

1.



Micro Stepping System
with Network Based Motion Controller



Step Motors with Integrated
Drive and Controller

사용자 메뉴얼

통신_Ver6 기능편

(Rev.08.05.029)



- 목 차 -


1. 통신 프로토콜	4
1.1 통신 기능	4
1-1-1. 통신사양	4
1-1-2. RS-485 통신 프로토콜(Ver6)	4
1-1-3. CRC 계산 예제	5
1-1-4. 답신 Frame 구조와 통신 에러(Ver6)	8
1.2 Frame 의 구성	9
1-2-1. Frame type 별 송수신 내용	9
1-2-2. 파라미터 목록표	23
1-2-3. 제어 출력핀의 bit 설정	24
1-2-4. 제어 입력핀의 bit 설정	24
1-2-5. 상태 Flag 의 bit 설정	25
1-2-6. 포지션테이블 항목	26
1-2-7. 모터의 종류	26
1.3 프로그램의 종류	27
2. PC 프로그램용 라이브러리(Ver6)	28
2.1 라이브러리의 구성	28
2.2 통신상태 표시용 창	29
2.3 드라이브 연결함수	33
FAS_Connect	
FAS_Close	
FAS_GetSlaveInfo	
FAS_GetMotorInfo	
FAS_IsSlaveExist	
FAS_EnableLog	
FAS_SetLogPath	
2.4 파라미터 제어함수	40
FAS_SaveAllParameters	
FAS_SetParameter	
FAS_GetParameter	
FAS_GetROMParameter	
2.5 알람 제어함수	46
FAS_ServoAlarmReset	
2.6 제어 입출력 함수	48
FAS_SetIOInput	
FAS_GetIOInput	
FAS_SetIOOutput	
FAS_GetIOOutput	
FAS_GetIOAssignMap	
FAS_SetIOAssignMap	
FAS_IOAssignMapReadROM	
2.7 위치 제어함수	58
FAS_SetCommandPos	
FAS_SetActualPos	
FAS_GetCommandPos	
FAS_GetActualPos	
FAS_GetPosError	
FAS_GetActualVel	
FAS_ClearPosition	
2.8 드라이브 상태 제어함수	67
FAS_GetIOAxisStatus	

FAS_GetMotionStatus	
FAS_GetAllStatus	
FAS_GetAxisStatus	
2.9 운전 제어함수	72
FAS_IsMotioning	
FAS_MoveStop	
FAS_EmergencyStop	
FAS_MoveOriginSingleAxis	
FAS_MoveSingleAxisAbsPos	
FAS_MoveSingleAxisIncPos	
FAS_MoveToLimit	
FAS_MoveVelocity	
FAS_PositionAbsOverride	
FAS_PositionIncOverride	
FAS_VelocityOverride	
FAS_AllMoveStop	
FAS_AllEmergencyStop	
FAS_AllMoveOriginSingleAxis	
FAS_AllMoveSingleAxisAbsPos	
FAS_AllMoveSingleAxisIncPos	
FAS_MoveSingleAxisAbsPosEx	
FAS_MoveSingleAxisIncPosEx	
FAS_MoveVelocityEx	
FAS_MoveLinearAbsPos / FAS_MoveLinearIncPos	
2.10 포지션테이블 제어함수	98
FAS_PosTableReadItem	
FAS_PosTableWriteItem	
FAS_PosTableWriteROM	
FAS_PosTableReadROM	
FAS_PosTableRunItem	
2.11 기타 제어함수	107
FAS_TriggerOutput_RunA	
FAS_TriggerOutput_Status	
3. PLC 프로그램용 프로토콜	110

1. 통신 프로토콜

1.1 통신기능

Ezi-STEP Plus-R 은 RS-485 (2 선식)에서의 Daisy-Chain 링크에 의해 16 축까지의 제어가 가능합니다.

 주의	Windows 가 대기모드 및 절전모드로 들어갈 경우 기본적으로 Serial 통신이 두절됨 을 주의하십시오. 이 모드에서 복구 후에는 Serial 통신을 재연결 해야 합니다. 이 사항은 제공되는 라이브러리도 같이 적용됩니다.
---	---

1-1-1. 통신사양

통신 규격	RS-485
통신 방식	비동기식
	반이중 통신
baud rate [bps]	9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600
데이터 형식	8bit ASCII 코드, HEX
패리티	No
stop bit	1bit
CRC 검사	있음
최대 배선 길이 (컨버터<->끝단 드라이브)	30 m 이내
드라이브간 최소 배선 길이	60 cm 이상
접속 축수	16 축 (No. 0~F)

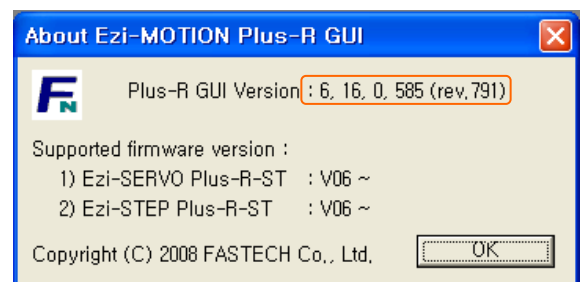
1-1-2. RS-485 통신 프로토콜(Ver6)

사용자 프로그램(GUI)의 version 에는 다음의 2 가지 지원되며, 본 메뉴얼은 **verison 6** 에 해당됩니다.

종류	Firmware version	호환	사용자(GUI) version
1	6 번째 (V06.0x.0xx.xx)	<->	6 번째 (6.xx.x.xxx)
2	8 번째 (V08.xx.0xx.xx)	<->	8 번째 (8.xx.x.xxx)

현재 사용중인 veriosn 은 다음과 같이 확인할 수 있습니다.

사용자 프로그램(GUI) 와 통신 연결후
'Help' 메뉴의 'About Plus-R GUI...' 메뉴를
클릭하면 오른쪽 창에서 확인



1) 통신 FRAME 의 개요



2) FRAME 의 기본 구조

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

- ① 0xAA : 구분 표시용 byte
- ② 0xAA 0xCC : Frame 의 시작(Header)임을 표시합니다.
- ③ 0xAA 0xEE : Frame 의 종료(Tail)임을 표시합니다.
- ④ Frame Data 항의 데이터중 실제 데이터가 '0xAA' 인 경우에는 바로 뒤에 '0xAA' 를 추가하여 줍니다. (byte stuffing)
- ⑤ '0xAA' 다음의 데이터가 '0xAA' , '0xCC' 또는 '0xEE' 이 아니면 에러가 발생한 상황입니다.

위의 기본 구조중 **Frame Data** 항의 세부 구성은 다음과 같습니다.

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

- ① Slave ID : PC 의 통신 port 에 연결된 드라이브 모듈의 번호(0~15)입니다.
- ② Frame type : 해당 Frame 의 명령어 종류를 지정합니다.
그 종류는 아래의 「Frame type 과 Data 구성」 절을 참조하십시오.
- ③ Data : 이 항의 데이터의 구조 및 길이 등은 Frame type 에 따라 정해집니다.
그 자세한 구조는 아래의 「Frame type 과 Data 구성」 절을 참조하십시오.
- ④ CRC : 통신중 에러 상태를 확인하기 위하여 **CRC(Cyclic Redundancy Check)** 규격중 다항식(polynomial) 인자값으로 CRC16(2 byte)의 '0xA001' 을 사용하거나 CRC-16-IBM 의 생성다항식 'X¹⁶+X¹⁵+X²+1' 을 사용합니다.
CRC 계산은 CRC 항 이전의 모든 항(Slave ID, Frame type, Data)에 대해 수행합니다.

1-1-3. CRC 계산 예제

아래의 프로그램 소스 내용은 제품과 함께 제공된 파일(파일명 : CRC_Checksum.c)에 포함되어 있습니다.

1)CRC16 의 '0xA001' 사용방법

```
const unsigned short TABLE_CRCVALUE[] =
{
    0X0000, 0XC0C1, 0XC181, 0X0140, 0XC301, 0X03C0, 0X0280, 0XC241,
    0XC601, 0X06C0, 0X0780, 0XC741, 0X0500, 0XC5C1, 0XC481, 0X0440,
    0XCC01, 0X0CC0, 0X0D80, 0XCD41, 0X0F00, 0XCF81, 0XCE81, 0X0E40,
    0X0A00, 0XCAC1, 0XCB81, 0X0B40, 0XC901, 0X09C0, 0X0880, 0XC841,
    0XD801, 0X18C0, 0X1980, 0XD941, 0X1B00, 0XDBC1, 0XDA81, 0X1A40,
    0X1E00, 0XDEC1, 0XDF81, 0X1F40, 0XDD01, 0X1DC0, 0X1C80, 0XDC41,
    0X1400, 0XD4C1, 0XD581, 0X1540, 0XD701, 0X17C0, 0X1680, 0XD641,
    0XD201, 0X12C0, 0X1380, 0XD341, 0X1100, 0XD1C1, 0XD081, 0X1040,
    0XF001, 0X30C0, 0X3180, 0XF141, 0X3300, 0XF3C1, 0XF281, 0X3240,
    0X3600, 0XF6C1, 0XF781, 0X3740, 0XF501, 0X35C0, 0X3480, 0XF441,
    0X3C00, 0XFCC1, 0XFD81, 0X3D40, 0XFF01, 0X3FC0, 0X3E80, 0XFE41,
```

```

0XFA01, 0X3AC0, 0X3B80, 0XFB41, 0X3900, 0XF9C1, 0XF881, 0X3840,
0X2800, 0XE8C1, 0XE981, 0X2940, 0XEB01, 0X2BC0, 0X2A80, 0XEA41,
0XEE01, 0X2EC0, 0X2F80, 0XEF41, 0X2D00, 0XEDC1, 0XEC81, 0X2C40,
0XE401, 0X24C0, 0X2580, 0XE541, 0X2700, 0XE7C1, 0XE681, 0X2640,
0X2200, 0XE2C1, 0XE381, 0X2340, 0XE101, 0X21C0, 0X2080, 0XE041,
0XA001, 0X60C0, 0X6180, 0XA141, 0X6300, 0XA3C1, 0XA281, 0X6240,
0X6600, 0XA6C1, 0XA781, 0X6740, 0XA501, 0X65C0, 0X6480, 0XA441,
0X6C00, 0XACC1, 0XAD81, 0X6D40, 0XAF01, 0X6FC0, 0X6E80, 0XAE41,
0XAA01, 0X6AC0, 0X6B80, 0XAB41, 0X6900, 0XA9C1, 0XA881, 0X6840,
0X7800, 0XB8C1, 0XB981, 0X7940, 0XBB01, 0X7BC0, 0X7A80, 0XBA41,
0XBE01, 0X7EC0, 0X7F80, 0XBF41, 0X7D00, 0XBD C1, 0XBC81, 0X7C40,
0XB401, 0X74C0, 0X7580, 0XB541, 0X7700, 0XB7C1, 0XB681, 0X7640,
0X7200, 0XB2C1, 0XB381, 0X7340, 0XB101, 0X71C0, 0X7080, 0XB041,
0X5000, 0X90C1, 0X9181, 0X5140, 0X9301, 0X53C0, 0X5280, 0X9241,
0X9601, 0X56C0, 0X5780, 0X9741, 0X5500, 0X95C1, 0X9481, 0X5440,
0X9C01, 0X5CC0, 0X5D80, 0X9D41, 0X5F00, 0X9FC1, 0X9E81, 0X5E40,
0X5A00, 0X9AC1, 0X9B81, 0X5B40, 0X9901, 0X59C0, 0X5880, 0X9841,
0X8801, 0X48C0, 0X4980, 0X8941, 0X4B00, 0X8BC1, 0X8A81, 0X4A40,
0X4E00, 0X8EC1, 0X8F81, 0X4F40, 0X8D01, 0X4DC0, 0X4C80, 0X8C41,
0X4400, 0X84C1, 0X8581, 0X4540, 0X8701, 0X47C0, 0X4680, 0X8641,
0X8201, 0X42C0, 0X4380, 0X8341, 0X4100, 0X81C1, 0X8081, 0X4040
};

```

```

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

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

```

2) CRC-16-IBM 의 'X16+X15+X2+1' 사용방법

```

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

    /* Initialize CRC */
    wCrc = 0xffff;

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

```

```

/* Loop through all 8 data bits */

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

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

```


1-1-4. 답신 Frame 구조와 통신 에러(Ver6)


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

Slave ID	Frame type	Data		CRC	
1 byte	1 bytes	1 bytes.	0 ~ 247bytes.	2 bytes	
		통신 상태	답신 데이터	Low byte	High byte

- ① Slave ID : 송신 Frame 과 동일.
(송신시의 데이터와 일치하지 않으면 에러 상태로 인식하십시오)
- ② Frame type : 송신 Frame 과 동일.
(송신시의 데이터와 일치하지 않으면 에러 상태로 인식하십시오)
- ③ Data : 단순 실행 명령등의 송신시에는 이 항목의 데이터는 없을 수 있습니다. 그러나
답신때에는 통신 상태(에러/정상 여부)를 나타내는 1 byte 의 데이터가 포함됩니다.

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 범위외의 데이터입니다.
0x82	130	수신 Frame 에러 : 수신된 Frame 이 규격에 맞지않는 데이터입니다.
0x85	133	운전 명령 실패 : 다음과 같은 상태에서 새로운 운전을 실행하려고 했습니다. 1) 현재 모터가 운전중 2) 정지 명령중 3) STEP OFF 상태 4) 외부 엔코더 없이 Z-pulse Origin 을 시도
0x86	134	RESET 실패 : 다음과 같은 상태에서 Reset 을 실행하려고 했습니다. 1) STEP ON 상태 2) 외부입력신호에 의해 이미 Reset 상태임
0xAA	170	CRC 에러 : 수신된 Frame 의 data 가 주변 노이즈등의 영향으로 CRC 에러가 발생했습니다. 이 경우에는 송신측 DLL Library 에서 자동으로 1 회 더 통신을 시도합니다.

 주의	1) 송신 Frame 의 ‘Header’ 와 ‘Slave ID’ 값이 정상이 아니면, 드라이브측에서 답신을 하지 않습니다. 2) 통신 상태가 ‘130’ 번의 에러가 발생한 경우에는 답신 데이터의 크기는 0 byte 입니다.
---	---

1.2 Frame 의 구성(ver6)

1-2-1. Frame type 별 송수신 내용

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

Frame type	라이브러리 명	내용										
0x01 (1)	FAS_ GetSlaveInfo	<p>연결된 slave 의 종류와 프로그램 version 정보를 의뢰함.</p> <p>송신 : 0 byte 답신 : 1~248 bytes</p> <table><tr><td>1 byte</td><td>1 bytes</td><td colspan="2">0~246 bytes</td></tr><tr><td>통신 상태</td><td>Slave 종류</td><td colspan="2">ASCII string with NULL byte (strlen() + 1 bytes)</td></tr></table> <p>◆ Slave 종류 : 20 : Ezi-STEP Plus-R ST 60 : Ezi-STEP Plus-R MINI 1 : Ezi-SERVO Plus-R ST</p>			1 byte	1 bytes	0~246 bytes		통신 상태	Slave 종류	ASCII string with NULL byte (strlen() + 1 bytes)	
1 byte	1 bytes	0~246 bytes										
통신 상태	Slave 종류	ASCII string with NULL byte (strlen() + 1 bytes)										
0x05 (5)	FAS_ GetMotor Info	<p>Slave 에 연결된 모터의 종류에 대한 정보를 의뢰함.</p> <p>송신 : 0 byte 답신 : 1~246 bytes</p> <table><tr><td>1 byte</td><td>1 bytes</td><td colspan="2">0~246 bytes</td></tr><tr><td>통신 상태</td><td>Motor 번호 (1~255)</td><td colspan="2">ASCII string with NULL byte (strlen() + 1 bytes)</td></tr></table> <p>◆ Motor 종류 : 「1-2-7. 모터 종류」의 표를 참조.</p>			1 byte	1 bytes	0~246 bytes		통신 상태	Motor 번호 (1~255)	ASCII string with NULL byte (strlen() + 1 bytes)	
1 byte	1 bytes	0~246 bytes										
통신 상태	Motor 번호 (1~255)	ASCII string with NULL byte (strlen() + 1 bytes)										
0x10 (16)	FAS_ SaveAllParameters	<p>현재 설정된 파라미터 값과 입출력 신호의 할당값들을 드라이브의 ROM 메모리에 저장함. 이는 드라이브 전원을 OFF 해도 값들이 저장되도록 한다.</p> <p>따라서 ‘FAS_SetParameter’ 와 ‘FAS_SetIOAssignMap’ 에서 설정된 값들이 함께 저장됨.</p> <p>송신 : 0 byte 답신 : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table>			1 byte	통신 상태						
1 byte												
통신 상태												
0x11 (17)	FAS_ GetRomParameter	<p>ROM 메모리 영역의 특정 파라미터값을 읽어들임.</p> <p>송신 : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>파라미터 번호 (0~28)</td></tr></table> <p>답신 : 5 bytes</p> <table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>통신 상태</td><td>파라미터 값</td></tr></table> <p>「1-2-2. 파라미터 목록표」를 참조.</p>			1 byte	파라미터 번호 (0~28)	1 byte	4 bytes	통신 상태	파라미터 값		
1 byte												
파라미터 번호 (0~28)												
1 byte	4 bytes											
통신 상태	파라미터 값											

0x12 (18)	FAS_ SetParameter	<p>특정 파라미터 값을 드라이브의 RAM 메모리에 저장함.</p> <p>송신 : 5 bytes</p> <table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>파라미터 번호 (0~28)</td><td>파라미터 값</td></tr></table> <p>답신 : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table> <p>「1-2-2. 파라미터 목록표」를 참조.</p>	1 byte	4 bytes	파라미터 번호 (0~28)	파라미터 값	1 byte	통신 상태
1 byte	4 bytes							
파라미터 번호 (0~28)	파라미터 값							
1 byte								
통신 상태								
0x13 (19)	FAS_ GetParameter	<p>RAM 메모리 영역의 특정 파라미터값을 읽어들이м.</p> <p>송신 : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>파라미터 번호 (0~28)</td></tr></table> <p>답신 : 5 bytes</p> <table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>통신 상태</td><td>파라미터 값</td></tr></table> <p>「1-2-2. 파라미터 목록표」를 참조.</p>	1 byte	파라미터 번호 (0~28)	1 byte	4 bytes	통신 상태	파라미터 값
1 byte								
파라미터 번호 (0~28)								
1 byte	4 bytes							
통신 상태	파라미터 값							
0x20 (32)	FAS_ SetIIOOutput	<p>제어 출력단의 출력 신호 레벨을 설정함.</p> <p>송신 : 8 bytes</p> <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> <p>Set mask의 특정 bit 값이 1이면 해당 출력단 신호는 [ON]이 된다. Clear mask의 특정 bit 값이 1이면 해당 출력단 신호는 [OFF]가 된다. 자세한 사항은 「1-2-3. 제어출력핀의 bit 설정」항을 참조.</p> <p>답신 : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table>	4 bytes	4 bytes	I/O set mask 값	I/O clear mask 값	1 byte	통신 상태
4 bytes	4 bytes							
I/O set mask 값	I/O clear mask 값							
1 byte								
통신 상태								
0x21 (33)	FAS_ SetIIOInput	<p>제어 입력단의 입력 신호 레벨을 설정함.</p> <p>송신 : 8 bytes</p> <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> <p>Set mask의 특정 bit 값이 1이면 해당 입력단 신호는 [ON]이 된다. Clear mask의 특정 bit 값이 1이면 해당 입력단 신호는 [OFF]가 된다. 자세한 사항은 「1-2-4. 제어입력핀의 bit 설정」항을 참조.</p> <p>답신 : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table>	4 bytes	4 bytes	I/O set mask 값	I/O clear mask 값	1 byte	통신 상태
4 bytes	4 bytes							
I/O set mask 값	I/O clear mask 값							
1 byte								
통신 상태								

0x22 (34)	FAS_ GetIOInput	<p>제어 입력단의 현재 입력신호 상태를 읽어들이м.</p> <p>송신 : 0 byte</p> <p>답신 : 5 byte</p> <table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>통신 상태</td><td>입력 상태 값</td></tr></table> <p>각 입력 신호별 해당 bit 는 「1-2-4. 제어입력핀의 bit 설정」항을 참조.</p>	1 byte	4 bytes	통신 상태	입력 상태 값		
1 byte	4 bytes							
통신 상태	입력 상태 값							
0x23 (35)	FAS_ GetIOOutput	<p>제어 출력단의 현재 출력신호 상태를 읽어들이м.</p> <p>송신 : 0 byte</p> <p>답신 : 5 byte</p> <table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>통신 상태</td><td>출력 상태 값</td></tr></table> <p>각 출력 신호별 해당 bit 는 「1-2-3. 제어출력핀의 bit 설정」항을 참조.</p>	1 byte	4 bytes	통신 상태	출력 상태 값		
1 byte	4 bytes							
통신 상태	출력 상태 값							
0x24 (36)	FAS_ SetIOAssignMap	<p>제어 입출력 신호를 CN1 단자대의 pin 에 할당하며 동시에 신호 level 을 설정해줍니다. 이 설정 값을 ROM 메모리에 저장하기 위해서는 ‘FAS_SaveAllParameters’ 를 실행 함.</p> <p>송신 : 6 bytes</p> <table><tr><td>1 byte</td><td>4 bytes</td><td>1 byte</td></tr><tr><td>I/O number</td><td>I/O pin masking data</td><td>설정 level</td></tr></table> <p>◆ I/O number : ‘0~11’ 은 각각 ‘Limit+,Limit-,Org,IN1,...,IN9’ 에 해당됨. </p>	1 byte	4 bytes	1 byte	I/O number	I/O pin masking data	설정 level
1 byte	4 bytes	1 byte						
I/O number	I/O pin masking data	설정 level						

0x26 (38)	FAS_ IOAssignMapReadROM	<p>제어입출력 신호의 설정상태와 신호의 레벨 설정값을 ROM 메모리 영역에서 읽어들임.</p> <p>송신 : 0 byte 답신 : 2 byte</p> <table><tr><td>1 byte</td><td>1 bytes</td></tr><tr><td>통신 상태</td><td>명령 수행여부 (0:완료, 0 이외값:에러)</td></tr></table>	1 byte	1 bytes	통신 상태	명령 수행여부 (0:완료, 0 이외값:에러)												
1 byte	1 bytes																	
통신 상태	명령 수행여부 (0:완료, 0 이외값:에러)																	
0x27 (39)	FAS_ TriggerOutput_RunA	<p>제어출력신호(Compare Out)를 발생시키기 위한 명령.</p> <p>송신 : 18 bytes</p> <table><tr><td>1 byte</td><td>4 bytes</td><td>4 byte</td></tr><tr><td>출력 시작/종료 명령 (1:시작 0:종료)</td><td>출력 시작위치 [pulse]</td><td>Pulse 주기 [pulse]</td></tr></table> <table><tr><td>4 byte</td><td>1 bytes</td><td>4 byte</td></tr><tr><td>Pulse 폭 [msec]</td><td>출력 pin 번호 (0 으로 고정됨)</td><td>여분</td></tr></table> <p>답신 : 2 byte</p> <table><tr><td>1 byte</td><td>1 byte</td></tr><tr><td>통신 상태</td><td>명령 수행여부 (0:완료, 0 이외의값:에러)</td></tr></table>	1 byte	4 bytes	4 byte	출력 시작/종료 명령 (1:시작 0:종료)	출력 시작위치 [pulse]	Pulse 주기 [pulse]	4 byte	1 bytes	4 byte	Pulse 폭 [msec]	출력 pin 번호 (0 으로 고정됨)	여분	1 byte	1 byte	통신 상태	명령 수행여부 (0:완료, 0 이외의값:에러)
1 byte	4 bytes	4 byte																
출력 시작/종료 명령 (1:시작 0:종료)	출력 시작위치 [pulse]	Pulse 주기 [pulse]																
4 byte	1 bytes	4 byte																
Pulse 폭 [msec]	출력 pin 번호 (0 으로 고정됨)	여분																
1 byte	1 byte																	
통신 상태	명령 수행여부 (0:완료, 0 이외의값:에러)																	
0x28 (40)	FAS_ TriggerOutput_Status	<p>현재 신호(Compare Out)출력 기능이 작동중인지 여부를 확인하는 명령.</p> <p>송신 : 0 byte 답신 : 2 byte</p> <table><tr><td>1 byte</td><td>1 bytes</td></tr><tr><td>통신 상태</td><td>현재상태 (1:출력중, 0 :종료)</td></tr></table>	1 byte	1 bytes	통신 상태	현재상태 (1:출력중, 0 :종료)												
1 byte	1 bytes																	
통신 상태	현재상태 (1:출력중, 0 :종료)																	
0x2A (42)	FAS_ StepEnable	<p>Step ON/OFF 상태를 설정함</p> <p>송신 : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>0:OFF, 1:ON</td></tr></table> <p>답신 : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table>	1 byte	0:OFF, 1:ON	1 byte	통신 상태												
1 byte																		
0:OFF, 1:ON																		
1 byte																		
통신 상태																		
0x2C (44)	FAS_ StepAlarmReset	<p>Step Alarm 상태를 reset 또는 reset 해제 시킴. Alarm을 reset 시키기위해서는 ‘reset’ 과 ‘reset 해제’를 차례대로 실행하여 주십시오.</p> <p>송신 : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>Reset 실시(1) Reset 해제(0)</td></tr></table>	1 byte	Reset 실시(1) Reset 해제(0)														
1 byte																		
Reset 실시(1) Reset 해제(0)																		

		<div>답신 : 1 byte</div> <table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table>	1 byte	통신 상태		
1 byte						
통신 상태						
0x2E (46)	FAS_ GetAlarmType	<div>현재 Alarm 상태 및 정보를 의뢰함.</div> <div>송신 : 0 byte</div> <div>답신 : 2 byte</div> <table><tr><td>1 byte</td><td>1 byte</td></tr><tr><td>통신 상태</td><td>Alarm 상태 (0:No alarm, 0 이외의값:Alarm 번호)</td></tr></table> <div>◆ Alarm type: No alarm (0) OverCurrent(1) OverSpeed(2) StepOut(3) OverTemperature(5) BackEMF(6) MotorConnect(7) MotorPower(9) Inposition(10)</div>	1 byte	1 byte	통신 상태	Alarm 상태 (0:No alarm, 0 이외의값:Alarm 번호)
1 byte	1 byte					
통신 상태	Alarm 상태 (0:No alarm, 0 이외의값:Alarm 번호)					

0x31 (49)	FAS_ MoveStop	현재 운전중인 모터의 정지를 의뢰함. 송신 : 0 byte 답신 : 1 byte <table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table>	1 byte	통신 상태				
1 byte								
통신 상태								
0x32 (50)	FAS_ EmergencyStop	현재 운전중인 모터의 비상정지를 의뢰함. 송신 : 0 byte 답신 : 1 byte <table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table>	1 byte	통신 상태				
1 byte								
통신 상태								
0x33 (51)	FAS_ MoveOriginSingleAxis	현재 설정된 파라미터의 조건으로 원점복귀 운동 시작을 의뢰함. 송신 : 0 byte 답신 : 1 byte <table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table>	1 byte	통신 상태				
1 byte								
통신 상태								
0x34 (52)	FAS_ MoveSingleAxisAbsPos	절대값[pulse] 위치만큼의 이동 운전 시작을 의뢰함. 송신 : 8 bytes <table><tr><td>4 bytes</td><td>4 bytes</td></tr><tr><td>절대 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>	4 bytes	4 bytes	절대 위치값	운전속도 [pps]	1 byte	통신 상태
4 bytes	4 bytes							
절대 위치값	운전속도 [pps]							
1 byte								
통신 상태								
0x35 (53)	FAS_ MoveSingleAxisIncPos	상대값[pulse] 위치만큼의 이동 운전 시작을 의뢰함. 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>	4 bytes	4 bytes	상대 위치값	운전속도 [pps]	1 byte	통신 상태
4 bytes	4 bytes							
상대 위치값	운전속도 [pps]							
1 byte								
통신 상태								

0x36 (54)	FAS_ MoveToLimit	<p>현재 설정된 파라미터의 조건으로 리미트 운동 시작을 의뢰함.</p> <p>송신 : 5 bytes</p> <table><tr><td>4 bytes</td><td>1 byte</td></tr><tr><td>운전속도[pps]</td><td>운전방향 (0:-Limit 1:+Limit)</td></tr></table> <p>답신 : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table>	4 bytes	1 byte	운전속도[pps]	운전방향 (0:-Limit 1:+Limit)	1 byte	통신 상태
4 bytes	1 byte							
운전속도[pps]	운전방향 (0:-Limit 1:+Limit)							
1 byte								
통신 상태								
0x37 (55)	FAS_ MoveVelocity	<p>현재 설정된 파라미터의 조건으로 Jog 운동 시작을 의뢰함.</p> <p>송신 : 5 bytes</p> <table><tr><td>4 bytes</td><td>1 byte</td></tr><tr><td>운전속도[pps]</td><td>운전방향 (0:-Jog 1:+Jog)</td></tr></table> <p>답신 : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table>	4 bytes	1 byte	운전속도[pps]	운전방향 (0:-Jog 1:+Jog)	1 byte	통신 상태
4 bytes	1 byte							
운전속도[pps]	운전방향 (0:-Jog 1:+Jog)							
1 byte								
통신 상태								
0x38 (56)	FAS_ PositionAbsOverride	<p>운전중인 상태에서 목표 절대위치값[pulse]의 변경을 의뢰함.</p> <p>송신 : 4 bytes</p> <table><tr><td>4 bytes</td></tr><tr><td>변경된 목표위치값[pulse]</td></tr></table> <p>답신 : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table>	4 bytes	변경된 목표위치값[pulse]	1 byte	통신 상태		
4 bytes								
변경된 목표위치값[pulse]								
1 byte								
통신 상태								
0x39 (57)	FAS_ PositionIncOverride	<p>운전중인 상태에서 목표 상대위치값[pulse]의 변경을 의뢰함.</p> <p>송신 : 4 bytes</p> <table><tr><td>4 bytes</td></tr><tr><td>변경된 목표위치값[pulse]</td></tr></table> <p>답신 : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table>	4 bytes	변경된 목표위치값[pulse]	1 byte	통신 상태		
4 bytes								
변경된 목표위치값[pulse]								
1 byte								
통신 상태								

0x3A (58)	FAS_ VelocityOverride	<p>운전중인 상태에서 운전 속도값[pps]의 변경을 의뢰함.</p> <p>송신 : 4 bytes</p> <table><tr><td>4 bytes</td></tr><tr><td>변경된 운전속도[pps]</td></tr></table> <p>답신 : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table>	4 bytes	변경된 운전속도[pps]	1 byte	통신 상태
4 bytes						
변경된 운전속도[pps]						
1 byte						
통신 상태						
0x3B (59)	FAS_ AllMoveStop	<p>동일 port 에 연결된 모든 운전중인 모터의 정지를 의뢰함.</p> <p>송신 : 0 byte (Slave 번호를 '99' 로 지정해야 합니다)</p> <p>답신 : 없음 (모든 Slave로부터 동시에 답신을 받을 수 없으므로 모든 Slave에서 답신을 보내지 않습니다)</p>				
0x3C (60)	FAS_ AllEmergencyStop	<p>동일 port 에 연결된 모든 운전중인 모터의 비상정지를 의뢰함.</p> <p>송신 : 0 byte (Slave 번호를 '99' 로 지정해야 합니다)</p> <p>답신 : 없음 (모든 Slave로부터 동시에 답신을 받을 수 없으므로 모든 Slave에서 답신을 보내지 않습니다)</p>				
0x3D (61)	FAS_All MoveOriginSingleAxis	<p>동일 port 에 연결된 모든 드라이브의 현재 설정된 파라미터 조건으로 원점복귀 운동 시작을 의뢰함.</p> <p>송신 : 0 byte (Slave 번호를 '99' 로 지정해야 합니다)</p> <p>답신 : 없음 (모든 Slave로부터 동시에 답신을 받을 수 없으므로 모든 Slave에서 답신을 보내지 않습니다)</p>				
0x3E (62)	FAS_All SingleAxisAbsPos	<p>동일 port 에 연결된 모든 드라이브에 절대값[pulse] 위치만큼의 이동 운전 시작을 의뢰함.</p> <p>송신 : 8 bytes (Slave 번호를 '99' 로 지정해야 합니다)</p> <table><tr><td>4 bytes</td><td>4 bytes</td></tr><tr><td>절대 위치값</td><td>운전속도[pps]</td></tr></table> <p>답신 : 없음 (모든 Slave로부터 동시에 답신을 받을 수 없으므로 모든 Slave에서 답신을 보내지 않습니다)</p>	4 bytes	4 bytes	절대 위치값	운전속도[pps]
4 bytes	4 bytes					
절대 위치값	운전속도[pps]					

0x3F (63)	FAS_All SingleAxisIncPos	<p>동일 port 에 연결된 모든 드라이브에 상대값[pulse] 위치만큼의 이동 운전 시작을 의뢰함.</p> <p>송신 : 8 bytes (Slave 번호를 '99' 로 지정해야 합니다)</p> <table><tr><td>4 bytes</td><td>4 bytes</td></tr><tr><td>상대 위치값</td><td>운전속도[pps]</td></tr></table> <p>답신 : 없음 (모든 Slave 로부터 동시에 답신을 받을 수 없으므로 모든 Slave 에서 답신을 보내지 않습니다)</p>	4 bytes	4 bytes	상대 위치값	운전속도[pps]								
4 bytes	4 bytes													
상대 위치값	운전속도[pps]													
0x80 (128)	FAS_ MoveSingleAxisAbsPosEx	<p>가감속값[msec]을 포함한 절대값[pulse] 위치만큼의 이동 운전 시작을 의뢰함.</p> <p>송신 : 40 bytes</p> <table><tr><td>4 bytes</td><td>4 bytes</td><td>2 bytes</td><td>2 bytes</td></tr><tr><td>절대 위치값</td><td>운전속도[pps]</td><td>Flag option</td><td>Custom Accel. Time (1~9999)</td></tr></table> <table><tr><td>2 bytes</td><td>26 bytes</td></tr><tr><td>Custom Decel. Time (1~9999)</td><td>Reserved</td></tr></table> <p>Flag option : 0x0001 : reserved 0x0002 : Custom Accel. time 값을 적용함. 0x0004 : Custom Decel. time 값을 적용함.</p> <p>해당 bit 값이 OFF 상태(0)인 경우에는 컨트롤러 내부에 저장된 값이 적용됩니다.</p> <p>답신 : 1 byte</p>	4 bytes	4 bytes	2 bytes	2 bytes	절대 위치값	운전속도[pps]	Flag option	Custom Accel. Time (1~9999)	2 bytes	26 bytes	Custom Decel. Time (1~9999)	Reserved
4 bytes	4 bytes	2 bytes	2 bytes											
절대 위치값	운전속도[pps]	Flag option	Custom Accel. Time (1~9999)											
2 bytes	26 bytes													
Custom Decel. Time (1~9999)	Reserved													
0x81 (129)	FAS_ MoveSingleAxisIncPosEx	<p>가감속값[msec]을 포함한 상대값[pulse] 위치만큼의 이동 운전 시작을 의뢰함.</p> <p>송신 : 40 bytes</p> <table><tr><td>4 bytes</td><td>4 bytes</td><td>2 bytes</td><td>2 bytes</td></tr><tr><td>상대 위치값</td><td>운전속도[pps]</td><td>Flag option</td><td>Custom Accel. Time (1~9999)</td></tr></table> <table><tr><td>2 bytes</td><td>26 bytes</td></tr><tr><td>Custom Decel. Time (1~9999)</td><td>Reserved</td></tr></table> <p>Flag option : 0x0001 : reserved 0x0002 : Custom Accel. time 값을 적용함. 0x0004 : Custom Decel. time 값을 적용함.</p> <p>해당 bit 값이 OFF 상태(0)인 경우에는 컨트롤러 내부에 저장된 값이 적용됩니다.</p> <p>답신 : 1 byte</p>	4 bytes	4 bytes	2 bytes	2 bytes	상대 위치값	운전속도[pps]	Flag option	Custom Accel. Time (1~9999)	2 bytes	26 bytes	Custom Decel. Time (1~9999)	Reserved
4 bytes	4 bytes	2 bytes	2 bytes											
상대 위치값	운전속도[pps]	Flag option	Custom Accel. Time (1~9999)											
2 bytes	26 bytes													
Custom Decel. Time (1~9999)	Reserved													

0x82 (130)	FAS_ MoveVelocityEx	<p>가감속값[msec]을 포함한 Jog 운동 시작을 의뢰함.</p> <p>송신 : 37 bytes</p> <table><tr><td>4 bytes</td><td>1 bytes</td><td>2 bytes</td></tr><tr><td>운전속도[pps]</td><td>운전방향 (0:-Jog 1:+Jog)</td><td>Flag option</td></tr></table> <table><tr><td>2 bytes</td><td>26 bytes</td></tr><tr><td>Custom Accel./Decel. Time (1~9999)</td><td>Reserved</td></tr></table> <p>Flag option : 0x0001 : reserved 0x0002 : Custom Accel./Decel. time 값을 적용함.</p> <p>해당 bit 값이 OFF 상태(0)인 경우에는 컨트롤러 내부에 저장된 값이 적용됩니다.</p> <p>답신 : 1 byte</p>	4 bytes	1 bytes	2 bytes	운전속도[pps]	운전방향 (0:-Jog 1:+Jog)	Flag option	2 bytes	26 bytes	Custom Accel./Decel. Time (1~9999)	Reserved		
4 bytes	1 bytes	2 bytes												
운전속도[pps]	운전방향 (0:-Jog 1:+Jog)	Flag option												
2 bytes	26 bytes													
Custom Accel./Decel. Time (1~9999)	Reserved													
0x40 (64)	FAS_ GetAxisStatus	<p>운전 상태를 표시해주는 Flag 값을 의뢰함.</p> <p>송신 : 0 byte 답신 : 5 byte</p> <table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>통신 상태</td><td>상태 Flag 값</td></tr></table> <p>각 Flag 별 해당 bit 는 「1-2-5. 상태 Flag 의 bit 설정」의 표를 참조.</p>	1 byte	4 bytes	통신 상태	상태 Flag 값								
1 byte	4 bytes													
통신 상태	상태 Flag 값													
0x41 (65)	FAS_ GetIOAxisStatus	<p>제어 입출력 상태와 운전 Flag 상태를 의뢰함. (Frame type 0x22, 0x23, 0x40 을 묶어 놓은것임)</p> <p>송신 : 0 byte 답신 : 13 byte</p> <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>	1 byte	4 bytes	4 bytes	4 bytes	통신 상태	입력 상태 값	출력 상태 값	상태 Flag 값				
1 byte	4 bytes	4 bytes	4 bytes											
통신 상태	입력 상태 값	출력 상태 값	상태 Flag 값											
0x42 (66)	FAS_ GetMotionStatus	<p>현재 운전 진행상황 및 운전중인 PT 번호를 의뢰함. (Frame type 0x51, 0x53, 0x54, 0x55 를 묶어 놓은것임)</p> <p>송신 : 0 byte 답신 : 21 byte</p> <table><tr><td>1 byte</td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td></tr><tr><td>통신 상태</td><td>Command Position 값</td><td>Actual Position 값</td><td>Position 차이값</td><td>운전 속도값</td><td>현재 운전중인 PT 번호</td></tr></table> <p>*Actual Position 값 : 외부 엔코더 연결시</p>	1 byte	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	통신 상태	Command Position 값	Actual Position 값	Position 차이값	운전 속도값	현재 운전중인 PT 번호
1 byte	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes									
통신 상태	Command Position 값	Actual Position 값	Position 차이값	운전 속도값	현재 운전중인 PT 번호									

0x43 (67)	FAS_ GetAllStatus	<p>현재의 운전 상태를 모두 포함하여 의뢰함 . (Frame type 0x41, 0x42 를 묶어 놓은것임)</p> <p>송신 : 0 byte 답신 : 33 byte</p> <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 차이값</td><td>운전 속도값</td><td>현재 운전중인 PT 번호</td></tr></table> <p>*Actual Position 값 : 외부 엔코더 연결시</p>	1 byte	4 bytes	4 bytes	4 bytes	통신 상태	입력 상태 값	출력 상태 값	상태 Flag 값	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	Command Position 값	Actual Position 값	Position 차이값	운전 속도값	현재 운전중인 PT 번호
1 byte	4 bytes	4 bytes	4 bytes																	
통신 상태	입력 상태 값	출력 상태 값	상태 Flag 값																	
4 bytes	4 bytes	4 bytes	4 bytes	4 bytes																
Command Position 값	Actual Position 값	Position 차이값	운전 속도값	현재 운전중인 PT 번호																
0x50 (80)	FAS_ SetCommandPos	<p>운전 목표 위치값을 설정합니다. 이 목표위치값 (Command position)을 운전 시작전에 설정한 후 추후 목표위치값의 변화를 확인할 수 있다.</p> <p>송신 : 4 bytes</p> <table><tr><td>4 bytes</td></tr><tr><td>목표 위치 설정 count 값</td></tr></table> <p>답신 : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table>	4 bytes	목표 위치 설정 count 값	1 byte	통신 상태														
4 bytes																				
목표 위치 설정 count 값																				
1 byte																				
통신 상태																				
0x51 (81)	FAS_ GetCommandPos	<p>현재 추종중인 목표위치(Command position) 값[pulse]을 의뢰함 .</p> <p>송신 : 0 byte 답신 : 5 byte</p> <table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>통신 상태</td><td>목표 위치값</td></tr></table>	1 byte	4 bytes	통신 상태	목표 위치값														
1 byte	4 bytes																			
통신 상태	목표 위치값																			
0x52 (82)	FAS_ SetActualPos	<p>외부 엔코더를 연결할 경우 페루프가 되므로, 운전중 실제 위치값이 계속 갱신된다. 이 실제위치값(Actual position)을 운전 시작전에 이 값으로 설정하여 추후 실제위치값의 변화를 확인할 수 있다. * 외부 엔코더 연결시</p> <p>송신 : 4 bytes</p> <table><tr><td>4 bytes</td></tr><tr><td>실제 위치 count 값</td></tr></table> <p>답신 : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table>	4 bytes	실제 위치 count 값	1 byte	통신 상태														
4 bytes																				
실제 위치 count 값																				
1 byte																				
통신 상태																				

0x53 (83)	FAS_ GetActualPos	<p>현재의 실제위치(Actual position) 값[pulse]을 의뢰함.</p> <p>* 외부 엔코더 연결시</p> <p>송신 : 0 byte</p> <p>답신 : 5 byte</p> <table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>통신 상태</td><td>실제 위치값</td></tr></table>	1 byte	4 bytes	통신 상태	실제 위치값
1 byte	4 bytes					
통신 상태	실제 위치값					
0x54 (84)	FAS_ GetPosError	<p>현재의 목표위치값(Command position)값과 실제위치값(Actual position)값의 차이값[pulse]을 의뢰함.</p> <p>* 외부 엔코더 연결시</p> <p>송신 : 0 byte</p> <p>답신 : 5 byte</p> <table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>통신 상태</td><td>위치 차이값</td></tr></table>	1 byte	4 bytes	통신 상태	위치 차이값
1 byte	4 bytes					
통신 상태	위치 차이값					
0x55 (85)	FAS_ GetActualVel	<p>현재의 운전 속도값[pps]을 의뢰함.</p> <p>송신 : 0 byte</p> <p>답신 : 5 byte</p> <table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>통신 상태</td><td>속도값</td></tr></table>	1 byte	4 bytes	통신 상태	속도값
1 byte	4 bytes					
통신 상태	속도값					
0x56 (86)	FAS_ ClearPosition	<p>목표위치값(Command position)과 실제위치값(Actual position)을 운전 시작전에 모두 ‘0’ 으로 설정해줌.</p> <p>송신 : 0 byte</p> <p>답신 : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table> <p>*Actual Position 값 : 외부 엔코더 연결시</p>	1 byte	통신 상태		
1 byte						
통신 상태						
0x58 (88)	FAS_ MovePause	<p>현재의 운전상태를 일시 정지 및 일시정지 해제를 의뢰함</p> <p>송신 : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>0:일시정지 해제, 1:일시 정지</td></tr></table> <p>답신 : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table>	1 byte	0:일시정지 해제, 1:일시 정지	1 byte	통신 상태
1 byte						
0:일시정지 해제, 1:일시 정지						
1 byte						
통신 상태						

0x60 (96)	FAS_ PosTableReadItem	<p>드라이브의 RAM 메모리 영역의 PT 항목값들을 읽어들이м.</p> <p>송신 : 2 bytes</p> <table><tr><td>2 bytes</td></tr><tr><td>읽어들일 PT 번호(0~255)</td></tr></table> <p>답신 : 65 byte</p> <table><tr><td>1 byte</td><td>64 bytes</td></tr><tr><td>통신 상태</td><td>해당 PT 의 항목값</td></tr></table> <p>각 PT 별 항목 내용은 「1-2-6. 포지션테이블 항목」 참조.</p>	2 bytes	읽어들일 PT 번호(0~255)	1 byte	64 bytes	통신 상태	해당 PT 의 항목값		
2 bytes										
읽어들일 PT 번호(0~255)										
1 byte	64 bytes									
통신 상태	해당 PT 의 항목값									
0x61 (97)	FAS_ PosTableWriteItem	<p>드라이브의 RAM 메모리 영역에 PT 항목값들을 저장함.</p> <p>송신 : 66 byte</p> <table><tr><td>2 bytes</td><td>64 bytes</td></tr><tr><td>PT 번호(0~255)</td><td>해당 PT 의 항목값</td></tr></table> <p>각 PT 별 항목 내용은 「1-2-6. 포지션테이블 항목」 참조.</p> <p>답신 : 2 byte</p> <table><tr><td>1 byte</td><td>1 bytes</td></tr><tr><td>통신 상태</td><td>명령 수행여부 (0 이외값:완료, 0:에러)</td></tr></table>	2 bytes	64 bytes	PT 번호(0~255)	해당 PT 의 항목값	1 byte	1 bytes	통신 상태	명령 수행여부 (0 이외값:완료, 0:에러)
2 bytes	64 bytes									
PT 번호(0~255)	해당 PT 의 항목값									
1 byte	1 bytes									
통신 상태	명령 수행여부 (0 이외값:완료, 0:에러)									
0x62 (98)	FAS_ PosTableReadROM	<p>드라이브의 ROM 메모리 영역의 모든(256 개) PT 항목값들을 읽어들이м.</p> <p>송신 : 0 byte</p> <p>답신 : 2 byte</p> <table><tr><td>1 byte</td><td>1 bytes</td></tr><tr><td>통신 상태</td><td>명령 수행여부 (0:완료, 0 이외값:에러)</td></tr></table>	1 byte	1 bytes	통신 상태	명령 수행여부 (0:완료, 0 이외값:에러)				
1 byte	1 bytes									
통신 상태	명령 수행여부 (0:완료, 0 이외값:에러)									
0x63 (99)	FAS_ PosTableWriteROM	<p>드라이브의 ROM 메모리 영역에 모든(256 개) PT 항목값들을 저장함.</p> <p>송신 : 0 byte</p> <p>답신 : 2 byte</p> <table><tr><td>1 byte</td><td>1 bytes</td></tr><tr><td>통신 상태</td><td>명령 수행여부 (0:완료, 0 이외값:에러)</td></tr></table>	1 byte	1 bytes	통신 상태	명령 수행여부 (0:완료, 0 이외값:에러)				
1 byte	1 bytes									
통신 상태	명령 수행여부 (0:완료, 0 이외값:에러)									
0x64 (100)	FAS_ PosTableRunItem	<p>지정된 PT 번호에서부터 포지션테이블 운전을 시작함.</p> <p>송신 : 2 byte</p> <table><tr><td>2 byte</td></tr><tr><td>PT 번호(0~255)</td></tr></table> <p>답신 : 1 byte</p> <table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table>	2 byte	PT 번호(0~255)	1 byte	통신 상태				
2 byte										
PT 번호(0~255)										
1 byte										
통신 상태										

0x6A (106)	FAS_ PosTableReadOneItem	<div>드라이브의 RAM 메모리 영역의 PT 항목중 특정값들 읽어들이м.</div> <div>송신 : 4 byte</div> <table><tr><td>2 byte</td><td>2 byte</td></tr><tr><td>읽어들일 PT 번호(0~255)</td><td>읽어들일 특정 항목의 Offset 값(0~40)</td></tr></table> <div>특정 항목 offset 값은 「1-2-6. 포지션테이블 항목」 참조.</div> <div>답신 : 5 byte</div> <table><tr><td>1 byte</td><td>4 byte</td></tr><tr><td>통신 상태</td><td>해당 항목의 값</td></tr></table>	2 byte	2 byte	읽어들일 PT 번호(0~255)	읽어들일 특정 항목의 Offset 값(0~40)	1 byte	4 byte	통신 상태	해당 항목의 값		
2 byte	2 byte											
읽어들일 PT 번호(0~255)	읽어들일 특정 항목의 Offset 값(0~40)											
1 byte	4 byte											
통신 상태	해당 항목의 값											
0x6B (107)	FAS_ PosTableWriteOneItem	<div>드라이브의 RAM 메모리 영역의 PT 항목중 특정값을 저장함.</div> <div>송신 : 8 byte</div> <table><tr><td>2 byte</td><td>2 byte</td><td>4 byte</td></tr><tr><td>저장할 PT 번호(0~255)</td><td>저장할 특정 항목의 Offset 값(0~40)</td><td>저장값</td></tr></table> <div>특정 항목 offset 값은 「1-2-6. 포지션테이블 항목」 참조.</div> <div>답신 : 2 byte</div> <table><tr><td>1 byte</td><td>1 byte</td></tr><tr><td>통신 상태</td><td>명령 수행여부 (0 이외값:완료, 0:에러)</td></tr></table>	2 byte	2 byte	4 byte	저장할 PT 번호(0~255)	저장할 특정 항목의 Offset 값(0~40)	저장값	1 byte	1 byte	통신 상태	명령 수행여부 (0 이외값:완료, 0:에러)
2 byte	2 byte	4 byte										
저장할 PT 번호(0~255)	저장할 특정 항목의 Offset 값(0~40)	저장값										
1 byte	1 byte											
통신 상태	명령 수행여부 (0 이외값:완료, 0:에러)											

* Frame Type '0x0E, 0x0F, 0x65 ~ 0x69' 는 내부 사용 목적으로 할당되어 있습니다.

1-2-2. 파라미터 목록표

번호	이름	단위	하한	상한	출하치
0	Pulse per Revolution		0	15	10
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	35,000	1
8	Jog Acc Dec Time	[msec]	1	9,999	100
9	Alarm Logic		0	1	0
10	Run/Stop Logic		0	1	0
11	Alarm Reset Logic		0	1	0
12	S/W Limit Plus Value	[pulse]	-134,217,727	+134,217,727	+134,217,727
13	S/W Limit Minus Value	[pulse]	-134,217,727	+134,217,727	-134,217,727
14	S/W Limit Stop Method		0	1	1
15	H/W Limit Stop Method		0	1	1
16	Limit Sensor Logic		0	1	0
17	Org Speed	[pps]	1	500,000	5,000
18	Org Search Speed	[pps]	1	500,000	1,000
19	Org Acc Dec Time	[msec]	1	9,999	50
20	Org Method		0	2	0
21	Org Dir		0	1	0
22	Org Offset	[pulse]	-134,217,727	+134,217,727	0
23	Org Position Set	[pulse]	-134,217,727	+134,217,727	0
24	Org Sensor Logic		0	1	0
25	Stop current	[%]	20	100	50
26	Motion Dir		0	1	0
27	Limit Sensor Dir		0	1	0
28	Encoder Multiply Value		0	3	0
29	Encoder Dir		0	1	0
30	Pos. Value Counting Method		0	1	0

1-2-3. 제어 출력핀의 bit 설정

‘0x20’ 번 Frame type 에 대한 세부 설명입니다.

이 명령은 제어 출력단의 24 개의 신호 종류중 「User Output0」 ~ 「User Output8」 까지의 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 Output 1	0x00010000
reserved	0x00000002	reserved	0x00000200	User Output 2	0x00020000
Alarm	0x00000004	reserved	0x00000400	User Output 3	0x00040000
Run/Stop	0x00000008	reserved	0x00000800	User Output 4	0x00080000
Acc/Dec	0x00000010	PT Output0	0x00001000	User Output 5	0x00100000
ACK	0x00000020	PT Output1	0x00002000	User Output 6	0x00200000
END	0x00000040	PT Output2	0x00004000	User Output 7	0x00400000
AlarmBlink	0x00000080	User Output 0	0x00008000	User Output 8	0x00800000

【예 1】 User Output 5 의 단자대를 ON 시키고자할 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 시키고자할 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 입력단의 32 개의 신호에 대해서 적용됩니다. 실제 입력 신호가 없는 상태에서 신호가 입력된 것처럼 시험용으로 사용할 수 있습니다.

다음표는 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	AlarmReset	0x00010000	JPT input2	0x01000000
Limit-	0x00000002	PT A5	0x00000200	reserved	0x00020000	JPT Start	0x02000000
Origin	0x00000004	PT A6	0x00000400	Pause	0x00040000	User Input0	0x04000000
Clear Position	0x00000008	PT A7	0x00000800	Org Search	0x00080000	User Input1	0x08000000
PT A0	0x00000010	PT Start	0x00001000	Teaching	0x00100000	User Input2	0x10000000
PT A1	0x00000020	Stop	0x00002000	E-stop	0x00200000	User Input3	0x20000000
PT A2	0x00000040	Jog+	0x00004000	JPT input0	0x00400000	User Input4	0x40000000
PT A3	0x00000080	Jog-	0x00008000	JPT input1	0x00800000	User Input5	0x80000000

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

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

1-2-5. 상태 Flag 의 bit 설정

include folder 중 ‘motion_define.h’ 의 ‘EZISTEP_AXISSTATUS’ 구조체를 참조

Flag define 명	내용	해당 bit 위치
FFLAG_ERRORALL	여러 에러중 하나 이상의 에러가 발생함.	0X00000001
FFLAG_HWPOSILMT	+방향 리미트 센서가 ON 이 된경우	0X00000002
FFLAG_HWNEGALMT	-방향 리미트 센서가 ON 이 된경우	0X00000004
FFLAG_SWPOGILMT	+방향 프로그램 리미트를 초과한 경우	0X00000008
FFLAG_SWNEGALMT	-방향 프로그램 리미트를 초과한 경우	0X00000010
reserved		0X00000020
reserved		0X00000040
FFLAG_ERRSTEPALARM	Step Alarm(이하 8 가지)중 한가지 이상 발생.	0X00000080
FFLAG_ERROVERCURRENT	모터 구동소자에 과전류 이상 Alarm 발생.	0X00000100
FFLAG_ERROVERSPEED	모터의 속도가 3000[rpm]을 초과한 Alarm 발생.	0X00000200
FFLAG_ERRSPEED	모터가 펄스 입력에 정상적으로 추종하지 않는 Alarm 발생.	0X00000400
reserved		0X00000800
FFLAG_ERROVERHEAT	드라이브의 내부온도가 55° C 를 초과하는 Alarm 발생.	0X00001000
FFLAG_ERRREVWPR	모터의 역기전력 전압이 70V 를 초과하는 Alarm 발생.	0X00002000
FFLAG_ERRMOTORPOWER	모터 전압 이상 Alarm 발생.	0X00004000
FFLAG_ERRLOWPOWER	드라이브 전압 이상 Alarm 발생.	0X00008000
FFLAG_EMGSTOP	모터가 비상 정지 상태임.	0X00010000
FFLAG_SLOWSTOP	모터가 일반 정지 상태임.	0X00020000
FFLAG_ORIGINRETURNING	원점복귀 운전중임.	0X00040000
reserved		0X00080000
reserved		0X00100000
FFLAG_ALARMRESET	AlarmReset 명령이 실시된 상태임.	0X00200000
FFLAG_PTSTOPED	포지션테이블 운전이 종료된 상태임.	0X00400000
FFLAG_ORIGINSENSOR	원점센서가 ON 되어 있는 상태임.	0X00800000
FFLAG_ZPULSE	모터의 엔코더 신호중 Z 상신호가 ON 상태인 경우.	0X01000000
FFLAG_ORIGINRETOK	원점복귀 운전이 완료된 상황임.	0X02000000
FFLAG_MOTIONDIR	모터의 운전 방향 (+방향:OFF, -방향:ON)	0X04000000
FFLAG_MOTIONING	모터가 현재 운전중임.	0X08000000
FFLAG_MOTIONPAUSE	운전중 Pause 명령으로 정지 상태임.	0X10000000
FFLAG_MOTIONACCEL	가속구간의 운전중임.	0X20000000
FFLAG_MOTIONDECEL	감속구간의 운전중임.	0X40000000
FFLAG_MOTIONCONST	가/감속 구간이 아닌 정속도 운전중인 상태임.	0X80000000

1-2-6. 포지션테이블 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]	-134217728	+134217728
Low Speed	dwStartSpd	4 (unsigned)	4	[pps]	0	500000
High Speed	dwMoveSpd	4 (unsigned)	8	[pps]	0	500000
Accel. Time	wAccelRate	2 (unsigned)	12	[msec]	1	9999
Decel. Time	wDecelRate	2 (unsigned)	14	[msec]	1	9999
Command	wCommand	2 (unsigned)	16		0	9
Wait time	wWaitTime	2 (unsigned)	18	[msec]	0	600000
Continuous Action	wContinuous	2 (unsigned)	20		0	1
Jump Table No.	wBranch	2 (unsigned)	22		0 10000	255 10255
Jump PT 0	wCond_branch0	2 (unsigned)	24		0 10000	255 10255
Jump PT 1	wCond_branch1	2 (unsigned)	26		0 10000	255 10255
Jump PT 2	wCond_branch2	2 (unsigned)	28		0 10000	255 10255
Loop Count	wLoopCount	2 (unsigned)	30		0	100
Loop Jump Table No.	wBranchAfterLoop	2 (unsigned)	32		0 10000	255 10255
PT set	wPTSet	2 (unsigned)	34		0	15
Loop Counter Clear	wLoopCountCLR	2 (unsigned)	36		0	255
Compare Position	lTriggerPos	4 (signed)	38	[pulse]	-134217728	+134217728
Compare Width	wTriggerOnTime	2 (unsigned)	42	[msec]	1	9999
Blank		20 (unsigned)	44	0x00		

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

1-2-7. 모터 종류

앞 부분에 표시되는 숫자와 영문은 각 각 모터의 크기와 길이를 나타냅니다.

예 : 56XL : 56 각 모터의 Extra Long 사이즈의 모터.

뒷 부분은 모터의 제조사에 대한 정보로서 다음표를 참조하십시오.

표기	제조사
빈칸	JapanServo
SD	Sanyo Denki
POR	Portescap
NPM	NPM
FUL	Fulling
YK	Yunkong
MIN	Minebia
Lin	Linear Step 류

1.3 프로그램의 종류

Ezi-STEP Plus-R 을 사용하기 위한 프로그램 방법은 2 가지가 있습니다.

첫번째는 일반적으로 많이 사용하는 방법으로서 PC 의 window system 하에서 Visual C++ 언어를 사용하는 방법입니다. 이때에는 제품과 함께 제공된 라이브러리(「2. PC 프로그램용 라이브러리」를 참조)를 사용합니다.

두번째는 라이브러리 함수를 사용하지 않고 사용자가 직접 명령어 command (character 들)을 전송 하는 방법입니다. Low level 의 protocol program 을 사용자가 직접 작성해야 하며 주로 상위 제어기로 PLC 등을 사용할 경우에 적용됩니다.

구체적인 프로그램 방법은 제품과 함께 제공된 ‘ProtocolTest_PlusR.exe’ 라는 사용자 GUI 프로그램을 사용하여 시험할 수 있고, 「3. PLC 프로그램용 프로토콜」을 참조하십시오.

2. PC 프로그램용 라이브러리(Ver6)

2.1 라이브러리의 구성

이 라이브러리를 사용하기 위해서는 C++ header file(*.h)와 library file(*.lib or *.dll) 이 필요합니다. 이 파일들은 “[WWFASTECHWW EziMOTION PlusRWWincludeWW](#)” 에 있으며 개발용 source file 에 다음 내용을 포함시키십시오.

```
#include "WWFASTECHWW EziMOTION PlusRWWincludeWWFAS_EziMotionPlusR.h"
#include "WWFASTECHWW EziMOTION PlusRWWincludeWWReturnCodes_Define.h"
#include "WWFASTECHWW EziMOTION PlusRWWincludeWWMOTION_DEFINE.h"
#include "WWFASTECHWW EziMOTION PlusRWWincludeWWCOMM_Define.h"
```

또한 라이브러리 파일은 다음과 같습니다.

“[WWFASTECHWW EziMOTION PlusRWWincludeWWEziMotionPlusR.lib](#)”
 “[WWFASTECHWW EziMOTION PlusRWWincludeWWEziMotionPlusR.dll](#)”

이 라이브러리를 사용한 sample program source 등이

“[WWFASTECHWW EziMOTION PlusRWWExamplesWW](#)” 폴더에 포함되어 있습니다

(1) 다음의 표는 각 라이브러리 함수 사용시 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	함수가 정상적으로 명령을 수행하였습니다
입력에러	FMM_NOT_OPEN	1	잘못된 Port 번호를 입력하였습니다.
	FMM_INVALID_PORT_NUM	2	연결되지 않은 Port 번호입니다.
	FMM_INVALID_SLAVE_NUM	3	잘못된 Slave 번호를 입력하였습니다
운전에러	FMM_POSTABLE_ERROR	9	포지션테이블 읽기/쓰기중 에러가 발생하였습니다.
연결에러	FMC_DISCONNECTED	5	해당 Board 가 연결 해제 되었습니다
	FMC_TIMEOUT_ERROR	6	정해진 시간(100msec)동안 응답이 없습니다.
	FMC_CRCFAILED_ERROR	7	통신중 Checksum 에러가 발생하였습니다.
	FMC_RECVPACKET_ERROR	8	드라이브로부터 받은 패킷에서 프로토콜 레벨의 에러가 발생하였습니다..

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

구분	명칭	리턴값	내용
정상	FMP_OK	0	통신이 정상적으로 수행되었습니다.
입력에러	FMP_FRAMETYPEERROR	128	Board 가 인식하지 못하는 명령어 입니다.
	FMP_DATAERROR	129	입력한 데이터가 범위를 벗어났습니다.
운전에러	FMP_BUSMOTOR	133	현재 모터가 운전 중이므로 동작이 완료될 때 까지 대기 하십시오.
연결에러	FMP_PACKETERROR	130	드라이브가 받은 패킷에서 프로토콜 레벨의 에러가 발생하였습니다.
	FMP_PACKETCRCERROR	170	드라이브가 받은 패킷의 CRC 계산값이 일치하지 않습니다.

2.2 통신상태 표시용 창

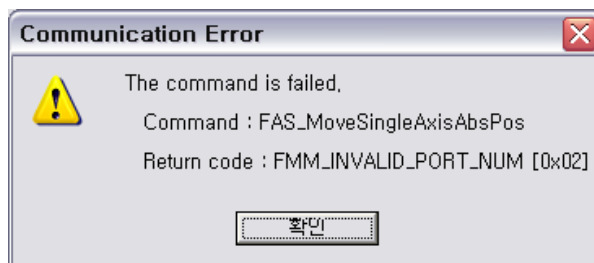
위의 통신 상태는 크게 3 가지 분류할 수 있다.

(1) Communication Error



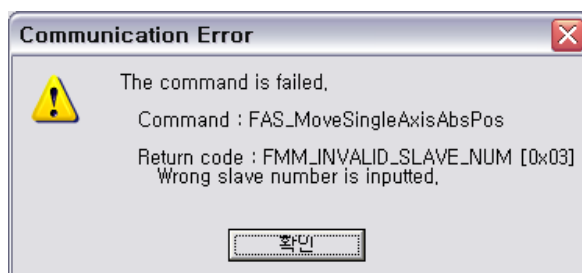
FMM_NOT_OPEN,

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



FMM_INVALID_PORT_NUM,

입력된 번호의 COM Port 는 연결되지 않은 Port 입니다. (GUI 에서는 발생할 수 없는 에러 입니다.)



FMM_INVALID_SLAVE_NUM,

해당 번호의 Slave 는 존재하지 않습니다. (GUI 에서는 발생할 수 없는 에러 입니다.)



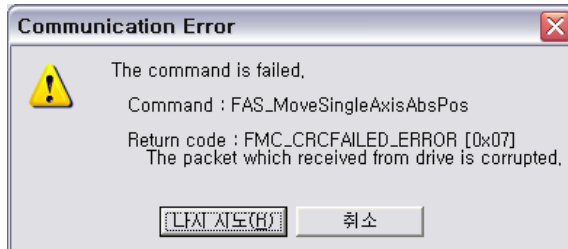
FMC_DISCONNECTED = 5,

해당 Port 가 연결이 끊어졌습니다.



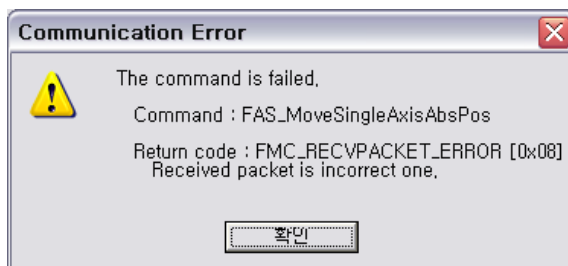
FMC_TIMEOUT_ERROR,

해당 명령의 응답이 도착하지 않았습니다.



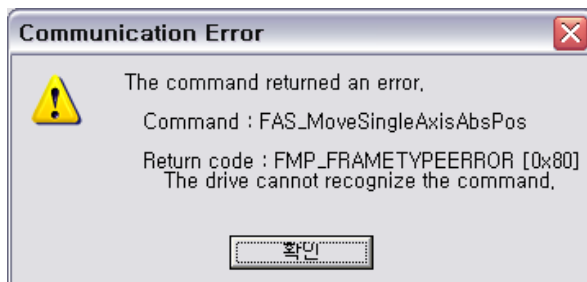
FMC_CRCFAILED_ERROR,

Drive 로부터 받은 Packet 의 CRC 가 일치하지 않습니다.



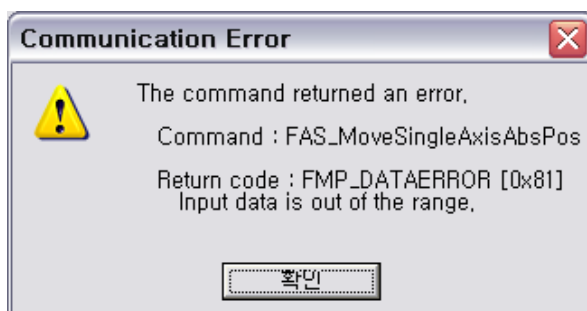
FMC_RECVPACKET_ERROR,

받은 Packet 의 길이가 일치하지 않습니다.



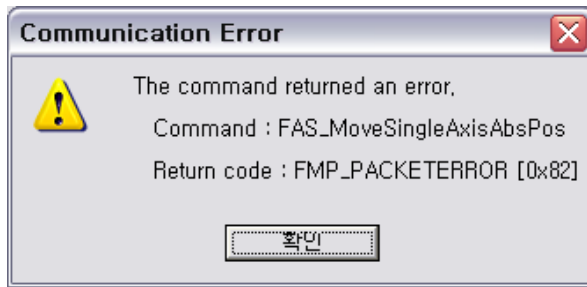
FMP_FRAMETYPEERROR = 0x80,

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



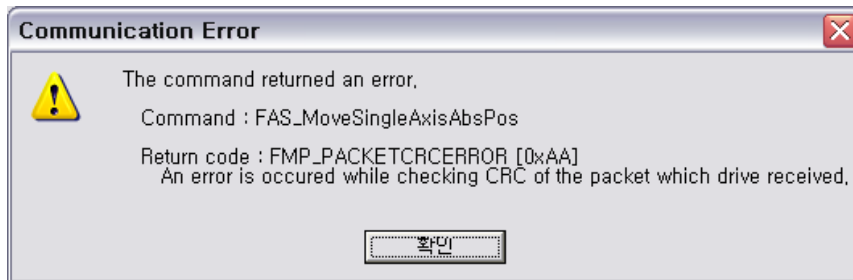
FMP_DATAERROR,

입력된 데이터의 범위가 Drive 가 지원하는 범위를 벗어났습니다.



FMP_PACKETERROR,

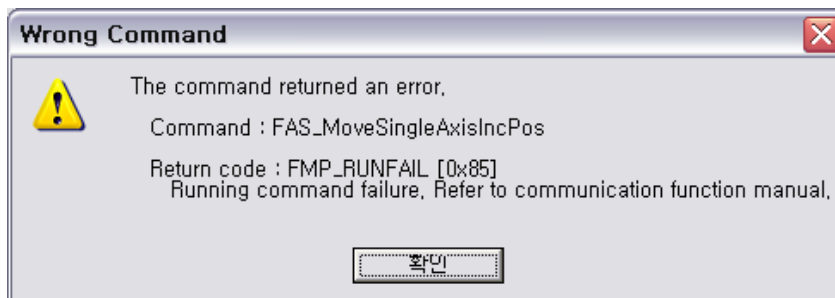
수신된 Frame 이 규격에 맞지 않는 데이터입니다.(Drive 로 보낸 Packet 의 길이가 일치하지 않습니다.)



FMP_PACKETCRCERROR = 0xAA,

Drive 로 보낸 Packet 의 CRC 계산이 일치하지 않습니다.

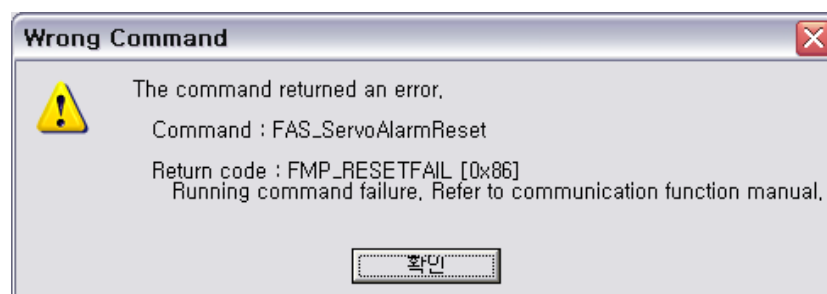
(2) Wrong Command



FMP_RUNFAIL = 0x85,

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

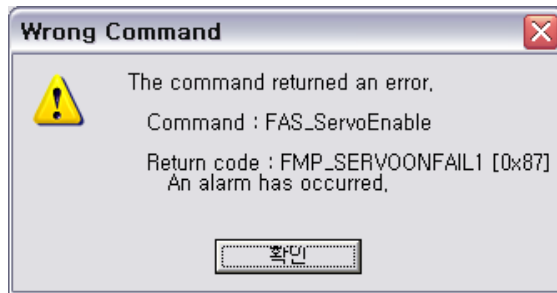
- . 현재 모터가 운전 중
- . 정지 명령 중
- . Servo OFF 상태
- . 외부 엔코더 없이 Z-pulse Origin 을 시도



FMP_RESETFAIL,

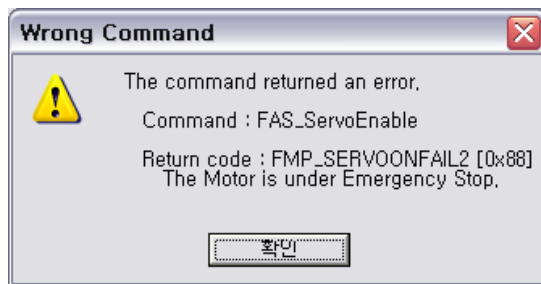
다음과 같은 상태에서 Reset 을 실행하려고 했습니다.

- Servo ON 상태
- 외부입력신호에 의해 이미 Reset 상태임



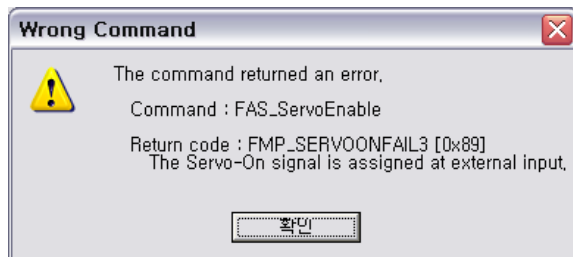
FMP_SERVOONFAIL1,

알람 발생 중에 Servo ON 명령을 실행하려고 했습니다.



FMP_SERVOONFAIL2,

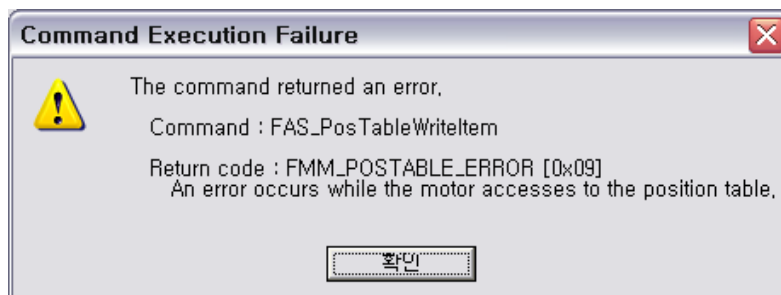
비상 정지 중에 Servo ON 명령을 실행하려고 했습니다.



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	드라이브 모듈과 통신 연결을 시도합니다 : 성공적으로 접속했다면 TRUE 를, 접속에 실패를 했다면 FALSE 를 리턴합니다.
FAS_Close	드라이브 모듈과 통신 해지를 시도합니다.
FAS_GetSlaveInfo	드라이브의 종류 와 프로그램 version 을 읽어들이니다 : 드라이브 종류와 version 정보를 리턴합니다.
FAS_GetMotorInfo	드라이브에 연결된 모터의 종류와 제조사에 대한 정보를 읽어들입니다.
FAS_IsSlaveExist	해당 드라이브의 존재 여부를 확인합니다. 현재 연결상태가 아닌 FAS_Connect 시의 존재 여부입니다. : 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_Connect

FAS_Connect 함수는 Ezi-STEP Plus-R 을 접속하는 함수이다.

Syntax

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

Parameters

nPortNo

접속하려는 Serial Port 번호를 선택한다.

dwBaud

Serial Port 의 Baudrate 를 입력한다.

Return Value

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

Remarks

Example

```
#include "FAS_EziMOTIONPlusR.h"

void funcInit()
{
    BYTE nPortNo = 1; // COMM Port 번호
    DWORD dwBaudrate = 115200; // Baudrate 값. (설정에 의해 변경될 수 있습니다)
    BYTE iSlaveNo = 0; // Slave No (0 ~ 15)
    char lpBuff[256];
    int nBuffSize = 256;
    BYTE nType;
    int nRtn;

    // 연결합니다.
    if (FAS_Connect(nPortNo, dwBaudrate) == FALSE)
    {
        // 연결이 실패하였습니다.
        // 연결된 Port 가 아니거나 Baudrate 가 틀릴 수 있습니다.
        return;
    }

    if (FAS_IsSlaveExist(nPortNo, iSlaveNo) == FALSE)
    {
        // 해당 Slave 번호는 존재하지 않습니다.
        // Ezi-STEP 의 Slave 번호를 확인하십시오.
        return;
    }

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

    printf("Port : %d (Slave %d) Wn", nPortNo, iSlaveNo);
```

```
printf("WtType : %d Wn", nType);  
printf("WtVersion : %d Wn", lpBuff);  
  
// 연결을 종료합니다.  
FAS_Close(nPortNo);
```

```
}
```

See Also

FAS_Close

FAS_Close

사용하던 Serial Port 를 연결 해제 한다.

Syntax

```
void FAS_Close(  
    BYTE nPortNo  
);
```

Parameters

nPortNo

연결을 해제할 Port 번호.

Remarks

Example

FAS_Connect 라이브러리 참조

See Also

FAS_Connect

FAS_GetSlaveInfo

해당 Board 의 Version 정보 문자열을 받아온다.

Syntax

```
int FAS_GetSlaveInfo(
    BYTE nPortNo,
    BYTE iSlaveNo,
    BYTE pType,
    LPSTR lpBuff,
    int nBuffSize
);
```

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

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 : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

Remarks

Example

FAS_Connect 라이브러리 참조

See Also

FAS_GetMotorInfo

해당 Board 에 연결된 모터의 정보 문자열을 받아온다.

Syntax

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

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

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 : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

Remarks

Example

FAS_Connect 라이브러리 참조

See Also

FAS_IsSlaveExist

현재 드라이브가 연결 상태인지를 확인한다.

Syntax

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

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

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_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-R 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 모든 Port 를 닫으면 Log 의 출력은 종료됩니다. Log 출력의 기본 설정 값은 TRUE 입니다.

Example

```
#include "FAS_ EziMOTIONPlusR.h"

void funcDisableLog()
{
    BYTE nPortNo = 1; // COMM Port 번호

    FAS_EnableLog(FALSE);

    // 이 후 함수들의 Log 는 출력되지 않습니다.

    // 연결을 시도합니다.
    if (FAS_Connect(nPortNo, 115200) == FALSE)
    {
        // 연결이 실패하였습니다.
        // 연결된 Port 가 아니거나 Baudrate 가 틀릴 수 있습니다.
        return;
    }

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

See Also

FAS_SetLogPath

FAS_SetLogPath

출력될 Log 가 저장될 경로를 설정합니다.

Syntax

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

Parameters

IpPath

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.

IpPath 값이 NULL 이거나 길이가 0 인 문자열을 입력할 경우 Log 경로는 현재 Ezi-MOTION Plus-R Library 를 사용하는 프로그램이 존재하는 폴더로 설정됩니다.

Log 경로의 기본값은 NULL 로서 현재 프로그램이 존재하는 폴더입니다.

Example

```
#include "FAS_ EziMOTIONPlusR.h"

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

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

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

    // 모든 함수들의 Log 는 C:\\WLogs 폴더에 출력됩니다.

    // 연결을 시도합니다.
    if (FAS_Connect(nPortNo, 115200) == FALSE)
    {
        // 연결이 실패하였습니다.
        // 연결된 Port 가 아니거나 Baudrate 가 틀릴 수 있습니다.
        return;
    }

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

See Also

FAS_EnableLog

2.4 파라미터 제어 함수

함수명	내용
FAS_SaveAllParameters	현재상태의 Parameter 들을 ROM 에 저장합니다 : 운전속도, 가감속 시간, 원점복귀 관련등의 파라미터를 드라이브 전원 OFF 후에도 보존되도록 저장합니다.
FAS_SetParameter	지정한 Parameter 값을 RAM 영역에 저장합니다 : 특정 파라미터 값을 저장합니다.
FAS_GetParameter	지정한 Parameter 값을 RAM 영역에서 읽어들입니다 : 특정 파라미터 값을 읽어 들입니다.
FAS_GetROMParameter	지정한 Parameter 값을 ROM 영역에서 읽어들입니다 : ROM 영역의 특정 파라미터 값을 읽어 들입니다.

FAS_SaveAllParameters

현재까지 수정된 모든 Parameter 값과 입출력신호의 할당값들을 ROM 영역에 저장한다.

Syntax

```
Int FAS_SaveAllParameters(
    BYTE nPortNo,
    BYTE iSlaveNo
);
```

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

Remarks

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

Example

```
#include "FAS_EziMOTIONPlusR.h"

void funcModifyParameter()
{
    BYTE nPortNo = 1; // COMM Port 번호
    BYTE iSlaveNo = 0; // Slave No (0 ~ 15)
    long IParamVal;
    int nRtn;

    // 연결합니다.
    if (FAS_Connect(nPortNo, 115200) == FALSE)
    {
        // 연결이 실패하였습니다.
        // 연결된 Port 가 아니거나 Baudrate 가 틀릴 수 있습니다.
        return;
    }

    // Axis Start Speed Parameter 값을 확인합니다.
    nRtn = FAS_GetParameter(nPortNo, iSlaveNo, STEP_AXISSTARTSPEED, &IParamVal);
    if (nRtn != FMM_OK)
    {
        // 명령이 정상적으로 수행되지 않았습니다.
        // ReturnCodes_Define.h 를 참조하십시오.
        _ASSERT(FALSE);
    }
    else
    {
        // 현재 Ezi-STEP 에 저장되어 있는 Parameter 값 입니다.
        printf("Parameter [before] : Start Speed = %d Wn", IParamVal);
    }
}
```

```

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

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

// ROM 에 저장되어 있는 값을 확인합니다.
nRtn = FAS_GetROMParameter(nPortNo, iSlaveNo, STEP_AXISSTARTSPEED, &IParmVal);
_ASSERT(nRtn == FMM_OK);    // 명령이 정상적으로 수행되지 않았다면 멈춥니다.
printf("Parameter [ROM] : Start Speed = %d Wn", IParmVal);

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

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

// 연결을 종료합니다.
FAS_Close(nPortNo);
}

```

See Also

FAS_GetRomParameter

FAS_SetParameter

해당 Parameter 값을 수정(RAM 영역에 저장)한다.

Syntax

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

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

iParamNo

수정 하려는 Parameter 의 번호.

IParamValue

수정하려는 Parameter 값.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

FMM_INVALID_PARAMETER_NUM : 지정한 iParamNo 의 Parameter 는 존재하지 않습니다.

Remarks

지정된 하나의 파라미터에 대해서만 작용합니다.

드라이브상의 파라미터는 2개의 메모리 영역에 저장하고 있습니다. 즉 ROM에는 전원 오프 시 영구히 저장하는 파라미터이고, 전원 온 시 ROM 의 파라미터를 DSP 의 RAM 상으로 복사 하여 사용합니다. 사용자가 파라미터의 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_GetParameter

Board 의 특정 Parameter 값을 불러온다.

Syntax

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

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

iParamNo

가져오려는 Parameter 의 번호.

lParamValue

Parameter 값.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

FMM_INVALID_PARAMETER_NUM : 지정한 iParamNo 의 Parameter 는 존재하지 않습니다.

Remarks

지정된 하나의 파라미터에 대해서만 작용합니다.

드라이브상의 파라미터는 2개의 메모리 영역에 저장하고 있습니다. 즉 ROM에는 전원 오프 시 영구히 저장하는 파라미터이고, 전원 온 시 ROM 의 파라미터를 DSP 의 RAM 상으로 복사 하여 사용합니다. 사용자가 파라미터의 변경 시 ROM 의 파라미터가 변경되는 것이 아니고, RAM 상의 파라미터가 변경됩니다. 이 함수는 RAM 영역의 지정된 번호의 파라미터를 읽어들이입니다.

Example

FAS_SaveAllParameter 라이브러리 참조

See Also

FAS_SetParameter

FAS_GetROMParameter

ROM 영역에 저장되어 있는 Parameter 를 불러온다.

Syntax

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

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

iParamNo

가져오려는 Parameter 의 번호.

lRomParam

ROM 에 저장되어 있던 Parameter 값.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 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 알람 제어 함수

함수명	내용
FAS_StepAlarmReset	알람이 발생한 드라이브의 알람을 해제시킵니다 : 알람이 발생한 원인을 제거한 후 실시하십시오.
FAS_GetAlarmType	현재 알람의 발생 여부 및 알람의 종류를 확인합니다.

FAS_StepAlarmReset

AlarmReset 명령을 보낸다.

Syntax

```
int FAS_StepAlarmReset(
    BYTE nPortNo,
    BYTE iSlaveNo,
    BYTE bReset
);
```

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

bReset

Reset 명령 (1: reset, 0:reset 해제)

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

Remarks

이 명령을 보내기 전에 알람이 발생한 원인을 먼저 제거하십시오.
알람의 원인에 대하여는 ‘사용자 매뉴얼_본문편’ 을 참조하십시오.

알람의 해제를 위해서는 두번의 명령이 필요합니다.

즉 bReset 값을 ‘1’ 과 ‘0’ 을 차례대로 실행해야 합니다.

‘1’ 의 명령만 실행하면 모터의 lock 상태가 해제된 상태입니다.

Example

See Also

2.6 제어 입출력 함수

함수명	내용
FAS_SetI0Input	제어 입력단의 입력 신호 레벨을 설정합니다 : 입력 신호를 [ON] 또는 [OFF] 상태로 만들어 줍니다.
FAS_GetI0Input	제어 입력단의 현재 입력신호 상태를 읽어 들입니다 : 각 입력 신호에 대해 bit 단위로 리턴합니다.
FAS_SetI0Output	제어 출력단의 출력 신호 레벨을 설정합니다 : 출력 신호를 [ON] 또는 [OFF] 상태로 만들어 줍니다.
FAS_GetI0Output	제어 출력단의 현재 입력신호 상태를 읽어 들입니다 : 각 출력 신호에 대해 bit 단위로 리턴합니다.
FAS_GetI0AssignMap	CN1 단자대의 pin 의 설정상태를 읽어들입니다 : 입력 및 출력단의 가변 설정 가능한 9 개씩의 신호에 대한 설정 상태 값등을 bit 단위로 리턴합니다.
FAS_SetI0AssignMap	제어 입출력 신호를 CN1 단자대의 pin 에 할당하며 동시에 신호 level 을 설정해줍니다 : 입력 및 출력단의 가변 설정 가능한 9 개씩의 신호에 대한 설정등을 명령 합니다.
FAS_I0AssignMapReadROM	제어 입력 및 출력단의 설정상태와 신호레벨 상태값을 ROM 영역으로부터 RAM 영역으로 읽어 들입니다

FAS_SetIOInput

I/O Input 을 설정한다. 자세한 설정 방법은 ‘1-1-5. Frame type 과 Data 의 구성’ 을 참조하십시오.

Syntax

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

Parameters

nPortNo
해당 Board 의 Port 번호.

iSlaveNo
해당 Board 의 Slave 번호.

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 : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

Remarks

dwIOSetMask 와 dwIOCLRMask 의 bit 값이 중복되지 않도록 주의하십시오.

Example

```
#include "FAS_EziMOTIONPlusR.h"

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

    // 연결합니다.
    if (FAS_Connect(nPortNo, 115200) == FALSE)
    {
        // 연결이 실패하였습니다.
        // 연결된 Port 가 아니거나 Baudrate 가 틀릴 수 있습니다.
        return;
    }

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

```

    }

    if (dwInput & STEP_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(nPortNo, iSlaveNo, STEP_IN_BITMASK_CLEARPOSITION |
STEP_IN_BITMASK_USERIN1, STEP_IN_BITMASK_PJOG);
    _ASSERT(nRtn == FMM_OK);

    // I/O Output 을 확인합니다.
    nRtn = FAS_GetIOOutput(nPortNo, iSlaveNo, &dwOutput);
    _ASSERT(nRtn == FMM_OK);
    if (dwOutput & STEP_OUT_BITMASK_USEROUT0)
    {
        // User Output 0 신호가 ON 되어 있습니다.
    }

    // User Output 1 과 2 신호를 OFF 합니다.
    nRtn = FAS_SetIOOutput(nPortNo, iSlaveNo, 0, STEP_OUT_BITMASK_USEROUT1 |
STEP_OUT_BITMASK_USEROUT2);
    _ASSERT(nRtn == FMM_OK);

    // 연결을 종료합니다.
    FAS_Close(nPortNo);
}

```

See Also

FAS_GetIOInput

FAS_GetIOInput

I/O Input 값을 읽어온다. 자세한 설정 방법은 ‘1-1-5. Frame type 과 Data 의 구성’ 을 참조하십시오.

Syntax

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

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

dwIOInput

Input 값이 저장될 변수 포인터.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

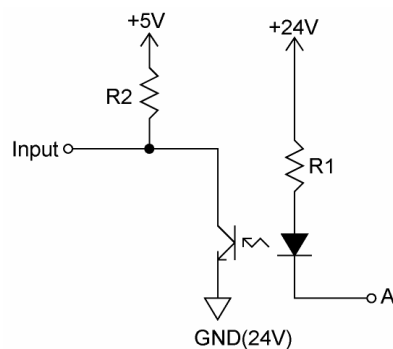
FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

Remarks

EziSTEP PlusR 에는 총 12 개의 입력단이 있으며 이중에서 9 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)



외부입력으로부터 A 점의 전압이 24V 이면 Input 은 5V(High)로 인식된다

Example

FAS_SetIOInput 라이브러리를 참조

See Also

FAS_SetIOInput

FAS_SetIOOutput

I/O Output 값을 설정한다. 자세한 설정 방법은 ‘1-1-5. Frame type 과 Data 의 구성’ 을 참조하십시오.

Syntax

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

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

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.

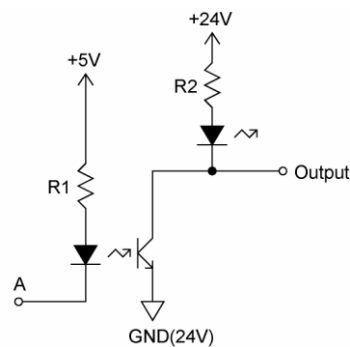
FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

Remarks

EziSTEP PlusR 에는 총 10 개의 출력단이 있으며 이중에서 9 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_GetI0Output

I/O Output 값을 읽어온다. 자세한 설정 방법은 ‘1-1-5. Frame type 과 Data 의 구성’ 을 참조하십시오.

Syntax

```
int FAS_GetI0Output(
    BYTE nPortNo,
    BYTE iSlaveNo,
    DWORD* dwI0Output
);
```

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

dwI0Input

Output 값이 저장될 변수 포인터.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

Remarks

Example

FAS_SetI0Input 라이브러리를 참조

See Also

FAS_SetI0Output

FAS_GetIOAssignMap

I/O Assign Map 을 읽어온다. 자세한 설정 방법은 ‘1-1-5. Frame type 과 Data 의 구성’ 을 참조하십시오.

Syntax

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

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

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 : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

Remarks

nIOLogic 에 대해서는 ‘Motion_define.h’ 를 참조하십시오.

Example

```
#include "FAS_EziMOTIONPlusR.h"

void funcIOAssign()
{
    BYTE nPortNo = 1; // COMM Port 번호
    BYTE iSlaveNo = 0; // Slave No (0 ~ 15)
    BYTE iPinNo;
    DWORD dwLogicMask;
    BYTE bLevel;
    BYTE i;
    int nRtn;

    // 연결합니다.
    if (FAS_Connect(nPortNo, 115200) == FALSE)
    {
        // 연결이 실패하였습니다.
        // 연결된 Port 가 아니거나 Baudrate 가 틀릴 수 있습니다.
        return;
    }
}
```

```

// Input pin 할당 정보를 확인합니다.
for (i=0; i</*Input Pin Count*/12; i++)
{
    nRtn = FAS_GetIOAssignMap(nPortNo, iSlaveNo, i, &dwLogicMask, &bLevel);
    _ASSERT(nRtn == FMM_OK);

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

// Input pin 3에 E-Stop Logic (Low Active)을 할당함.
iPinNo = 3;          // 0 ~ 11의 값이 가능함 (주의 : 0 ~ 2는 고정되어 있음).
nRtn = FAS_SetIOAssignMap(nPortNo, iSlaveNo, iPinNo, STEP_IN_BITMASK_ESTOP,
LEVEL_LOW_ACTIVE);
_ASSERT(nRtn == FMM_OK);

// Output pin 할당 정보를 확인합니다.
for (i=0; i<10/*Output Pin Count*/; i++)
{
    nRtn = FAS_GetIOAssignMap(nPortNo, iSlaveNo, 12/*Input Pin Count*/ + i,
&dwLogicMask, &bLevel);
    _ASSERT(nRtn == FMM_OK);

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

// Output pin 9에 ALARM Logic (High Active)을 할당함.
iPinNo = 9;          // 0 ~ 9의 값이 가능함 (주의 : 0은 COMPOUT으로 고정되어
있음).
nRtn = FAS_SetIOAssignMap(nPortNo, iSlaveNo, 12/*Input Pin Count*/ + iPinNo,
STEP_OUT_BITMASK_ALARM, LEVEL_HIGH_ACTIVE);
_ASSERT(nRtn == FMM_OK);

// 연결을 종료합니다.
FAS_Close(nPortNo);
}

```

See Also

FAS_SetIOAssignMap

FAS_SetIOAssignMap

I/O Assign Map 을 설정한다. 자세한 설정 방법은 ‘1-1-5. Frame type 과 Data 의 구성’ 을 참조하십시오.

Syntax

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

Parameters

nPortNo
해당 Board 의 Port 번호.

iSlaveNo
해당 Board 의 Slave 번호.

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 : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 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_I0AssignMapReadROM

현재 ROM 영역에 저장된 입출력 설정상태 및 신호레벨 값들을 읽어들인다.

Syntax

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

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 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 설정합니다.
FAS_SetActualPos	현재의 위치를 실제(Actual) 위치값으로 설정합니다.
FAS_GetCommandPos	현재의 목표(Command) 위치값을 읽어들입니다.
FAS_GetActualPos	현재의 실제(Actual) 위치값을 읽어들입니다.
FAS_GetPosError	현재의 실제(Actual) 위치값과 목표위치(Command)값의 차이를 읽어들입니다.
FAS_GetActualVel	현재 이동중인 운전의 실제 운전 속도값을 읽어 들입니다.
FAS_ClearPosition	목표(Command) 위치값과 실제(Actual) 위치값을 '0' 으로 설정합니다.

FAS_SetCommandPos

Motor 의 Command Position 값을 설정한다.

Syntax

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

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

lCmdPos

설정할 Command Position 값.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

Remarks

위치 지령 (펄스 출력 카운터) 값을 사용자가 설정한다.

주로 현재의 위치를 사용자가 원하는 좌표 값으로 설정 시 사용된다

Example

```
#include "FAS_ EzIMOTIONPlusR.h"

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

    // 연결합니다.
    if (FAS_Connect(nPortNo, 115200) == FALSE)
    {
        // 연결이 실패하였습니다.
        // 연결된 Port 가 아니거나 Baudrate 가 틀릴 수 있습니다.
        return;
    }

    // Command Position, Actual Position 값을 초기화합니다.
    nRtn = FAS_SetCommandPos(nPortNo, iSlaveNo, 0);
    _ASSERT(nRtn == FMM_OK);
    nRtn = FAS_SetActualPos(nPortNo, iSlaveNo, 0);
    _ASSERT(nRtn == FMM_OK);

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

See Also

FAS_SetActualPos

FAS_SetActualPos

Motor 의 Actual Position 값을 설정한다.

Syntax

```
int FAS_SetActualPos(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long lActPos
);
```

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

lActPos

설정할 Actual Position 값.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

Remarks

외부 엔코더 연결시 사용할 수 있습니다.

엔코더 피드백 카운터 값을 사용자가 원하는 값으로 설정한다.

Example

FAS_GetActualPos 라이브러리 참조

See Also

FAS_SetCommandPos

FAS_GetCommandPos

현재 Motor 의 Command Position 값을 읽어온다.

Syntax

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

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

lCmdPos

Command Position 값이 저장될 변수 포인터.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

Remarks

위치 지령 (펄스 출력 카운터) 값을 읽어 온다

Example

```
#include "FAS_ EziMOTIONPlusR.h"

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

    // 연결합니다.
    if (FAS_Connect(nPortNo, 115200) == FALSE)
    {
        // 연결이 실패하였습니다.
        // 연결된 Port 가 아니거나 Baudrate 가 틀릴 수 있습니다.
        return;
    }

    // Ezi-STEP 의 Position 에 관한 정보를 확인합니다.
    nRtn = FAS_GetCommandPos(nPortNo, iSlaveNo, &lValue);
    _ASSERT(nRtn == FMM_OK);
    printf("CMDPOS : %d Wn", lValue);
    nRtn = FAS_GetActualVel(nPortNo, iSlaveNo, &lValue);
    _ASSERT(nRtn == FMM_OK);
    printf("ACTVEL : %d Wn", lValue);

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


See Also

FAS_GetActualPos

FAS_GetActualPos

현재 Motor 의 Actual Position 값을 읽어온다.

Syntax

```
int FAS_GetActualPos(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long* lActPos
);
```

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

lActPos

Actual Position 값이 저장될 변수 포인터.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

Remarks

외부 엔코더 연결시 사용할 수 있습니다.

주로 위치 결정 완료 후 실제의 위치를 확인 시 사용한다.

Example

FAS_GetCommandPosition 라이브러리 참조

See Also

FAS_GetCommandPos

FAS_GetPosError

Motor 의 Position Error 값을 읽어온다.

Syntax

```
int FAS_GetPosError(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long* lPosErr
);
```

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

lPosErr

Position Error 값이 저장될 변수 포인터.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

Remarks

외부 엔코더 연결시 사용할 수 있습니다.

Example

FAS_GetCommandPosition 라이브러리 참조

See Also

FAS_GetCommandPos,

FAS_GetActualPos

FAS_GetActualVel

Motor 의 Actual Velocity 값을 읽어온다.

Syntax

```
int FAS_GetActualVel(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long* lActVel
);
```

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

lActVel

Actual Velocity 값이 저장될 변수 포인터.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

Remarks

Example

FAS_GetCommandPosition 라이브러리 참조

See Also

FAS_ClearPosition

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

Syntax

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

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

Remarks

위치 값을 사용자가 설정한다.

주로 시스템 초기화시에 사용된다

Example

```
#include "FAS_ EzIMOTIONPlusR.h"

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

    // 연결합니다.
    if (FAS_Connect(nPortNo, 115200) == FALSE)
    {
        // 연결이 실패하였습니다.
        // 연결된 Port 가 아니거나 Baudrate 가 틀릴 수 있습니다.
        return;
    }

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

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

See Also

FAS_SetActualPos, FAS_SetCommandPos

2.8 드라이브 상태 제어 함수

함수명	내용
FAS_GetIOAxisStatus	제어 입출력 상태와 운전 상태 flag 값을 읽어 들입니다 : 현재의 입력 상태값, 출력 설정 상태값 및 운전상태 Flag 값을 리턴합니다.
FAS_GetMotionStatus	현재 운전 진행상황 및 운전중인 PT 번호를 읽어들입니다 : Command position, Actual position, 속도값등을 리턴합니다.
FAS_GetAllStatus	현재 상태를 모두 포함하여 한꺼번에 읽어 들입니다 : 'FAS_GetIOAxisStatus' 함수와 'FAS_GetMotionStatus' 함수를 결합한 것입니다.
FAS_GetAxisStatus	해당 드라이브의 운전 상태 flag 값을 읽어 들입니다.

FAS_GetIOAxisStatus

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

Syntax

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

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

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 : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

Remarks

Example

FAS_MoveSingleAxisAbsPos 라이브러리를 참조

See Also

FAS_GetMotionStatus

현재 Motor 의 Motion Status 들을 한번에 읽어온다.

Syntax

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

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

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 : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 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 읽어온다.

Syntax

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

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

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 : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

Remarks

Example

FAS_MoveSingleAxisAbsPos 라이브러리를 참조

See Also

FAS_GetAxisStatus

FAS_GetMotionStatus

FAS_GetAxisStatus

Motor 의 Axis Status 값을 읽어온다. Status Flag 값에 대한 설명은 ‘1-1-5. Frame type 과 Data 의 구성’ 을 참조하십시오.

Syntax

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

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

dwAxisStatus

해당 Motor 의 Axis Status 값이 저장될 변수 포인터.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 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	원점복귀 운전을 시작합니다.
FAS_MoveSingleAxisAbsPos	주어진 절대위치(Absolute)값 만큼 운전을 실시합니다.
FAS_MoveSingleAxisIncPos	주어진 상대위치(Incremental)값 만큼 운전을 실시합니다.
FAS_MoveToLimit	리미트 센서가 감지되는 위치까지 운전을 실시합니다.
FAS_MoveVelocity	주어진 속도와 방향으로 운전을 시작합니다 : Jog 운전등에 사용됩니다.
FAS_PositionAbsOverride	운전중인 상태에서 목표 절대위치값[pulse]을 변경합니다.
FAS_PositionIncOverride	운전중인 상태에서 목표 상대위치값[pulse]을 변경합니다.
FAS_VelocityOverride	운전중인 상태에서 운전 속도값[pps]을 변경 합니다.
FAS_AllMoveStop	동일 port 에 연결된 모든 운전중인 모터를 감속하면서 정지시킵니다
FAS_AllEmergencyStop	동일 port 에 연결된 모든 운전중인 모터를 감속없이 정지시킵니다
FAS_AllMoveOriginSingleAxis	동일 port 에 연결된 모든 모터에 원점복귀 운전을 시작합니다
FAS_AllMoveSingleAxisAbsPos	동일 port 에 연결된 모든 모터에 주어진 절대위치(Absolute)값 만큼 운전을 실시합니다.
FAS_AllMoveSingleAxisIncPos	동일 port 에 연결된 모든 모터에 주어진 상대위치(Increment)값 만큼 운전을 실시합니다.
FAS_MoveLinearAbsPos	동일 port 에 연결된 2 개의 모터간에 주어진 절대위치(Absolute)값 만큼 Linear Interpolation 운전을 실시합니다
FAS_MoveLinearIncPos	동일 port 에 연결된 2 개의 모터간에 주어진 상대위치(Incremental)값 만큼 Linear Interpolation 운전을 실시합니다
FAS_MoveSingleAxisAbsPosEx	주어진 절대위치(Absolute)값 만큼 운전을 실시합니다. 가속 및 감속 시간을 설정할 수 있습니다.
FAS_MoveSingleAxisIncPosEx	주어진 상대위치(Incremental)값 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 일시정지 상태에서의 운전 재개를 실시합니다.

FAS_IsMotioning

현재 Motor 가 동작 중인지를 확인한다.

Syntax

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

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

Return Value

TRUE : 운전중인 상태.

FALSE : 정지상태.

Remarks

FAS_GetAxisStatus 함수의 flag bit 들중 Motioning, Motion Accel, Motion Decel, motion Continuous 들을 확인하여 결과값을 리턴합니다.

Example

FAS_MoveSingleAxisAbsPos 라이브러리를 참조

See Also

FAS_GetAxisStatus

FAS_MoveStop

Motor 를 정지시킨다.

Syntax

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

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

Remarks

Example

FAS_MoveSingleAxisAbsPos 라이브러리를 참조

See Also

FAS_EmergencyStop

Motor 를 급정지 시킨다.

Syntax

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

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

Remarks

이 함수는 감속 단계가 없으므로 기계계의 충격에 주의하십시오.

Example

FAS_MoveSingleAxisAbsPos 라이브러리를 참조

See Also

FAS_MoveOriginSingleAxis

시스템의 원점(Origin)을 찾는다.

자세한 사항은 ‘[사용자 매뉴얼_본문편의 9.3 원점복귀](#)’ 항목을 참조하십시오.

Syntax

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

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

Remarks

Example

FAS_MoveSingleAxisAbsPos 라이브러리를 참조

See Also

FAS_MoveSingleAxisAbsPos

Motor 를 절대좌표 값으로 이동시킨다.

Syntax

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

Parameters

nPortNo
해당 Board 의 Port 번호.

iSlaveNo
해당 Board 의 Slave 번호.

lAbsPos
이동할 위치의 절대좌표 값.

lVelocity
이동 시 속도 값.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

Remarks

Example

```
#include "FAS_EziMOTIONPlusR.h"

void funcMove()
{
    BYTE nPortNo = 1; // COMM Port 번호
    BYTE iSlaveNo = 0; // Slave No (0 ~ 15)
    DWORD dwAxisStatus, dwInput;
    EZISTEP_AXISSTATUS stAxisStatus;
    long lAbsPos, lIncPos, lVelocity;
    int nRtn;

    // 연결합니다.
    if (FAS_Connect(nPortNo, 115200) == FALSE)
    {
        // 연결이 실패하였습니다.
        // 연결된 Port 가 아니거나 Baudrate 가 틀릴 수 있습니다.
        return;
    }

    // Error 상태를 체크합니다.
    nRtn = FAS_GetAxisStatus(nPortNo, iSlaveNo, &dwAxisStatus);
    _ASSERT(nRtn == FMM_OK);
    stAxisStatus.dwValue = dwAxisStatus;
```



```

//if (dwAxisStatus & 0x00000001)
if (stAxisStatus.FFLAG_ERRORALL)
    FAS_StepAlarmReset(nPortNo, iSlaveNo);

// Input 상태를 확인합니다.
nRtn = FAS_GetIOInput(nPortNo, iSlaveNo, &dwInput);
_ASSERT(nRtn == FMM_OK);

if (dwInput & (STEP_IN_BITMASK_STOP | STEP_IN_BITMASK_PAUSE | STEP_IN_BITMASK
_ESTOP))
    FAS_SetIOInput(nPortNo, iSlaveNo, 0, STEP_IN_BITMASK_STOP | STEP_IN_
BITMASK_PAUSE | STEP_IN_BITMASK_ESTOP);

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

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

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

// Motor 를 0 으로 옮깁니다.
lAbsPos = 0;
lVelocity = 20000;
nRtn = FAS_MoveSingleAxisAbsPos(nPortNo, iSlaveNo, lAbsPos, lVelocity);
_ASSERT(nRtn == FMM_OK);

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

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

// 연결을 종료합니다.
FAS_Close(nPortNo);
}

```

See Also

FAS_MoveSingleAxisIncPos

Motor 를 상대좌표 값으로 이동시킨다.

Syntax

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

Parameters

nPortNo
해당 Board 의 Port 번호.

iSlaveNo
해당 Board 의 Slave 번호.

lIncPos
이동할 위치의 상대좌표 값.

lVelocity
이동 시 속도 값.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

Remarks

Example

FAS_MoveSingleAxisAbsPos 라이브러리를 참조

See Also

FAS_MoveToLimit

Motor 에게 해당 Limit 를 찾도록 명령한다.

Syntax

```
int FAS_MoveToLimit(
    BYTE nPortNo,
    BYTE iSlaveNo,
    DWORD lVelocity,
    int iLimitDir,
);
```

Parameters

nPortNo
해당 Board 의 Port 번호.

iSlaveNo
해당 Board 의 Slave 번호.

lVelocity
이동 시 속도 값.

iLimitDir
찾아갈 Limit 의 방향.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

Remarks

Example

FAS_MoveSingleAxisAbsPos 라이브러리를 참조

See Also

FAS_MoveVelocity

Motor 를 해당 방향, 해당 속도로 이동시킨다. Jog 운전시 사용된다.

Syntax

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

Parameters

nPortNo
해당 Board 의 Port 번호.

iSlaveNo
해당 Board 의 Slave 번호.

lVelocity
이동 시 속도 값.

iVelDir
이동 할 방향.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

Remarks

Example

FAS_MoveSingleAxisAbsPos 라이브러리를 참조

See Also

FAS_PositionAbsOverride

Motor 의 절대위치 이동 중 설정되었던 절대 위치 값을 변경한다.

Syntax

```
int FAS_PositionAbsOverride(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long lOverridePos
);
```

Parameters

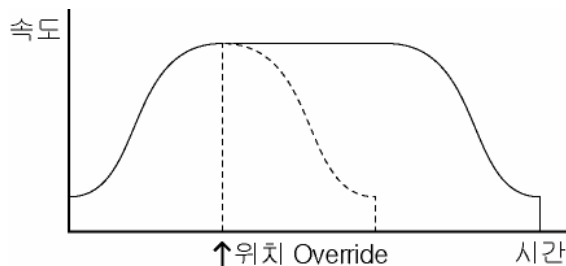
nPortNo
해당 Board 의 Port 번호.
iSlaveNo
해당 Board 의 Slave 번호.
lOverridePos
변경할 절대좌표 위치값..

Return Value

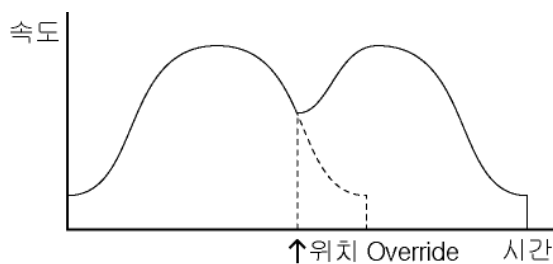
FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.
FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.
FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

Remarks

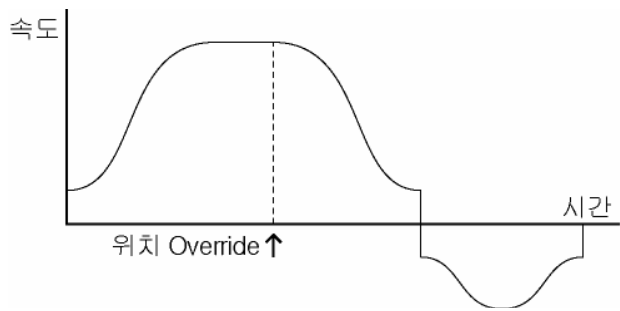
- 1) 가속 혹은 등속 중에 목표위치를 원래의 목표 위치보다 먼 좌표로 설정하면 속도 패턴은 그 때까지의 속도 패턴으로 동작하여 변경한 목표 위치에서 정지합니다.



- 2) 감속 중에 목표위치를 변경하면 속도 패턴은 다시 등속속도 까지 가속한 후 변경한 목표 위치에서 정지합니다.



- 3) 변경한 목표 위치가 원래의 목표 위치 보다 가까운 곳에 설정한 경우는 변경된 목표 위치로 이동한후 정지 합니다.



Example

FAS_MoveSingleAxisAbsPos 라이브러리를 참조

See Also

FAS_PositionIncOverride

FAS_PositionIncOverride

Motor 의 상대위치 이동 중 설정되었던 상대 위치 값을 변경한다.

Syntax

```
int FAS_PositionIncOverride(
    BYTE nPortNo,
    BYTE iSlaveNo,
    long lOverridePos
);
```

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

lOverridePos

변경할 상대좌표 위치값..

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

Remarks

FAS_PositionAbsOverride 라이브러리를 참조

Example

FAS_MoveSingleAxisAbsPos 라이브러리를 참조

See Also

FAS_PositionAbsOverride

FAS_VelocityOverride

Motor 의 이동 중 설정 되었던 속도 값을 변경한다.

Syntax

```
int FAS_VelocityOverride(
    BYTE nPortNo,
    BYTE iSlaveNo,
    DWORD lVelocity
);
```

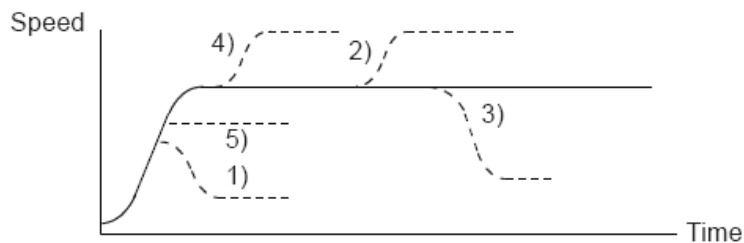
Parameters

nPortNo
해당 Board 의 Port 번호.
iSlaveNo
해당 Board 의 Slave 번호.
lVelocity
변경할 속도 값..

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.
FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.
FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

Remarks



- 1) ((change speed) < (speed before change)) 인 경우 새로운 속도패턴을 이용한 가감속을 통하여 change speed 까지 도달합니다.
- 5) ((change speed) ≥ (speed before change)) 인 경우 속도패턴이 없는 가감속을 통하여 change speed 까지 도달합니다.
- 4) speed before change 까지는 속도패턴의 변경없이 도달한후, change speed 까지는 새로운 속도패턴 특성으로 도달합니다.
- 2),3) 가감속이 완료된후에는 change speed 의 속도패턴 특성에 맞추어서 change speed 까지 도달합니다.

Example

FAS_MoveSingleAxisAbsPos 라이브러리를 참조

See Also

FAS_AIIMoveStop

동일 port 에 연결된 모든 Motor 를 정지시킨다.

Syntax

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

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호로서 '99' 를 지정해야합니다.

Return Value

없음.

Remarks

Example

FAS_MoveSingleAxisAbsPos 라이브러리를 참조

See Also

FAS_AIIEmergencyStop

동일 port 에 연결된 모든 Motor 를 급정지 시킨다.

Syntax

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

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호로서 '99' 를 지정해야합니다.

Return Value

없음.

Remarks

이 함수는 감속 단계가 없으므로 기계계의 충격에 주의하십시오.

Example

FAS_MoveSingleAxisAbsPos 라이브러리를 참조

See Also

FAS_AIIMoveOriginSingleAxis

동일 port 에 연결된 모든 Motor 의 원점(Origin)을 찾는다.

자세한 사항은 ‘[사용자 매뉴얼_본문편의 9.3 원점복귀](#)’ 항목을 참조하십시오.

Syntax

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

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호로서 ‘99’ 를 지정해야 합니다.

Return Value

없음.

Remarks

Example

FAS_MoveSingleAxisAbsPos 라이브러리를 참조

See Also

FAS_AIIMoveSingleAxisAbsPos

동일 port 에 연결된 모든 Motor 를 절대좌표 값으로 이동시킨다.

Syntax

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

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호로서 '99' 를 지정해야합니다.

lAbsPos

이동할 위치의 절대좌표 값.

lVelocity

이동 시 속도 값.

Return Value

없음.

Remarks

Example

FAS_MoveSingleAxisAbsPos 라이브러리를 참조

See Also

FAS_AIIMoveSingleAxisIncPos

동일 port 에 연결된 모든 Motor 를 상대좌표 값으로 이동시킨다.

Syntax

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

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호로서 '99' 를 지정해야합니다.

lIncPos

이동할 위치의 상대좌표 값.

lVelocity

이동 시 속도 값.

Return Value

없음.

Remarks

Example

FAS_MoveSingleAxisAbsPos 라이브러리를 참조

See Also

FAS_MoveLinearAbsPos

동일 port 에 연결된 2 개 이상의 Motor 를 절대좌표 값으로 Linear Interpolation 기능으로 이동시킨다.

Syntax

```
int FAS_MoveLinearAbsPos(
    BYTE nPortNo,
    BYTE nNoOfslaves,
    BYTE *iSlavesNo,
    long *lAbsPos,
    DWORD lFeedrate,
    DWORD wAccelTime,
);
```

Parameters

nPortNo
해당 Board 의 Port 번호.

nNoOfSlaves
Linear motion 을 실행하고자 하는 Slave 의 수.

iSlavesNo
해당 Slave 들의 번호의 배열.

lAbsPos
해당 Slave 들의 이동위치의 배열.

lFeedrate
이동 시 속도 값.

wAccelTime
이동 시 가감속 구간의 시간 값.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

Remarks

FAS_MoveLinear IncPos

동일 port 에 연결된 2 개 이상의 Motor 를 상대좌표 값으로 Linear Interpolation 기능으로 이동시킨다.

Syntax

```
int FAS_MoveLinear IncPos(
    BYTE nPortNo,
    BYTE nNoOfslaves,
    BYTE *iSlavesNo,
    long *lIncPos,
    DWORD lFeedrate,
    DWORD wAccelTime,
);
```

Parameters

nPortNo
해당 Board 의 Port 번호.

nNoOfSlaves
Linear motion 을 실행하고자 하는 Slave 의 수.

iSlavesNo
해당 Slave 들의 번호의 배열.

lIncPos
해당 Slave 들의 이동거리의 배열.

lFeedrate
이동 시 속도 값.

wAccelTime
이동 시 가감속 구간의 시간 값.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

Remarks

FAS_MoveSingleAxisAbsPosEx

Motor 를 특정 절대좌표 값으로 이동시킨다. (운전 가속 및 감속 시간 지정 가능)

Syntax

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

Parameters

nPortNo
해당 Board 의 Port 번호.

iSlaveNo
해당 Board 의 Slave 번호.

lAbsPos
이동할 위치의 상대좌표 값

lVelocity
이동 시 속도 값.

lpExOption
Custom option.

Return Value

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

Remarks

Refer to MOTION_OPTION_EX struct.

Example

```
#include "FAS_ EzIMOTIONPlusR.h"

void funcMoveEx()
{
    BYTE nPortNo = 1; // COMM Port 번호
    BYTE iSlaveNo = 0; // Slave No (0 ~ 15)
    DWORD dwAxisStatus, dwInput;
    EZISTEP_AXISSTATUS stAxisStatus;
    long lAbsPos, lIncPos, lVelocity;
    MOTION_OPTION_EX opt = {0};
    int nRtn;

    // 연결합니다.
    if (FAS_Connect(nPortNo, 115200) == FALSE)
    {
        // 연결이 실패하였습니다.
        // 연결된 Port 가 아니거나 Baudrate 가 틀릴 수 있습니다.
        return;
    }

    // 특정 가속속 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(nPortNo, iSlaveNo, lIncPos, lVelocity, &opt);
_ASSERT(nRtn == FMM_OK);

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

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

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

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

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

// 연결을 종료합니다.
FAS_Close(nPortNo);
}

```

See Also

FAS_MoveSingleAxisIncPosEx

Motor 를 특정 상대좌표 값으로 이동시킨다. (운전 가속 및 감속 시간 지정 가능)

Syntax

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

Parameters

nPortNo
해당 Board 의 Port 번호.

iSlaveNo
해당 Board 의 Slave 번호.

lIncPos
이동할 위치의 상대좌표 값.

lVelocity
이동 시 속도 값.

lpExOption
Custom option.

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

Remarks

Example

See Also

FAS_MoveVelocityEx

Motor 를 해당 방향, 해당 속도, 해당 가감속도로 이동시킨다. Jog 운전시 사용된다.

Syntax

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

Parameters

nPortNo
해당 Board 의 Port 번호

iSlaveNo
해당 Board 의 Slave 번호.

lVelocity
이동 시 속도 값.

iVelDir
이동 할 방향 (0: -Jog, 1: +Jog)

lpExOption
Custom option.

Return Value

FMM_OK : Command has been normally performed.

FMM_NOT_OPEN : The drive has not been connected yet.

FMM_INVALID_PORT_NUM : There is no nPort in the connected ports.

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

Remarks

Refer to VELOCITY_OPTION_EX struct.

Example

```
#include "FAS_ EzIMOTIONPlusR.h"

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

    // 연결합니다.
    if (FAS_Connect(nPortNo, 115200) == FALSE)
    {
        // 연결이 실패하였습니다.
        // 연결된 Port 가 아니거나 Baudrate 가 틀릴 수 있습니다.
        return;
    }

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

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

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

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

See Also

2.10 포지션테이블 제어 함수

함수명	내용
FAS_PosTableReadItem	특정 포지션테이블의 RAM 영역의 항목값들을 읽어들이니다.
FAS_PosTableWriteItem	특정 포지션테이블의 항목값을 RAM 영역에 저장합니다.
FAS_PosTableWriteROM	모든 포지션테이블 값을 ROM 영역에 저장합니다 : 256 개의 모든 PT 값이 저장됩니다.
FAS_PosTableReadROM	ROM 영역의 포지션테이블 값들을 읽어 들입니다 : 256 개의 모든 PT 값을 읽어 들입니다.
FAS_PosTableRunItem	지정된 포지션테이블에서 부터 순차적으로 운전을 시작합니다.
FAS_PosTableReadOneItem	특정 포지션테이블의 특정 항목의 RAM 영역의 값을 읽어 들입니다.
FAS_PosTableWriteOneItem	특정 포지션테이블의 특정 항목 값을 RAM 영역에 저장합니다.

FAS_PosTableReadItem

Position Table의 특정 Item을 읽어온다.

Syntax

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

Parameters

nPortNo
해당 Board의 Port 번호.

iSlaveNo
해당 Board의 Slave 번호.

wItemNo
읽어 올 Item 번호.

lpItem
Item 값이 저장될 Item 구조체 포인터.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port에 iSlaveNo의 Slave는 존재하지 않습니다.

FMM_INVALID_PARAMETER_NUM : wItemNo가 범위를 벗어났습니다.

Remarks

Example

```
#include "FAS_EziMOTIONPlusR.h"

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

    // 연결합니다.
    if (FAS_Connect(nPortNo, 115200) == FALSE)
    {
        // 연결이 실패하였습니다.
        // 연결된 Port가 아니거나 Baudrate가 틀릴 수 있습니다.
        return;
    }

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

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

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

// 연결을 종료합니다.
FAS_Close(nPortNo);
}

```

See Also

FAS_PosTableWriteItem

FAS_PosTableWriteItem

Position Table 의 특정 Item 을 수정한다.

Syntax

```
int FAS_PosTableWriteItem(
    BYTE nPortNo,
    BYTE iSlaveNo,
    WORD wItemNo,
    LPITEM_NODE lpItem
);
```

Parameters

nPortNo
해당 Board 의 Port 번호.

iSlaveNo
해당 Board 의 Slave 번호.

wItemNo
수정 할 Item 번호.

lpItem
수정 할 Item 구조체 포인터.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

FMC_POSTABLE_ERROR : Position Table 을 쓰는 도중에 에러가 발생하였습니다.

FMM_INVALID_PARAMETER_NUM : wItemNo 가 범위를 벗어났습니다.

Remarks

포지션테이블 데이터가 저장되는 영역은 RAM 영역과 ROM 영역이 있으며,
이 함수는 RAM 영역에 저장하는 기능이고, 전원 OFF 시에는 데이터가 삭제됩니다.

Example

See Also

FAS_PosTableReadItem

FAS_PosTableWriteROM

현재의 Position Table Item 들을 ROM 영역에 모두 저장한다.

Syntax

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

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

FMC_POSTABLE_ERROR : Position Table 을 저장하는 중에 에러가 발생하였습니다.

Remarks

포지션테이블 데이터가 저장되는 영역은 RAM 영역과 ROM 영역이 있으며,
이 함수는 ROM 영역에 저장하는 기능이고, 전원 OFF 시에도 데이터가 보존됩니다.

Example

See Also

FAS_PosTableReadROM

FAS_PosTableReadROM

현재 ROM 영역에 저장된 Position Table Item 값들을 읽어들인다.

Syntax

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

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 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 수행한다.

Syntax

```
int FAS_PosTableRunItem(
    BYTE nPortNo,
    BYTE iSlaveNo,
    WORD wItemNo
);
```

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

wItemNo

동작을 시작할 Item 번호.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 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 읽어온다.

Syntax

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

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

wItemNo

읽어 올 Item 번호.

wOffset

읽어 올 항목의 위치 offset 값. (‘1-2-6. 포지션테이블 항목’ 을 참조)

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 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 수정한다.

Syntax

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

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

wItemNo

수정 할 Item 번호.

wOffset

읽어 올 항목의 위치 offset 값. (‘1-2-6. 포지션테이블 항목’ 을 참조)

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 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

2.11 기타 제어 함수

함수명	내용
FAS_TriggerOutput_RunA	특정 위치에서 출력신호를 발생시키도록하는 기능
FAS_TriggerOutput_Status	출력신호(COMP)의 발생여부를 확인하는 기능

FAS_TriggerOutput_RunA

위치명령에 의한 운전중 특정위치에서 디지털 출력 신호(COMP pin)를 발생/종료 시킴.

Syntax

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

Parameters

nPortNo
해당 Board 의 Port 번호.

iSlaveNo
해당 Board 의 Slave 번호.

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 : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

Remarks

Example

See Also

FAS_TriggerOutput_Status

FAS_ TrggerOutput_Status

현재 신호출력 기능이 작동중인지 여부를 확인하고자 하는 명령

Syntax

```
int FAS_TrggerOutput_Status(
    BYTE nPortNo,
    BYTE iSlaveNo,
    BYTE* bTriggerStatus
);
```

Parameters

nPortNo

해당 Board 의 Port 번호.

iSlaveNo

해당 Board 의 Slave 번호.

bTriggerStatus

현재의 신호 출력 상태.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : 아직 Board 를 연결하기 전 입니다.

FMM_INVALID_PORT_NUM : 연결한 Port 중에 nPort 는 존재하지 않습니다.

FMM_INVALID_SLAVE_NUM : 해당 Port 에 iSlaveNo 의 Slave 는 존재하지 않습니다.

Remarks

Example

See Also

FAS_ TrggerOutput_RunA

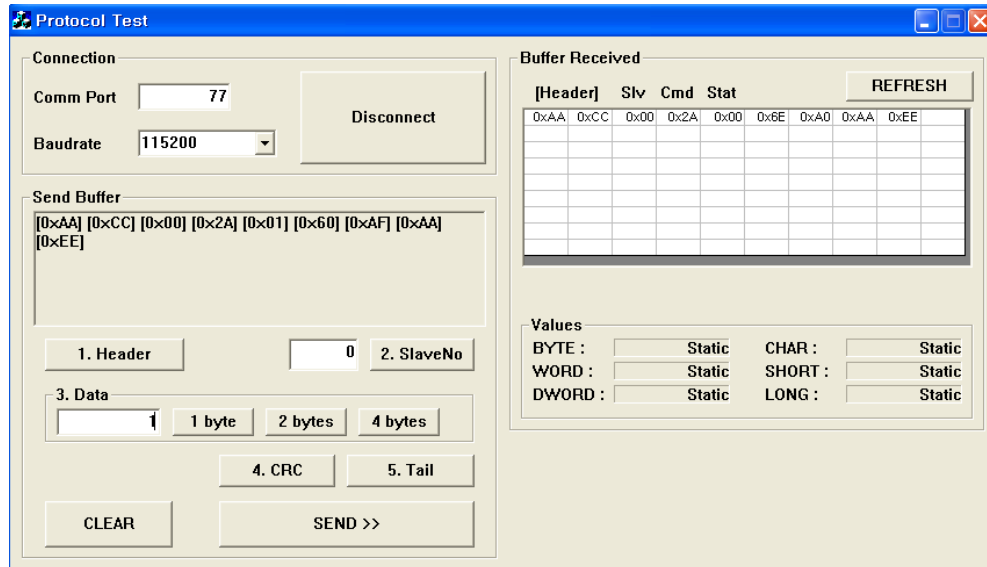
3. PLC 프로그램용 프로토콜

사용자 프로그램(GUI)이 설치된 폴더에서  을 클릭하면 다음과 같은 창이 활성화 됩니다.

아래의 예를 통하여 프로토콜 프로그램의 방법을 이해할 수 있습니다.

(1) Servo ON/OFF 명령용 프로토콜

- * Ezi-STEP Plus-R의 경우에는 전원 인가시 모터 통전 상태이므로 곧바로 **‘(2)모터 동작 명령용 프로토콜’**을 실행하면 됩니다.



모든 프로토콜에는 Header, Slave No., Frame type, Data, CRC, Tail의 내용이 포함되어 있어야 합니다.

- 1) 'Comm Port'와 'Baudrate'를 선택한 후 'Connect'를 클릭합니다.
- 2) Header : 'Header'를 클릭하면 '[0xAA][0xCC]'가 'Send Buffer'창에 나타납니다.
- 3) Slave ID : 연결된 Slave 번호를 입력하고 'Slave No'를 클릭합니다.
- 4) Frame type : 명령의 종류를 선택합니다.
 '1-2-1. Frame type 별 송수신 내용'에서 Servo ON/OFF 명령에 대해 다음표를 확인할 수 있습니다.

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

따라서 '42'를 입력하고 송신 Frame type의 크기가 1 byte이므로, '1 byte'를 클릭합니다

- 5) Data : Servo ON을 시키기 위한 지정 명령은 '1'이므로 '1'을 입력하고 '1 byte'를 클릭합니다.
- 6) CRC : 이것을 클릭하면 자동으로 checksum 값을 계산하여 2 byte 크기로 표현됩니다.
- 7) Tail : 'Tail'을 클릭하면 '[0xAA][0xEE]'가 표현됩니다.
- 8) 마지막으로 'Send'를 클릭하면 'Send Buffer'창에 입력된 data들이 Ezi-SERVO Plus-R로 전달됩니다.

이때 모터의 Servo ON(Torque)상태와 LED 점멸 상태로 확인할 수 있습니다.

- 9) 명령이 Ezi-SERVO Plus-R 에 정상적으로 전달되고 Ezi-SERVO Plus-R로부터 정상적으로 받은 내용은 'Buffer Received' 창에 표현됩니다.

(2)모터 동작 명령용 프로토콜

The image shows a 'Protocol Test' window with the following sections:

- Connection:** Comm Port: 77, Baudrate: 115200, Disconnect button.
- Send Buffer:**
 - Header: 0xAA, 0xCC, 0x00, 0x35, 0x10, 0x27, 0x00, 0x00
 - SlaveNo: 0
 - Data: 5000, 1 byte, 2 bytes, 4 bytes
 - CRC: 4. CRC, 5. Tail
 - Buttons: CLEAR, SEND >>
- Buffer Received:**
 - Table with columns: [Header], Slv, Cmd, Stat. Data: 0xAA, 0xCC, 0x00, 0x35, 0x00, 0x66, 0x90, 0xAA, 0xEE.
 - Values section: BYTE, WORD, DWORD (Static); CHAR, SHORT, LONG (Static).
 - REFRESH button.

- 1) Header
- 2) Slave No.
- 3) Frame type : 'Incremental Move' 명령을 위해서는 '53' 을 입력합니다.
- 4) Data(Position value) : '10000' 을 입력하고 '4byte' 를 클릭합니다.
- 5) Data(Running speed) : '5000' 을 입력하고 '4 byte' 를 클릭합니다.
- 6) CRC
- 7) Tail
- 8) Send : 파라미터가 'default' 값으로 설정되어 있을 경우에 모터가 1 회전합니다.
'53' 명령은 incremental move 명령이므로 다시 'Send' 를 클릭하면 동일한 거리 만큼 모터가 다시 이동합니다.

(3)PLC Programming

실제 프로토콜 프로그램과 상기의 예는 다음과 차이가 있습니다.

상기의 시험용 GUI 프로그램에서는 다음의 기능이 자동적으로 수행되고 있습니다.

- 1) Header 와 Tail 의 확실한 구분을 위해 'Byte stuffing' 기법을 사용.

「1-1-2. RS-485 통신 프로토콜」을 참조 하십시오.

- 2) 통신 에러 확인용의 CRC 기능을 사용하십시오.

- 3) 「1-1-3. CRC 계산 예제」를 참조 하십시오.



FASTECH Co., Ltd.

경기도 부천시 원미구 약대동 193번지
 부천테크노파크 401동 1202호 (우)420-734
 TEL:032)234-6300~1 FAX:032)234-6302
 Email : fastech@fastech.co.kr
 Homepage : www.fastech.co.kr

- 사용자설명서의 일부 또는 전부를 무단 기재하거나 복제하는 것은 금지되어 있습니다.
- 손상이나 분실 등으로 사용자설명서가 필요할 때에는 본사 또는 가까운 대리점에 문의하여 주십시오.
- 사용자설명서는 제품의 개량이나 사양 변경 및 사용자설명서의 개선을 위해서 예고 없이 변경되는 경우가 있습니다.
- Ez-SERVO는 국내에 등록된 FASTECH CO., LTD.의 등록상표입니다.

© Copyright 2006 FASTECH Co.,Ltd.

All Rights Reserved. Dec 17, 2017 Rev.08.05.029