
        {
```

```
            // 연결이 실패하였습니다.

            return;
        }

        // 특정 가감속 시간으로 모터를 움직입니다. : FAS_MoveSingleAxisIncPosEx
        lVelocity = 30000;

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

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

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

See Also

2-10. 포지션 테이블 제어 함수

함수명	내용
FAS_PosTableReadItem	특정 포지션 테이블의 RAM 의 항목 값들을 읽어들입니다.
FAS_PosTableWriteItem	특정 포지션 테이블의 항목 값을 RAM 에 저장합니다.
FAS_PosTableWriteROM	모든 포지션 테이블 값을 ROM 에 저장 합니다 : 256 개의 모든 PT 값이 저장됩니다.
FAS_PosTableReadROM	ROM 의 포지션 테이블 값들을 읽어들입니다 : 256 개의 모든 PT 값을 읽어들입니다.
FAS_PosTableRunItem	지정된 포지션 테이블에서 부터 순차적으로 운전을 시작 합니다.
FAS_PosTableReadOneItem	특정 포지션 테이블의 특정 항목의 RAM 의 값을 읽어 들입니다.
FAS_PosTableWriteOneItem	특정 포지션 테이블의 특정 항목 값을 RAM 에 저장합니다.

FAS_PosTableReadItem

Positin Table 의 특정 Item 을 읽어옵니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

wItemNo

읽어 올 Item 번호.

lpItem

Item 값이 저장될 Item 구조체 포인터.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.
FMM_INVALID_PARAMETER_NUM : wItemNo 가 범위를 벗어났습니다.

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;          //  Board의 고유 번호
        WORD wItemNo;
        ITEM_NODE nodeItem;
        int nRtn;

        // 연결합니다.
        if (FAS_Connect(sb1, sb2, sb3, sb4, iBdID) == FALSE)
        {
```

```
        // 연결이 실패하였습니다.

        return;
    }

    // 20번의 Position Table 값을 읽어와 위치 값을 수정합니다.
    wItemNo = 20;
    nRtn = FAS_PosTableReadItem(iBdID, wItemNo, &nodeItem);
    _ASSERT(nRtn == FMM_OK);

    nodeItem.lPosition = 260000;  // 위치 값을 260000으로 변경합니다.
    nodeItem.wBranch = 23;                  // 다음 명령을 23번으로 설정합니다.
    nodeItem.wContinuous = 1;               // 다음 명령이 감속 없이 바로 이어지도록
합니다.

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

    // 현재 수정된 Position Table 데이터를 무시하고 ROM에 값을 불러옵니다.
    nRtn = FAS_PosTableReadROM(iBdID);
    _ASSERT(nRtn == FMM_OK);

    // 현재 수정된 Position Table 데이터를 ROM에 저장합니다.
    nRtn = FAS_PosTableWriteROM(iBdID);
    _ASSERT(nRtn == FMM_OK);

    // 연결을 종료합니다.
    FAS_Close(iBdID);
}
```

See Also

FAS_PosTableWriteItem

FAS_PosTableWriteItem

Position Table 의 특정 Item 을 수정합니다.

Syntax

```
int FAS_PoslableWriteItem(
    int iBdID,
    WORD wItemNo,
    LPITEM_NODE lpItem
);
```

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

wItemNo

수정 할 Item 번호.

lpItem

수정 할 Item 구조체 포인터.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.
FMC_POSTABLE_ERROR : Position Table 을 쓰는 도중에 에러가 발생하였습니다.
FMM_INVALID_PARAMETER_NUM : wItemNo 가 범위를 벗어났습니다.

Remarks

포지션 테이블 데이터가 저장되는 은 RAM 과 ROM 이 있으며,
이 함수는 RAM 에 저장하는 기능이고, 전원 OFF 시에는 데이터가 삭제됩니다.

Example

See Also

FAS_PosTableWriteROM

현재의 Position Table Item 들을 ROM 에 모두 저장합니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.
FMC_POSTABLE_ERROR : Position Table 을 저장하는 중에 에러가 발생하였습니다.

Remarks

포지션 테이블 데이터가 저장되는 은 RAM 과 ROM 이 있으며,
이 함수는 ROM 에 저장하는 기능이고, 전원 OFF 시에도 데이터가 보존됩니다.

Example

See Also

FAS_PosTableReadROM

FAS_PosTableReadROM

현재 ROM 에 저장된 Position Table Item 값들을 읽어들입니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.
FMC_POSTABLE_ERROR : Position Table 을 읽는 도중에 에러가 발생하였습니다.

Remarks

Example

See Also

FAS_PosTableWriteROM

FAS_PosTableRunItem

Position Table 의 특정 Item 을 시작으로 명령을 수행합니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

wItemNo

동작을 시작할 Item 번호.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.
FMM_INVALID_PARAMETER_NUM : wItemNo 가 범위를 벗어났습니다.

Remarks

Example

See Also

FAS_GetAllStatus
FAS_MoveStop
FAS_EmergencyStop

FAS_PosTableReadOneItem

Positin Table 의 특정 Item 의 특정 항목의 값을 읽어옵니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

wItemNo

읽어 올 Item 번호.

wOffset

읽어 올 항목의 위치 offset 값. ('1-2-6. 포지션 테이블 항목'을 참조)

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.
FMM_INVALID_PARAMETER_NUM : wItemNo 가 범위를 벗어났습니다.

Remarks

Example

See Also

FAS_PosTableReadItem

FAS_PosTableWriteOneItem

FAS_PosTableWriteOneItem

Position Table 의 특정 Item 의 특정 항목의 값을 수정합니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

wItemNo

수정 할 Item 번호.

wOffset

읽어 올 항목의 위치 offset 값. ('1-2-6. 포지션 테이블 항목'을 참조)

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.
FMC_POSTABLE_ERROR : Position Table 을 쓰는 도중에 에러가 발생하였습니다.
FMM_INVALID_PARAMETER_NUM : wItemNo 가 범위를 벗어났습니다.

Remarks

Example

See Also

FAS_PosTableWriteItem

FAS_PosTableReadOneItem

2 - 1 1 . 기타 제어 함수

함수명	내용
FAS_TriggerOutput_RunA	특정 위치에서 출력 신호를 발생시키도록 하는 기능
FAS_TriggerOutput_Status	출력 신호(COMP)의 발생 여부를 확인하는 기능
FAS_MovePush	특정 위치에서부터 지정된 힘을 유지하며 이동하도록 하는 기능
FAS_GetPushStatus	현재의 push motion 의 상태를 확인하는 기능

FAS_TriggerOutput_RunA

위치 명령에 의한 운전중 특정 위치에서 디지털 출력 신호(COMP pin)를 발생/종료 시킵니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

bStartTrigger

출력 시작/종료 명령 (1:시작, 0:종료)

long lStartPos

출력 시작 위치 [pulse]

DWORD dwPeriod

출력 신호의 주기 [pulse]

DWORD dwPulseTime

출력 신호의 펄스 폭 [msec]

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : Board 를 연결하기 전입니다.

FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

Example

See Also

FAS_TriggerOutput_Status

FAS_ TriggerOutput_Status

현재 신호 출력 기능이 작동 중인지 여부를 확인하고자 하는 명령입니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

bTriggerStatus

현재의 신호 출력 상태.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

Example

See Also

FAS_ TriggerOutput_RunA

FAS_MovePush

위치 명령에 의해 이동 중 특정 위치에서부터 정해진 힘을 유지하면서 이동하며, 이동 중 물체(work)에 접촉되면 이동을 멈추지만 그 힘은 계속 유지하는 기능입니다.

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

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

DWORD dwStartSpd

위치 명령의 시작 속도.

DWORD dwMoveSpd

위치 명령의 운전 속도.

long lPosition

위치 명령의 목표 절대 위치값.

WORD wAccel

위치 명령의 가속 시간.

WORD wDecel

위치 명령의 감속 시간.

WORD wPushRate

Push 운전시의 힘의 비율.

DWORD dwPushSpd

Push 운전시의 운전 속도.

long lEndPosition

Push 운전시의 목표 절대 위치값.

WORD wPushMode

Push 운전시의 mode 지정

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : Board 를 연결하기 전입니다.

FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

Example

See Also

FAS_GetPushStatus

FAS_GetPushStatus

현재의 push motion 의 진행 상태를 확인하는 기능입니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

BYTE nPushStatus*

읽어 올 push motion 상태 값. ('1-2-1. FrameType 별 송수신 내용'을 참조)

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : Board 를 연결하기 전입니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

Example

See Also

FAS_Move Push

3. 부록 – DHCP 를 이용한 Network 정보 설정

3 - 1 . DHCP 기능

1) DHCP(Dynamic Host Configuration Protocol)란?

→ IP 주소와 같은 TCP/IP 통신을 수행하기 위한 Network 정보들을 동적으로 설정하기 위해 사용되는 표준 Network Protocol 입니다.

(Network 정보 : Gateway, Subnet, IP address)

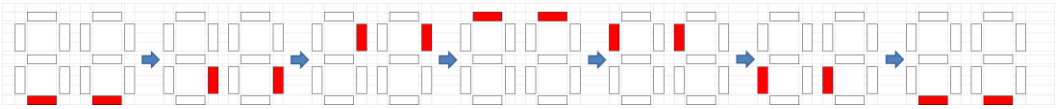
2) DHCP 를 사용하지 않을 경우

→ DRIVE 기본으로 설정된 Gateway, Subnet, IP address 를 사용하지 않으면, GUI 를 이용하여 설정을 변경하고 저장해야 하며 현재 네트워크 정보를 알고 있어야 합니다.

→ DHCP 를 사용할 경우 Gateway, Subnet, IP address 를 제품 내에서 자동으로 설정합니다. 다만, GUI 를 이용하여 자동으로 설정된 Network 정보의 저장이 필요 합니다.

3 - 2 . DHCP 을 이용한 Network 설정(Plus-E series)

1) IP 설정 스위치(SW1, SW2)를 F,F 로 설정
2) Ethernet IN Connector 에 Ethernet 연결
3) 전원 인가
4) 7-segment 가 아래와 같이 점멸



5) Network 정보가 설정되면 7-segment 에 IP address 를 표시
(aaa.bbb.ccc.ddd 를 표시 후에 ddd 에 해당하는 Hex.값 표시)
6) GUI 를 접속하여 Network 정보를 저장한 후에 전원 차단
(Config Slave ID / IP Address 를 이용)
7) IP 설정 스위치(SW1, SW2)의 값을 1~254 에서 Gateway 와 중복되지 않게 설정
8) 전원 인가(설정 완료)

● DHCP 는 동적으로 Network 정보를 설정하기 때문에 전원을 인가할 때마다 IP address – aaa.bbb.ccc.ddd 에서 ddd 는 변경될 수 있습니다. 따라서 DHCP 의 방법으로 Network 정보를 설정한 후, 반드시 6)을 수행해야 합니다.

● **DHCP 서버 기능이 있는 PC 또는 공유기를 사용할 경우에만 DHCP 를 이용한 Network 정보 설정이 가능합니다.**

FASTECH

Fast, Accurate, Smooth Motion

FASTECH Co., Ltd.
경기도 부천시 원미구 약대동 193번지
부천테크노파크 401동 1202호 (우)420-734
TEL : 032)234-6300,6301 FAX : 032)234-6302
E-mail : fastech@fastech.co.kr
Homepage : www.fastech.co.kr