

# **Ezi-SERVO<sup>®</sup> II Plus-E**

**Closed Loop Stepping System**

개발자 매뉴얼

리눅스편

( Rev.01 )



## 목차

목 차.....	2
1. 리눅스 프로그램용 라이브러리.....	3
1-1. 라이브러리의 구성.....	3
1-2. 함수.....	4
1-3. 윈도우와 리눅스의 차이점.....	12
2. 사용 방법.....	13
2-1. 사용 방법.....	13
-라이브러리 구조.....	13
-컴파일 및 so 파일 생성 방법.....	15
2-2. 사용 예시	
1) ProtocolTest.....	17
2) CommandPosition_ActualPosition.....	18

# 1. 리눅스 프로그램용 라이브러리

## 1-1. 라이브러리의 구성

(1) C용

C header file(\*.h)와 library file(\*.so) 이 필요합니다. 이 파일들은 “WWincludeWW”에 있으며 개발용 source file 에 다음 내용을 포함 시키십시오.

```
#include "WWincludeWWETHERNET_DEFINE.h"

#include "WWincludeWWEthernetInterface.h"

#include "WWincludeWWFAS_EziMOTIONPlusE.h"

#include "WWincludeWWFsocket.h"

#include "WWincludeWWINTERNAL_MOTION_DEFINE.h"

#include "WWincludeWWPROTOCOL_FRAME_DEFINE.h"

#include "WWincludeWWReturnCodes_Define.h"
```

또한 라이브러리 파일은 다음과 같습니다.

[WWincludeWWlibEziMOTION.so](#)

이 라이브러리를 사용한 sample program source 등이  
“WWExamplesWW”폴더에 포함되어 있습니다.

## 1-2. 함수

함수명	내용
<b>FAS_ConnectUDP(char *server_ip)</b>	드라이브와 UDP Protocol 로 연결을 시도합니다. : 성공적으로 접속했다면 TRUE를, 접속에 실패를 했다면 FALSE를 리턴합니다.
<b>FAS_ConnectTCP(char *server_ip)</b>	드라이브와 TCP Protocol 로 연결을 시도합니다. : 성공적으로 접속했다면 TRUE를, 접속에 실패를 했다면 FALSE를 리턴합니다.
<b>FAS_ServoEnable(bool bOnOff)</b>	지정한 드라이브의 Servo 상태를 ON/OFF 시킵니다.
<b>FAS_ServoAlarmReset()</b>	알람이 발생한 드라이브의 알람을 해제시킵니다. : 알람이 발생한 원인을 제거한 후 실시하십시오.
<b>FAS_MoveStop()</b>	운전중인 모터를 감속하면서 정지시킵니다
<b>FAS_GetAllStatus()</b>	제어 입출력 상태와 운전 상태 flag 값, 현재 운전 진행 상황 및 운전중인 PT 번호를 읽어 들입니다. : 현재의 입력 상태값, 출력 설정 상태값 및 운전상태 Flag, Command position, Actual position, 속도값 등을 리턴합니다.
<b>FAS_MoveVelocity(DWORD IVelocity, int iVelDir)</b>	주어진 속도와 방향으로 운전을 시작합니다 : Jog 운전 등에 사용됩니다.

\*FAS\_EziMOTIONPlusE.c와 EthernetInterface.c를 수정하여 원하는 모터 명령 함수들을 추가할 수 있습니다.

## FAS\_ConnectUDP

---

FAS\_ConnectUDP 함수는 Ezi-SERVOII Plus-E에 UDP Protocol로 접속하는 함수입니다.

### Syntax

```
int FAS_ConnectUDP(  
    char *server_ip  
);
```

### Parameters

server\_ip

접속하려는 드라이브의 IP 주소를 문자열 형식으로 입력합니다.

ex) "192.168.0.171"

### Return Value

성공적으로 접속했다면 TRUE를, 접속에 실패를 했다면 FALSE를 리턴합니다.

FMM\_NOT\_OPEN: 잘못된 Port 번호를 입력하였습니다.

## FAS\_ConnectTCP

---

FAS\_ConnectTCP 함수는 Ezi-SERVOII Plus-E에 TCP Protocol로 접속하는 함수입니다.

### Syntax

```
int FAS_ConnectTCP(  
    char *server_ip  
);
```

### Parameters

server\_ip

접속하려는 드라이브의 IP 주소를 문자열 형식으로 입력합니다.

ex) "192.168.0.171"

### Return Value

성공적으로 접속했다면 TRUE를, 접속에 실패를 했다면 FALSE를 리턴합니다.

FMM\_NOT\_OPEN: 잘못된 Port 번호를 입력하였습니다.

## FAS\_ServoEnable

---

드라이브를 Servo ON/OFF 합니다.

### Syntax

```
int FAS_ServoEnable(  
    bool bOnOff  
);
```

### Parameters

bOnOff

1: Enable

2: Disable

### Return Value

FMM\_OK: 명령이 정상적으로 수행되었습니다

FMC\_DISCONNECTED: 해당 Board 가 연결 해제되었습니다.

FMM\_NOT\_OPEN: 잘못된 Port 번호를 입력하였습니다.

## FAS\_ServoAlarmReset

AlarmReset 명령을 보냅니다.

## Syntax

```
int FAS_ServoAlarmReset(
);
```

## Parameters

### Return Value

FMM\_OK: 명령이 정상적으로 수행되었습니다

FMC\_DISCONNECTED: 해당 Board 가 연결 해제되었습니다.

FMM\_NOT\_OPEN: 잘못된 Port 번호를 입력하였습니다.

## Remarks

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



## FAS\_ FAS\_MoveStop

Motor를 정지시킵니다.

## Syntax

```
int FAS_MoveStop(
);
```

## Parameters

### Return Value

FMM\_OK: 명령이 정상적으로 수행되었습니다

FMC\_DISCONNECTED: 해당 Board 가 연결 해제되었습니다.

FMM\_NOT\_OPEN: 잘못된 Port 번호를 입력하였습니다.

## Remarks

## FAS\_MoveVelocity

---

Motor를 해당 방향, 해당 속도로 이동시킨다. Jog 운전시 사용됩니다.

### Syntax

```
int FAS_ MoveVelocity (
    DWORD lVelocity,
    int iVelDir
);
```

### Parameters

lVelocity

이동 시 속도 값.

iVelDir

이동 할 방향.

### Return Value

FMM\_OK: 명령이 정상적으로 수행되었습니다

FMC\_DISCONNECTED: 해당 Board 가 연결 해제되었습니다.

FMM\_NOT\_OPEN: 잘못된 Port 번호를 입력하였습니다.

## FAS\_GetAllStatus

해당 Board의 I/O Input, Output 값과 Motor Axis Status, Motor의 Motion Status 값들을 모두 읽어옵니다

## Syntax

```
int FAS_ GetAllStatus (
);
```

## Parameters

### Return Value

FMM\_OK: 명령이 정상적으로 수행되었습니다

FMC\_DISCONNECTED: 해당 Board 가 연결 해제되었습니다.

FMM\_NOT\_OPEN: 잘못된 Port 번호를 입력하였습니다.

## 1-3. 윈도우와 리눅스의 차이점

### -헤더파일 차이

윈도우 운영 체제에서 제공하는 헤더파일이 리눅스 계열의 운영 체제에서는 지원하지 않을 수 있습니다.

모터 드라이브와 통신 부분에서 사용되는 헤더파일이 다릅니다.

<arpa/inet.h> 이 헤더파일은 리눅스 계열의 운영체제에서 사용하는 헤더파일입니다. 인터넷 주소 변환 및 관련 함수를 사용하기 위한 라이브러리를 포함합니다.

Windows에서 네트워크 관련 작업을 수행하기 위해서는 <winsock2.h> 또는 <windows.h>와 같은 Windows 소켓 라이브러리 관련 헤더 파일을 사용해야 합니다.

### -컴파일러 차이

윈도우 환경에서는 Microsoft Visual Studio를 사용하거나 MinGW로 C 언어를 컴파일 하게 됩니다.

반면 리눅스 환경에서는 기본적으로 GCC가 포함되어 있어 별도의 설치 과정 없이 GCC를 C 및 C++프로그램을 컴파일하고 빌드하는 데 사용합니다.

## 2. 사용 방법

### 2-1. 사용 방법

#### -라이브러리 구조:

FAS\_EziMOTIONPlusE.h 헤더파일에 아래 함수들이 들어있습니다.

```
int FAS_ConnectUDP(char *server_ip);
int FAS_ConnectTCP(char *server_ip);
int FAS_ServoEnable(bool bOnOff);
int FAS_ServoAlarmReset();
int FAS_MoveStop();
int FAS_GetAllStatus();
int FAS_MoveVelocity(DWORD lVelocity, int iVelDir);
```

**FAS\_ConnectUDP** 함수를 실행시키게 되면 소켓을 생성하고 서버 주소를 설정해 패킷을 보낼 준비를 합니다.

**FAS\_ConnectTCP** 함수를 실행시키게 되면 소켓을 생성하고 서버 주소를 설정하고 모터드라이버와 연결을 합니다.

**FAS\_ServoEnable**를 실행하게 되면 **DoCmdServoEnable** 함수를 호출합니다.

```
int FAS_ServoEnable(bool bOnOff)
{
    int nRtn = FMM_OK;
    if (m_socket == NULL)
        return FMM_NOT_OPEN;
    nRtn = DoCmdServoEnable(bOnOff);
    return nRtn;
}
```

EthernetInterface.h 헤더파일에 아래 함수들이 들어있습니다.

```
int DoCmdServoEnable(bool bOnOff);
int DoCmdServoAlarmReset();
int DoCmdMoveStop();
int DoCmdMoveVelocity(DWORD lVelocity, int iVelDir);
int DoCmdGetAllStatus();
```

**DoCmdServoEnable** 함수에서는 **DoSendCommand** 함수를 호출합니다.

```
int DoCmdServoEnable(bool bOnOff)
{
    BYTE byValue = (BYTE)(bOnOff) ? 0x01 : 0x00;

    return DoSendCommand(FRAME_FAS_SERVOENABLE, &byValue, 1, NULL, 0);
}
```

**Fsocket.h** 헤더파일에 아래 함수들이 들어있습니다.

```
int DoSendCommand(BYTE byCmd, LPVOID lpIN, DWORD dwINlen, LPVOID
lpOUT, DWORD dwOUTlen);
int SendTCPPacket(BYTE FrameType, LPBYTE lpData, DWORD dwLen);
int RecvTCPPacket(BYTE FrameType, LPBYTE lpData, DWORD dwLen);
int SendUDPPacket(BYTE FrameType, LPBYTE lpData, DWORD dwLen);
int RecvUDPPacket(BYTE FrameType, LPBYTE lpData, DWORD dwLen);
```

**DoSendCommand** 함수에서는 **m\_bTCP** 값에 따라 TCP통신으로 통신할 지, UDP통신으로 통신할 지 정하게 됩니다. TCP통신인 경우, **SendTCPPacket** 함수와 **RecvTCPPacket** 함수를 통해 패킷을 주고받습니다.

```
int DoSendCommand(BYTE byCmd, LPVOID lpIN, DWORD dwINlen, LPVOID
lpOUT, DWORD dwOUTlen)
{
    int nRtn = FMM_OK;
    m_nSyncNo++;
    if (m_bTCP)
    {
        if (SendTCPPacket(byCmd, (BYTE *)lpIN, dwINlen))
        {
            // Recv
            nRtn = RecvTCPPacket(byCmd, (BYTE *)lpOUT, dwOUTlen);
        }
        else
        {
            nRtn = FMC_DISCONNECTED;
        }
    }
    else // UDP
    {
        if (SendUDPPacket(byCmd, (BYTE *)lpIN, dwINlen))
        {
            // Recv
            nRtn = RecvUDPPacket(byCmd, (BYTE *)lpOUT, dwOUTlen);
        }
    }
}
```

```

    }
    else
    {
        nRtn = FMC_DISCONNECTED;
    }
}
return nRtn;
}

int SendTCPPacket(BYTE FrameType, LPBYTE lpData, DWORD dwLen)
{
    m_BuffSend[SENDOFFSET_HEADER] = 0xAA;
    m_BuffSend[SENDOFFSET_LENGTH] = (unsigned char)(dwLen + 3);
    m_BuffSend[SENDOFFSET_SYNC] = m_nSyncNo;
    m_BuffSend[SENDOFFSET_AXIS] = 0x00;
    m_BuffSend[SENDOFFSET_CMD] = FrameType;
    if (dwLen > 0)
        memcpy(&(m_BuffSend[SENDOFFSET_DATA]), lpData, dwLen);
    int send_result = send(m_socket, m_BuffSend, strlen(m_BuffSend), 0);
}

int RecvTCPPacket(BYTE FrameType, LPBYTE lpData, DWORD dwLen)
{
    received_bytes = recv(m_socket, m_BuffRecv, MAX_BUFFER_SIZE, 0);
    if (received_bytes <= 0)
    {
        perror("recv 실패");
    }
    m_BuffRecv[received_bytes] = '\0';
}

```

## -컴파일 및 so 파일 생성 방법:

### 1) so파일 생성 및 개발용 프로그램 실행 방법

1. 개발용 소스파일과 [www.include.cc](http://www.include.cc)에 있는 c 소스파일 및 h 헤더파일을 한 폴더에 저장하고 터미널에서 해당 폴더의 디렉토리로 이동합니다.

-개발용 소스파일의 이름은 test.c라고 가정하겠습니다.

2. 터미널에 아래와 같이 입력해서 c 소스파일들을 컴파일 합니다.

```
gcc -c -fpic EthernetInterface.c -o EthernetInterface.o
```

```
gcc -c -fpic FAS_EziMOTIONPlusE.c -o FAS_EziMOTIONPlusE.o
```

```
gcc -c -fpic Fsocket.c -o Fsocket.o
```

3. 터미널에 아래와 같이 입력해서 공유 라이브러리 생성 (이름을 "libEziMOTION.so"로 지정)

```
gcc -shared -o libEziMOTION.so EthernetInterface.o FAS_EziMOTIONPlusE.o Fsocket.o
```

4. 터미널에 아래와 같이 입력해서 개발용 소스파일 test.c를 컴파일 하고 라이브러리와 링크합니다.

```
gcc -o test test.c -L. -lEziMOTION
```

5. 터미널에 아래와 같이 입력해서 라이브러리를 시스템 디렉토리에 복사합니다.

```
sudo cp libEziMOTION.so /usr/lib
```

```
sudo ldconfig
```

6. 터미널에 아래와 같이 입력해서 test 프로그램을 실행시킵니다.

```
./test
```

## 2) 컴파일 및 개발용 프로그램 실행 방법

1. 개발용 소스파일과 [www.include.cc](http://www.include.cc)에 있는 c 소스파일 및 h 헤더파일을 한 폴더에 저장하고 터미널에서 해당 폴더의 디렉토리로 이동합니다.

-개발용 소스파일의 이름은 test.c라고 가정하겠습니다.

2. 터미널에 아래와 같이 입력해서 컴파일하고 test 프로그램을 실행시킵니다.

```
gcc -c EthernetInterface.c -o EthernetInterface.o
```

```
gcc -c FAS_EziMOTIONPlusE.c -o FAS_EziMOTIONPlusE.o
```

```
gcc -c Fsocket.c -o Fsocket.o
```

```
gcc -c test.c -o test.o
```

```
gcc test.o EthernetInterface.o FAS_EziMOTIONPlusE.o Fsocket.o -o test
```

```
./test
```



## 2-2. 사용 예시

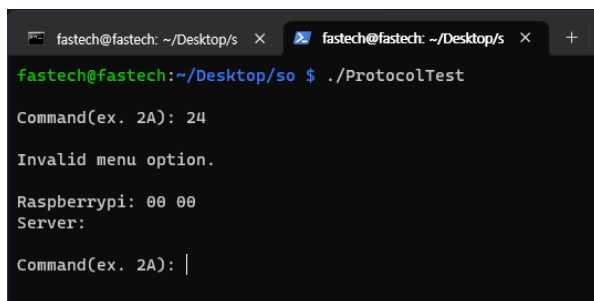
“WWExamples\WW”폴더에 포함되어 있습니다.

### 1) ProtocolTest

내장되어 있는 명령들을 실행시켜 모터를 제어할 수 있는 프로그램입니다.

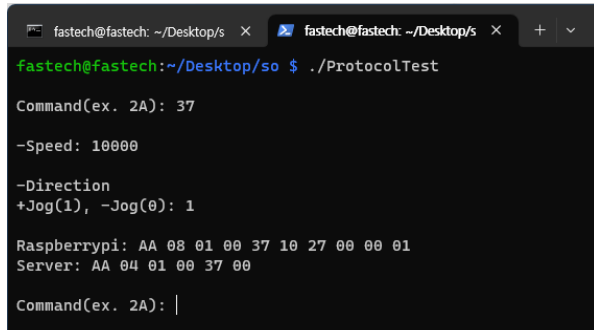
프로그램을 실행하면 자동적으로 UDP프로토콜로 연결하게 됩니다. 기본 ip주소가 “192.168.0.171”로 설정되어 있습니다. 필요할 경우 c소스파일에 ip주소 부분을 수정해야 합니다.

1. 원하는 모터 명령을 입력합니다. 잘못된 입력을 했을 경우 다시 입력하게 됩니다.



```
fastech@fastech: ~/Desktop/s
fastech@fastech:~/Desktop/so $ ./ProtocolTest
Command(ex. 2A): 24
Invalid menu option.
Raspberrypi: 00 00
Server:
Command(ex. 2A): |
```

2. 정상적으로 입력했을 경우 보낸 패킷과 받은 패킷을 보여줍니다.



```
fastech@fastech: ~/Desktop/s
fastech@fastech:~/Desktop/so $ ./ProtocolTest
Command(ex. 2A): 37
-Speed: 10000
-Direction
+Jog(1), -Jog(0): 1
Raspberrypi: AA 08 01 00 37 10 27 00 00 01
Server: AA 04 01 00 37 00
Command(ex. 2A): |
```

## 2) CommandPosition\_ActualPosition

모터의 Command Position과 Actual Position을 비교할 수 있도록 하였고 프로그램 실행 시 각 데이터를 10ms마다 받아 각각의 csv파일에 저장하게 할 수 있습니다. 필요할 경우 수신 주기를 수정해서 사용하면 됩니다.

프로그램을 실행하면 자동적으로 UDP프로토콜로 연결하게 됩니다. 기본 ip주소가 "192.168.0.171"로 설정되어 있습니다. 필요할 경우 c소스파일에 ip주소 부분을 수정해야 합니다.


1. command position값, actual position값, position 차이값을 보여줍니다.

```
fastech@fastech: ~/Desktop/$
```

```
Raspberrypi: AA 03 87 00 43
Server: AA 24 87 00 43 00 00 00 00 00 40 00 00 00 00 00 40 00 F6 76 02 00 F0 76 02 00 06 00 00 00 00 00 00 FF FF FF FF

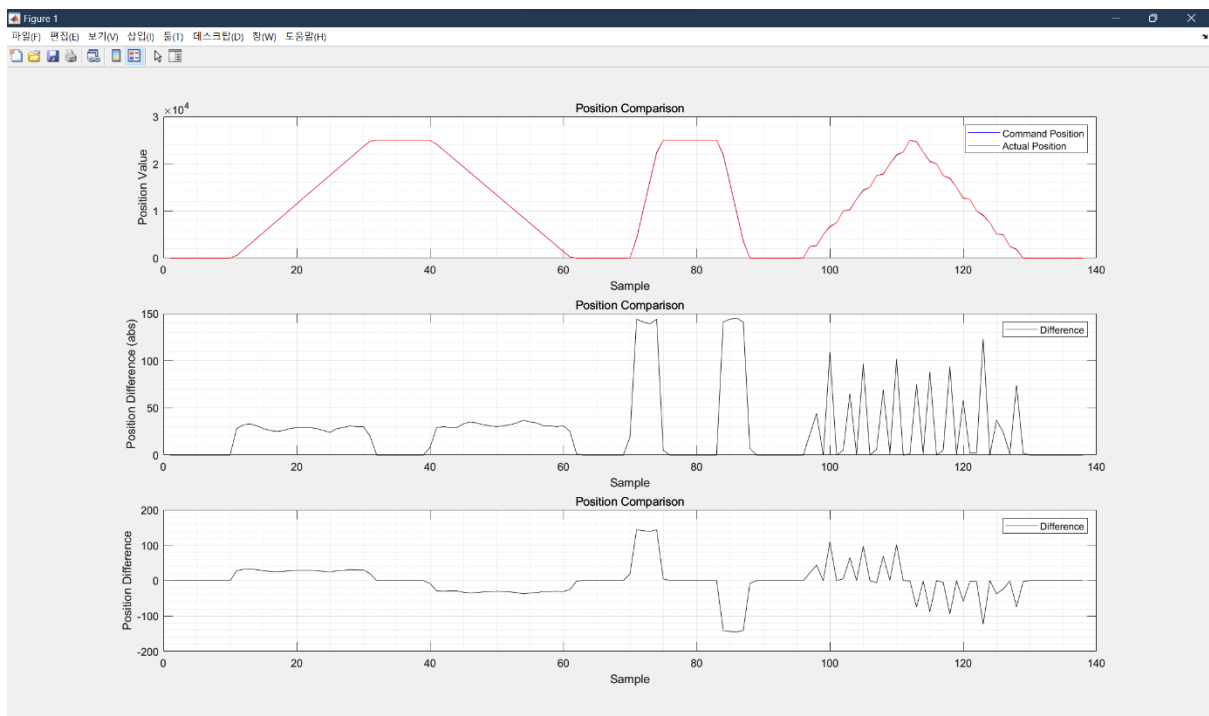
command pos: 161526
actual pos: 161520
pos: 6
```

2. 프로그램이 들어있는 폴더에 아래와 같이 쉽표로 구분된 데이터가 들어있는 csv파일이 생성됩니다.

 outputactual.csv

 outputcommand.csv

3. 이 파일을 활용해서 그래프를 그리게 되면 다음과 같이 그릴 수 있습니다.





*Fast, Accurate, Smooth Motion*

### **FASTECH Co., Ltd.**

경기도 부천시 원미구 약대동 193번지

부천테크노파크 401동 1202호 (우)420-734

TEL : 032)234-6300,6301 FAX : 032)234-6302

E-mail : fastech@fastech.co.kr

Homepage : [www.fastech.co.kr](http://www.fastech.co.kr)

- 사용자 설명서의 일부 또는 전부를 무단 기재하거나 복제하는 것은 금지되어 있습니다.
- 손상이나 분실 등으로 사용자 설명서가 필요할 경우에는 본사 또는 가까운 대리점에 문의하여 주십시오.
- 사용자 설명서는 제품의 계량이나 사양 변경 및 사용자 설명서의 개선을 위해 예고 없이 변경되는 경우가 있습니다.
- Ezi-SERVOⅡ Plus-E 은 국내에 등록된 FASTECH Co.,Ltd.의 등록 상표입니다.

© Copyright 2016 FASTECH Co.,Ltd.

[www.fastech.co.kr](http://www.fastech.co.kr)