Ezi-IO® Ethernet
Input/Output Module DIO

사용자 매뉴얼

통신 기능편

(**Rev.04**)

FASTECH

목 차

1 . 통신 프로토콜

1 - 1 . 통신 기능

Ezi-IO Ethernet DIO 제품군은 Ethernet 통신을 이용하여 최대 254(1~254)개의 모듈을 제어 가능합니다.

1 - 1 - 1 . 통신 사양

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

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

1 - 1 - 2 . Ethernet IP 주소

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

- PC 또는 Ethernet 장치에서 직접 Ezi-IO Ethernet DIO 에 연결할 경우 반드시 Network 설정을 위의 IP 주소에 준하여 설정하여 주십시오.
  설정이 안되었거나 상이할 경우 연결할 수 없습니다.
- 스위치를 255(FF)로 설정하면 IP Address 는 자동으로 설정됩니다.
  DHCP 를 사용하기 때문에 공유기를 사용할 경우에만 IP Address 가 자동으로 설정 됩니다. (Ethernet IN 커넥터에 Ethernet 을 연결 하십시오.)
- 제어기(PC, PLC 등)에서 직접 연결할 경우에는 반드시 스위치로 IP Address 를 설정 하십시오.
- 상기의 기본 IP Address 를 사용하지 않을 경우에만 IP Address 를 자동으로 설정 하십시오. 자동으로 IP 가 설정 되면 사용자 프로그램(GUI)을 접속하여 IP Address 를 저장한 후에 전원을 차단하고 스위치로 IP 의 마지막 번호를 설정 하십시오.
- IP 설정 스위치를 0(00)으로 하면 IP 설정이 상기 값으로 초기화 됩니다.

1 - 1 - 3 . Ethernet 프로토콜

1) 통신 FRAME 의 개요

2) FRAME 의 기본 구조

UDP Header	***Frame Data***
8 bytes	5~257 bytes

UDP Header 에는 다음의 정보가 포함됩니다.

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

1 - 1 - 4 . 수신 Frame Data 구조

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

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

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

1 - 1 - 5 . 답신 Frame Data 구조와 통신 에러

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

Header	Length	Sync No.	Reserved	Frame type	Data	
1 byte	1 byte	1 byte	1 byte (0x00)	1 byte	1 byte	0 ~ 251 bytes
					통신 상태	답신 Data

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

(수신시의 데이터와 일치하지 않으면 에러 상태로 인식하십시오.)

④ Reserved : 1 byte(0x00)

⑤ Frame type : 수신 Frame 과 동일

(송신시의 데이터와 일치하지 않으면 에러 상태로 인식하십시오.)

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

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

Hexa 값	Decimal 값	내 용
0x00	0	통신이 정상 상태입니다.
0x80	128	Frame type 에러 : 수신한 Frame type 명령을 인식할 수 없습니다.
0x81	129	데이터 에러, ROM 데이터 읽기, 쓰기 에러 : 수신한 데이터의 값이 정해진 범위외의 데이터입니다.
0x82	130	수신 Frame 에러 : 수신된 Frame 이 규격에 맞지 않는 데이터입니다.
0xAA	170	CRC 에러 : 수신된 Frame 의 data 가 주변 노이즈등의 영향으로 CRC 에러가 발생 했습니다. 이 경우에는 송신측 DLL Library 에서 자동으로 통신을 1 회 더 시도합니다.

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

1 - 2 . Frame 의 구성

1 - 2 - 1 . Frame type 별 송수신 내용

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

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

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

송신 : 0 byte

답신 : 1~248 bytes
<table>
<tr><td>1 byte</td><td>1 byte</td><td>0~246 bytes</td></tr>
<tr><td>통신 상태</td><td>Slave 종류</td><td>ACII string with NULL byte
(strlen() + 1 byte)</td></tr>
</table>
◆ Slave 종류
150 : Ezi-IO-EN-I16 series
155 : Ezi-IO-EN-I8O8 series
160 : Ezi-IO-EN-O16 series
151 : Ezi-IO-EN-I32 series
156 : Ezi-IO-EN-I16O16 series
161 : Ezi-IO-EN-O32 series
</td></tr>
<tr><td>0xC0
(192)</td><td>FAS_GetInput</td><td>
Input / Latch 상태를 읽어 들임.

송신 : 0 byte

수신 : 9 bytes
<table>
<tr><td>1 byte</td><td>4 bytes</td><td>4 bytes</td></tr>
<tr><td>통신 상태</td><td>Input 상태</td><td>Latch 상태</td></tr>
</table>
수신 DATA 구조
MSB LSB
<table>
<tr><td>Latch31</td><td>Latch30</td><td>Latch29</td><td>Latch28</td><td>Latch27</td><td>Latch26</td><td>Latch25</td><td>Latch24</td><td>63~56</td></tr>
<tr><td>Latch23</td><td>Latch22</td><td>Latch21</td><td>Latch20</td><td>Latch19</td><td>Latch18</td><td>Latch17</td><td>Latch16</td><td>55~48</td></tr>
<tr><td>Latch15</td><td>Latch14</td><td>Latch13</td><td>Latch12</td><td>Latch11</td><td>Latch10</td><td>Latch9</td><td>Latch8</td><td>47~40</td></tr>
<tr><td>Latch7</td><td>Latch6</td><td>Latch5</td><td>Latch4</td><td>Latch3</td><td>Latch2</td><td>Latch1</td><td>Latch0</td><td>39~32</td></tr>
<tr><td>Input31</td><td>Input30</td><td>Input29</td><td>Input28</td><td>Input27</td><td>Input26</td><td>Input25</td><td>Input24</td><td>31~24</td></tr>
<tr><td>Input23</td><td>Input22</td><td>Input21</td><td>Input20</td><td>Input19</td><td>Input18</td><td>Input17</td><td>Input16</td><td>23~16</td></tr>
<tr><td>Input15</td><td>Input14</td><td>Input13</td><td>Input12</td><td>Input11</td><td>Input10</td><td>Input9</td><td>Input8</td><td>15~8</td></tr>
<tr><td>Input7</td><td>input6</td><td>Input5</td><td>Input4</td><td>Input3</td><td>Input2</td><td>Input1</td><td>Input0</td><td>7~0</td></tr>
</table>
0: Off, 1: On
</td></tr>
</table>

<table>
<tr><th>Frame type</th><th>라이브러리 명</th><th>내용</th></tr>
<tr><td></td><td></td><td>◆ Ezi-IO-EN-I16 series : Input0~15, Latch 0~15 상태를 읽어 들임.
Ezi-IO-EN-I8O8 series : Input0~7, Latch 0~7 상태를 읽어 들임.
Ezi-IO-EN-I32 series : Input0~31, Latch 0~31 상태를 읽어 들임.
Ezi-IO-EN-I16O16 series : Input0~15, Latch 0~15 상태를 읽어 들임.</td></tr>
<tr><td>0xC1
(193)</td><td>FAS_ClearLatch</td><td>해당 bit 의 Latch 를 Clear(Reset)함.

송신 : 4 bytes
<table><tr><td>4 bytes</td></tr><tr><td>Data</td></tr></table>
송신 Data 구조
MSB LSB
<table><tr><td>Latch31</td><td>Latch30</td><td>Latch29</td><td>Latch28</td><td>Latch27</td><td>Latch26</td><td>Latch25</td><td>Latch24</td><td>31~24</td></tr><tr><td>Latch23</td><td>Latch22</td><td>Latch21</td><td>Latch20</td><td>Latch19</td><td>Latch18</td><td>Latch17</td><td>Latch16</td><td>23~16</td></tr><tr><td>Latch15</td><td>Latch14</td><td>Latch13</td><td>Latch12</td><td>Latch11</td><td>Latch10</td><td>Latch9</td><td>Latch8</td><td>15~8</td></tr><tr><td>Latch7</td><td>Latch6</td><td>Latch5</td><td>Latch4</td><td>Latch3</td><td>Latch2</td><td>Latch1</td><td>Latch0</td><td>7~0</td></tr></table>0 : 현재 상태 유지
1 : Latch Clear(Reset)
◆ Ezi-IO-EN-I16 series : Latch 0~15
Ezi-IO-EN-I8O8 series : Latch 0~7
Ezi-IO-EN-I32 series : Latch 0~31
Ezi-IO-EN-I16O16 series : Latch 0~16

수신 : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table></td></tr>
<tr><td>0xC2
(194)</td><td>FAS_GetLatchCount</td><td>Latch Count 를 읽어 들임(0~15 번)

송신 : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>Data</td></tr></table>Data : Latch bit 번호(0~15 번)
◆ Ezi-IO-EN-I16 series : 0~15
Ezi-IO-EN-I8O8 series : 0~7
Ezi-IO-EN-I32 series : 0~31
Ezi-IO-EN-I16O16 series : 0~16

수신 : 5 bytes
<table><tr><td>1 byte</td><td>4 bytes</td></tr><tr><td>통신 상태</td><td>Data</td></tr></table>Data : $0 \sim 2^{32}$</td></tr>
</table>

<table>
<tr><th>Frame type</th><th>라이브러리 명</th><th>내용</th></tr>
<tr><td>0xC3
(195)</td><td>FAS_GetLatchCount
All</td><td>
0~15CH 의 Latch Count 를 읽어 들임

송신 : 0 byte

수신 : 65 bytes
<table>
<tr><td>1 byte</td><td>64 bytes</td></tr>
<tr><td>통신 상태</td><td>Data</td></tr>
</table>
Data 구조
<table>
<tr><td colspan="3">MSB</td><td></td><td></td><td colspan="3">LSB</td></tr>
<tr><td>COUNT15</td><td>COUNT14</td><td>COUNT13</td><td>・・・</td><td>・・・</td><td>COUNT2</td><td>COUNT1</td><td>COUNT0</td></tr>
<tr><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td><td></td><td></td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td></tr>
</table>
◆ Ezi-IO-EN-I16 series : Count 0~15
Ezi-IO-EN-I8O8 series : Count 0~7
Ezi-IO-EN-I16O16 series : Count 0~15
</td></tr>
<tr><td>0xBD</td><td>FAS_GetLatchCount
All32</td><td>
0~31CH 의 Latch Count 를 읽어 들임

송신 : 0 byte

수신 : 129 bytes
<table>
<tr><td>1 byte</td><td>128 bytes</td></tr>
<tr><td>통신 상태</td><td>Data</td></tr>
</table>
Data 구조
<table>
<tr><td colspan="3">MSB</td><td></td><td></td><td colspan="3">LSB</td></tr>
<tr><td>COUNT31</td><td>COUNT30</td><td>COUNT29</td><td>・・・</td><td>・・・</td><td>COUNT2</td><td>COUNT1</td><td>COUNT0</td></tr>
<tr><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td><td></td><td></td><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td></tr>
</table>
◆ Ezi-IO-EN-I16 series : Count 0~15
Ezi-IO-EN-I8O8 series : Count 0~7
Ezi-IO-EN-I32 series : Count 0~31
Ezi-IO-EN-I16O16 series : Count 0~15
</td></tr>
</table>

<table>
<tr><th>Frame type</th><th>라이브러리 명</th><th>내용</th></tr>
<tr><td>0xC4
(196)</td><td>FAS_ClearLatch Count</td><td>
해당 bit 의 Latch Count 를 Reset 함

송신 : 4 bytes
<table>
<tr><td>4 bytes</td></tr>
<tr><td>Data</td></tr>
</table>

Data 구조
<table>
<tr><td colspan="4">MSB</td><td colspan="4">LSB</td><td></td></tr>
<tr><td>Latch31</td><td>Latch30</td><td>Latch29</td><td>Latch28</td><td>Latch27</td><td>Latch26</td><td>Latch25</td><td>Latch24</td><td>31~24</td></tr>
<tr><td>Latch23</td><td>Latch22</td><td>Latch21</td><td>Latch20</td><td>Latch19</td><td>Latch18</td><td>Latch17</td><td>Latch16</td><td>23~16</td></tr>
<tr><td>Latch15</td><td>Latch14</td><td>Latch13</td><td>Latch12</td><td>Latch11</td><td>Latch10</td><td>Latch9</td><td>Latch8</td><td>15~8</td></tr>
<tr><td>Latch7</td><td>Latch6</td><td>Latch5</td><td>Latch4</td><td>Latch3</td><td>Latch2</td><td>Latch1</td><td>Latch0</td><td>7~0</td></tr>
</table>
◆ Ezi-IO-EN-I16 series : Latch 0~15
Ezi-IO-EN-I8O8 series : Latch 0~7
Ezi-IO-EN-I32 series : Latch 0~31
Ezi-IO-EN-I16O16 series : Latch 0~16

수신 : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>통신 상태</td></tr>
</table>
</td></tr>
<tr><td>0xC5
(197)</td><td>FAS_GetOutput</td><td>
Output / Trigger(Run/Stop) 상태를 읽어 들임

송신 : 0 byte

수신 : 9 bytes
<table>
<tr><td>1 byte</td><td>8 bytes</td></tr>
<tr><td>통신 상태</td><td>Data</td></tr>
</table>

Data 구조
<table>
<tr><td colspan="4">MSB</td><td colspan="4">LSB</td><td></td></tr>
<tr><td>Run/Stop31</td><td>Run/Stop30</td><td>Run/Stop29</td><td>Run/Stop28</td><td>Run/Stop27</td><td>Run/Stop26</td><td>Run/Stop25</td><td>Run/Stop24</td><td>63~56</td></tr>
<tr><td>Run/Stop23</td><td>Run/Stop22</td><td>Run/Stop21</td><td>Run/Stop20</td><td>Run/Stop19</td><td>Run/Stop18</td><td>Run/Stop17</td><td>Run/Stop16</td><td>55~48</td></tr>
<tr><td>Run/Stop15</td><td>Run/Stop14</td><td>Run/Stop13</td><td>Run/Stop12</td><td>Run/Stop11</td><td>Run/Stop10</td><td>Run/Stop9</td><td>Run/Stop8</td><td>47~40</td></tr>
<tr><td>Run/Stop7</td><td>Run/Stop6</td><td>Run/Stop5</td><td>Run/Stop4</td><td>Run/Stop3</td><td>Run/Stop2</td><td>Run/Stop1</td><td>Run/Stop0</td><td>39~32</td></tr>
<tr><td>Output31</td><td>Output30</td><td>Output29</td><td>Output28</td><td>Output27</td><td>Output26</td><td>Output25</td><td>Output24</td><td>31~24</td></tr>
<tr><td>Output23</td><td>Output22</td><td>Output21</td><td>Output20</td><td>Output19</td><td>Output18</td><td>Output17</td><td>Output16</td><td>23~16</td></tr>
<tr><td>Output15</td><td>Output14</td><td>Output13</td><td>Output12</td><td>Output11</td><td>Output10</td><td>Output9</td><td>Output8</td><td>15~8</td></tr>
<tr><td>Output7</td><td>Output6</td><td>Output5</td><td>Output4</td><td>Output3</td><td>Output2</td><td>Output1</td><td>Output0</td><td>7~0</td></tr>
</table>
0 : Off (Trigger Stop)
1 : On (Trigger Run)

◆ Ezi-IO-EN-O16 series : Output0~15, Trigger(Run/Stop)0~15 의 상태를 읽어 들임
</td></tr>
</table>

<table>
<tr><th>Frame type</th><th>라이브러리 명</th><th>내용</th></tr>
<tr><td></td><td></td><td>Ezi-IO-EN-I8O8 series : Output8~15, Trigger(Run/Stop)8~15 의 상태를 읽어 들임
Ezi-IO-EN-O32 series : Output0~31, Trigger(Run/Stop)0~31 의 상태를 읽어 들임
Ezi-IO-EN-I16O16 series : Output8~15, Trigger(Run/Stop)16~31 의 상태를 읽어 들임</td></tr>
<tr><td>0xC6
(198)</td><td>FAS_SetOutput</td><td>
Output 을 설정 함(Set : On, Reset : Off)

송신 : 8 bytes
<table>
<tr><td>4 bytes</td><td>4 bytes</td></tr>
<tr><td>Set Output</td><td>Reset Output</td></tr>
</table>
Data 구조
<table>
<tr><td>MSB</td><td></td><td></td><td></td><td></td><td></td><td></td><td>LSB</td><td></td></tr>
<tr><td>Reset Output31</td><td>Reset Output30</td><td>Reset Output29</td><td>Reset Output28</td><td>Reset Output27</td><td>Reset Output26</td><td>Reset Output25</td><td>Reset Output24</td><td>63~56</td></tr>
<tr><td>Reset Output23</td><td>Reset Output22</td><td>Reset Output21</td><td>Reset Output20</td><td>Reset Output19</td><td>Reset Output18</td><td>Reset Output17</td><td>Reset Output16</td><td>55~48</td></tr>
<tr><td>Reset Output15</td><td>Reset Output14</td><td>Reset Output13</td><td>Reset Output12</td><td>Reset Output11</td><td>Reset Output10</td><td>Reset Output9</td><td>Reset Output8</td><td>47~40</td></tr>
<tr><td>Reset Output7</td><td>Reset Output6</td><td>Reset Output5</td><td>Reset Output4</td><td>Reset Output3</td><td>Reset Output2</td><td>Reset Output1</td><td>Reset Output0</td><td>39~32</td></tr>
<tr><td>Set Output31</td><td>Set Output30</td><td>Set Output29</td><td>Set Output28</td><td>Set Output27</td><td>Set Output26</td><td>Set Output25</td><td>Set Output24</td><td>31~24</td></tr>
<tr><td>Set Output23</td><td>Set Output22</td><td>Set Output21</td><td>Set Output20</td><td>Set Output19</td><td>Set Output18</td><td>Set Output17</td><td>Set Output16</td><td>23~16</td></tr>
<tr><td>Set Output15</td><td>Set Output14</td><td>Set Output13</td><td>Set Output12</td><td>Set Output11</td><td>Set Output10</td><td>Set Output9</td><td>Set Output8</td><td>15~8</td></tr>
<tr><td>Set Output7</td><td>Set Output6</td><td>Set Output5</td><td>Set Output4</td><td>Set Output3</td><td>Set Output2</td><td>Set Output1</td><td>Set Output0</td><td>7~0</td></tr>
</table>
1 : Set Output or Reset Out 실행
(Set 과 Reset 의 같은 bit 가 1 일 경우 Reset 으로 실행)
0 : Set Output or Reset Out 실행하지 않음
◆ Ezi-IO-EN-O16 series : Output0~15, Reset Output0~15 설정 가능
Ezi-IO-EN-I8O8 series : Output8~15, Reset Output8~15 설정 가능
Ezi-IO-EN-O32 series : Output0~32, Reset Output0~31 설정 가능
Ezi-IO-EN-I16O16 series : Output16~31, Reset Output16~31 설정 가능

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

<table>
<tr><th>Frame type</th><th>라이브러리 명</th><th>내용</th></tr>
<tr><td>0xC7
(199)</td><td>FAS_SetTrigger</td><td>
Trigger Out 설정.

송신 : 13 bytes
<table>
<tr><td>13 bytes</td></tr>
<tr><td>Data</td></tr>
</table>

Data 구조
<table>
<tr><td>MSB</td><td></td><td></td><td></td><td></td><td>LSB</td></tr>
<tr><td>Count</td><td>Blank</td><td>On time</td><td>Blank</td><td>Period</td><td>Output No.</td></tr>
<tr><td>4 bytes</td><td>2 bytes</td><td>2 bytes</td><td>2 bytes</td><td>2 bytes</td><td>1 byte</td></tr>
</table>
Output No. : 트리거 설정 출력 번호
◆ Ezi-IO-EN-O16 series : 0~15
Ezi-IO-EN-I8O8 series : 8~15
Ezi-IO-EN-O32 series : 0~31
Ezi-IO-EN-I16O16 series : 16~31
Period : 트리거 출력 주기, 단위 [ms],
On time 보다 반드시 큰 값을 입력 (2~65,535)
On time : 트리거 출력 On time, 단위[ms],
Period 보다 반드시 작은 값을 입력 (1~65,534)
Count : 트리거 출력 반복 횟수 (1~4,294,967,295)

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

<table>
<tr><th>Frame type</th><th>라이브러리 명</th><th>내용</th></tr>
<tr><td>0xC8
(200)</td><td>FAS_SetRunStop</td><td>
해당 bit 의 Trigger 출력을 Run or Stop 함.
Run 중 Stop bit 가 On 되면 count 만큼 출력되지 않아도 Stop 함.

송신 : 8 bytes
<table>
<tr><td>4 bytes</td><td>4 bytes</td></tr>
<tr><td>Run Output</td><td>Stop Output</td></tr>
</table>

Data 구조
<table>
<tr><td>MSB</td><td></td><td></td><td></td><td></td><td></td><td></td><td>LSB</td><td></td></tr>
<tr><td>Stop31</td><td>Stop30</td><td>Stop29</td><td>Stop28</td><td>Stop27</td><td>Stop26</td><td>Stop25</td><td>Stop24</td><td>63~56</td></tr>
<tr><td>Stop23</td><td>Stop22</td><td>Stop21</td><td>Stop20</td><td>Stop19</td><td>Stop18</td><td>Stop17</td><td>Stop16</td><td>55~48</td></tr>
<tr><td>Stop15</td><td>Stop14</td><td>Stop13</td><td>Stop12</td><td>Stop11</td><td>Stop10</td><td>Stop9</td><td>Stop8</td><td>47~40</td></tr>
<tr><td>Stop7</td><td>Stop6</td><td>Stop5</td><td>Stop4</td><td>Stop3</td><td>Stop2</td><td>Stop1</td><td>Stop0</td><td>39~32</td></tr>
<tr><td>Run31</td><td>Run30</td><td>Run29</td><td>Run28</td><td>Run27</td><td>Run26</td><td>Run25</td><td>Run24</td><td>31~24</td></tr>
<tr><td>Run23</td><td>Run22</td><td>Run21</td><td>Run20</td><td>Run19</td><td>Run18</td><td>Run17</td><td>Run16</td><td>23~16</td></tr>
<tr><td>Run15</td><td>Run14</td><td>Run13</td><td>Run12</td><td>Run11</td><td>Run10</td><td>Run9</td><td>Run8</td><td>15~8</td></tr>
<tr><td>Run7</td><td>Run6</td><td>Run5</td><td>Run4</td><td>Run3</td><td>Run2</td><td>Run1</td><td>Run0</td><td>7~0</td></tr>
</table>
◆ Ezi-IO-EN-O16 series : Run0~15, Stop0~15
Ezi-IO-EN-I8O8 series : Run8~15, Stop8~15
Ezi-IO-EN-O32 series : Run0~31, Stop0~31
Ezi-IO-EN-I16O16 series : Run16~15, Stop16~31

수신 : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>통신 상태</td></tr>
</table>
</td></tr>
<tr><td>0xC9
(201)</td><td>FAS_GetTrigger</td><td>
현재의 Trigger 출력 횟수를 읽어 들임(출력 번호 : 0~15).

송신 : 1 byte
<table>
<tr><td>1 byte</td></tr>
<tr><td>Data</td></tr>
</table>
Data : Trigger 출력 bit 번호
◆ Ezi-IO-EN-O16 series : 출력 Bit 번호 0~15
Ezi-IO-EN-I8O8 series : 출력 Bit 번호 8~15
Ezi-IO-EN-O32 series : 출력 Bit 번호 0~31
Ezi-IO-EN-I16O16 series : 출력 Bit 번호 16~31

수신 : 5 bytes
<table>
<tr><td>1 byte</td><td>4 bytes</td></tr>
<tr><td>통신 상태</td><td>Data</td></tr>
</table>
Data : 0~2^{32}
</td></tr>
</table>

<table>
<tr><th>Frame type</th><th>라이브러리 명</th><th>내용</th></tr>
<tr><td>0xCA
(202)</td><td>FAS_GetIOLevel</td><td>
Input or Output Active Level 을 읽어 들임.

송신 : 0 byte

수신 : 5 bytes
<table>
<tr><td>1 byte</td><td>4 bytes</td></tr>
<tr><td>통신 상태</td><td>Data</td></tr>
</table>

수신 Data 구조
<table>
<tr><td colspan="4">MSB</td><td colspan="4">LSB</td><td></td></tr>
<tr><td>LEVEL31</td><td>LEVEL30</td><td>LEVEL29</td><td>LEVEL28</td><td>LEVEL27</td><td>LEVEL26</td><td>LEVEL25</td><td>LEVEL24</td><td>31~24</td></tr>
<tr><td>LEVEL23</td><td>LEVEL22</td><td>LEVEL21</td><td>LEVEL20</td><td>LEVEL19</td><td>LEVEL18</td><td>LEVEL17</td><td>LEVEL16</td><td>23~16</td></tr>
<tr><td>LEVEL15</td><td>LEVEL14</td><td>LEVEL13</td><td>LEVEL12</td><td>LEVEL11</td><td>LEVEL10</td><td>LEVEL9</td><td>LEVEL8</td><td>15~8</td></tr>
<tr><td>LEVEL7</td><td>LEVEL6</td><td>LEVEL5</td><td>LEVEL4</td><td>LEVEL3</td><td>LEVEL2</td><td>LEVEL1</td><td>LEVEL0</td><td>7~0</td></tr>
</table>
0 : Low Active Level (Latch : Falling edge)
1 : High Active Level (Latch : Rising edge)

◆ Ezi-IO-EN-I16 series : Input Actcive Level 0~15 을 읽어 들임
Ezi-IO-EN-O16 series : Output Actcive Level 0~15 을 읽어 들임
Ezi-IO-EN-I8O8 series : Input Actcive Level 0~7, Output Actcive Level 8~15 을 읽어 들임
Ezi-IO-EN-I32 series : Input Actcive Level 0~32 을 읽어 들임
Ezi-IO-EN-O32 series : Output Actcive Level 0~32 을 읽어 들임
Ezi-IO-EN-I16O16 series : Input Actcive Level 0~15, Output Actcive Level 16~31 을 읽어 들임
</td></tr>
</table>

<table>
<tr><th>Frame type</th><th>라이브러리 명</th><th>내용</th></tr>
<tr><td>0xCB
(203)</td><td>FAS_SetIOLevel</td><td>
Input or Output Active Level 을 설정.

송신 : 4 bytes

<table><tr><td>4 bytes</td></tr><tr><td>Data</td></tr></table>
송신 Data 구조
MSB LSB
<table>
<tr><td>LEVEL31</td><td>LEVEL30</td><td>LEVEL29</td><td>LEVEL28</td><td>LEVEL27</td><td>LEVEL26</td><td>LEVEL25</td><td>LEVEL24</td><td>31~24</td></tr>
<tr><td>LEVEL23</td><td>LEVEL22</td><td>LEVEL21</td><td>LEVEL20</td><td>LEVEL19</td><td>LEVEL18</td><td>LEVEL17</td><td>LEVEL16</td><td>23~16</td></tr>
<tr><td>LEVEL15</td><td>LEVEL14</td><td>LEVEL13</td><td>LEVEL12</td><td>LEVEL11</td><td>LEVEL10</td><td>LEVEL9</td><td>LEVEL8</td><td>15~8</td></tr>
<tr><td>LEVEL7</td><td>LEVEL6</td><td>LEVEL5</td><td>LEVEL4</td><td>LEVEL3</td><td>LEVEL2</td><td>LEVEL1</td><td>LEVEL0</td><td>7~0</td></tr>
</table>
0 : Low Active Level (Latch : Falling edge)
1 : High Active Level (Latch : Rising edge)
◆ Ezi-IO-EN-I16 series : Input Actcive Level 0~15 을 설정
Ezi-IO-EN-O16 series : Output Actcive Level 0~15 을 설정
Ezi-IO-EN-I8O8 series : Input Actcive Level 0~7, Output Actcive Level 8~15 을 설정
Ezi-IO-EN-I32 series : Input Actcive Level 0~31 을 설정
Ezi-IO-EN-O32 series : Output Actcive Level 0~31 을 설정
Ezi-IO-EN-I16O16 series : Input Actcive Level 0~15, Output Actcive Level 16~31 을 설정
수신 : 1 byte
<table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table>
</td></tr>
<tr><td>0xCC
(204)</td><td>FAS_
LoadIOLevel</td><td>
ROM 메모리 Active Level 을 읽어 들임.
Board 의 RAM 의 data 도 읽어 들인 data 로 변경함.

송신 : 0 byte

답신 : 1 byte

<table><tr><td>1 byte</td></tr><tr><td>통신 상태</td></tr></table>
</td></tr>
<tr><td>0xCD
(205)</td><td>FAS_
SaveIOLevel</td><td>
현재 설정된 Active level 값을 ROM 메모리에 저장함. 이는 전원을 OFF 해도 해당 값들이 저장되도록 합니다.

송신 : 0 byte

답신 : 1 byte

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

1 - 3 . 프로그램의 종류

Ezi-IO Ethernet DIO 을 사용하기 위한 프로그램 방법은 두 가지가 있습니다.

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

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

2 . PC 프로그램용 라이브러리

2 - 1 . 라이브러리의 구성

(1) C++용

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

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

또한 라이브러리 파일은 다음과 같습니다.
1) 32bit 용
"₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include₩₩EziMotionPlusE.lib"
"₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include₩₩EziMotionPlusE.dll"
2) 64bit 용
"₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include_x64₩₩EziMotionPlusE.lib"
"₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include_x64₩₩EziMotionPlusE.dll"

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

(2) C#용

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

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

또한 라이브러리 파일은 다음과 같습니다.
1) 32bit 용
"₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include₩₩LIB_EziMOTIONPlusE.cs"
2) 64bit 용
"₩₩FASTECH₩₩Ezi-MOTION Plus-E V6₩₩include_x64₩₩LIB_EziMOTIONPlusE.cs"

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

(3) 다음의 표는 각 라이브러리 함수 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(0x00)	함수가 정상적으로 명령을 수행하였습니다.
입력 에러	FMM_INVALID_SLAVE_NUM	3(0x03)	잘못된 Slave 번호를 입력하였습니다.
연결 에러	FMC_DISCONNECTED	5(0x05)	해당 Board 가 연결 해제 되었습니다.
	FMC_TIMEOUT_ERROR	6(0x06)	정해진 시간(100msec)동안 응답이 없습니다.
	FMC_CRCFAILED_ERROR	7(0x07)	통신중 Checksum 에러가 발생하였습니다.
	FMC_RECVPACKET_ERROR	8(0x08)	받은 패킷에서 프로토콜 레벨의 에러가 발생하였습니다.

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

구분	명칭	리턴값	내용
정상	FMP_OK	0(0x00)	통신이 정상적으로 수행되었습니다.
입력 에러	FMP_FRAMETYPEERROR	128 (0x80)	Board 가 인식하지 못하는 명령어 입니다.
	FMP_DATAERROR	129 (0x81)	입력한 데이터가 범위를 벗어났습니다.
연결 에러	FMP_PACKETERROR	130 (0x82)	Board 가 받은 패킷에서 프로토콜 레벨의 에러가 발생하였습니다.
	FMP_PACKETCRCERROR	170 (0xAA)	Board 가 받은 패킷의 CRC 계산값이 일치하지 않습니다.

2 - 2 . Board 연결함수

함수명	내용
FAS_Connect	드라이브 모듈과 UDP Protocol 로 연결을 시도합니다. : 성공적으로 접속했다면 TRUE 를, 접속에 실패 했다면 FALSE 를 리턴합니다.
FAS_ConnectTCP	드라이브 모듈과 TCP Protocol 로 연결을 시도합니다. : 성공적으로 접속했다면 TRUE 를, 접속에 실패 했다면 FALSE 를 리턴합니다.
FAS_Reconnect	기존의 IP, Protocol, iBdID 로 다시 연결합니다.
FAS_Close	드라이브 모듈과 통신 해지를 시도합니다.
FAS_GetSlaveInfo	드라이브의 종류와 프로그램 Version 을 읽어들입니다. : 드라이브 종류와 Version 정보를 리턴합니다.
FAS_IsSlaveExist	해당 드라이브의 존재 여부를 확인합니다. : 존재하면 TRUE 를, 접속에 실패 했다면 FALSE 를 리턴합니다.
FAS_IsBdIDExist	해당 IP Address 에 BdID 의 사용 여부를 확인합니다. : 존재하면 TRUE 를, 접속에 실패를 했다면 FALSE 를 리턴합니다.
FAS_IsIPAddressExist	해당 BdID 에 IP Address assign 여부를 확인합니다. : 존재하면 TRUE 를, 접속에 실패를 했다면 FALSE 를 리턴합니다.
FAS_EnableLog	통신 오류 관련 Log 의 출력을 제어합니다.
FAS_SetLogPath	출력될 Log 가 저장될 경로를 설정합니다. : 경로가 존재하면 TRUE 를, 경로가 존재하지 않거나 접근할 수 없으면 FALSE 를 리턴합니다.
FAS_SetLogLevel	Log 를 설정된 Level 에 따라 출력합니다. : 기본으로 내부 통신 오류 관련 Log 만 출력하게 설정되어 있습니다. (LOG_LEVEL_COMM)
FAS_PrintCustomLog	임의의 Log 를 출력 합니다.

- **다음의 함수는 F/W Ver V06.01.020.04 Library Ver 2.0.0.10 이상에서 지원합니다**

1) FAS_ConnectTCP

2) FAS_Reconnect

3) FAS_SetLogLevel

4) FAS_PrintCustomLog

FAS_Connect

FAS_Connect 함수는 Ezi-IO Ethernet DIO 을 접속하는 함수입니다.

Syntax

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

Parameters

sb1~4

접속하려는 드라이브의 IP 주소를 입력합니다.

ex) 192.168.0.2 일경우

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

iBdID

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

IP 주소 처럼 동일한 ID 사용이 불가합니다.

Return Value

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

Remarks

Example

```
#include "FAS_ EziMOTIONPlusE.h"

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

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

```
        }

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

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

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

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

See Also

FAS_Close

FAS_ConnectTCP

FAS_ConnectTCP 함수는 Ezi-IO Ethernet DIO 에 TCP Protocol 으로 접속하는 함수입니다.

Syntax

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

Parameters

sb1~4

접속하려는 드라이브의 IP 주소를 입력합니다.

ex) 192.168.0.2 일경우

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

iBdID

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

IP 주소와 같이 동일한 ID 사용이 불가합니다.

Return Value

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

Remarks

Example

```
#include "FAS_ EziMOTIONPlusE.h"

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

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

```
        }

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

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

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

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

See Also

FAS_Close

FAS_Reconnect

FAS_Connect()를 사용하지 않고 재연결하는 함수입니다.

Syntax

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

Parameters

iBdID

다시 연결할 드라이브 ID 번호.

Remarks

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

Example

FAS_Connect 라이브러리 참조.

See Also

FAS_Connect

FAS_Close

해당 ID 의 통신 연결을 해제 합니다.

Syntax

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

Parameters

iBdID

연결을 해제할 ID 번호.

Remarks

Example

FAS_Connect 라이브러리 참조.

See Also

FAS_Connect

FAS_GetSlaveInfo

해당 Board 의 Version 정보 문자열을 받아옵니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID.

pType

해당 Board 의 Type 번호.

lpBuff

Version 정보 문자열을 입력받을 Buffer Pointer.

nBuffSize

lpBuff 의 메모리 할당 크기 값.

Return Value

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

Remarks

Example

FAS_Connect 라이브러리 참조.

See Also

FAS_IsSlaveExist

현재 드라이브가 연결 상태인지를 확인합니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID.

Return Value

TRUE : 연결 상태.
FALSE : 해지 상태.

Remarks

이 함수는 라이브러리에서만 제공되며 프로토콜 프로그램 방식에는 지원되지 않습니다.

Example

FAS_Connect 라이브러리 참조.

See Also

FAS_Connect

FAS_IsBdIDExist

현재 드라이브가 연결 상태인지를 확인합니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

sb1, sb2, sb3, sb4

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

Return Value

TRUE : 해당 BdID 사용

FALSE : 해당 BdID 사용하지 않음

Remarks

이 함수는 라이브러리에서만 제공되며 프로토콜 프로그램 방식에는 지원되지 않습니다.

Example

FAS_Connect 라이브러리 참조.

See Also

FAS_Connect

FAS_IsIPAddressExist

현재 드라이브가 연결 상태인지를 확인합니다.

Syntax

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

Parameters

sb1, sb2, sb3, sb4

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

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

Return Value

TRUE : 해당 IP Address 사용.

FALSE : 해당 IP Address 사용하지 않음.

Remarks

이 함수는 라이브러리에서만 제공되며 프로토콜 프로그램 방식에는 지원되지 않습니다.

Example

FAS_Connect 라이브러리 참조.

See Also

FAS_Connect

FAS_EnableLog

통신 오류 관련 Log 의 출력을 제어합니다.

Syntax

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

Parameters

bEnable

Log 를 출력할지 설정합니다.

Remarks

현재 Process 에서 Ezi-MOTION Ethernet 함수를 사용하는 중 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_ EziMOTIONPlusE.h"

void funcDisableLog()
{
	FAS_EnableLog(FALSE);
	 // 이 후 함수들의 Log 는 출력되지 않습니다.

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

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

		return;
	}

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

See Also

FAS_SetLogPath

FAS_SetLogPath

출력될 Log 가 저장될 경로를 설정합니다.

Syntax

BOOL FAS_SetLogPath(LPCTSTR lpPath);

Parameters

lpPath

Log 가 저장될 절대 경로 문자열.

Return Value

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

Remarks

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

Example

```
#include "FAS_ EziMOTIONPlusE.h"

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

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

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

	// 모든 함수들의 Log 는 C:₩Logs 폴더에 출력됩니다.

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

		return;
	}

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

See Also

FAS_EnableLog

FAS_SetLogLevel

출력될 Log 가 저장될 경로를 설정합니다.

Syntax

BOOL FAS_SetLogLevel(enum LOG_LEVEL level);

Parameters

level

Log 출력의 범위 설정

Return Value

단계 설정값 이외의 값을 입력시에 FALSE 를 리턴합니다.

Remarks

LOG_LEVEL_COMM : 통신 오류 관련 Log 만 출력합니다.

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

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

LOG_LEVEL_ALL : 출력될수 있는 모든 Log 가 출력됩니다.

Example

```
#include "FAS_ EziMOTIONPlusE.h"

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

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

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

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

                return;
        }

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

See Also

FAS_EnableLog

FAS_PrintCustomLog

출력될 Log 가 저장될 경로를 설정합니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

level

Log 출력의 범위 설정

lpszMsg

출력될 Log 의 문자열

Return Value

단계 설정값 이외의 값을 입력시에 FALSE 를 리턴합니다.

Remarks

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

Example

```
#include "FAS_ EziMOTIONPlusE.h"

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

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

}
```

See Also

FAS_ SetLogLevel

2 - 3 . Active Level 관련 함수

함수명	내용
FAS_GetIOLevel	Input or Output Active Level 을 읽어 들입니다.
FAS_SetIOLevel	Input or Output Active Level 을 설정 합니다.
FAS_SaveIOLevel	현재의 Input Active Level or Output Active Level 을 ROM 에 저장합니다. 전원 OFF 후에도 data 가 보존되도록 저장합니다.
FAS_LoadIOLevel	Input Active Level or Output Active Level 을 ROM 영역에서 RAM 으로 읽어 들입니다.

FAS_GetIOLevel

Input or Output Active Level 을 읽어 들입니다.

Syntax

```
int FAS_GetInputLevel(
    BYTE iBdID,
    unsigned long* ulOLevel
);
```

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

ulOLevel

Active Level 값이 저장될 변수 포인터.

Return Value

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

Remarks

Example

```
#include "FAS_ EziMOTIONPlusE.h"

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

        int nRtn;

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

                return;
        }

        nRtn = FAS_GetInputLevel(m_ iBdID, &ulOLevel);
```

```
        _ASSERT(nRtn == FMM_OK);

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

See Also

FAS_SetIOLevel

Input or Output Active Level 을 설정 합니다.

Syntax

```
int FAS_SetIOLevel(
    BYTE iBdID,
    unsigned long uIOLevel
);
```

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID.

uIOLevel

Active Level 을 설정할 값.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : Board 를 연결하기 전 입니다.

FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcInputStatus()
{
        BYTE sb1 = 192, sb2 = 168, sb3 = 0, sb4 = 2; // IP :192.168.0.2
        BYTE iBdID = 0;;     //  Board 의 고유 번호
        unsigned long uIOLevel = 0x0000FF00; // 0~7: low active,   8~15 : high active

        int nRtn;

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

                return;
        }

        nRtn = FAS_SetIOLevel(iBdID, uIOLevel);
```

```
        _ASSERT(nRtn == FMM_OK);

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

See Also

FAS_SaveIOLevel

현재의 Input or Output Active Level 값을 ROM 에 저장합니다.

Syntax

```
Int FAS_SaveIOLevel(
    BYTE iBdID,
);
```

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID.

Return Value

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

Remarks

Example

```
#include "FAS_ EziMOTIONPlusE.h"

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

        int nRtn;

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

                return;
        }
        // ROM 에 저장합니다.
        nRtn = FAS_SavetIOLevel(nPortNo, iSlaveNo)
        _ASSERT(nRtn == FMM_OK); // 명령이 정상적으로 수행되지 않았다면 멈춥니다.

        FAS_Close(iBdID);
}
```

See Also

FAS_LoadIOLevel

ROM 에 저장되어 있는 Input or Output Active Level 값을 불러옵니다.

Syntax

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

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_NOT_OPEN : Board 를 연결하기 전 입니다.

FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

Example

```
#include "FAS_ EziMOTIONPlusE.h"

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

        int nRtn;

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

                return;
        }

         // ROM 에서 Input or Output Level 값을 ROM 에서 RAM 으로불러옵니다
        nRtn = FAS_LoadIOLevel(nPortNo, iSlaveNo)
        _ASSERT(nRtn == FMM_OK); // 명령이 정상적으로 수행되지 않았다면 멈춥니다.

        FAS_Close(iBdID);
}
```

See Also

2 - 4 . Input 제어 함수

함수명	내용
FAS_GetInput	Input / Latch 상태를 읽어 들입니다.
FAS_ClearLatch	해당 bit 의 Latch 를 Clear 합니다.
FAS_GetLatchCount	해당 bit 의 Latch Count 를 읽어 들입니다.
FAS_GetLatchCountAll	Latch Count 전체를 읽어 들입니다. (0~15CH)
FAS_GetLatchCountAll32	Latch Count 전체를 읽어 들입니다. (0~31CH)
FAS_ClearLatchCount	해당 bit 의 Latch Count 를 Reset(0)합니다.

- Ezi-IO-EN-I16 series : 0~15
  Ezi-IO-EN-I8O8 series : 0~7
  Ezi-IO-EN-I32 series : 0~31
  Ezi-IO-EN-I16O16 series : 0~15

FAS_GetInput

Input / Latch 상태를 읽어 들입니다.

Syntax

```
int FAS_GetInput(
    BYTE iBdID,
    unsigned long* uInput,
    unsigned long* uLatch
);
```

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID.

uInput

Input 값이 저장될 변수 포인터.

uLatch

Latch 값이 저장될 변수 포인터.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcInputStatus()
{
        BYTE sb1 = 192, sb2 = 168, sb3 = 0, sb4 = 2; // IP :192.168.0.2
        BYTE iBdID = 0;;      //  Board 의 고유 번호
        unsigned long uInput;
        unsigned long uLatch;

        int nRtn;

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

```
            return;
        }

        nRtn = FAS_GetInput(iBdID, &uInput, &uLatch);

        _ASSERT(nRtn == FMM_OK);

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

See Also

FAS_ClearLatch

해당 bit 의 Latch 를 Clear 합니다.

Syntax

```
int FAS_ClearLatch(
    BYTE iBdID,
    unsigned long uLatchMask
);
```

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID

uLatchMask

Latch 를 Clear 할 bitmask 값.

Return Value

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

Remarks

Example

```
#include "FAS_ EziMOTIONPlusE.h"

void funcInputStatus()
{
        BYTE sb1 = 192, sb2 = 168, sb3 = 0, sb4 = 2; // IP :192.168.0.2
        BYTE iBdID = 0;;     //  Board 의 고유 번호
        unsigned long uLatchMask = 0x0000FFFF;   // (Latch 0~15 Clear)

        int nRtn;

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

                return;
        }

        nRtn = FAS_ClearLatch(iBdID, uLatchMask);
```

```
        _ASSERT(nRtn == FMM_OK);

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

See Also

FAS_GetLatchCount

Latch Count 를 읽어 들입니다.

Syntax

```
int FAS_ClearLatchCount(
    BYTE iBdID,
    unsigned char iInputNo,
    unsigned long* uCount
);
```

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID.

iInputNo

Latch count 를 읽어 들일 Latch 번호

uCount

Latch count 값이 저장될 변수 포인터.

Return Value

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

Remarks

Example

```
#include "FAS_ EziMOTIONPlusE.h"

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

        unsigned iInputNo =0;   // (Latch count 번호, 0~15)
        unsigned long* uCount;

        int nRtn;

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

```
            // 연결이 실패하였습니다.

            return;
        }

        nRtn = FAS_GetLatchCount(iBdID, iInputNo, &uCount);

        _ASSERT(nRtn == FMM_OK);

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

See Also

FAS_GetLatchCountAll

Latch Count 전체를 읽어 들입니다.(0~15 번)

Syntax

```
int FAS_GetLatchCountAll(
    BYTE iBdID,
    unsigned long** ppuAllCount
);
```

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID.

ppuAllCount

Latch count 값(0~15)이 저장될 변수 포인터.

Return Value

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

Remarks

Example

```
#include "FAS_ EziMOTIONPlusE.h"

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

        unsigned long Latch_All_Count[16] = 0 ; //Latch count 0~15 을 저장할 변수

        int nRtn;

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

                return;
        }
```

```
        nRtn = FAS_GetLatchCountAll(iBdID,
                (unsigned long**) Latch_All_Count));

        _ASSERT(nRtn == FMM_OK);

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

See Also

FAS_GetLatchCountAll32

Latch Count 전체를 읽어 들입니다.(0~31 번)

Syntax

```
int FAS_GetLatchCountAll32(
    BYTE iBdID,
    unsigned long** ppuAllCount
);
```

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID.

ppuAllCount

Latch count 값(0~31)이 저장될 변수 포인터.

Return Value

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

Remarks

Example

```
#include "FAS_ EziMOTIONPlusE.h"

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

        unsigned long Latch_All_Count[32] = 0 ; //Latch count 0~31 을 저장할 변수

        int nRtn;

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

                return;
        }
```

```
        nRtn = FAS_GetLatchCountAll(iBdID,
                (unsigned long**) Latch_All_Count));

        _ASSERT(nRtn == FMM_OK);

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

See Also

FAS_ClearLatchCount

해당 bit 의 Latch Count 를 Reset(0)합니다.

Syntax

```
int FAS_ClearLatchCount(
    BYTE iBdID,
    unsigned long uInputMask
);
```

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID.

uInputmask

Latch count 를 Reset(0)할 Latch 번호의 Bitmask 값

Return Value

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

Remarks

Example

```
#include "FAS_ EziMOTIONPlusE.h"

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

        unsigned long uInputmask = 0x000000FF ;   //Latch count 0~7 을 Reset

        int nRtn;

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

                return;
        }
```

```
        nRtn = FAS_ClearLatchCount (iBdID, uInputMask);

        _ASSERT(nRtn == FMM_OK);

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

See Also

2 - 5 . Output 제어 함수

함수명	내용
FAS_GetOutput	Output 과 Trigger(Run or Stop)상태를 읽어 들입니다.
FAS_SetOutput	Output 을 설정합니다.
FAS_SetTrigger	Trigger 에 대한 설정을 합니다.
FAS_SetRunStop	Trigger 의 실행 또는 정지 설정을 합니다.
FAS_GetTriggerCount	선택한 번호의 Trigger 횟수를 읽어 들입니다.

- Ezi-IO-EN-O16 series : 0~15
  Ezi-IO-EN-I8O8 series : 8~15
  Ezi-IO-EN-O32 series : 0~31
  Ezi-IO-EN-I16O16 series : 16~31

FAS_GetOutput

Output 과 Trigger(Run or Stop)상태를 읽어 들입니다.

Syntax

```
int FAS_GetOutput(
    BYTE iBdID,
    unsigned long* uOutput,
    unsigned long* uStatus
);
```

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID.

uOutput

Output 값이 저장될 변수 포인터.

uStatus

Trigger Run/Stop 값이 저장될 변수 포인터.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

Example

```
#include "FAS_ EziMOTIONPlusE.h"

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

        unsigned long uOutput;
        unsigned long uStatus;

        int nRtn;

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

```
            return;
        }

        nRtn = FAS_GetOutput(iBdID, &uOutput, &uStatus);

        _ASSERT(nRtn == FMM_OK);

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

See Also

FAS_SetOutput

Output 을 설정합니다.

Syntax

```
int FAS_SetOutput(
    BYTE iBdID,
    unsigned long uSet,
    unsigned long uClear
);
```

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID.

uSet

Output 을 On 할 bitmask 값.

uClear

Output 을 Off 할 bitmask 값.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

uSet 과 uClear 가 같은 bit 를 설정할 경우 uClear 로 처리합니다.

Example

```
#include "FAS_ EziMOTIONPlusE.h"

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

        unsigned long uSet = 0x0000FF00; (Output 8~15 번을 On 합니다.)
        unsigned long uClear = 0x000000FF; (Output 0~7 번을 Off 합니다.)

        int nRtn;

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

```
            return;
        }

        nRtn = FAS_SetOutput(iBdID, uSet, uClear);

        _ASSERT(nRtn == FMM_OK);

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

See Also

FAS_SetTrigger

Trigger 에 대한 설정을 합니다.

Syntax

```
int FAS_SetTrigger(
    BYTE iBdID,
    unsigned char uOutputNo,
    TRIGGER_INFO* pTrigger
);
```

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID.

uOutputNo

Trigger 를 설정하는 Output 번호

pTrigger

Trigger 를 설정할 구조체 변수.(Count, OnTime, Period)

Return Value

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

Remarks

Ontime 은 [ms]단위입니다.

Example

```
#include "FAS_ EziMOTIONPlusE.h"

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

        TRIGGER_INFO Trg_info;
        unsigned char uOutputNo = 0;

        int nRtn;
        Trg_info.wCount   = 100; //Trigger 출력 횟수 설정
        Trg_info.wOnTime   = 10; // On 시간 설정
        Trg_info.wPeriod   = 30 ; //주기 설정, 30[ms]마다 On : 10[ms] Off : 20[ms]
        Trg_info.wReserved1 = 0;
```

```
        Trg_info.wReserved2 = 0;

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

        nRtn = FAS_SetTrigger(iBdID, uOutputNo, &Trg_info);

        _ASSERT(nRtn == FMM_OK);

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

See Also

FAS_SetRunStop

Trigger 의 실행 또는 정지 설정을 합니다.

Syntax

```
int FAS_SetRunStop(
    BYTE iBdID,
    unsigned long uRun,
    unsigned long uStop
);
```

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID.

uRun

Trigger 를 실행할 bitmask 값.

uStop

Trigger 를 정지할 bitmask 값.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.
FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

uRun 과 uStop 이 같은 bit 를 설정할 경우 uStop 으로 처리합니다.

Example

```
#include "FAS_ EziMOTIONPlusE.h"

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

        unsigned long uRun = 0x0000FF00; (Trigger 8~15 번을 실행합니다.)
        unsigned long uStop = 0x000000FF; (Trigger 0~7 번을 정지합니다.)


        int nRtn;

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

```
            // 연결이 실패하였습니다.

            return;
        }

        nRtn = FAS_SetRunStop(iBdID, uRun, uStop);

        _ASSERT(nRtn == FMM_OK);

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

See Also

FAS_GetTriggerCount

선택한 번호의 Trigger 횟수를 읽어 들입니다.

Syntax

```
int FAS_GetTriggerCount(
    BYTE iBdID,
    unsigned char uOutputNo,
    unsigned long* uCount
);
```

Parameters

iBdID

해당 Board 의 ID 번호. FAS_Connect 함수에서 설정한 iBdID.

uOutputNo

Trigger count 를 읽어 들일 Output 번호.

uCount

Trigger Count 값을 저장할 변수 포인터.

Return Value

FMM_OK : 명령이 정상적으로 수행되었습니다.

FMM_INVALID_SLAVE_NUM : 해당 iBdID 의 board 는 존재하지 않습니다.

Remarks

Example

```
#include "FAS_ EziMOTIONPlusE.h"

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

        unsigned char uOutputNo = 1; (Output 번호. 0~15)
        unsigned long* uCount;

        int nRtn;

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

```
            return;
        }

        nRtn = FAS_GetTriggerCount(iBdID, uOutputNo, &uCount);

        _ASSERT(nRtn == FMM_OK);

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

See Also

3 . 부록 – DHCP 를 이용한 Network 정보 설정

3 - 1 . DHCP 기능

1) DHCP(Dynamic Host Configuration Protocol)란?

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

(Network 정보 : Gateway, Subnet, IP address)

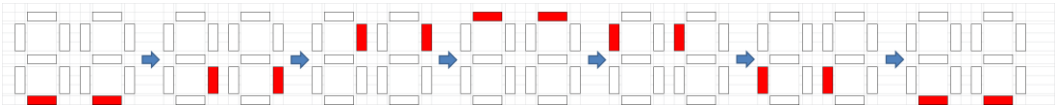
2) DHCP 를 사용하지 않을 경우

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

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

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

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



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

● DHCP 는 동적으로 Network 정보를 설정하기 때문에 전원을 인가할 때마다 IP address – aaa.bbb.ccc.ddd 에서 ddd 는 변경될 수 있습니다. 따라서 DHCP 의 방법으로 Network 정보를 설정한 후, 반드시 3-2. 6)을 수행해야 합니다.

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

FASTECH
Fast, Accurate, Smooth Motion

㈜파스텍
경기도 부천시 평천로 655
부천테크노파크 401동 1202호 (우: 14502)
TEL : 032-234-6300 FAX : 032-234-6302
E-mail : team_sales@fastech-motions.com
Homepage : www.fastech-motions.com

- 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-IO Ethernet DIO 은 국내에 등록된 FASTECH Co.,Ltd.의 등록 상표입니다.

ⓒ Copyright 2016 FASTECH Co.,Ltd. Dec 10, 2021 Rev.04