MoonWalker

i-STEP VSCB24D3S

User's Manual

http://ntrexgo.com

※ 사용자 매뉴얼에 포함된 정보는 정확하고 신뢰성이 있는 내용입니다. 그러나 출판 당시 발견되지 않은 오류가 있을 수 있으니 사용자는 자신의 제품 검증을 수행하시기 바라며, 전적으로 사용자 매뉴얼에 포함된 정보에 의존하지 마시기 바랍니다.

사용 전 유의 사항

본 설명서는 제품의 취급 방법과 안전상의 주의 사항이 기재되어 있습니다. 제품을 사용하시기 전에 설명서를 반드시 숙지하시기 바랍니다.

1. 인명이나 재산상의 큰 기기, 또는 이동형 로봇에 사용할 경우 안전장치(Fuse, 기계적 브레이크 장치)를 설치 후 사용 하시기 바랍니다. (인명사고의 위험이 있습니다)

2. 제품을 분해 또는 개조하지 마십시오. 점검 또는 수리해야 하는 상황 발생 시 본사에 문의해 주시기 바랍니다. (화재, 감전, 파손의 위험이 있습니다)

3. 제품의 주변에 폭발성 물질이나, 인화성 가스가 있는 장소, 부식성 분위기, 물이 닿을 가능성이 있는 장소, 가연성 물질이 있는 부근에서는 사용을 피하여 주십시오.

4. 전원을 연결할 경우 반드시 극성(전원공급기의 단자)을 확인하여 주시기 바랍니다.

5. 모든 배선장치의 연결 또는 해제 시 전원이 꺼져 있는지 확인하시기 바랍니다.

6. 제품을 장시간 운전하였을 경우, 화상에 주의하십시오

7. 전원이 인가된 상태에서는 이동, 설치, 접속, 점검 등의 작업을 하지 마십시오.

8. 비정상적으로 전원을 차단하였을 경우, 다시 전원 인가 시 전체적인 설정 상태를 재확인하시기 바랍니다. (갑작스러운 모터 기동으로 인한 상해의 위험이 있습니다)

9. 방열 및 성능 향상을 위하여 통풍이 잘되도록 유지하여 주시기 바랍니다.

10. 장시간 연속 구동 시 되도록 방열판의 열을 흡수할 수 있는 형태의 샷시에 장착시켜 주시고, 2차 방열판을 권장합니다.

11. 제품의 정격(전압 또는 전력) 이내의 범위에서 사용하시기 바랍니다.

12. 본 설명서를 장비기기와 가까운 곳에 비치해 두시기 바랍니다.

13. 보조 제동장치가 설치되지 않은 조건에서 수직 상하 운동 장치(Z축)에 사용하지 마십시오.

1 제품 소개 1

1.1 제품의 특징 1

1.2 제품의 사양 1

1.3 제품의 규격 2

1.4 컨넥터 핀 정보 3

1.4.1 전원 및 모터 3

1.4.2 D-Sub 커넥터 3

1.4.3 Ethernet 커넥터 4

1.4.4 LED 표시등 4

2 통신 포트 연결 6

2.1 시리얼 포트 설정 6

2.1.1 RS-232 연결 6

2.1.2 RS-485 연결 7

2.2 RJ-45 포트 연결 7

2.2.1 Ethernet 연결 8

2.3 통신 포트의 기능 9

2.3.1 명령의 충돌 9

2.3.2 명령 와치독 타이머 10

3 센서 및 액츄에이터 연결 11

3.1 제어기의 입출력 포트 11

3.2 디지털 출력 포트 11

3.3 디지털 입력 포트 12

3.4 아날로그 입력 포트 13

3.4.1 전압을 출력하는 아날로그 센서 연결 13

3.4.2 포텐셔미터 연결 13

3.5 펄스 입력 포트 **14**

4 제어기의 구조 **15**

4.1 모터 제어기 **16**

4.2 전력단 **16**

4.3 가상 머신 **16**
4.3.1 스택 기반 연산 17
4.3.2 데이터 메모리 17

4.4 오브젝트 **17**
4.4.1 오브젝트의 종류 18
4.4.2 오브젝트의 저장과 읽어오기 18
4.4.3 제어기의 모든 설정 리셋 19

4.5 통신 포트 **20**

4.6 입출력 포트 **20**

5 모터제어기 인터페이스 **21**

5.1 모터 구동 명령 **21**
5.1.1 통신으로 명령 전달 22
5.1.2 스크립트에 의한 명령 전달 22
5.1.3 아날로그/펄스 입력 채널로부터 명령 전달 22

6 I/O 신호 처리 **24**

6.1 디지털 출력 매핑 **25**

6.2 아날로그 입력 캡쳐 **25**

6.3 펄스 입력 캡쳐 **26**
6.3.1 펄스 신호의 캡쳐 26

6.4 아날로그/펄스 입력의 정규화 **27**
6.4.1 정규화된 값으로의 캘리브레이션 28
6.4.2 Min/Max 안전 검사 29

6.4.3 지수/로그 변환 30
6.4.4 센터 안전 검사 30

6.5 입력 값의 매핑, 중재 31
6.5.1 매핑 32
6.5.2 중재 33

7 모터제어기 오브젝트 34

7.1 제품 ID 및 버전 정보 34
7.1.1 vendor_id - Vendor ID 35
7.1.2 product_id - Product ID 35
7.1.3 software_version - Software Version 35
7.1.4 hardware_version - Hardware Version 36

7.2 모터제어기 명령 및 상태 36
7.2.1 system_status - System Status 36
7.2.2 system_command - System Command 37
7.2.3 startup_command - Startup Command 38

7.3 배터리 상태 38
7.3.1 power_source_voltage - Power Source Voltage 38

7.4 시리얼 통신 설정 39
7.4.1 device_id - Device ID 39
7.4.2 serial_baudrate - Serial Baudrate 40
7.4.3 serial_watchdog - Serial Watchdog 40

7.5 이더넷 통신 설정 41
7.5.1 ip_address - IP Address 41
7.5.2 subnet_mask - Subnet Mask 42
7.5.3 gateway_ip - Gateway IP 42
7.5.4 mac_address - MAC Address 42
7.5.5 connection_mode - Connection Mode 42
7.5.6 connection_ip - Connection IP 43
7.5.7 connection_port - Connection Port 44

7.6 사용자/임시 변수 44

7.6.1 user_variable - User Variable 44
7.6.2 temp_variable - Temp Variable 45

7.7 스크립트 다운로드와 실행 45
7.7.1 script_stop_action - Script Stop Action 45
7.7.2 script_size - Script Size 46
7.7.3 script_code - Script Bytecode 46
7.7.4 script_variable - Script Variable 46

8 모터제어기 오브젝트 48

8.1 모터 개수 48
8.1.1 num_motors - Number of Motors 49

8.2 모터 명령 49
8.2.1 command - Command 49
8.2.2 go_position - Go Position 50
8.2.3 go_velocity - Go Velocity 51

8.3 모터 상태 51
8.3.1 status - Status 52
8.3.2 fault - Fault 54
8.3.3 temperature - Temperature 55
8.3.4 velocity - Velocity 55
8.3.5 position - Position 55

8.4 모터의 위치 및 Homing 설정 55
8.4.1 min_position, max_position - Min, Max Position 56
8.4.2 home_position - Home Position 56
8.4.3 use_soft_limit - Use Soft Limit 56
8.4.4 homing_method - Homing Method 57
8.4.5 homing_velocity - Homing Velocity 57

8.5 모터 속성 설정 57
8.5.1 acceleration, deceleration - Acceleration, Deceleration 58

8.6 모터 구동 한계 설정 58
8.6.1 overheat_limit - Overheat Limit 59

8.6.2 overvoltage_limit - Overvoltage Limit 59

8.6.3 undervoltage_limit - Undervoltage Limit 59

8.7 모터 방향 및 정지 지연시간 설정 59

8.7.1 direction - Direction 60

8.8 모터 전류, 분해능, 전류감쇄 설정 60

8.8.1 idle_current_delay - Idle Current Delay 60

8.8.2 idel_current - Idle Current 60

8.8.3 run_current - Run Current 61

8.8.4 excitation_mode - Excitation Mode 61

8.8.5 decay_ratio - Decay Ratio 61

9 I/O 오브젝트 63

9.1 디지털 입력 63

9.1.1 num_dis - Number of DI 64

9.1.2 di_all_values - DI All Values 64

9.2 디지털 입력 채널 64

9.2.1 di_flags - DI Flags 65

9.2.2 di_value - DI Value 65

9.2.3 di_mapping - DI Mapping 65

9.3 디지털 출력 65

9.3.1 num_do - Number of DO 66

9.3.2 do_all_values - DO All Values 66

9.4 디지털 출력 채널 66

9.4.1 do_flags - DO Flags 67

9.4.2 do_value - DO Value 67

9.4.3 do_mapping - DO Mapping 67

9.5 아날로그 입력 67

9.5.1 num_ai - Number of AI 68

9.6 아날로그 입력 채널 68

9.6.1 ai_flags - AI Flags 68

9.6.2 ai_value - AI Value 69
9.6.3 ai_converted_value - AI Converted Value 69
9.6.4 ai_linearity - AI Linearity 69
9.6.5 ai_mapping - AI Mapping 69
9.6.6 ai_input_min, ai_input_center, ai_input_max - AI Input Min, Center, Max 70
9.6.7 ai_input_deadband - AI Input Deadband 70

9.7 펄스 입력 70
9.7.1 num_pi - Number of PI 71

9.8 펄스 입력 채널 71
9.8.1 pi_flags - PI Flags 72
9.8.2 pi_value - PI Value 72
9.8.3 pi_converted_value - PI Converted Value 72
9.8.4 capture_type - PI Capture Type 72
9.8.5 pi_linearity - PI Linearity 73
9.8.6 pi_mapping - PI Mapping 73
9.8.7 pi_input_min, pi_input_center, pi_input_max - PI Input Min, Center, Max 74
9.8.8 pi_input_deadband - PI Input Deadband 74

10 통신 프로토콜 75

10.1 용어의 정의 75

10.2 시리얼 바이너리 패킷 76
10.2.1 바이너리 패킷의 메시지 구조 76
10.2.2 바이너리 패킷의 기본 구조 77
10.2.3 오브젝트 읽기 요청 78
10.2.4 오브젝트 쓰기 요청 78
10.2.5 오브젝트 읽기/쓰기 요청에 대한 성공 응답 78
10.2.6 오브젝트 읽기/쓰기 요청에 대한 실패 응답 79

10.3 시리얼 텍스트 패킷 79
10.3.1 오브젝트 읽기 요청 79
10.3.2 오브젝트 쓰기 요청 80
10.3.3 오브젝트 읽기/쓰기 요청에 대한 성공 응답 80
10.3.4 오브젝트 읽기/쓰기 요청에 대한 실패 응답 81

10.3.5 Device ID의 부여 81

11 MINI-C 스크립트 언어 83

11.1 스크립트 관련 구성 83

11.2 C언어와의 차이 83

11.3 문장 85
11.3.1 수식과 문장 85
11.3.2 복합문 85
11.3.3 제어문 85

11.4 주석 86
11.4.1 블록 주석문 86
11.4.2 라인 주석문 86

11.5 상수 86
11.5.1 정수형 상수 87
11.5.2 실수형 상수 87
11.5.3 미리 정의된 상수 88

11.6 변수 91
11.6.1 변수명 91
11.6.2 변수의 선언과 초기화 91
11.6.3 시스템 변수 92
11.6.4 자료형 변환 94

11.7 연산자 95
11.7.1 산술, 부호 연산자 95
11.7.2 증감 연산자 96
11.7.3 관계 연산자 96
11.7.4 논리 연산자 97
11.7.5 비트 연산자 97
11.7.6 대입 연산자 98
11.7.7 연산자 우선순위 99

11.8 제어문 100

11.8.1 if 문 100

11.8.2 if-else 문 100

11.8.3 if-else-if-else 문 101

11.8.4 while 문 102

11.8.5 for 문 102

11.8.6 do-while 문 103

11.8.7 break 문 103

11.8.8 continue 문 104

11.8.9 goto 문 105

11.9 내장 함수 105

11.9.1 clock 106

11.9.2 rand 106

11.9.3 sleep 107

11.9.4 getv 107

11.9.5 setv 108

11.9.6 int 108

11.9.7 sin 108

11.9.8 cos 109

11.9.9 tan 109

11.9.10 asin 109

11.9.11 acos 109

11.9.12 atan 110

11.9.13 sinh 110

11.9.14 cosh 110

11.9.15 tanh 110

11.9.16 fabs 111

11.9.17 floor 111

11.9.18 ceil 111

11.9.19 sqrt 112

11.9.20 exp 112

11.9.21 log 112

11.9.22 log10 113

11.9.23 atan2 113

11.9.24 pow 114

11.9.25 min 114

11.9.26 max 114

12 프로그램의 작성과 실행 115

12.1 프로그램의 작성 115

12.1.1 소스코드 작성 115

12.1.2 소스코드 구조 115

12.1.3 스크립트 예제 (다음 적절한 예제 코드를 하나 편집 부탁합니다.) 116

12.2 빌드 118

12.2.1 컴파일 118

12.2.2 컴파일 오류 메시지 119

12.3 다운로드 및 실행 120

12.3.1 다운로드 120

12.3.2 실행 120

13 MOTOR CONTROL UI 유틸리티 122

13.1 소프트웨어 다운로드 및 실행 122

13.1.1 시스템 요구사항 122

13.1.2 다운로드 122

13.1.3 실행 122

13.2 메인 화면 구성 123

13.2.1 헤더 123

13.2.2 탭 윈도우 124

13.3 Firmware Update 124

13.3.1 펌웨어 다운로드 125

13.3.2 펌웨어 업데이트 125

13.3.3 펌에어 업데이트 확인 128

13.4 Connection 128

13.5 Real-time Plot 131

13.6 Jog Control **132**

13.7 Motor Control 탭 **133**

13.7.1 Motor Control Status 134

13.7.2 Motor Fault Flags 135

13.7.3 Step Motor Control 135

13.7.4 Power Stage Monitoring 136

13.8 Configuration 탭 **136**

13.8.1 Product Information 137

13.8.2 Startup 138

13.8.3 Connection 139

13.8.4 Motor 141

13.8.5 Motor - Position Limits 141

13.8.6 Motor - Homing 142

13.8.7 Motor - Velocity Profile 142

13.8.8 Motor - Fault Conditions 143

13.8.9 Motor - Stepper Properties 143

13.8.10 Digital Inputs 144

13.8.11 Digital Outputs 145

13.8.12 Analog Inputs 146

13.8.13 Pulse Inputs 148

13.8.14 Calibration 149

13.9 I/O Monitoring 탭 **151**

13.9.1 Digital Inputs 151

13.9.2 Digital Outputs 152

13.9.3 Analog Inputs 152

13.9.4 Pulse Inputs 152

13.10 Script 탭 **152**

13.10.1 Script 작성 153

13.10.2 빌드 및 다운로드 154

13.10.3 실행 154

13.10.4 Variables 154

14 관련 자료 **156**

1 제품 소개

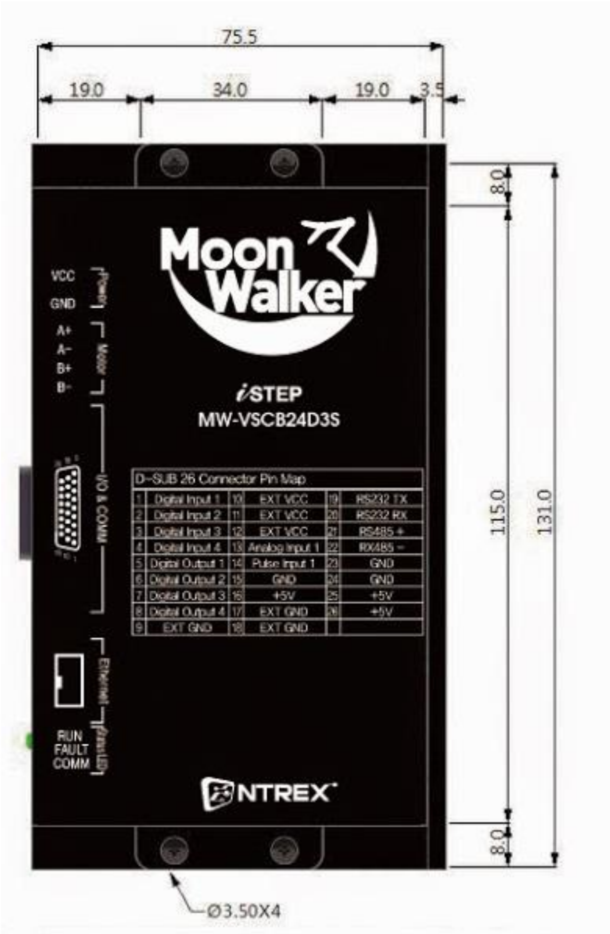
1.1 제품의 특징

- 2상 바이폴라 PWM 정전류 쵸퍼 드라이버
- 마이크로 스텝 설정 가능 (x2, x8, x16, x32, x64, x128, x10, x20)
- 12 ~ 30V의 입력 전압
- 0~3A의 구동전류 및 정지전류 설정 가능
- 5A 안전퓨즈 내장
- 포토 커플러를 이용한 입력 제어 신호 적용으로 외부 노이즈 최소화
- 저전압, 과전압, 과전류 설정 가능
- RS-232, RS-485, Ethernet 인터페이스
- RUN, FAULT, COMM LED 상태 표시
- iSTEP Motion Controller UI 제공

1.2 제품의 사양

모델명	MW-VSCB24D3S
공급 전원	24VDC (Typ), 12 ~ 30V (±20%)
구동 전류	0 ~ 3A
정지 전류	0 ~ 3A
구동 방식	Bipolar Constant current Drive
분해능	x2, x8, x16, x32, x64, x128, x10, x20
최소 펄스폭	Min 0.5u Sec
펄스 간격	Min 0.5u Sec
상승 하강 시간	Max 1u Sec
최대 구동 주파수	120kbps
인터페이스	RS232, RS485, Ethernet
I/O	Digital Output x4, Digital Input x4, Analog Input x1, Pulse Input x1
Mini-C Program Size	~1500 lines of C-language code, 256 user variables
주변 온도	0 ~ 50℃
주변 습도	35 ~ 85% RH
제품 크기	142mm x 75mm x 35mm
제품 무게	g

1.3 제품의 규격


75.5
19.0
34.0
19.0
3.5
8.0
115.0
131.0
8.0
Ø3.50X4
VCC
GND
Power
A+
A-
B+
B-
Motor
I/O & COMM
Ethernet
RUN
FAULT
COMM
Status LED
Moon Walker
iSTEP
MW-VSCB24D3S
D-SUB 26 Connector Pin Map
1 Digital Input 1
2 Digital Input 2
3 Digital Input 3
4 Digital Input 4
5 Digital Output 1
6 Digital Output 2
7 Digital Output 3
8 Digital Output 4
9 EXT GND
10 EXT VCC
11 EXT VCC
12 EXT VCC
13 Analog Input 1
14 Pulse Input 1
15 GND
16 +5V
17 EXT GND
18 EXT GND
19 RS232 TX
20 RS232 RX
21 RS485 +
22 RX485 -
23 GND
24 GND
25 +5V
26 +5V
NTREX

1.4 컨넥터 핀 정보

1.4.1 전원 및 모터

[전원 및 모터 커넥터 이미지 적용]

표 1-1 전원 및 모터 커넥터

No.	Function	Description
1	VCC	Power VCC
2	GND	Power GND
3	A+	Motor A+
4	A-	Motor A-
5	B+	Motor B+
6	B-	Motor B-

1.4.2 D-Sub 커넥터

[D-Sub 커넥터 이미지 적용]

표 1-2 D-Sub 커넥터

No.	Function	Description
1	DIN 1	Digital Input 1
2	DIN 2	Digital Input 2
3	DIN 3	Digital Input 3
4	DIN 4	Digital Input 4
5	DOUT 1	Digital Output 1
6	DOUT 2	Digital Output 2
7	DOUT 3	Digital Output 3
8	DOUT 4	Digital Output 4
9	EXT GND	External (-) Power
10	EXT VCC	External (+) Power
11	EXT VCC	External (+) Power
12	EXT VCC	External (+) Power
13	AIN 1	Analog Input 1
14	PIN 1	Pulse Input 1
15	GND	Ground
16	+5V	+5V
17	EXT GND	External (-) Power

18	EXT GND	External (-) Power
19	TX Data	RS-232 Tx
20	Rx Data	RS-232 Rx
21	Data (B) +	RS-485 +
22	Data (A) -	RS-485 -
23	GND	Ground
24	GND	Ground
25	+5V	+5V
26	+5V	+5V

1.4.3 Ethernet 커넥터

[Ethernet 커넥터 와 LED 이미지 적용]

표 1-3 Ethernet 커넥터

No.	Function	Description
1	RJ-45	Ethernet Connector

1.4.4 LED 표시등

[Ethernet 커넥터 와 LED 이미지 적용]

제어기는 동작 상태를 표시하는 청색, 적색, 녹색(혹은 주황색)의 LED를 가지고 있습니다. 각각의 LED가 표시하는 상태는 표 1-4에서 정리하고 있습니다.

제어기에 전원이 투입되면 3개의 LED가 동시에 0.5초 동안 켜졌다가 꺼지며, 이후 동작 상태를 표시하는 청색 LED만 깜박이게 됩니다. 이는 정상적으로 제어기가 켜진 상태를 표시하는 것입니다.

제어기에 전원을 투입 후, 3개의 LED가 모두 꺼져있거나 켜져 있는 경우는 제어기가 정상적으로 초기화되지 않은 상태입니다. 만일 제어기가 정상적으로 초기화 되었다면 동작 상태를 표시하는 LED는 항상 1초 주기로 깜박이게 됩니다.

3개의 LED가 모두 0.25초 주기로 깜박이는 경우는 제어기의 하드웨어나 소프트웨어가 오작동을 일으켜 소프트웨어 실행이 중단된 상황으로 제어기를 하드웨어적으로 리셋 하거나 전원을 껐다 켜야 합니다. 근본적으로 제어기의 소프트웨어나 하드웨어에 문제가 있는 것을 의미하며 개발자에게 상황을 리포트 하여 소프트웨어나 하드웨어 수준에서 문제를 수정해야 합니다. 이러한 상황은 다음 경우에 발생하게 됩니다:

- 펌웨어 버그로 인한 오작동
- 펌웨어 내부 파라미터 설정의 오류
- 펌웨어에서 함수로 잘못된 파라미터의 전달
- 제어기 하드웨어와 맞지 않는 펌웨어를 다운로드하고 실행

제어기에 새로운 펌웨어를 업데이트하다가 중단하고 제어기를 재시작한 경우, 제어기는 완전한 펌웨어가 설치되지 않아 정상적으로 동작하지 않게 됩니다. 이때는 3개의 LED가 동시에 천천히 깜박입니다. 이런 경우 Motor Control UI 유틸리티를 통해 새로운 펌웨어 다운로드를 완료하여야 합니다.

표 1-4 LED 상태 표시

Status	Pattern	Description
동작 (Run)	1000000000	모터 Disable 상태
	1111100000	모터 Enable 상태
	1111111110	모터 Running 상태
폴트 (Fault)	0000000000	제어기가 정상적으로 동작 중인 상태, 폴트가 발생하지 않은 상태
	1000000000	Overheat 폴트 발생
	1010000000	Undervoltage 폴트 발생
	1010100000	Overvoltage 폴트 발생
통신 (Comm)	0	패킷을 주고받지 않는 상태
	1	마스터 PC가 시리얼 포트로 제어기에 연결되어 패킷을 주고받는 상태

상기 표에서 이진수로 표시된 패턴은 100ms마다 수행되면서 0이면 꺼진 상태를 1이면 켜진 상태를 나타냅니다. 한 패턴이 표시되는 주기는 1초가 됩니다.

2 통신 포트 연결

이 장에서는 PC(Personal Computer)나 마이크로컨트롤러(Microcontroller)를 제어기의 통신 포트에 연결하는 방법에 대해 설명합니다. 제어기는 RS-232, RS-485, Ethernet을 사용할 수 있습니다.

2.1 시리얼 포트 설정

시리얼 포트 설정은 RS-232, RS-485에 공통으로 해당됩니다. 제어기의 기본 시리얼 포트 설정은 다음과 같습니다:

- 115200 bps (제품 초기 설정 값)
- 8-bit data
- 1 start bit
- 1 stop bit
- No parity
- No flow control

제어기의 RS-232, RS-485 통신 속도는 제품 초기 설정 값(Factory Default Value)으로 115200 bps로 설정되어 있습니다. 사용자는 **'Serial Baudrate'** 파라미터 설정을 통해, 통신 속도를 각각 다른 속도로 변경할 수 있습니다. 하지만 통신 속도를 제외한 다른 통신 관련 설정(data bits, start bit, stop bit, parity, flow control)은 변경할 수 없습니다. 사용자는 시리얼 통신을 사용하는 PC 또는 마이크로컨트롤러의 통신 파라미터를 제어기와 일치하도록 설정해야 합니다.

시리얼(RS-232, RS-485) 통신은 흐름 제어 없이 수행되는데, 이는 제어기가 항상 데이터를 수신할 준비가 되어있고 언제든지 데이터를 전송할 수 있다는 것을 의미합니다.

2.1.1 RS-232 연결

제어기는 RS-232 시리얼 통신을 통해서 PC에 접속할 수 있습니다. 이 연결은 제어기에 명령을 전송하고 제어기에서 다양한 상태 정보를 읽어오는데 사용됩니다.

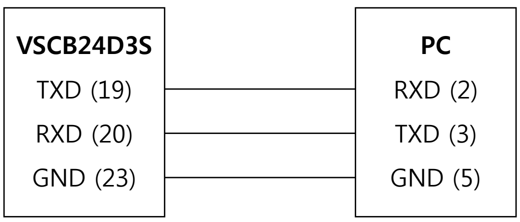


그림 2-1 RS-232 연결

제어기는 Rx, Tx, GND 핀을 통해서 RS-232 통신을 하게 됩니다. 제어기의 Rx 포트를 PC의 COM

포트 Tx)에, Tx 포트를 Rx에 연결한 후 RS-232 통신을 사용하면 됩니다.

2.1.2 RS-485 연결

제어기는 RS-485 시리얼 통신을 통해서 PC에 접속할 수 있습니다. RS-485는 RS-232의 직렬 통신 방법이 아닌 병렬 통신 방법을 사용힙니다. 이를 보통은 EIA 다중 직렬 연결 방법이라고 합니다. RS-232와 RS-485의 차이는 표 2-1를 참조하시기 바랍니다.

표 2-1 RS-232 와 RS-485커넥터

항목	RS-232	RS-485
송수신 방식	비차동	차동
최대 Node 수	1	32
통신 방법	1:1	1:31
최대 통신 거리	15m	1200m
네트웍 그룹핑	불가	불가
암호화	없음	없음
최대 통신 속도	20kbps	10Mbps
최대 출력 전압	±25V	-7 ~ +12V
수신 감도	±3V	±200mV

제어기는 Data +, Data -, GND 핀을 통해서 RS-485 통신을 하게 됩니다. 만약 PC와 연결해서 사용하고 싶다면 **"RS-232 to RS-485 컨버터"** 또는 **"USB to RS-485 컨버터"**를 이용합니다. 제어기의 Data +와 Data – 포트를 컨버터에 연결하고 컨버터는 PC에 연결해서 RS-485 통신을 사용하면 됩니다.

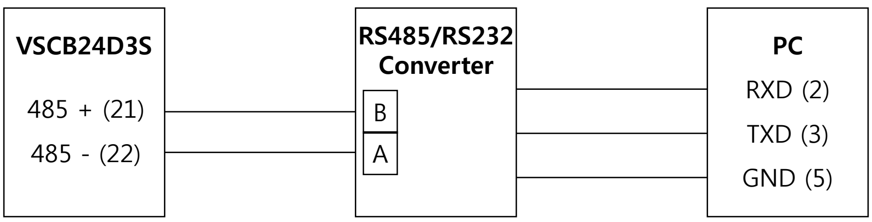


그림 2-2 RS-485 연결

2.2 RJ-45 포트 연결

RJ-45 포트 설정은 Ethernet에 해당됩니다. 제어기의 기본 RJ-45 포트 설정은 다음과 같습니다:

네트워크 설정:

- Device's IP address : 192.168.1.100

- Subnet mask : 255.255.255.0
- Gateway IP address : 192.168.1.1
- Physical(MAC) address : 00-08-DC-XX-XX-XX

이더넷 연결:

- Connection Mode : TCP server
- IP address : 192.168.1.100
- Port number : 5000

2.2.1 Ethernet 연결

RJ-45 포트에는 Ethernet의 UTP 케이블을 연결하면 10Mbit와 100Mbit 이더넷을 자동으로 감지합니다.

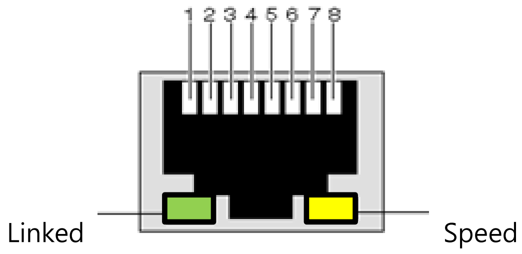


그림 2-3 RJ-45 커넥터

표 2-2 RJ-45 커넥터 핀 정보

PIN	Function	Wire Color
1	Tx +	주황색과 흰색
2	Tx -	주황색
3	Rx +	녹색과 흰색
4	Not used	청색
5	Not used	청색과 흰색
6	Rx -	녹색
7	Not used	갈색과 흰색
8	Not used	갈색

RJ-45 커넥터는 2가지의 LED 표시등이 있습니다. 녹색 LED는 연결 상태를 표시하고 노랑색 LED는 연결 속도를 표시합니다.

표 2-3 RJ-45 커넥터 LED 색깔에 따른 기능

LED Color	Function	Description
녹색	연결 상태 표시	Low : Link (10/100M) High : Un-Link
노랑색	연결 속도 표시	Low : 100Mbps High : 10Mbps

2.3 통신 포트의 기능

PC를 제어기에 통신으로 연결하는 것은 제어기를 용도에 맞게 구성하고 운용하기 위함입니다. RS-232, RS-485, Ethernet 연결은 제어기 오브젝트들의 모든 기능을 엑세스 할 수 있도록 합니다.

사용자가 제어기를 설정하고 제어기를 간단하게 운용 및 모니터링 하기 위해 PC에서 제어기로 RS-232 연결을 하고 Motor Control UI 유틸리티를 실행합니다. PC 유틸리티로 다음과 같은 일들을 쉽게 할 수 있습니다:

- 제어기의 모델과 하드웨어/소프트웨어 버전 읽기
- 모터에 구동 명령을 전송
- 모터의 상태 읽기
- 모터제어기의 구성 파라미터를 설정
- 제어기의 I/O 입력을 읽고 출력을 쓰기

그리고 제어기의 통신 연결을 통해 제어기를 운용하는 마스터 PC나 PLC, 마이크로컨트롤러가 연결될 수 있습니다.

2.3.1 명령의 충돌

제어기의 RS-232, RS-485, Ethernet 통신 포트로 여러 장치와 연결하여 동시에 메시지를 주고받는 것이 가능합니다. 통신 포트로 수신되는 메시지는 동일한 우선순위를 가지며 먼저 도착한 메시지를 먼저 처리하게 됩니다.

이때, 서로 다른 통신 포트들로부터 수신되는 명령이 충돌하지 않도록 주의해야 합니다. 상수나 상태, 구성 파라미터를 읽는 메시지는 서로 충돌하지 않습니다.

※ 만일 명령이 충돌하는 경우, 명령 아비터(arbiter)에 의해 명령이 중재 됩니다. 하지만 사용자가 의도치 않은 동작이 발생할 수 있기 때문에, 하나의 모터 채널에 두 마스터가 동시에 명령을 내리는 상황이 발생하지 않도록 주의하여야 합니다.

2.3.2 명령 와치독 타이머

명령 와치독 타이머는 제어기에 명령을 내리는 마스터 PC나 마이크로컨트롤러가 의도치 않게 명령을 중단하거나 연결이 끊어져 제어기가 더 이상 명령을 수신할 수 없는 상황을 감지하기 위해 사용합니다.

사용자가 명령 와치독 타이머인 **'Serial Watchdog'** 파라미터에 0보다 큰 값을 설정한 경우, RS-232, RS-485, Ethernet 통신을 통해 타임아웃 기간 내에 모터 구동 명령이 도착하면 명령 활성 상태로 간주합니다. 타이머가 타임아웃 되면 비활성으로 간주하며 제어기에 연결된 모든 모터에 대해 정지 명령이 내려집니다. 이때 스크립트나 아날로그 입력, 펄스 입력 포트로 계속 명령이 수신되고 있는 상태라면 모터에 대한 정지 명령은 해제되고 가장 최근에 수신된 명령을 수행하게 됩니다.

RS-232, RS-485, Ethernet 통신으로 수신되는 상수, 상태, 구성 파라미터 읽기/쓰기 메시지는 명령 와치독 타이머에 영향을 주지 않습니다. 또한, 스크립트에 의한 명령과 디지털 입력, 아날로그 입력, 펄스 입력 포트에 의한 명령도 명령 와치독 타이머에 영향을 주지 않습니다.

3 센서 및 액츄에이터 연결

이 장에서는 제어기의 입출력 포트에 센서와 액츄에이터, 기타 액세서리를 연결하는 방법에 대해 설명합니다.

3.1 제어기의 입출력 포트

제어기는 각종 센서와 액츄에이터를 연결할 수 있는 디지털 입력(Digital Input), 디지털 출력 (Digital Output), 아날로그 입력(Analog Input), 펄스 입력(Pulse Input) 포트를 가지고 있습니다. 이러한 입출력 포트는 제어기에서 설정한 기능에 따라 다음과 같은 용도로 사용됩니다:

디지털 입력(Digital Input):
센서(또는 스위치)를 연결해서 모터의 구동/정지, 정방향/역방향을 명령하는데 사용합니다. 또는, 리미트와 홈 센서(또는 스위치)를 연결해서 모터의 이동 한계와 홈 위치를 감지하는데 사용합니다.

디지털 출력(Digital Output):
경고등, 부저를 연결하여 모터의 현재 구동 상태를 외부에 알리는데 사용합니다.

아날로그 입력(Analog Input):
조이스틱으로부터 모터에 내려지는 구동 명령을 읽어 들입니다.

펄스 입력(Pulse Input):
RC 조종기와 수신기로부터 모터에 내려지는 구동 명령을 읽어 들입니다.

제어기의 입출력 포트는 D-Sub 커넥터로 구성되어 있습니다. 사용자는 환경과 목적에 맞게 입출력 포트를 구성하여 사용해야 합니다.

3.2 디지털 출력 포트

디지털 출력(Digital Output) 포트는 모터의 구동 상태(Motor Enable, Motor Running, Motor Fault) 외부에 알리는데 사용합니다.

디지털 출력 포트 말단에는 마이크로프로세서의 출력 신호에 비해 높은 전력의 부하를 구동하고 제어기와의 절연을 위해 PHOTO-TRANSISTOR가 사용됩니다. 다음은 PHOTO-TRANSISTOR 소자 정보입니다:

- Collector-Emitter Voltage : 50V
- Collector Current : 50mA

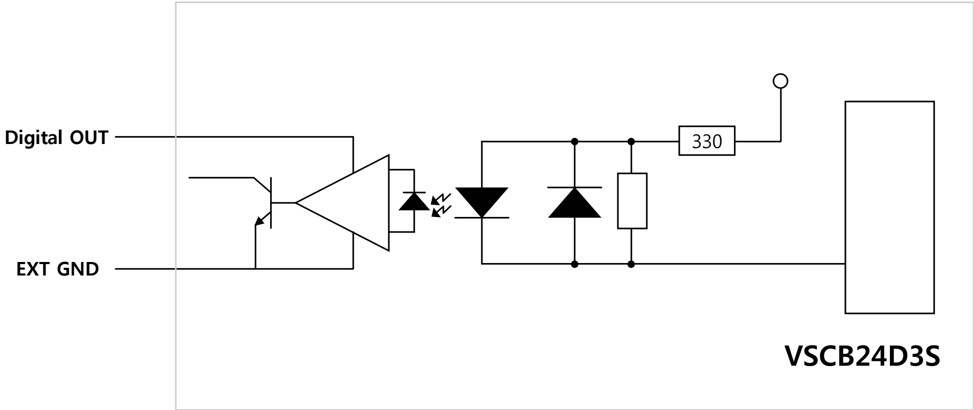


그림 3-1 디지털 출력

※주의※ 비유도성 부하를 연결할 때, 디지털 출력 단의 PHOTO-TRANSISTOR가 구동 가능한 범위 내의 부하를 사용하십시오.

※주의※ 유도성 부하 소자가 스위칭 할 때 유기된 고전압 스파크에 의해 PHOTO-TRANSISTOR가 파괴될 수 있으니 외부적으로 보호 다이오드를 사용하시기 바랍니다.

3.3 디지털 입력 포트

디지털 입력(Digital Input) 포트는 근접센서, 터치센서 등의 외부 디지털 입력 신호를 받아서 모터 정지, 정/역방향 전환, 리미트 등의 기능을 수행하는데 사용합니다.

디지털 입력은 외부의 신호가 마이크로프로세서에 직접 입력되지 않도록 포토트랜지스터(PHOTO-TRANSISTOR)에 연결되어 있습니다. 그리고 외부 전원과 직렬로 22Ω 저항이 연결되어 있습니다. 사용자는 외부 입력 전원에 따라 저항을 추가해 주시기 바랍니다.

- LED Forward Current : 50mA 이하

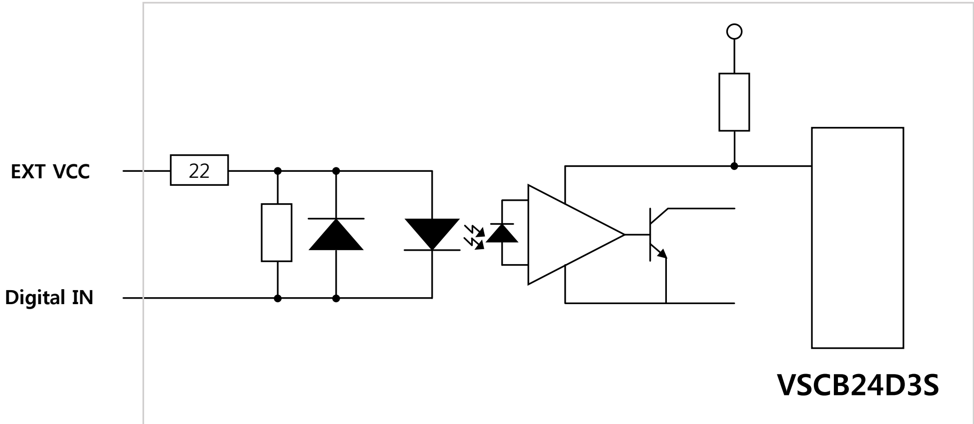


그림 3-2 디지털 입력

3.4 아날로그 입력 포트

제어기의 아날로그 입력(Analog Input) 포트는 전압을 출력하는 아날로그 센서를 연결해서 사용합니다.

아날로그 입력 포트의 입력 버퍼(Input Buffer) 전단에는 100KΩ의 풀업(Pull-Up)과 풀다운(Pull-Down) 저항이 연결되어 있습니다. 또한 포트와 입력 버퍼 간에 직렬로 22Ω 저항이 연결되어 있습니다. 풀업과 풀다운 저항은 아날로그 입력 포트에 아무런 센서도 연결되지 않은 경우 중앙값 2.5V를 읽도록 하기 위해 사용됩니다.

이런 저항들로 인해 아날로그 입력 신호에 약간의 왜곡이 발생하게 됩니다. 하지만 아날로그 입력 포트에 연결된 센서가 파손되거나 끊어진 경우에 모터를 안전하게 구동할 수 있도록 해줍니다.

3.4.1 전압을 출력하는 아날로그 센서 연결

제어기에 전압을 출력하는 아날로그 센서를 연결할 때는 그림 3-3와 같이 아날로그 입력에 직접 연결하면 됩니다.

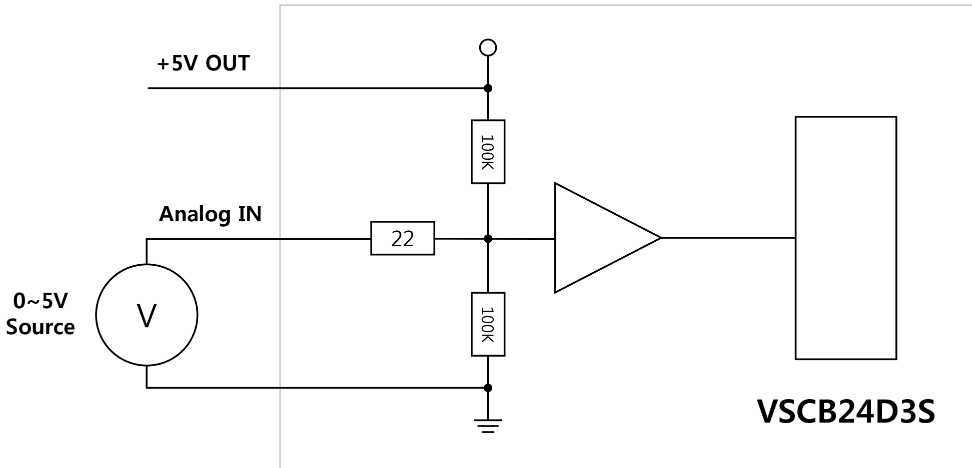


그림 3-3 전압을 출력하는 아날로그 센서 연결

전압을 출력하는 센서를 연결한 경우 입력 버퍼(Input Buffer) 전단의 100KΩ 풀다운(Pull-Down) 저항과 22Ω 저항 때문에 입력 전압을 분배하는 효과가 있습니다. 이로 인한 오차는 대략 22/100K = 0.00022 정도로 마이크로 프로세서의 AD 변환 해상도보다 낮아서 무시할 수 있습니다.

3.4.2 포텐셔미터 연결

포텐셔미터(Potentiometer)는 흔히 우리가 알고 있는 가변저항입니다. 가변 저항은 돌린 위치에 따라 저항값이 바뀌기 때문에 종종 모터의 회전 위치를 검출하는데 사용됩니다. 또한, 조이스틱의 내부에 설치되어 스틱의 위치를 검출하는데도 사용됩니다.

포텐셔미터는 그림 3-4과 같이 제어기의 아날로그 입력 포트에 연결되어, 일반적으로 속도제어나 위치제어 모드에서 제어기에 속도 명령을 내리는데 사용합니다.

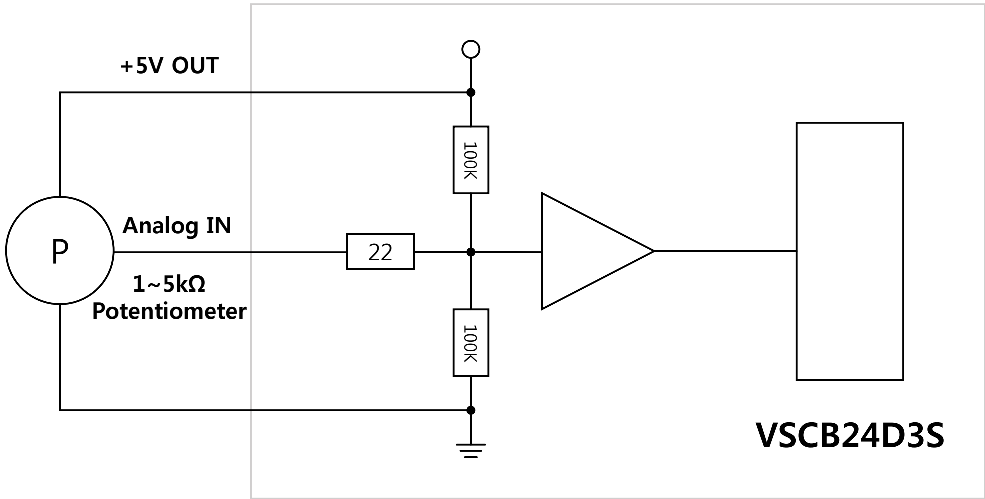


그림 3-4 포텐셔미터 연결

포텐셔미터의 저항 값을 선정할 때는 포텐셔미터에 흐르는 전류를 고려하여도 되도록 낮은 수치의 저항 값을 가지도록 해야 합니다. 저항 값이 너무 높으면 입력 버퍼(Input Buffer) 전단의 풀업(Pull-Up) 저항과 풀다운(Pull-Down) 저항에 의해 입력 전압이 왜곡됩니다. 일반적으로 1K~5K의 저항을 가지는 포텐셔미터를 사용하기를 권장합니다.

입력 포트와 포텐셔미터 간의 배선이 길어지는 경우 혹은 높은 저항 수치의 포텐셔미터를 사용하는 경우 아날로그 입력 신호에 노이즈 및 왜곡이 발생할 수 있으므로 주의해야 합니다.

3.5 펄스 입력 포트

펄스 입력(Pulse Input) 포트는 RC 조종기와 같이 펄스를 기반으로 하는 장치를 연결할 수 있습니다. 펄스 입력으로 펄스 폭(Pulse Width), 주파수(Frequency), 듀티비(Duty cycle) 중 하나를 사용자가 선택하여 캡쳐할 수 있습니다. 각 캡쳐 타입에 따른 입력 범위는 다음과 같습니다:

- Pulse Width : 0 ~ 50000 μs
- Frequency : 20Hz ~ 20kHz
- Duty Cycle : 0 ~ 1000 ‰:

캡쳐 타입은 연결하고자 하는 외부 장치의 출력 신호의 형태에 따라 결정되므로, 사용하고자 하는 장비의 종류를 선정할 때 캡쳐 타입과 입력 범위를 잘 고려해야 합니다.

4 제어기의 구조

이 장에서는 제어기의 내부 구조에 대하여 다룹니다. 제어기의 내부 구조를 파악하는 것은 제어기를 올바르게 운용하는데 꼭 필요한 내용이므로, 사용자는 본 장의 내용을 숙지하기 바랍니다.

제어기 하드웨어의 논리적인 구조는 다음그림 4-1과 같습니다. 제어기에 공급되는 전원(Power Source)은 전력단(Power Stage)을 통해 모터(Motor)를 구동하는데 사용됩니다. 또한, DC/DC 컨버터(Converter)를 거쳐 마이크로컨트롤러(Microcontroller)에 전원을 공급합니다.

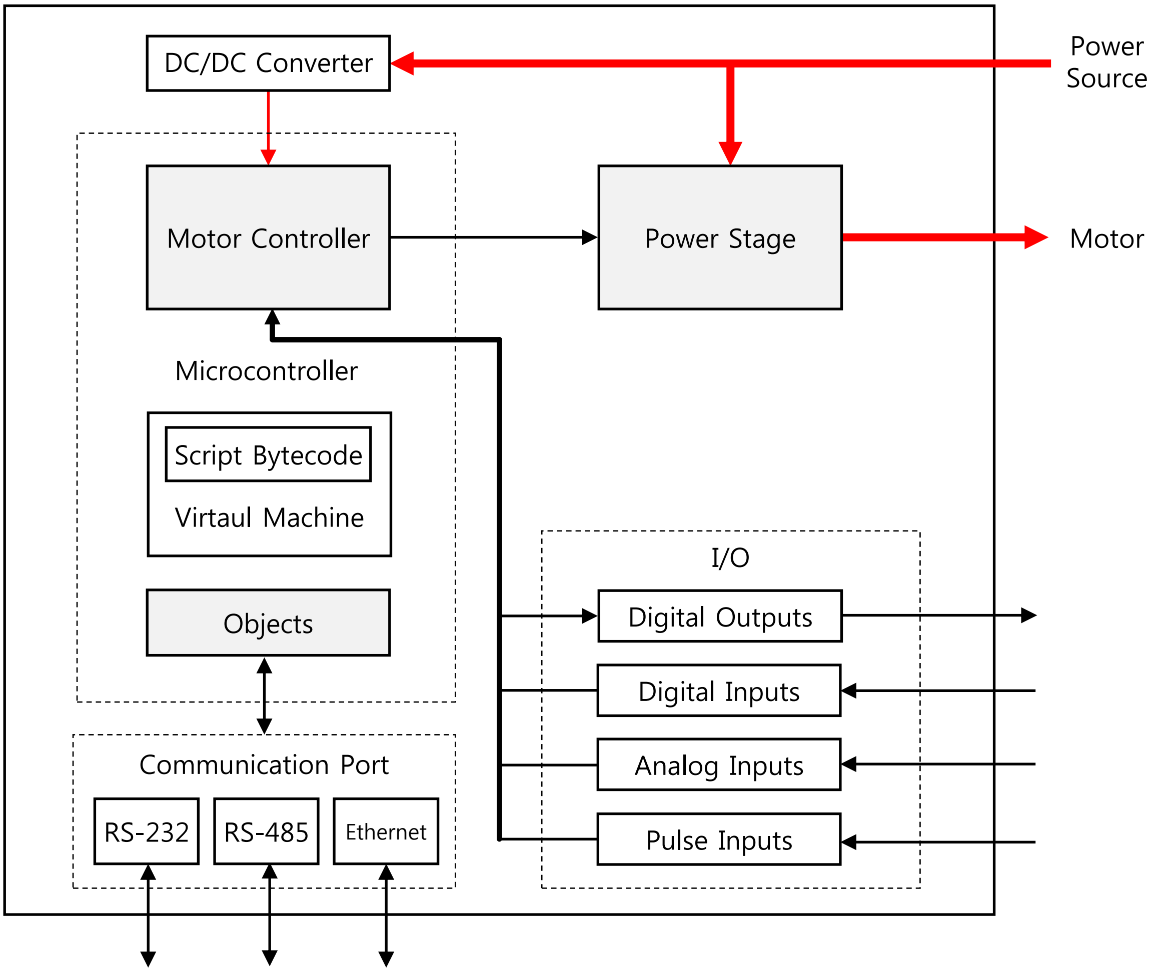


그림 4-1 제어기의 내부 구조

제어기의 마이크로컨트롤러에서 실행되는 프로그램은 모터제어기(Motor Controller)와 가상머신(Virtual Machine)으로 구성됩니다. 제어기에는 PC와 연결하기 위한 통신 포트가 있습니다. 그리고 각종 센서와 액츄에이터를 연결하는 I/O(Digital Outputs, Digital Inputs, Analog Inputs, Pulse Inputs)를 가집니다.

4.1 모터 제어기

제어기에서 가장 핵심이 되는 부분은 모터제어 알고리즘을 수행하는 모터제어기입니다. 여기서 모터제어기란 모터의 출력이 원하는 목표 값이 되도록 모터에 공급되는 전력을 적절히 조절하는 소프트웨어 알고리즘을 말합니다. 사용자는 모터제어기에 위치나 속도 명령을 내리고 모터제어기는 모터가 명령에 따르도록 속도를 출력합니다.

4.2 전력단

전력단(Power Stage)은 스테핑 모터의 구동을 위한 전력 변환 장치입니다. 다음 그림 4-2과 같이 MicroController에서 Pulse 구동 신호를 Power Stage에 보내면 모터 구동을 위한 출력 전압을 만들어 스테핑 모터를 구동합니다.

이는 2상 4선식 스테핑모터의 바이폴라 구조이며, 저속 구동시 토크가 높고 코일에 저장된 전류가 회생되어 효율이 높은 장점이 있습니다.

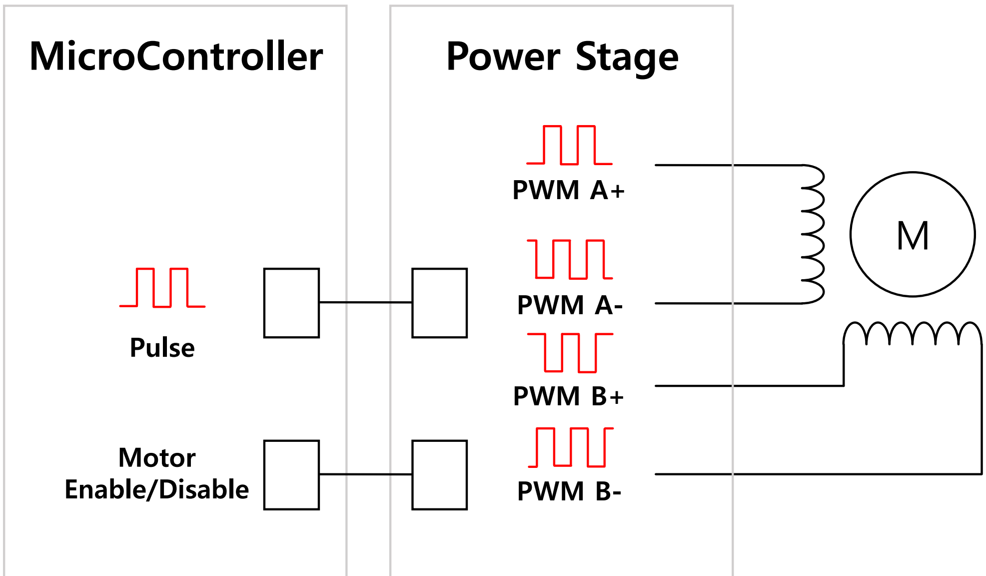


그림 4-2 2상 4선식 스테핑모터 구동을 위한 전력단 구조

4.3 가상 머신

가상머신(Virtual Machine)이란 어떤 기계 구조나 하드웨어 플랫폼을 모방한 환경, 또는 이것에 의해 실현된 가상적인 기계 환경을 의미합니다. 제어기는 사용자가 작성한 스크립트를 해석하여 실행하는 가상머신을 가지고 있습니다.

사용자가 스크립트 언어로 작성한 소스코드는 컴파일러에 의해 바이트코드로 컴파일 됩니다. 그

리고 바이트코드는 제어기에 다운로드 되어 플래시 메모리에 저장됩니다. 이때 최대 60Kbyte까지 저장 가능합니다. 이후, 가상머신은 플래시 메모리에서 바이트코드를 읽어와 해석하여 실행합니다.

4.3.1 스택 기반 연산

가상머신은 스택을 기반으로 동작합니다. 먼저 스택에 피연산자를 넣고 연산자가 스택의 피연산자를 꺼내 연산하고 결과는 다시 스택에 넣습니다. 하나의 수식문에 대한 연산이 끝날 때까지 이 과정을 반복하게 됩니다.

스택에는 최대 32개의 4byte 정수형 혹은 8byte 실수형 피연산자를 저장할 수 있습니다. 스택의 용량은 일반적인 수식문의 연산에 충분한 크기입니다. 하지만 너무 긴 수식문을 작성하게 되면 프로그램 실행 중 스택 오버플로우가 발생할 수 있습니다.

4.3.2 데이터 메모리

가상머신은 또한 프로그램 내의 전역 변수를 저장하기 위한 데이터 메모리를 가지고 있습니다. 데이터 메모리에는 최대 256개의 4byte 정수형 혹은 8byte 실수형 전역 변수를 담을 수 있습니다.

가상머신에서 프로그램이 실행될 때 사용되는 전역 변수 값은 Motor Control UI 유틸리티에서 읽어올 수 있습니다. 프로그램의 전역 변수를 읽는 것은 프로그램의 실행 상태를 모니터링하고 프로그램을 디버깅하는 것은 쉽게 합니다.

제어기의 전원을 켜거나 리셋 후 프로그램이 한 번도 실행되지 않았다면 모든 전역 변수는 0의 값을 가집니다. 프로그램의 실행이 종료되었다면, 전역변수에는 프로그램에서 마지막으로 저장한 값이 남아있습니다.

4.4 오브젝트

제어기에 명령(Command)을 내리거나 제어기의 구성 파라미터(Configuration Parameter)를 설정하는 것은 모터제어기 내부의 오브젝트(Object)에 값을 쓰는 것을 말하며, 제어기의 상수나 상태를 모니터링하는 것은 모터제어기 내부의 오브젝트에서 값을 읽는 것을 말합니다. 이처럼 제어기가 가진 각종 오브젝트는 제어기와 통신으로 연결된 다른 장치와 데이터를 교환하는 기본 단위가 됩니다.

4.4.1 오브젝트의 종류

모터제어기 내부의 오브젝트들(Objects)은 다음과 같이 5가지 속성을 가지는 그룹으로 나누어 집니다:

- 상수(Constant) 오브젝트: 읽기 전용, 모터제어기 실행 도중 변하지 않음
- 명령(Command) 오브젝트: 쓰기 전용, 제어기가 수행할 명령 전달
- 상태(Status) 오브젝트: 읽기 전용, 제어기의 구동 상태를 표시
- 구성 파라미터(Configuration Parameter) 오브젝트: 읽고 쓰기 가능, 플래시 메모리 저장
- 변수(Variable) 오브젝트: 읽고 쓰기 가능, 플래시 메모리에 저장 안됨

상수 오브젝트는 제어기에서 고정된 값으로 변하지 않는 오브젝트입니다. 제어기의 소프트웨어/하드웨어 버전, 모델 번호와 같은 것들이 여기에 해당합니다. 보통 마스터 PC가 제어기를 처음 연결할 때, 필요한 상수 오브젝트들을 한꺼번에 읽어옵니다.

명령 오브젝트는 제어기가 지정된 동작을 실행하도록 하는 명령 코드값을 쓰거나 명령 인자(argument)를 쓰는 오브젝트입니다. 명령 코드를 쓰는 것은, 명령 오브젝트가 이미 개별 코드에 대해 **'0 - Motor Disable, 1 - Motor Enable, 7 - Quick Stop Motor'**와 같이 기능이 부여된 상태에서 한 가지 기능을 선택하는 것과 같습니다. 예를 들면, 모터 Disable 명령을 "`command = 0`"과 같이 사용하는 것입니다.

명령 인자를 쓰는 것은, 특정 기능을 수행하도록 구성된 명령 오브젝트에 인자와 함께 실행 명령을 내리는 것과 같습니다. 예를 들면, 모터를 50 pulse/sec의 속도로 구동하라는 명령을 "`go_velocity = 50`"과 같이 사용하는 것입니다.

상태 오브젝트는 제어기의 내부 상태를 저장하고 있는 오브젝트로 마스터 PC는 읽기만 가능합니다. 이 값은 제어기가 실행되면서 계속 바뀌게 됩니다. 보통 마스터 PC가 제어기의 구동 상태를 알기 위해 상태 오브젝트를 주기적으로 읽기 됩니다.

구성 파라미터는 오브젝트는 제어기의 구동과 관련된 여러 파라미터를 설정하는데 사용됩니다. 제어기가 가진 대부분의 오브젝트가 구성 파라미터이며, 이 오브젝트를 어떻게 설정하는지에 따라 제어기를 다양한 방식으로 동작시킬 수 있게 됩니다.

4.4.2 오브젝트의 저장과 읽어오기

그림 4-3에서와 같이 통신 포트를 통해 엑세스 되는 오브젝트들은 대부분 RAM(Random Access Memory) 영역에 존재합니다. RAM 영역에는 수시로 값이 바뀔 수 있는 구성 파라미터(Configuration Parameter)와 명령(Command), 상태(Status), 변수(Variable) 오브젝트가 저장됩니다. 하지만 제어기의 전원이 꺼지더라도 사용자가 설정한 값이 유지되어야 하는 구성 파라미터의 경우 플래시 메모리(Flash Memory)에 저장되어야만 합니다. ROM(Read Only Memory) 영역에는 값이 변하지 않는 상수(Constant) 오브젝트가 저장됩니다. 그리고 제품 초기 설정 값(Factory Default

Value)도 저장되어 있어, 필요에 따라 제어기의 모든 구성 파라미터를 초기값으로 재설정할 수 있게 합니다.

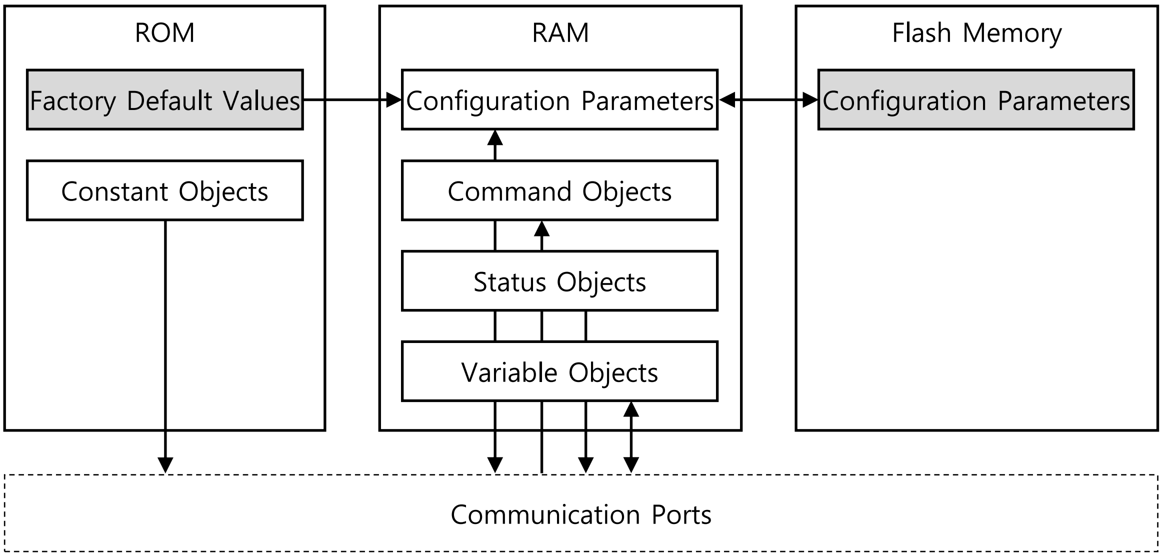


그림 4-3 오브젝트가 저장된 영역과 통신 포트를 통한 오브젝트의 엑세스

제어기가 시작되면서 플래시 메모리에 저장된 구성 파라미터(Configuration Parameters) 오브젝트들을 RAM 영역으로 읽어옵니다. 또한 RAM 영역의 다른 오브젝트들은 모두 0으로 초기화 됩니다.

사용자가 Motor Control UI 유틸리티를 사용하여 구성 파라미터의 값을 변경하면 RAM 영역에서만 반영됩니다. 만일 변경한 값이 제어기에 영구적으로 반영되어야 한다면, RAM 영역의 구성 파라미터 값을 플래시 메모리 영역으로 복사해야 합니다. 이는 제어기에 **'1 - Save properties in Flash Memory'** 명령을 내림으로 가능합니다.

제어기가 시작된 후에도 플래시 메모리에 저장된 구성 파라미터들을 RAM 영역으로 다시 읽어올 수 있습니다. 이는 제어기에 **'2 - Load properties from Flash Memory'** 명령을 내림으로 가능합니다.

Motor Control UI 유틸리티를 사용하면 제어기의 구성 파라미터 값을 변경하고 명령을 내리는 것을 쉽게 할 수 있습니다.

4.4.3 제어기의 모든 설정 리셋

제어기의 구성 파라미터가 엉켜 제어기가 의도치 않게 동작할 때는 제어기의 구성 파라미터를 제품 초기 설정 값(Factory Default Value)으로 되돌리고 새로 설정하는 것이 좋습니다. 이는 제어기에 **'98 - Reset to Factory Default'** 명령을 내림으로 가능합니다.

4.5 통신 포트

마스터 PC는 제어기의 RS-232, RS-485, Ethernet 으로 연결이 가능합니다.

4.6 입출력 포트

제어기는 각종 센서와 액츄에이터를 연결하는 입출력(Digital Outputs, Digital Inputs, Analog Inputs, Pulse Inputs) 포트를 가집니다.

5 모터제어기 인터페이스

이번 장에서는 모터제어기에 전달되는 명령과 피드백 신호의 처리를 다룹니다. 다음 그림 5-1은 모터제어기의 입출력과 관련된 내부와 외부 인터페이스 구조를 보여주고 있습니다.

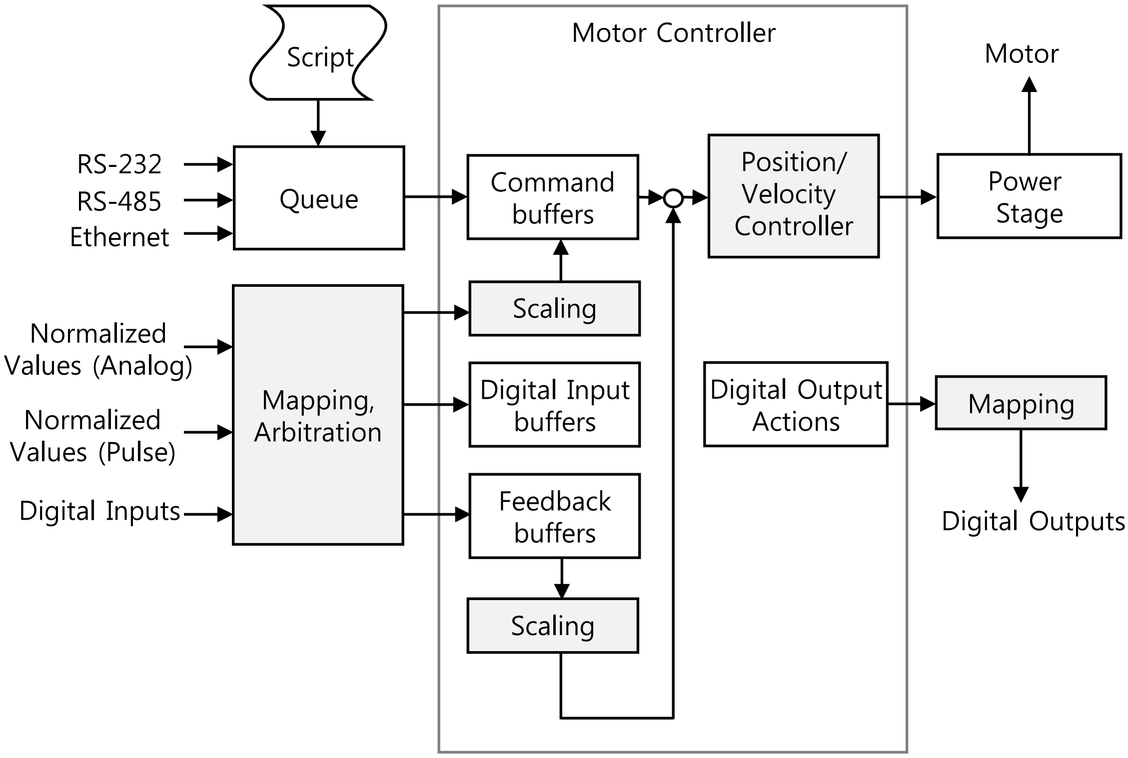


그림 5-1 모터제어기와 외부 I/O의 연결

5.1 모터 구동 명령

모터제어기는 다음 소스 중 하나에서 모터의 구동 명령을 받아들입니다:

- 통신 포트 (RS-232, RS-485)
- 스크립트
- 펄스 입력 채널 (조이스틱, RC수신기)
- 아날로그 입력 채널 (조이스틱)

모터제어기는 상기 모든 명령 입력 소스를 통해 동시에 하나 이상의 모터 구동 명령을 받아들일 수 있습니다. 여러 명령이 동시에 입력되면, 제어기는 명령이 들어온 순서대로 실행하게 됩니다. 이때 서로 다른 소스들로부터 들어오는 명령 간에 충돌하지 않도록 주의해야 합니다.

5.1.1 통신으로 명령 전달

통신(RS-232, RS-485, Ethernet) 포트를 통해 명령이 동시에 내려지기도 합니다. 이때는 도착한 순서대로 큐(queue)에 명령이 들어갑니다. 그리고 매 1ms마다 큐를 검사하여 큐에 명령이 있다면 도착한 순서대로 명령을 실행합니다.

명령을 실행하는 것은 모터제어기의 명령 버퍼(Command buffer)에 명령의 인수를 쓰는 것과 같다고 볼 수 있습니다. 명령 버퍼에는 위치와 속도, 전류, 전압 명령을 받아들일 수 있는 4개의 버퍼 메모리가 있습니다. 만일 사용자가 통신 포트로 모터를 500 pulse/sec으로 구동하라는 명령을 내렸다면, 속도 명령 버퍼에 500이라는 값을 쓰고 모터제어기를 속도모드로 바꿔 주게 됩니다.

5.1.2 스크립트에 의한 명령 전달

명령은 스크립트에 의해 생성되기도 하는데, 스크립트에 의해 생성된 명령도 통신 포트로 수신되는 명령과 같은 방법으로 처리됩니다. 스크립트에 의해 보내진 명령은 큐에 들어가고 이후 통신 포트를 통해 수신된 명령과 같게 처리됩니다.

만일 통신 포트로부터 들어온 명령과 스크립트가 생성한 두 명령이 같은 채널의 모터와 관련되어 있을 때는 충돌이 발생할 수 있습니다. 이러한 상황이 발생하지 않도록 주의가 필요합니다. 예를 들자면, 스크립트 명령이 모터를 특정 위치로 이동할 것을 명령하고 반면 통신 포트의 명령은 다른 위치로 이동할 것을 명령하는 것입니다.

5.1.3 아날로그/펄스 입력 채널로부터 명령 전달

아날로그나 펄스 입력 채널 또한 명령을 수신하는 용도로 사용될 수 있습니다. 예를 들자면, 아날로그 입력 포트에 조이스틱을 연결하여 모터를 조종하는 경우입니다. 이때도 아날로그 입력 채널로부터 입력되는 명령과 통신 포트에서 수신되는 명령 간에 충돌하지 않도록 주의해야 합니다. 만일 조이스틱으로 모터를 조종하고 있을 때는 RS-232, RS-485로 연결된 마스터 PC에서 명령을 내리지 않도록 조치하여야 합니다.

아날로그나 펄스 입력 채널로부터의 명령은 큐를 통해 입력되는 명령과 차이가 있습니다. 큐를 통하는 명령과 달리, 이 명령은 연속적으로 입력되는 명령이라 볼 수 있습니다. (실제로 아날로그나 펄스 입력도 주기적으로 샘플링 되는 이산 신호지만, 여기서는 다른 명령과 비교하기 위해서 연속 신호로 봅니다)

연속적인 명령이 스케일 변환(Scaling)을 거쳐 명령 버퍼에 지속적으로 입력되면 큐를 통해 입력되는 명령은 처리될 수 없게 됩니다. 그래서 연속적인 명령은 값이 변할 때만 명령 버퍼에 입력되도록 이산 명령으로 바뀌게 됩니다. 이산 형태로 바뀐 명령이 명령 버퍼에 입력되면 이후 처리 과정은 통신 포트나 스크립트에 의한 명령과 같아집니다.

6 I/O 신호 처리

제어기의 디지털, 아날로그, 펄스 입력과 디지털 출력은 다양한 용도로 이용 될 수 있습니다. 아래 표는 제어기에서 사용되는 I/O 형태와 기능 그리고 연결되는 센서와 액츄에이터를 보여줍니다.

표 6-1 제어기에 사용되는 I/O의 기능과 장치 연결

I/O Type	Function	Sensor, Actuator
Digital Output	- Motor Enable - Motor Running - Motor Fault	Buzzer, LED, Light, Brake, Shunt Load, Cooling Fan, Relay, Valve, Motor, Solenoid
Digital Input	- Emergency/Quick/Slowdown Stop - Run Script - Forward/Reverse Limit Switch - Home Sensor	Switch류
Analog Input	- Motor Command: Velocity, Position	Joystick, Potentiometer,
Pulse Input	- Motor Command: Velocity, Position	RC Radio, Absolute Encoder

아래 그림 6-1은 제어기의 I/O 입출력 신호의 전체 흐름을 단순화 된 블록 선도로 보여줍니다.

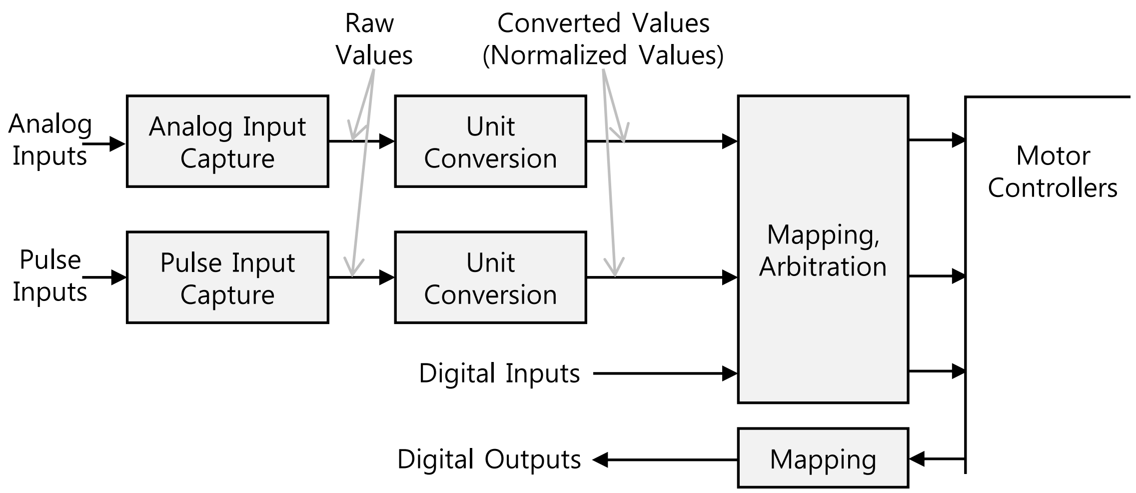


그림 6-1 I/O 입출력 신호의 전체 흐름을 보여주는 블록 선도

아날로그 입력과 펄스 입력 채널로부터 캡처된 신호는 처리 순서를 따라 조정되어 모터제어기에 입력됩니다. 캡처된 신호는 캡처 방법에 따라 값의 범위가 서로 다릅니다. 이 신호는 단위 변환(Unit Conversion) 블록에서 -1과 1사이의 정규화된 값으로 조정됩니다. 이후, 디지털 입력 신호와 함께 매핑, 중재(Mapping, Arbitration) 과정을 거쳐 최종적으로 모터제어기에 입력됩니다.

아날로그 입력이나 펄스 입력이 완전히 처리된 후에는 모터 명령으로 사용될 수 있습니다. 만일 모터제어기가 위치제어 모드나 속도제어 모드로 동작하는 경우는 피드백(속도, 위치)으로 사용될

수도 있습니다.

모터제어기의 디지털 출력 신호는 매핑(Mapping) 과정만을 거치고 제어기의 디지털 출력 채널로 나갑니다.

6.1 디지털 출력 매핑

모터제어기의 디지털 출력 버퍼 값은 매핑 경로를 따라 제어기의 디지털 출력 채널로 나갑니다.

디지털 출력 채널의 매핑과 관련된 구성 파라미터는 다음과 같습니다:

- **do_mapping - DO Mapping**

사용자는 **'DO Mapping'** 파라미터를 설정하여, 제어기의 디지털 출력 채널을 모터제어기의 디지털 출력 버퍼의 기능으로 매핑할 수 있습니다. 아래의 그림 6-2는 사용 가능한 기능을 매핑 한 예를 보여줍니다. 여기서 검은색 화살표는 신호의 흐름이고 파란색 화살표는 매핑의 방향입니다.

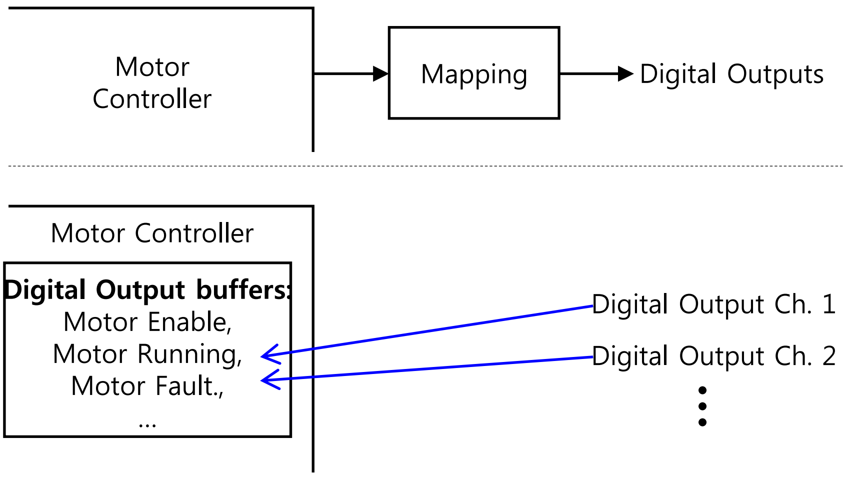


그림 6-2 디지털 출력 채널의 매핑 예

매핑의 설정은 디지털 출력 채널을 기준으로 모터제어기의 디지털 출력 버퍼의 기능을 선택하게 됩니다. 디지털 출력의 매핑은 출력 신호의 충돌을 만들지 않습니다. 즉, 하나의 소스를 여러 출력 채널로 내보내는 것은 문제가 되지 않습니다.

Motor Control UI 유틸리티를 사용하여 디지털 출력 채널의 매핑을 쉽게 할 수 있습니다.

6.2 아날로그 입력 캡쳐

아날로그 입력은 제어기에서 가장 간단하고 일반적인 입력 수단입니다. 보통 포텐셔미터나 포텐

셔미터를 내장한 조이스틱을 연결하여 모터를 조종하는데 사용됩니다.

아날로그 입력 채널은 0과 5V 사이의 전압을 읽습니다. 전압은 12bits AD 컨버터에 의해 디지털로 변환되어 마이크로컨트롤러가 읽게 됩니다. 읽은 값은 0과 4095 사이의 원시 값(raw value)이 됩니다.

6.3 펄스 입력 캡쳐

펄스 입력은 보통 RC 조종기와 수신기를 연결하여 로봇을 원격 조종하는데 사용됩니다. 그리고 모터의 회전각을 측정하는 절대 엔코더를 펄스 입력에 연결할 수도 있습니다.

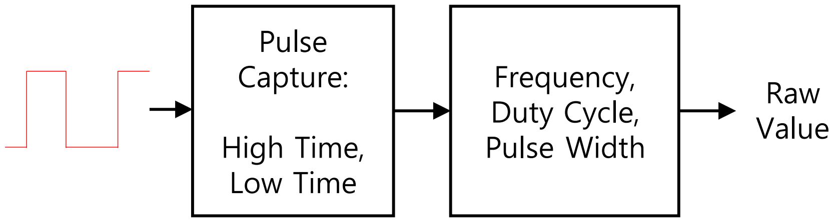


그림 6-3 펄스 신호 캡쳐

펄스 입력 캡쳐와 관련된 구성 파라미터는 다음과 같습니다:

- **PI Capture Type**

6.3.1 펄스 신호의 캡쳐

다음 그림 6-4과 같이, 펄스에서 ON 상태는 High time으로 OFF 상태는 Low Time으로 측정됩니다. Pulse Period는 High Time과 Low Time을 더한 시간입니다.

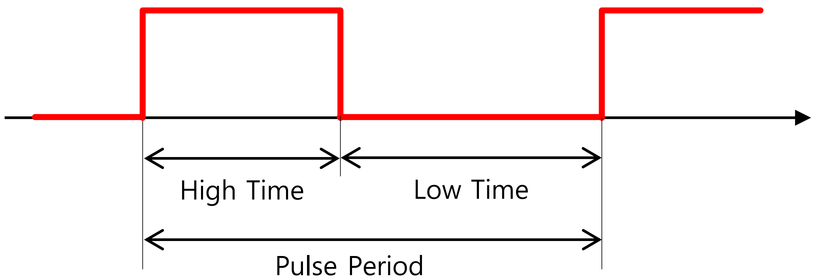


그림 6-4 펄스 신호의 구성

사용자가 **'PI Capture Type'** 파라미터의 설정에 의해, 다음과 같이 Frequency, Duty Cycle, Pulse Width 중 하나를 캡쳐 할 수 있습니다:

- Pulse Width = High Time X 1,000,000 (단위: μs)
- Frequency = 1 / Pulse Period (단위: Hz)

- Duty Cycle = 1000 X High Time / Pulse Period(단위: ‰)

Pulse Width는 펄스의 주기에 관계없이 펄스의 ON 시간(High Time)을 측정합니다. Frequency 캡쳐는 펄스 주기(Pulse Period)의 역수로부터 계산합니다. Duty Cycle은 펄스의 주기에 대해 상대적으로 펄스의 ON 시간을 측정합니다. Duty Cycle은 PWM 발진기의 주파수 유동을 보상함으로 일반적으로 Pulse Width 측정에 비해 좀 더 정확합니다.

펄스 캡쳐(Pulse Width, Frequency, Duty Cycle) 값은 펄스 입력이 최소 20Hz에서 최대 20kHz 사이의 주파수를 가져야 합니다. 그리고 펄스 폭은 최소 10us 이상이 되어야 합니다. 그렇지 않으면 캡쳐가 정상적으로 수행되지 않거나 캡쳐 값을 무효한 값으로 판단하고 원시 값(Raw Value)은 0이 출력 됩니다.

각각의 캡쳐 타입에 대한 값의 유효한 범위를 다시 한번 정리해 보면 다음과 같습니다:

- Pulse Width : 10μs ~ 50000μs
- Frequency : 20Hz ~ 20kHz
- Duty Cycle : 0‰ ~ 1000‰

6.4 아날로그/펄스 입력의 정규화

아날로그/펄스 입력의 원시 값(Raw Value)은 캡쳐 타입에 따라 다양한 스케일의 값을 가집니다. 모터제어기에서 이 값을 범용으로 사용하기 위해, -1과 1사이의 값을 가지는 정규화 과정을 거치게 됩니다. 그리고 입력 값의 Min/Max Safety, Center Safety를 검사합니다.

원시 값은 아래 그림 6-5와 같이 순차적으로 처리되어 변환된 값(Converted Value, Normalized Value)을 출력합니다.

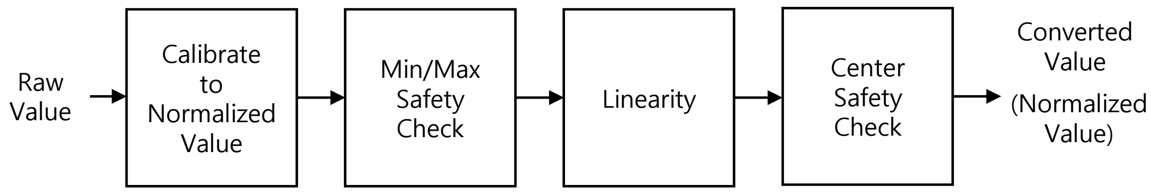


그림 6-5 아날로그/펄스 입력 값의 단위 변환

아날로그/펄스 펄스 입력의 정규화와 관련된 구성 파라미터는 다음과 같습니다:

- ai_flags - AI Flags
- ai_linearity - AI Linearity
- ai_input_min - AI Input Min
- ai_input_max - AI Input Max
- ai_input_center - AI Input Center
- ai_input_deadband - AI Input Deadband

- pi_flags - PI Flags
- pi_linearity - PI Linearity
- pi_input_min - PI Input Min
- pi_input_max - PI Input Max
- pi_input_center - PI Input Center
- pi_input_deadband - PI Input Deadband

AI_Flags와 PI_ Flags 파라미터는 아래와 같은 목록을 포함하고 있습니다.

- 0x01 - Enable
- 0x02 - Invert Input Signal
- 0x04 - Min/Max Safety
- 0x08 - Center Safety

6.4.1 정규화된 값으로의 캘리브레이션

센서로 측정하고자 하는 물리량의 값을 정확하게 측정하기 위해 센서를 연결한 후 반드시 캘리브레이션 해야 합니다.

만일 조이스틱의 움직임이 0V(Raw Value: 0)와 5V(Raw Value: 4095)에 도달하지 못하는 경우 또는 조이스틱의 중심점이 정확히 2.5V(Raw Value: 2047)를 출력하지 않는 경우, 아날로그 입력은 이를 보상하기 위해 캘리브레이션할 수 있습니다.

다음 그림 6-6과 같이, 원시 값은 사용자가 설정한 최소('**AI Input Min**', ' **PI Input Min**')와 최대('**AI Input Max**', ' **PI Input Max**'), 센터('**AI Input Center**', ' **PI Input Center**') 값을 사용하여 -1에서 1까지의 실수로 조정됩니다.

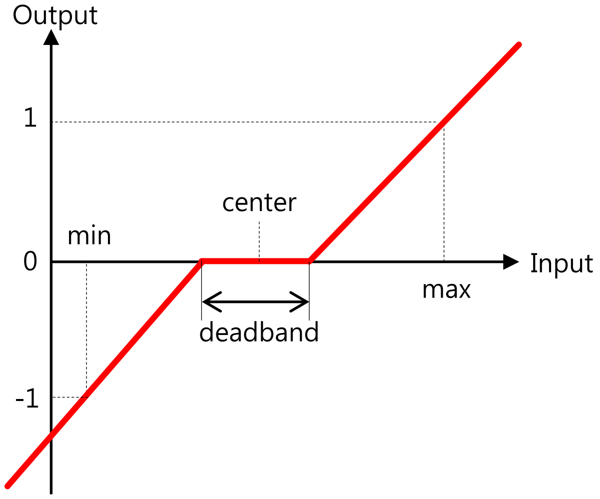


그림 6-6 아날로그 입력 값의 정규화

데드밴드('**AI Input Deadband', ' PI Input Deadband**')는 중심 부근의 움직임에 대해 0의 값을 가지도록 하는 영역을 지정합니다. 이는 조이스틱이나 RC 조종기의 중심 부근에서 약간의 움직임을 허용하게 합니다.

보통 스틱을 가만히 두어도 완전한 중립 위치로 복귀하지 않게 되는데, 사용자가 눈으로 보기에 스틱이 중간에 있는 것처럼 보여도 제어기에서는 마치 송신기의 스틱을 미세하게 움직이고 있는 상태로 인식하게 됩니다. 데드밴드 설정으로 이러한 문제를 해결합니다.

6.4.2 Min/Max 안전 검사

사용자가 ' **AI Flags**' 또는, '**PI Flags**' 파라미터를 통해, Min/Max Safety 검사 기능을 켜면, 정규화된 값이 -1과 1을 벗어날 때 입력 값을 비정상으로 판단합니다. 그리고 이후 처리 블록에는 0이 입력되는 것처럼 처리합니다. 벗어나는 값에는 5%의 허용 범위를 둡니다. 즉, 1에서 1.05까지는 1로 간주합니다. 그리고 -1에서 -1.05까지는 1로 간주합니다. 하지만 -1.05와 1.05 범위를 벗어나면 0으로 간주합니다(그림 6-7 참조).

Min/Max 안전 검사는 포텐셔미터의 결선이 끊어진 경우를 감지하는데 유용하게 사용할 수 있습니다. 이를 위해 포텐셔미터 양단에 직렬로 두 개의 저항을 삽입하여, 출력 범위를 0V ~ 5V보다 조금 좁히게 됩니다. 만약 체크 기능이 켜진 상태라면, 포텐셔미터에 연결된 하나 이상의 결선이 절단되는 경우 전압은 실제로는 0V 혹은 5V가 되고 입력 값은 0이 됩니다.

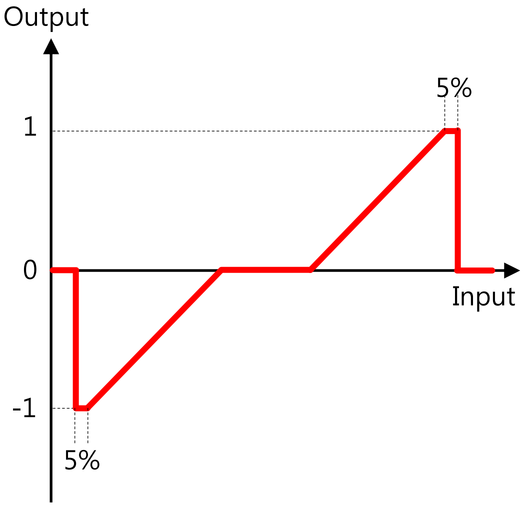


그림 6-7 Min/Max 범위 제한

※ Min/Max Safety 기능은 아날로그 입력 또는 펄스 입력이 모터의 속도 및 위치 명령이 매핑된 경우에만 사용 가능합니다.

6.4.3 지수/로그 변환

사용자는 **'AI Linearity'**와 **'PI Linearity'** 파라미터의 설정을 통해, 아날로그/펄스 입력 값에 선택적으로 지수 혹은 로그 변환을 수행할 수 있습니다. 지수 보정은 조이스틱의 중앙에서는 적게 변하도록 하고 양 끝에서는 크게 변하도록 합니다. 로그 보정은 조이스틱의 중앙에서는 크게 변하도록 하고 양 끝에서는 적게 변하도록 합니다. 'Linear'를 선택하면 입력 값이 변하지 않고 바로 출력 값이 됩니다. 또한, 약, 중, 강 등 3가지의 지수/로그 선택이 가능합니다.

아래 그림 6-8의 그래프는 지수(실선), 로그(점선) 변환에 의한 입력과 출력의 변화를 보여줍니다.

지수/로그 변환에 선택 가능한 옵션은 다음과 같습니다:

- linear - Output = Input
- exp weak - Output = exp(Input, 1.4)
- exp medium - Output = exp(Input, 2)
- exp strong - Output = exp(Input, 3)
- log weak - Output = exp(Input, 1/1.4)
- log medium - Output = exp(Input, 1/2)
- log strong - Output = exp(Input, 1/3)

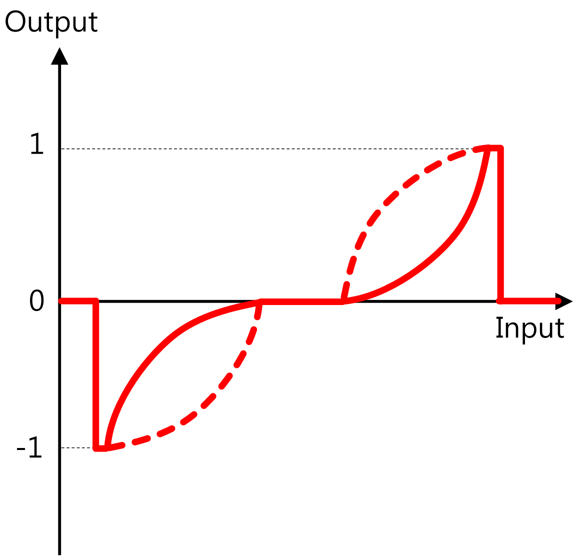


그림 6-8 Exponent 함수의 적용

6.4.4 센터 안전 검사

사용자가 **'AI Flags'** 또는, **'PI Flags'** 파라미터를 통해, Senter Safety 검사 기능을 켜면, 아날로그 입력이나 펄스 입력이 0이 될 때까지 명령이 실행되지 않도록 합니다.

이는 조이스틱이나 RC 조종기로 로봇을 조종하는 경우 유용하게 사용됩니다. 스틱이 중앙에 있지

않은 상태에서 제어기가 켜지면 로봇이 갑자기 움직여 위험한 상황을 초래할 수 있습니다. 이러한 상황을 방지하기 위해, 센터 안전 검사 기능은 조이스틱의 스틱이 중심에 올 때까지 모터가 구동하지 않도록 합니다. 중심 위치는 설정된 중심 주변의 데드밴드 영역 내가 됩니다.

모터에 전원이 공급되거나 조이스틱이나 RC 조종기가 연결될 때 속도 명령의 값이 100ms 동안 0에 머무른 후 입력 값을 받아들이기 시작합니다. 그리고 모터의 전원이 차단될 때까지 센터 안전 검사를 다시 수행하지는 않습니다.

※ 센터 안전 검사 기능은 아날로그 입력 또는 펄스 입력이 모터의 속도명령이나 전류 명령, 전압 명령으로 매핑된 경우에만 활성화됩니다.

6.5 입력 값의 매핑, 중재

아날로그/펄스 입력으로부터의 정규화 된 입력 값과 디지털 입력으로부터의 입력 값은 매핑, 중재(Mapping, Arbitration) 과정을 거쳐 특정 모터에 구동 명령을 내리거나 피드백 값을 전달하고 액션을 트리거 하기 위해 사용됩니다.

아날로그, 펄스, 디지털 입력 채널의 매핑과 관련된 구성 파라미터는 다음과 같습니다:

- AI Mapping, PI Mapping

그리고 다음 아날로그/펄스 입력 값

- AI Value
- PI Value

은 매핑, 중재 과정을 거쳐 다음의 변환된 값을 만들어냅니다:

- AI Converted Value
- PI Converted Value

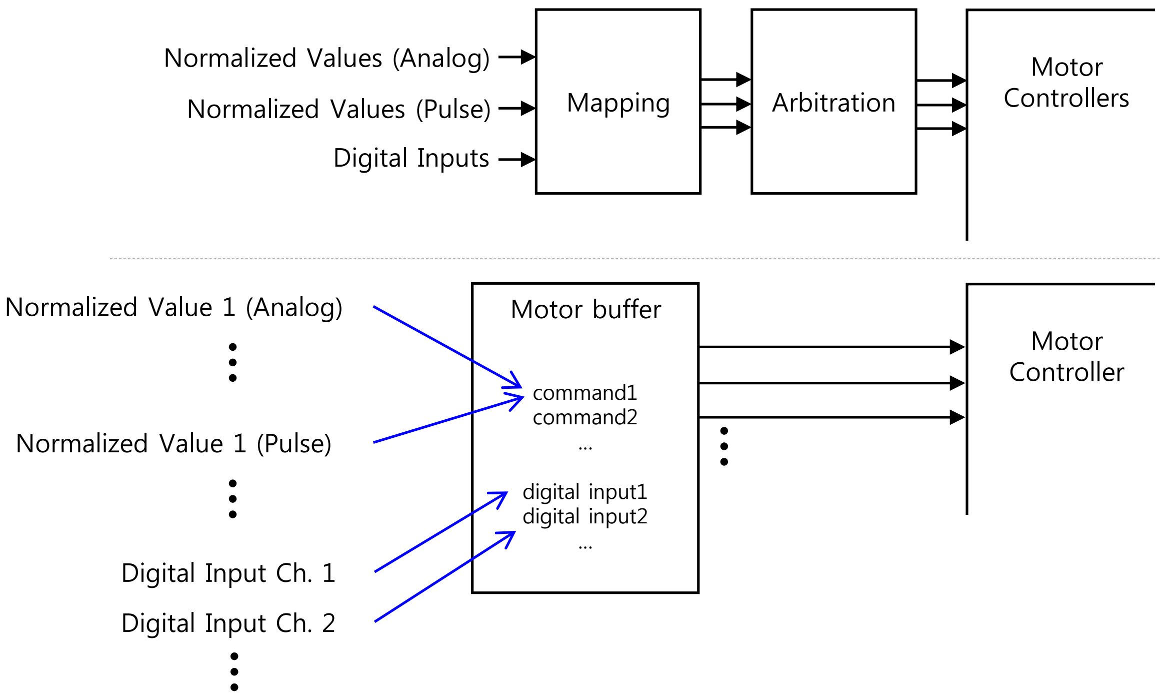


그림 6-9 아날로그/펄스/디지털 입력 신호의 흐름

매핑(Mapping)은 입력 채널을 모터의 특정 기능으로 연결합니다. 상기 그림에서 파란색 실선으로 매핑이 표시되어 있습니다. 검은색 실선은 데이터의 흐름을 표시합니다. 만일 매핑 과정에서 둘 이상의 입력이 하나의 모터 기능으로 연결되어 있다면, 두 충돌하는 입력 데이터에 대해 중재(Arbitration) 과정이 적용됩니다.

6.5.1 매핑

제어기의 아날로그나 펄스 입력 채널의 입력 값(**'AI Value'**, **'PI Value'**)은 사용자가 **'AI Mapping'**, **'PI Mapping'** 파라미터에 설정한 매핑 경로를 따라 모터제어기의 명령이나 피드백 값으로 전달됩니다. 그리고 디지털 입력 채널의 입력 값 또한 사용자가 **'DI Mapping'** 파라미터에 설정한 매핑 경로를 따라 모터제어기의 액션을 트리거 합니다.

매핑의 설정은 아날로그/펄스/디지털 입력 채널을 기준으로 모터제어기의 입력 기능(명령, 피드백, 디지털 입력) 중 하나를 선택하게 됩니다.

아날로그 입력이나 펄스 입력은 모터의 명령이나 피드백 버퍼로 연결되거나 디지털 입력 버퍼로 연결 가능합니다. 하지만 디지털 입력은 모터의 명령이나 피드백 버퍼로는 연결될 수 없고 디지털 입력 버퍼로만 연결 가능합니다.

Motor Control UI 유틸리티를 사용하여 I/O 입출력 채널의 매핑을 쉽게 할 수 있습니다.

6.5.2 중재

둘 이상의 아날로그, 펄스, 디지털 입력이 하나의 모터제어기 입력 버퍼로 매핑된 경우, 입력 신호의 충돌을 유발합니다. 이때, 중재(Arbitration) 과정에서 미리 정해진 규칙에 따라 여러 입력 소스의 값을 적절하게 섞게 됩니다.

제어기는 그림 6-9와 같이 모터와 기능에 따라 중재를 위한 메모리 버퍼를 가지고 있습니다. 이 버퍼에 하나 이상의 값이 입력될 때 다음 규칙에 따라 중재됩니다:

- 명령(command) 경우, 입력되는 값의 절대값으로 크기를 비교하여 제일 큰 값을 선택함
- 디지털 입력(digital input) 버퍼의 경우, OR 조건으로 병합

만일 정규화 된 아날로그 입력이나 펄스 입력이 디지털 입력 버퍼로 매핑될 경우, 0보다 크면 1이 되고, 0보다 같거나 작으면 0이 됩니다.

만일 아날로그나 펄스 입력 값으로 0.5, -0.2, 0의 세 값이 하나의 버퍼로 입력되었다면, 이 버퍼의 값은 절대값이 가장 큰 0.5 값을 가지게 됩니다. 이러한 상황은 조이스틱과 RC 조종기를 모터에 연결하여 동시에 조종하는 경우 발생할 수 있습니다. 이때, 조이스틱은 조종에 사용되지 않아 입력이 0이고 RC 조종기로 조종하고 있는 상황이라 입력이 0이 아니라면 모터는 RC 조종기에 의해 움직이게 됩니다.

7 모터제어기 오브젝트

이 장에서는 제어기의 제품 정보와 버전, 통신, 스크립트에 관련된 오브젝트들(상수와 명령, 상태, 구성 파라미터)에 대해서 설명합니다.

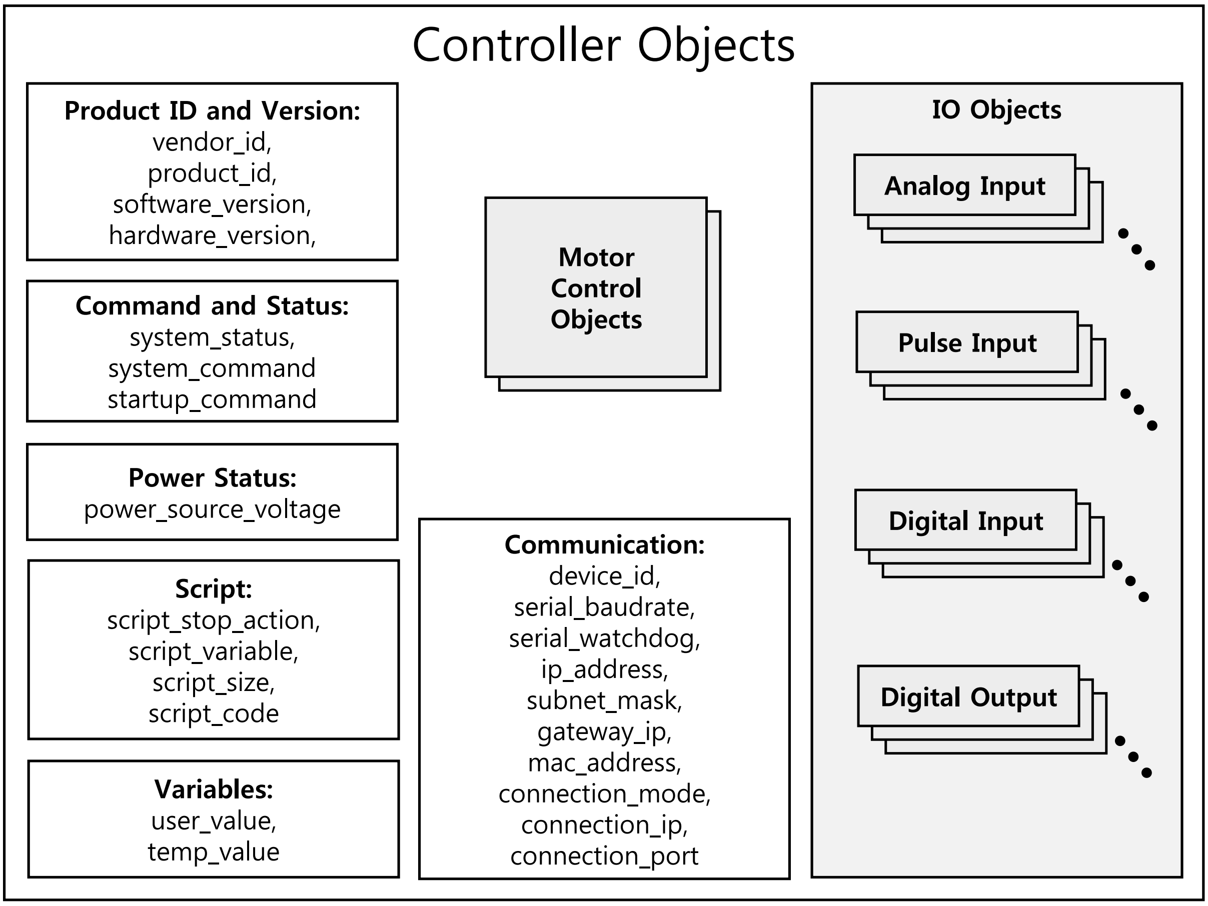


그림 7-1 Controller Objects

제어기가 가지고 있는 오브젝트들을 모두 표시하면 상기 그림과 같습니다. 이 중, 모터제어 오브젝트(Motor Control Object)와 디지털/아날로그 입출력 오브젝트(I/O Objects: Analog Input, Pulse Input, Digital Input, Digital Output)에 대해서는 8장과 9장에서 설명합니다. 이번 장에서는 이 둘을 제외한 오브젝트들에 대해 설명합니다.

7.1 제품 ID 및 버전 정보

제품 ID 및 소프트웨어/하드웨어 버전은 상수 오브젝트로 읽기만 가능합니다. 이 상수들은 제품 생산 시 결정되며 사용자가 바꿀 수 없습니다.

표 7-1 제품 ID 및 버전 정보 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
vendor_id, vid	1/0	I32	제품 공급자 ID
product_id, pid	2/0	I32	제품 ID
software_version, swv	3/0	F32	모터제어기 펌웨어 버전
hardware_version, hwv	4/0	F32	모터제어기 하드웨어 버전

상기 표에서 Long Name과 Short Name은 텍스트 모드에서 오브젝트를 엑세스 할 수 있도록 부여된 이름입니다. 그리고 Index와 Sub-index는 오브젝트를 엑세스 하기 위한 주소입니다.

Type 열은 오브젝트의 형식을 나타냅니다:

- I8 - 부호를 가지는 8bit 정수형 수
- I16 - 부호를 가지는 16bit 정수형 수
- I32 - 부호를 가지는 32bit 정수형 수
- F32 - 부호를 가지는 32bit 실수형 수

7.1.1 vendor_id - Vendor ID

'vendor_id'는 제품 공급자 ID를 가집니다. 이 값은 0으로 고정되어 있습니다.

7.1.2 product_id - Product ID

'product_id'는 제품의 ID를 가집니다.

- 2001 - 1축 STEP 모터 컨트롤러

Motor Control UI 유틸리티가 제어기의 종류를 구분하기 위해 사용합니다. 새로운 제품이 출시됨에 따라 제품 ID의 종류는 늘어날 수 있습니다.

7.1.3 software_version - Software Version

'software_version'은 제어기의 펌웨어 버전 넘버를 가집니다. 버전은 다음과 같이 소수점을 기준으로 상위 1~2자리와 하위 2자리로 구분됩니다.

소프트웨어 버전: XX.YY

XX는 메이저 버전이고 YY는 마이너 버전입니다. 메이저 버전이 변경된 경우에는 제어기와 Motor Control UI 유틸리티 및 기타 응용 소프트웨어들간에 호환되지 않습니다. 제어기와 관련 소프트웨어들이 서로 호환되는 범위 내에서 마이너 버전은 변경될 수 있습니다.

소프트웨어 버전은 제어기의 생산 일자에 따라 달라질 수 있으며 제어기에 최신 펌웨어를 업데이트 하는 경우에도 달라질 수 있습니다.

7.1.4 hardware_version - Hardware Version

'hardware_version'은 제어기의 하드웨어 버전 넘버를 가집니다. 버전은 다음과 같이 소수점을 기준으로 상위 1~2자리와 하위 2자리로 구분됩니다.

하드웨어 버전: XX.YY

XX는 메이저 버전이고 YY는 마이너 버전입니다. 메이저 버전이 변경된 경우에는 제어기와 Motor Control UI 유틸리티 및 기타 응용 소프트웨어들간에 호환되지 않습니다.

하드웨어 버전은 제어기의 생산 일자에 따라 달라질 수 있습니다.

7.2 모터제어기 명령 및 상태

제어기에 명령을 내리고 상태를 모니터링 합니다.

표 7-2 모터제어기 명령 및 상태 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
system_status, sst	5/0	I32	제어기의 운용 상태
system_command, sco	7/0	I16	제어기에 내려지는 명령
startup_command, suc	16/0	I8	모터제어기 시작 시 제어기에 내리는 명령 (Motor Disable, Motor Enable, Run Script, Motor Enable and Run Script)

7.2.1 system_status - System Status

'system_status'는 제어기의 운용 상태를 가집니다. 각 비트에 따라 운용 상태는 다음과 같습니다:

- 0x10000 - Script Running

0x10000 - Script Running:
스크립트가 실행 중임을 표시합니다. 모터제어기의 **status** 오브젝트의 'Script Running' 플래그와 동일합니다.

7.2.2 system_command - System Command

'system_command'에 명령 코드를 쓰는 것으로 제어기의 특정 기능을 실행할 수 있습니다. 다음은 명령 코드 목록입니다:

- 1 - Save properties in Flash Memory
- 2 - Load properties from Flash Memory
- 3 - Save Script's Byte codes in Flash Memory
- 8 - Script Run
- 9 - Script Stop
- 98 - Reset to Factory Default
- 99 - Reset Controller

1 - Save properties in Flash Memory:
제어기의 각종 오브젝트 값을 변경하면, 변경된 값들은 RAM에 기록되어 제어기가 켜져 있는 동안만 유지됩니다. 그리고 제어기가 재시작 되면 이 값들은 소실되고 플래시 메모리에 저장된 값을 RAM으로 읽어 들이게 됩니다. 그렇기 때문에 오브젝트 값을 변경하고 계속 유지가 되기를 바란다면, '1 - Save properties in Flash Memory' 명령 코드를 사용하여 변경한 오브젝트들의 값을 플래시 메모리에 저장해야 합니다.

2 - Load properties from Flash Memory:
이 명령 코드는 제어기의 오브젝트들에 현재 설정된 값을 무시하고 플래시 메모리에 저장된 값을 읽어와 덮어씁니다.

3 - Save Script's Bytecodes in Flash Memory:
스크립트의 바이트코드를 제어기에 다운로드 하면 일시적으로 RAM영역에 저장됩니다. 이 명령 코드로 RAM 영역에 저장된 바이트코드를 플래시 메모리 영역으로 옮겨야 합니다. 이는 PC에서 RAM 영역의 프로그램을 하드디스크에 저장하는 것과 같습니다.

8 - Script Run, 9 - Script Stop:
스크립트와 관련된 또 다른 명령으로는 '8 - Script Run'과 '9 - Script Stop' 명령 코드가 있습니다. 이 두 명령 코드는 각각 스크립트를 실행하고 중단하는 명령입니다.

98 - Reset to Factory Default:

이 명령 코드는 제어기에 현재 설정된 모든 오브젝트들의 값을 무시하고 제품 초기 설정 값 (Factory Default Value)으로 복구합니다. 이 명령을 실행하면, RAM 및 플래시 메모리에 있는 모든 값들이 제품 초기 설정 값으로 복구됩니다.

99 - Reset Controller:
이 명령 코드는 제어기를 소프트웨어적으로 리셋 시킵니다. 제어기의 전원을 껐다 켜는 것과 같습니다.

7.2.3 startup_command - Startup Command

사용자가 **'startup_command'** 파라미터를 변경하여 제어기가 시작될 때 한번 실행 되는 명령을 설정합니다. 이 값으로 다음 중 하나를 선택합니다:

- 0 - Motor Disable (기본값)
- 1 - Motor Enable
- 2 - Run Script
- 3 - Motor Enable and Run Script

7.3 배터리 상태

제어기에 공급되는 전원의 소스는 주로 배터리 혹은 파워서플라이가 되는데, 매뉴얼에서는 이를 따로 구분하지 않고 배터리로 통칭하여 설명합니다.

표 7-3 배터리 상태 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
power_source_voltage, psv	8/0	F32	전원 소스로부터 제어기에 공급되는 전압 (단위: V)

7.3.1 power_source_voltage - Power Source Voltage

'power_source_voltage'는 전원(파워서플라이, 배터리)으로부터 제어기에 공급되는 전압을 나타냅니다.

만약 **'power_source_voltage'**가 모터제어기의 **'undervoltage_limit'**과 **'overvoltage_limit'** 범위를 벗어나는 경우, 모터의 폴트(Fault)를 발생합니다.

7.4 시리얼 통신 설정

이 절에서는 PC 및 마이크로컨트롤러와 제어기를 연결하는 RS-232, RS-485 포트의 통신 설정에 대해 다룹니다.

표 7-4 시리얼 통신 설정 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
device_id, id	11/0	I32	장치 ID (범위: 1 ~ 255, 기본값: 1)
serial_baudrate, sb	13/1	I32	RS-232 통신 속도 (기본값: 115200)
	13/2	I32	RS-485 통신 속도 (기본값: 115200)
serial_watchdog, sw	14/0	I32	시리얼 통신의 위치/속도 구동 명령에 대한 와치독 타이머 타임아웃 값 (단위: ms, 범위: 0~10000, 기본값: 0)

※ device_id, serial_baudrate 값을 변경한 경우에는 제어기를 새로 시작(모터제어기 전원을 끄고 켜거나 제어기를 리셋)하여야 변경된 값이 적용됩니다.

7.4.1 device_id - Device ID

CAN 또는 RS-422, RS-485 버스에는 여러 제어기가 동시에 연결되어 통신 선로를 공유합니다. 이 때, Device ID는 각각의 제어기를 구분하기 위한 ID 값입니다. 그러므로 하나의 통신 선로에 연결된 제어기의 Device ID는 모두 달라야 합니다.

제품 초기 설정 값(Factory Default Value)으로 제어기의 **'device_id'** 파라미터는 1로 설정되어 있습니다. 이 값은 0과 255사이의 값으로 변경될 수 있습니다. 보통 마스터가 되는 PC나 마이크로 컨트롤러의 Device ID로 0을 설정하기 때문에, 제어기에서 **'device_id'**로 0을 사용하는 것은 피해야 합니다.

만일 CAN 버스에 한 개 이상의 장치(모터제어기 외 다른 장치를 포함)를 연결한다면, 먼저 제어기의 RS-232 포트로 Motor Control UI 유틸리티를 연결합니다. 그리고 Configuration 탭에서 Connection 그룹의 Device ID를 서로 중복되지 않도록 변경해야 합니다.

PC나 마이크로컨트롤러에 RS-232나 RS-485 포트를 통해 하나의 제어기만 연결하는 경우, 이 값을 변경하지 않고 기본 설정 값을 사용하여도 상관없습니다.

7.4.2 serial_baudrate - Serial Baudrate

PC나 마이크로컨트롤러의 RS-232 또는 RS-485 포트에 제어기를 연결하는 경우, '**serial_baudrate**'로 시리얼 통신 속도를 설정해야 합니다.

'**serial_baudrate**' 파라미터의 값으로 다음 중 하나를 설정하여 시리얼 통신 속도를 변경합니다:

- 9600 - 9600bps
- 19200 - 19200bps
- 38400 - 38400bps
- 57600 - 57600bps
- 115200 - 115200bps (기본값)
- 230400 - 230400bps
- 460800 - 460800bps
- 921600 - 921600bps

시리얼 통신속도는 9600에서 921600까지 자유롭게 설정 가능합니다. 하지만 상기 목록 이외의 속도는 다른 장치에서 대부분 사용되지 않습니다. 되도록 상기 목록 중 하나의 값을 사용하도록 합니다.

Sub-index가 1인 경우 RS-232의 통신 속도를 설정하게 되고, 2인 경우 RS-485의 통신 속도를 설정하게 됩니다. 예로, 제어기에 연결된 텍스트 터미널로 "`serial_baudrate1 = 9600`"와 같이 입력하면 RS-232의 시리얼 통신 속도를 9600bps로 설정하게 됩니다. 만약 "`serial_baudrate2 = 115200`"와 같이 입력하면 RS-485의 시리얼 통신 속도를 115200bps로 설정하게 됩니다.

RS-232 또는 RS-485 포트로 Motor Control UI 유틸리티가 연결된 경우, 제어기에 설정된 시리얼 통신 속도를 모르더라도 Motor Control UI 유틸리티에서 자동으로 검색이 가능합니다. 이를 위해 Motor Control UI 유틸리티의 Connection 대화상자에서 Serial Baudrate 항목을 '**Auto Detect**'로 설정해야 합니다.

7.4.3 serial_watchdog - Serial Watchdog

시리얼 와치독 타이머는 제어기에 연결된 PC가 오작동 하거나 통신 선로가 끊어져 명령 전송 중단된 상황을 대비하기 위함입니다. 이러한 상황에서 제어기는 마지막 명령을 유지하여 모터가 계속 구동중인 상황이 발생할 수 있습니다.

사용자가 '**serial_watchdog**' 파라미터에 설정한 타임아웃 기간 내에 RS-232, RS-485 포트를 통해 명령('**go_position**', '**go_velocity**' 중 하나)이 도착하면, 시리얼 통신은 정상으로 간주됩니다. 와치독 타이머가 타임아웃 되면 시리얼 통신은 비정상으로 간주되고 모터에 정지 명령을 내립니다.

'serial_watchdog' 값으로 0이 설정되면 와치독 타이머는 타임아웃을 검사하지 않습니다. 그리고 통신 포트는 항상 정상인 상태로 간주됩니다. 와치독 타이머를 활성화 하려면 타임아웃 값을 1보다 큰 값으로 설정해야 합니다. 값의 설정 범위는 0과 10,000ms 사이 입니다.

7.5 이더넷 통신 설정

이 절에서는 PC 및 마이크로컨트롤러와 제어기를 연결하는 이더넷 통신 설정에 대해 다룹니다.

표 7-5 이더넷 통신 설정 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
ip_address, ip	31/0	I32	제어기의 IP 주소 설정
subnet_mask, sm	32/0	I32	제어기의 서브넷 마스크 설정
gateway_ip, gi	33/0	I32	제어기의 게이트웨이의 IP 주소 설정
mac_address, mac	34/ 1~6	I8	제어기의 MAC 주소 설정
connection_mode, cm	36/0	I8	Ethernet 통신의 연결방법 설정 (TCP Server, TCP Client, UDF)
connection_ip, cip	37/0	I32	제어기의 연결방법 설정이 TCP Client로 설정된 경우, 제어기가 연결을 시도하는 원격 PC의 IP 주소 설정
connection_port, cpt	38/0	I16	TCP, UDP 연결에 사용되는 포트 번호 설정

7.5.1 ip_address - IP Address

제어기에 사용하는 IP 주소는 서브 네트워크에서 유일한 주소를 할당해야 합니다. 이를 위해 네트워크 관리자에게 문의하여 유효한 IP 주소를 설정해야 합니다. 그렇지 않으면 네트워크에 올바르게 연결될 수 없습니다.

고정 IP 모드에서 사용자는 IP 주소, 서브넷 마스크, 게이트웨이 IP 주소를 수동으로 올바르게 설정해야 합니다. IP 주소는 네트워크 상에 연결된 장비를 구분하고 서로 대화하는데 사용합니다. 그러므로 장비에 사용된 IP 주소는 네트워크 환경에서 고유하며 유효한 값을 가져야 합니다.

제어기에 설정된 IP 주소는 다음과 같으며 Motor Control UI 유틸리티에서 변경 가능합니다:

- IP Address: 192.168.1.100

7.5.2 subnet_mask - Subnet Mask

서브넷은 같은 지리적 위치, 한 건물 또는 동일한 지역 네트워크를 말합니다. 한 장비가 다른 장비에 패킷을 전송하고자 할 경우, 먼저 패킷을 수신할 장치가 로컬 네트워크 영역에 있는지 서브넷 마스크를 통해 확인합니다. 그리고 주소가 송신 장치와 동일한 네트워크 영역에 있는 경우는 직접 연결하여 전송합니다. 그렇지 않은 경우는 게이트웨이로 보내집니다.

제어기에 설정된 서브넷 마스크는 다음과 같으며 UI 유틸리티에서 변경 가능합니다:

- Subnet Mask: 255.255.255.0

7.5.3 gateway_ip - Gateway IP

게이트웨이는 다른 네트워크로 들어가는 입구 역할을 하는 네트워크 노드로, 일반적으로 AP, 라우터 등이 여기에 해당됩니다.

제어기에 설정된 게이트웨이 IP 주소는 다음과 같으며 UI 유틸리티에서 변경 가능합니다:

- Gatway IP Address: 192.168.1.1

7.5.4 mac_address - MAC Address

이더넷 장비들은 각각의 고유한 하드웨어 주소인 MAC(Media Access Control) 주소를 가지고 있습니다. MAC 주소는 6바이트의 하드웨어 주소로 이더넷 제품에 하나씩 부여되며, 전세계적으로 유일하게 구분되는 주소를 사용합니다. MAC 주소의 상위 3바이트는 IEEE에서 할당하고 하위 3바이트는 제품 생산자에서 할당하게 됩니다.

제어기에 설정된 MAC 주소는 다음과 같으며 Motor Control UI 유틸리티에서 변경 가능합니다:

- MAC address (Hexa): 00-08-DC-2E-E8-72

7.5.5 connection_mode - Connection Mode

'connection_mode' 파라미터는 제어기와 PC간 연결 방법을 결정하는데 사용합니다. 다음은 선택할 수 있는 연결 방법입니다.

- 0 - TCP Server
- 1 - TCP Client

- 2 - UDP

0 - TCP Server:
TCP는 연결과정이 필요한 프로토콜로 연결이 이루어진 후에 패킷 전송이 가능합니다. 두 장비가 연결하기 위해서 하나의 장비는 연결을 기다리고 다른 하나는 연결을 시도해야 합니다. 이러한 접속 방향에 따라 서버와 클라이언트가 결정됩니다. TCP 서버는 접속을 기다리고, TCP 클라이언트는 접속을 시도합니다.

TCP 연결은 PC와 모터제어기 간 일대일로 이루어 집니다. 따라서 만일 제어기가 이미 하나의 PC와 연결되어 있다면, 다른 PC는 연결할 수 없습니다.

TCP 서버는 제어기가 서버로 동작합니다. 호스트 PC로부터 TCP 연결이 요청되면 서버는 연결을 승낙합니다. 연결된 후 시리얼 포트로 들어오는 데이터는 TCP 패킷으로 포장되어 호스트 PC로 전송되며, 이와 반대의 과정도 진행됩니다.

1 - TCP Client:
TCP 클라이언트는 제어기가 클라이언트로 동작합니다. 호스트 PC가 서버가 되어 연결을 기다리고 제어기가 서버로 TCP 연결을 시도하여 서버가 연결을 승낙하면 연결이 되고, 이후에는 TCP 서버 모드와 동일한 과정으로 데이터를 주고 받습니다.

2 - UDP:
UDP는 연결 과정 없이 데이터를 블록 단위로 주고 받습니다. UDP에서는 연결이 이루어지지 않은 상태에서도 패킷 전송이 가능합니다. UDP 연결은 호스트 PC와 제어기가 동일한 서브넷 안에 있을 때 사용합니다. UDP 연결에서는 패킷의 손실이 발생할 수 있습니다.

UDP 모드에서는 연결에 상관없이 호스트 PC와 데이터를 주고 받습니다. 그래서 전송하는 UDP 패킷이 호스트 PC에서 수신되었는지 확인할 수 없습니다. 하지만 데이터를 여러 호스트 PC에 브로드캐스팅 할 때 사용할 수 있습니다.

7.5.6 connection_ip - Connection IP

'connection_ip' 파라미터는 제어기의 연결방법 설정이 TCP 클라이언트로 설정된 경우, 제어기가 연결을 시도하는 원격 PC의 IP 주소를 설정하는데 사용합니다.

제어기에 설정된 PC Connection IP 주소는 다음과 같으며 Motor Control UI 유틸리티에서 변경 가능합니다:

- IP Address: 192.168.1.100

7.5.7 connection_port - Connection Port

'connection_port; 파라미터는 제어기가 접속해야 할 PC의 포트 번호를 설정하는데 사용합니다. TCP 서버 모드에서 제어기가 접속을 기다리는 TCP 서버 포트 번호를 설정합니다.

제어기에 설정된 포트 번호는 다음과 같으며 Motor Control UI 유틸리티에서 변경 가능합니다:

- Port Number: 5000

표 7-6 통신 모드에 따른 IP 주소 및 포트 번호 설정

	IP Address	Port Number
TCP 서버	(사용 안함)	TCP 접속을 기다릴 포트 번호
TCP 클라이언트	원격 호스트 PC의 IP 주소	원격 호스트 PC의 포트 번호
UDP	원격 호스트 PC의 IP 주소 (0.0.0.0 으로 설정된 경우 broadcasting)	원격 호스트 PC의 포트 번호, UDP 패킷을 수신할 포트 번호

※주의※ TCP 서버 모드나 TCP 클라이언트 모드에서 호스트 PC로 연결되어있지 않은 경우 시리얼 포트로부터 수신된 데이터는 폐기됩니다.

※주의※ 제어기가 TCP Client 모드로 설정되어 호스트 PC로 연결을 시도하는 경우, 호스트 PC에 방화벽(firewall)이 설정되어 있다면 호스트 PC로의 접속 시도가 거절될 수 있습니다.

7.6 사용자/임시 변수

제어기는 사용자가 언제든지 읽고 쓸 수 있는 사용자 변수와 임시 변수를 가지고 있습니다.

표 7-7 사용자 및 임시 변수 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
user_variable, uv	56/1~32	I32	사용자가 읽고 쓰는 변수들, 플래시 메모리에 저장 가능
temp_variable, tv	57/1~32	I32	사용자가 읽고 쓰는 변수들, RAM에만 저장 됨

7.6.1 user_variable - User Variable

'**user_variable**' 파라미터는 제어기의 운용과 관련된 사용자 파라미터 값들을 저장하는데 사용합니다. 32개의 사용자 파라미터들을 저장할 수 있으며, 각각의 파라미터는 Sub-index에 의해 구분됩니다.

'**system_command**'에서 '**1 - Save properties in Flash Memory**' 명령으로 모터제어기 구성 파라미터들을 저장한 경우, 이 값들은 제어기의 전원이 꺼지더라도 유지됩니다.

7.6.2 temp_variable - Temp Variable

'**temp_variable**' 파라미터는 PC와 제어기에서 실행되는 스크립트간에 명령/상태를 주고받기 위해 사용합니다. 32개의 변수들을 저장할 수 있으며, 각각의 변수들은 Sub-index에 의해 구분됩니다. 이 값들은 모터제어기 전원이 꺼지면 소실됩니다.

7.7 스크립트 다운로드와 실행

스크립트 언어로 작성된 프로그램은 빌드 과정을 거쳐 바이트코드로 컴파일 되어 제어기로 다운로드 됩니다. 다운로드 된 바이트코드는 가상머신에 의해 해석되고 실행됩니다.

표 7-8 스크립트 다운로드와 실행 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
script_stop_action, ssa	17/0	I8	실행중인 스크립트를 중단할 때, 혹은 스크립트가 종료 이후 행동 선택
script_size, ss	250/0	I32	바이트코드를 다운로드하기 전에 크기 전송 (단위: byte)
script_code, sc	251/0	I32	바이트코드를 4-byte씩 묶어서 차례로 전송
script_variable, sv	252/ 0~255	F32	스크립트가 실행되는 동안 사용되는 변수

스크립트가 실행되기 위해서는 제어기에 미리 다운로드 되어 있어야 합니다. 스크립트는 하나만 다운로드 가능하며, 새로운 스크립트를 다운로드 하면 이전에 저장된 내용이 지워집니다.

7.7.1 script_stop_action - Script Stop Action

사용자가 '**script_stop_action**' 파라미터를 변경하여 제어기가 시작될 때 스크립트 실행 중지 후 모터 동작 여부를 설정합니다. 이 값으로 다음 중 하나를 선택합니다:

- 0 - (none)
- 1 - All Motor Stop (기본값)
- 2 - All Motor Quick Stop
- 3 - All Motor Disable

7.7.2 script_size - Script Size

스크립트의 소스코드는 제어기에 직접 다운로드 될 수 없고, 다운로드 되기 전에 반드시 바이트 코드로 컴파일 되어야 합니다. 스크립트의 바이트코드를 다운로드 하는 것은 '**script_size**'와 '**script_code**'에 의해 구현됩니다.

제어기에 스크립트를 전송하기에 앞서 '**script_size**'에 바이트코드의 크기를 설정해야 합니다. 크기는 byte 단위로 설정합니다.

스크립트 전송이 끝나면 '**script_size**'에 -1을 설정하여 전송이 끝났음을 알립니다.

7.7.3 script_code - Script Bytecode

다운로드 할 스크립트의 바이트코드 크기를 '**script_size**'에 설정하였다면, 바이트코드를 4-byte씩 묶어서 '**script_code**'에 차례로 전송합니다. 마지막 묶음이 4-byte가 되지 않을 때는 남는 공간에 0을 채워 4-byte를 전송합니다.

스크립트의 바이트코드를 다운로드 하기 위해 사용자가 '**script_size**'와 '**script_code**'를 직접 엑세스하는 것을 권장하지 않습니다. PC의 Motor Control UI 유틸리티를 사용하여 스크립트를 다운로드 하기를 권장합니다.

7.7.4 script_variable - Script Variable

사용자는 '**script_variable**' 변수를 읽음으로 스크립트가 실행되는 동안 사용되는 변수들의 값을 모니터링 합니다. 스크립트가 사용하는 변수는 최대 256개가 되며, 각각의 변수에는 컴파일 시에 변수 ID가 0부터 순차적으로 부여됩니다. 변수의 값을 읽을 때는 변수 ID를 Sub-index로 지정합니다.

하이퍼터미널과 같은 유틸리티로 제어기의 RS-232 또는 RS-485 포트에 연결하여, 스크립트 변수를 텍스트 형식으로 읽을 때는 터미널 유틸리티에서 다음과 같이 입력합니다.

```
sv1↵
sv1=1.342
sv2↵
sv2=30
sv3↵
sv3=0
```

여기서 ↵는 키보드의 Enter 키를 의미합니다. 터미널 설정에 따라 다르겠지만, 입력된 값은 일반적으로 터미널에 표시되지 않습니다. 입력 후 Enter 키를 누를 때마다 읽은 값을 보여줍니다.

만일 스크립트에서 사용하는 변수 ID보다 큰 ID를 지정하면 읽은 값은 0이 됩니다. 또한, 스크립

트가 실행되기 전에 읽은 값은 모두 0입니다. 만일 스크립트의 실행이 종료된 후 읽으면, 스크립트 변수에 마지막으로 저장된 값을 읽게 됩니다.

스크립트 변수는 Motor Control UI 유틸리티의 Script 탭에서 모니터링 할 수 있습니다.

8 모터제어기 오브젝트

이 장에서는 모터 제어부의 구성 파라미터 설정 및 명령과 상태에 관련된 오브젝트들에 대해 설명합니다.

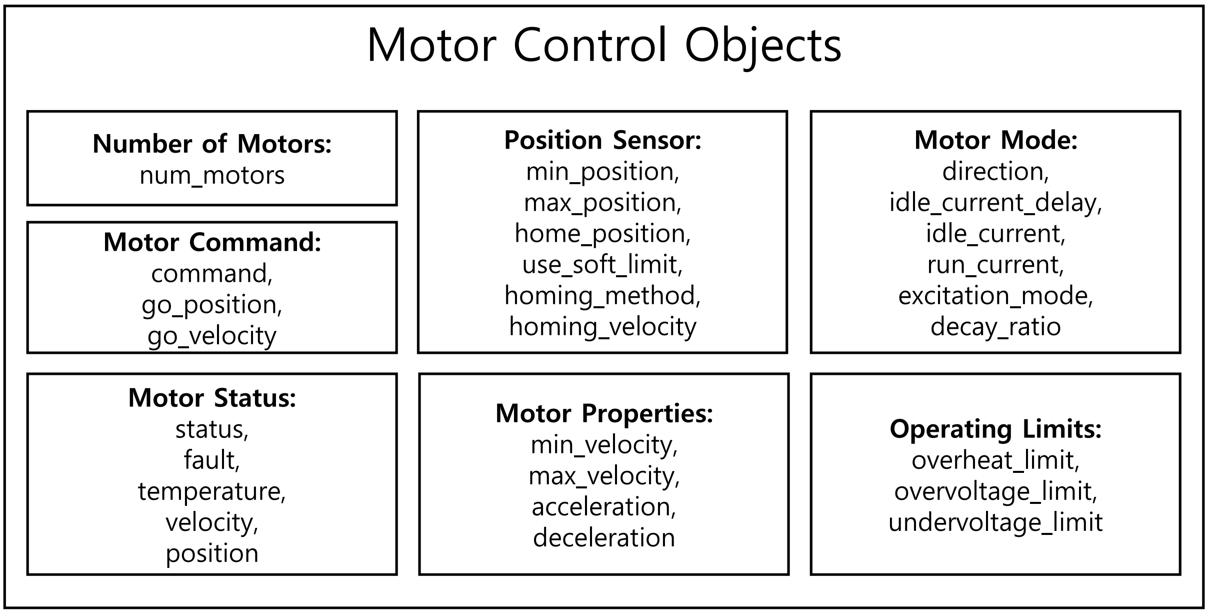


그림 8-1 Motor Controller Objects

싱글 채널 제어기는 하나의 모터제어기를 가집니다. 그리고 듀얼 채널 제어기는 두 개의 모터제어기를 가집니다. 듀얼 채널 제어기에서는 모터를 구분하기 위해 Sub-index를 사용합니다. 첫 번째 모터는 Sub-index가 1이고 두 번째 모터는 Sub-index가 2입니다. 싱글 채널 제어기에서는 모터가 하나이기 때문에 Sub-index로 1 또는 0을 사용할 수 있습니다.

8.1 모터 개수

모터 개수는 상수 오브젝트로 읽기만 가능합니다. 제품 생산 시 결정되며 사용자가 바꿀 수 없는 값입니다.

표 8-1 모터 개수 확인 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
num_motors, nm	80/0	I8	제어기에 연결 가능한 모터의 수

상기 표에서 Long Name과 Short Name은 텍스트 모드에서 오브젝트를 엑세스 할 수 있도록 부여된 이름입니다. 그리고 Index와 Sub-index는 오브젝트를 엑세스 하기 위한 주소입니다.

Type 열은 오브젝트의 형식을 나타냅니다:

- I8 - 부호를 가지는 8bit 정수형 수
- I16 - 부호를 가지는 16bit 정수형 수
- I32 - 부호를 가지는 32bit 정수형 수
- F32 - 부호를 가지는 32bit 실수형 수

8.1.1 num_motors - Number of Motors

제어기에 연결 가능한 모터의 수는 1개 입니다. '**num_motors**' 상수의 값으로 다음 중 하나를 가집니다:

- 1 - 싱글 채널 모터제어기

새로운 제품이 출시됨에 따라 제어기에 연결 가능한 모터 수는 늘어날 수 있습니다.

8.2 모터 명령

이 절에서는 모터에 내려지는 위치, 속도 명령에 대해 설명합니다.

명령을 내리면 모터제어기는 해당 명령을 수행하기 위한 모드로 변경되고 명령에 따라 제어기가 동작하여 모터를 제어합니다.

표 8-2 모터 명령 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
command, co	101/ 0~1	I16	모터제어기에 내려지는 명령 코드
go_position, gp	111/ 0~1	I32	모터의 폐루프 위치 제어 명령 (단위: pulse)
go_velocity, gv	112/ 0~1	I32	모터의 폐루프 속도 제어 명령 (단위: pulse/sec)

8.2.1 command - Command

'**command**'에 명령 코드를 쓰는 것으로 모터제어기의 해당 기능을 실행할 수 있습니다. 다음은 명령 코드 목록입니다:

- 0 - Motor Disable
- 1 - Motor Enable
- 2 - Clear Fault Flags

- 3 - Motor Homing
- 6 - Deceleration Stop
- 7 - Quick Stop

0 - Motor Disable:
모터의 전원을 차단하고 모터를 제어하지 않습니다. 모터의 전원을 차단한 것은 모터가 전원으로부터 분리된 것과 같은 효과를 냅니다.

1 - Motor Enable:
모터에 0V 전원만 공급하고 모터를 제어하지는 않습니다. 모터에 0V 전원을 공급하는 것은 모터의 +와 – 단자를 결선하는 것과 같은 효과를 냅니다. 만일 모터가 회전하고 있는 상태에서 모터에 0V 전원을 공급한다면, 모터는 급 감속하여 정지하게 됩니다.

2 - Clear Fault Flags:
모터에 발생한 모든 폴트(Fault) 플래그를 해제합니다. 폴트 플래그를 해제한다고 해서 폴트의 원인이 해결되지는 않습니다. 사용자가 원인을 파악하여 적절한 조치를 해야 합니다.

6 - Deceleration Stop:
모터의 속도를 **deceleration** 파라미터에 설정된 감속도로 감속하여 정지합니다.

7 - Quick Stop
모터를 최대한 빨리 정지합니다. 디지털 입력에서의 'Emergency Stop' 기능과는 구별됩니다.

모터가 Motor Enable될 때는 다음 사항을 체크합니다. 이 중 하나라도 통과되지 못하면 모터는 Motor Disable 상태에 머무르게 됩니다:

- 모터제어기의 디지털 입력 중 **'Emergency Stop'**이 켜진 상태
- 모터에 흐르는 전류(currnet)가 **'overcurrent_limit'** 보다 큰 경우
- 전원의 전압인(**power_source_voltage**)이 **'undervoltage_limit'**과 **'overvoltage_limit'** 범위를 벗어난 경우
- 방열반의 온도(**temperature**)가 **'overheat_limit'** 보다 큰 경우

Motor Enable을 성공하면 다음 과정이 진행됩니다:

1. 아날로그 입력과 펄스 입력의 센터 안전 검사 플래그를 리셋
2. 위치, 속도 명령을 모두 0으로 리셋

8.2.2 go_position - Go Position

모터의 폐루프 위치 제어기가 추종해야 할 위치 명령을 내립니다. 명령을 내리기 전에 모터제어기가 속도 제어나 전류 제어, 전압 출력 모드에서 구동 중인 상황에 있었더라도 위치 제어 모드

로 즉시 변경되고 **'go_position'**로 주어진 명령 인수를 추종합니다.

만일 **'use_soft_limit'**이 켜진 상태라면 위치 명령은 **'min_position'**과 **'max_position'** 범위 내로 제한됩니다. 또한 이동 중 최고 속도는 **'max_velocity'** 값에 의해 제한됩니다.

'go_position'를 사용하려면, 모터제어기는 다음 사항을 만족하여야 합니다:

- 모터는 Enable 상태에 있어야 함
- 모터제어기의 디지털 입력 중 **'Emergency Stop (Disable)'**, **'Quick Stop'**, **'Declaration Stop'** 가 꺼져 있거나 제어기의 디지털 입력 포트로 매핑되어있지 않은 상태여야 함
- 모터제어기의 디지털 입력 중 **'Forward Limit Switch'**가 켜져 있는 상태라면 위치 명령이 현재 위치보다 작아야 함
- 모터제어기의 디지털 입력 중 **'Reverse Limit Switch'**가 켜져 있는 상태라면 위치 명령이 현재 위치보다 커야 함

8.2.3 go_velocity - Go Velocity

모터의 폐루프 속도 제어기가 추종해야 할 속도 명령을 내립니다. 명령을 내리기 전에 모터제어기가 위치 제어나 전류 제어, 전압 출력 모드에서 구동 중인 상황에 있었더라도 속도 제어 모드로 즉시 변경되고 **'go_velocity'**로 주어진 명령 인수를 추종합니다.

속도 명령의 절대값은 **'max_velocity'**보다 클 수 없으며, -**'max_velocity'**와 **'max_velocity'**사이의 값을 가질 수 있습니다.

'go_velocity'를 사용하려면, 모터제어기는 다음 사항을 만족하여야 합니다:

- 모터는 Enable 상태에 있어야 함
- 모터제어기의 디지털 입력 중 **'Emergency Stop (Disable)'**, **'Quick Stop'**, **'Declaration Stop'** 가 꺼져 있거나 제어기의 디지털 입력 포트로 매핑되어있지 않은 상태여야 함
- 모터제어기의 디지털 입력 중 **'Forward Limit Switch'**가 켜져 있는 상태라면 속도 명령이 0보다 작아야 함
- 모터제어기의 디지털 입력 중 **'Reverse Limit Switch'**가 켜져 있는 상태라면 속도 명령이 0보다 커야 함

8.3 모터 상태

모터의 속도 및 위치 등의 모터의 운용 상태를 읽어오는 오브젝트입니다.

표 8-3 모터 상태 확인 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description

status, s	102/ 0~1	I32	모터제어기의 현재 상태
fault, f	103/ 0~1	I32	모터제어기 및 관련 I/O에서 발생한 폴트
temperature, tp	121/ 0~1	F32	방열판의 온도 (단위: °C)
velocity, v	124/ 0~1	I32	모터의 회전속도 (단위: pulse/sec)
position, p	125/ 0~1	I32	모터의 회전위치 (단위: pulse)

8.3.1 status - Status

'status'의 각 비트는 모터제어기의 구동 상태 및 제어기와 연결된 I/O의 상태를 표시합니다. 다음 목록을 참고하십시오:

- 0x0001 - Motor Enable
- 0x0002 - Motor Moving
- 0x0004 - Fault Detected
- 0x0008 - Motor Homing
- 0x0010 - DI Disable
- 0x0100 - AI not Center
- 0x0200 - PI not Center
- 0x0400 - AI loss Signal
- 0x0800 - PI loss Signal
- 0x1000 - DI Stop
- 0x2000 - AT Limit
- 0x4000 - Serial Go
- 0x8000 - Analog Go
- 0x10000 - Script Run

0x0001 - Motor Enable:
모터에 전원이 공급되고 위치와 속도 명령에 따라 모터제어기가 구동 가능한 상태입니다. 제어기에 명령을 내리기 전에 이 플래그가 켜진 상태인지 확인해야 합니다.

0x0002 - Motor Moving:
모터가 회전하고 있습니다. 즉, 모터의 속도가 0이 아님을 나타냅니다.

0x0004 - Fault Detected:
모터제어기에 폴트가 발생한 상태입니다. 폴트 발생 시 제어기와 모터, 주변회로를 보호하기 위해 모터에 공급되는 전원을 차단합니다. 폴트의 원인을 파악하려면 “**fault”** 상태를 읽어야 합니다.

0x0010 - Motor Homing:
모터가 Homing중 일 때 켜집니다. 홈포지션 이동이 완료되면 꺼집니다.

0x0020 - DI Disable:
IO에 연결된 ‘**Emergency stop (Motor Disable)**’ 버튼이 눌러진 상태입니다.

0x0100 - AI not Center:
모터제어기의 위치나 속도 명령에 매핑된 아날로그 입력 값이 모터 전원 공급 후 0에 있지 않습니다. 입력 값이 0이 될 때까지 입력 값은 강제로 0이 됩니다.

0x0200 - PI not Center:
모터제어기의 위치나 속도 명령에 매핑된 펄스 입력 값이 모터 전원 공급 후 0에 있지 않습니다. 입력 값이 0이 될 때까지 입력 값은 강제로 0이 됩니다.

0x0400 - AI loss Signal:
모터제어기의 위치나 속도 명령에 매핑된 아날로그 입력 값이 캘리브레이션의 Min/Max 범위를 벗어났습니다. 이 경우 입력 값은 강제로 0이 됩니다.

0x0800 - PI loss Signal:
모터제어기의 위치나 속도 명령에 매핑된 펄스 입력 값이 캘리브레이션의 Min/Max 범위를 벗어났습니다. 이 경우 입력 값은 강제로 0이 됩니다.

0x1000 - DI Stop:
모터제어기의 디지털 입력 포드 중 ‘**Emergency stop (Motor Disable)**’, '**Quick Stop**', ‘**Slowdown Stop**’이 켜진 상태입니다. 이 상태가 꺼질 때까지 모터제어기에 내려지는 위치와 속도 명령은 차단됩니다.

0x2000 - At Limit:
모터제어기의 디지털 입력 포트 중 '**Forward Limit Switch**'나 '**Reverse Limit Switch**'가 켜진 상태이거나 또는, 모터의 위치가 설정된 ‘**Min Position**’, ‘**Max Position**’ 범위를 벗어난 상태입니다. 이 상태가 꺼질 때까지 모터제어기에 내려지는 위치와 속도 명령은 제한적으로 실행됩니다. 즉, 리미트를 벗어나는 방향으로 움직이는 명령만 수행됩니다.

0x4000 - Serial Go:
시리얼(RS-232, RS-485) 포트로부터 모터제어기에 구동(위치, 속도) 명령이 내려지고 있습니다. 이 플래그는 구동 명령 외(구성 파라미터 읽고 쓰기와 같은 것들)에는 반응하지 않습니다.

0x8000 - Analog Go:
모터제어기에 매핑 된 아날로그 입력 혹은 펄스 입력 포트로부터 구동 명령이 입력되고 있습니다.

0x10000 - Script Run:
스크립트가 실행 중임을 표시합니다. 제어기의 **'system_status'** 상태의 **'Script Running'** 플래그와 동일합니다.

※주의※ 사용자는 상기 모터의 상태를 잘 확인하고 모터를 구동해야 합니다. 특히 '0x0004 – Fault Detected' 상태인 경우 폴트의 원인을 파악하여 적절한 조치를 한 후 모터 구동을 재개해야 합니다.

8.3.2 fault - Fault

모터제어기에서 폴트(Fault)가 발생하면 모터에 공급되는 전원은 차단됩니다. 폴트 상황이 해제되어야만 모터에 전력 공급이 가능합니다. 폴트 상태에서 모터제어기의 구성 파라미터 설정이나 상태 모니터링은 가능하지만 구동 명령은 실행되지 않습니다.

'fault'는 **'status'**의 **'0x0004 - Fault Detected'** 플래그가 켜졌을 때 폴트의 발생 원인을 파악하기 위한 오브젝트 입니다. 다음 목록을 참고하십시오:

- 0x0002 - Overvoltage
- 0x0004 - Undervoltage
- 0x0008 - Overheat

0x0002 - Overvoltage:
제어기에 공급되는 전압이 **'overvoltage_limit'**에서 설정한 값 이상으로 올라갔습니다. 과전압(Overvoltage)은 다음 중 하나가 원인이 될 수 있습니다:

- 모터제어기 전원(배터리, 파워서플라이)로부터 제어기에 높은 전압이 공급되는 경우
- 회전 관성이 큰 부하가 연결된 모터를 빠르게 감속하는 경우

과전압 상황은 일반적으로 전원으로 배터리가 아닌 파워서플라이가 사용될 경우 모터의 회생전류를 흡수하지 못하여 발생합니다. 이때에는 전원에 제동저항(Brake Resistor) 회로를 구성해야 합니다.

0x0004 - Undervoltage:
제어기에 공급되는 전압이 **'undervoltage_limit'**에서 설정한 값 이하로 내려갔습니다. 저전압(Undervoltage)은 다음 중 하나가 원인이 될 수 있습니다:

- 모터제어기 전원(배터리, 파워서플라이)으로부터 낮은 전압이 공급되는 경우
- 회전 관성이 큰 부하가 연결된 모터를 빠르게 가속하는 경우

저전압 상황은 일반적으로 전원의 용량이 낮아 모터의 초기 구동 시 흐르는 높은 전류를 감당하지 못하여 발생합니다.

0x0008 - Overheat:

Motor Driver Chip과 방열판의 온도가 **'overheat_limit'**에서 설정한 값 이상으로 올라갔습니다.

8.3.3 temperature - Temperature

'temperature' 상태는 Motor Driver Chip과 방열판의 온도를 나타냅니다.

이 값이 **'overheat_limit'**에서 설정한 값 이상으로 올라가면 **'0x0008 - Overheat'** 폴트를 발생하고 모터는 Disable 상태가 됩니다.

※ Motor Driver Chip의 내부 온도는 측정된 온도보다 훨씬 높을 수 있습니다.

8.3.4 velocity - Velocity

'velocity' 파라미터는 단위 초당 출력 펄스 변화량(회전속도)을 나타냅니다.

8.3.5 position - Position

'position' 파라미터는 모터의 위치를 나타냅니다.

8.4 모터의 위치 및 Homing 설정

이 절은 모터의 최소 최대 위치와 소프트 리미트 사용여부 그리고 Homing에 관련된 설정에 대해 설명합니다.

표 8-4 위치 센서 설정 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
min_position, np	141/ 0~1	I32	모터가 이동 가능한 최소 위치 (단위: pulse, 범위: 0 ~ 음수)
max_position, xp	142/ 0~1	I32	모터가 이동 가능한 최대 위치 (단위: pulse, 범위: 0 ~ 양수)
home_position, hp	143/ 0~1	I32	홈센서 감지 시 position에 로드 되는 위치 값 설정 (단위: pulse)
use_soft_limit, usl	146/ 0~1	I8	소프트 리미트 스위치 사용 여부 (기본값: 0)
homing_method,	147/	I8	모터제어기가 Homing을 하는 방법 설정

hm	0~1		
homing_velocity, hv	147/ 0~1	I32	Homing 시 모터의 구동 속도 설정 (단위: pulse/sec)

8.4.1 min_position, max_position - Min, Max Position

'min_position'과 **'max_position'** 파라미터는 모터가 이동 가능한 최소 위치와 최대 위치를 가집니다. 최소 위치는 양수가 될 수 없으며, 최대 위치는 음수가 될 수 없습니다.

'use_soft_limit'이 켜진 경우, **'go_position'**와 **'home_position'**는 최소 위치와 최대 위치 범위 내로 조정됩니다.

또한, **'min_position'**과 **'max_position'**은 정규화된 아날로그 입력이나 펄스 입력을 모터제어기의 위치 명령이나 위치 피드백으로 스케일을 변환합니다.

8.4.2 home_position - Home Position

'home_position' 파라미터는 디지털 입력 포트에 연결된 홈 센서 감지 시 **'position'**에 로드 되는 위치 값을 가집니다.

모터의 홈 위치는 다음과 같이 결정합니다:

1. **'home_position'**에 모터의 홈 위치를 설정합니다
2. 모터가 **'homing_velocity'** 속도값으로 움직입니다.
3. 홈 센서가 트리거 되면 **'position'**에 **'home_postion'** 값이 로드 됩니다.
4. 모터를 정지합니다.

8.4.3 use_soft_limit - Use Soft Limit

'use_soft_limit'는 소프트 리미트 스위치의 사용 여부를 결정합니다. 소프트 리미트 스위치는 실제 제어기의 디지털 입력 포트에 연결된 센서가 아니라, 소프트웨어적으로 설정된 리미트 범위를 벗어나는지에 대해 리미트 스위치의 ON/OFF 상태를 에뮬레이션 합니다.

이 값으로 다음 중 하나를 선택합니다:

- 0 - 소프트 리미트 사용하지 않음 (기본값)
- 1 - 소프트 리미트 사용함

이 값이 1로 설정된 경우, 모터의 위치(**position**)가 **'min_position'**과 **'max_position'** 범위를 벗어나면 리미트 스위치가 동작한 것과 같은 효과를 내도록 합니다.

8.4.4 homing_method - Homing Method

모터제어기가 Homing을 하는 방법은 디지털 입력을 통해서 센서값이 들어올 때까지 **'homing_velocity'** 속도값으로 이동하다가 센서 입력이 들어오면 제어기의 위치를 **'home_position'**으로 설정하고 정지하게 됩니다. **'homing_method'** 파라미터는 Homing하는 방법을 설정합니다.

Homing 방법은 다음과 같습니다.

- 0 - Home Sensor
- 1 - Forward Limit Switch
- 2 - Reverse Limit Switch

0 - Home sensor:
모터제어기의 디지털 입력 포트 중 **'Home Sensor'**가 감지되면 제어기의 위치를 **'home_position'**으로 설정하고 정지하게 됩니다.

1 - Forward Limit Switch
모터제어기의 디지털 입력 포트 중 **'Forward Limit Switch'**가 감지되면 제어기의 위치를 **'home_position'**으로 설정하고 정지하게 됩니다. 그리고 이 상태가 꺼질 때까지 모터제어기에 내려지는 위치와 속도 명령은 제한적으로 실행됩니다. 즉, 리미트를 벗어나는 방향(**Reverse**)으로 움직이는 명령만 수행됩니다.

2 - Reverse Limit Switch
모터제어기의 디지털 입력 포트 중 **'Reverse Limit Switch'**가 감지되면 제어기의 위치를 **'home_position'**으로 설정하고 정지하게 됩니다. 그리고 이 상태가 꺼질 때까지 모터제어기에 내려지는 위치와 속도 명령은 제한적으로 실행됩니다. 즉, 리미트를 벗어나는 방향(**Forward**)으로 움직이는 명령만 수행됩니다.

8.4.5 homing_velocity - Homing Velocity

'homing_velocity' 파라미터는 Homing 시 모터의 구동 속도(pulse/sec)를 설정합니다.

8.5 모터 속성 설정

이 절은 모터의 특성과 모터에 연결된 부하의 특성에 따라 결정되는 오브젝트들에 대하여 설명합니다.

표 8-5 모터 속성 설정 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
max_velocity, xv	153/ 0~1	I32	모터의 최대 회전 속도 (단위: pulse/sec)
acceleration, ac	154/ 0~1	I32	모터의 회전 가속도 (단위: pulse/sec^2)
deceleration, dc	155/ 0~1	I32	모터의 회전 감속도 (단위: pulse/sec^2)

※주의※ 모터의 속성에 관련된 구성 파라미터들은 제어기와 모터를 안전하게 운용하기 위해 올바른 값들로 설정되어야 합니다. 모터의 데이터시트를 참고하여 모터 정격에 맞게 설정하기 바랍니다.

8.5.1 acceleration, deceleration - Acceleration, Deceleration

'acceleration'과 **'deceleration'** 파라미터에는 각각 모터의 회전 가속도와 감속도를 설정합니다. 가속도와 감속도는 위치제어, 속도제어 모드에서 속도 프로파일을 만드는데 사용됩니다.

8.6 모터 구동 한계 설정

이 절에서는 모터의 구동 한계를 설정하는 오브젝트들에 대해 다룹니다.

표 8-6 모터 구동 한계 설정 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
overheat_limit, ohl	161/ 0~1	F32	방열판의 과열 한계 (단위: °C)
overvoltage_limit, ovl	165/ 0~1	F32	전원단의 과전압 한계 (단위: V)
undervoltage_limit, uvl	166/ 0~1	F32	전원단의 저전압 한계 (단위: V)

모터의 상태(전압, 열 등)가 한계를 벗어나는 경우에 폴트(**Fault**)를 발생하고 모터는 Disable 상태가 됩니다.

8.6.1 overheat_limit - Overheat Limit

'overheat_limit' 파라미터는 방열판의 과열(Overheat) 상황을 감지하기 위한 최대 한계 온도를 설정합니다.

방열판의 온도(**temperature**)가 **'overheat_limit'**를 넘어갈 경우, **'0x0008- Overheat'** 폴트를 발생하고 모터는 Disable 상태가 됩니다.

이 파라미터는 방열판의 열 전달을 고려하여 값을 설정해야 합니다.

8.6.2 overvoltage_limit - Overvoltage Limit

'overvoltage_limit' 파라미터는 모터의 과전압(Overvoltage) 상황을 감지하기 위한 최대 한계 전압을 설정합니다.

전원에서 공급되는 전압이 **'overvoltage_limit'**를 넘어갈 경우, **'0x0002 - Overvoltage'** 폴트를 발생하고 모터는 Disable 상태가 됩니다.

8.6.3 undervoltage_limit - Undervoltage Limit

'undervoltage_limit' 파라미터는 모터의 저전압(Undervoltage) 상황을 감지하기 위한 최소 한계 전압을 설정합니다.

전원에서 공급되는 전압이 **'undervoltage_limit'** 아래로 내려갈 경우, **'0x0004 - Undervoltage'** 폴트를 발생하고 모터는 Disable 상태가 됩니다.

8.7 모터 방향 및 정지 지연시간 설정

이 절에서는 모터 방향 및 모터 상태 전환 시 지연시간에 관련된 오브젝트들에 대해 설명합니다.

표 8-7 모터 및 I/O 구동 관련 설정

Long name, Short name	Index/ Sub-index	Type	Description
direction, dir	174/ 0~1	I8	모터의 정방향/역방향 회전 설정 (기본값: 0)

8.7.1 direction - Direction

'direction'은 모터의 회전 방향을 결정합니다. 이 파라미터의 값으로 다음 중 하나를 선택합니다:

- 0 - Normal (기본값)
- 1 - Invert

8.8 모터 전류, 분해능, 전류감쇄 설정

이 절은 모터 구종 및 정지 전류, 분해능, 전류 감쇄 설정에 대해 설명 합니다.

표 8-8 스텝 모터제어기 각종 모드 설정 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
idle_current_delay, icd	179/ 0~1	I32	모터 Run 상태에서 Idle 상태로 전환시 지연시간 설정 (단위: ms, 기본값: 100, 범위: 0 ~ 3000)
idle_current, idc	181/ 0~1	I8	모터의 정지(시 흐르는 최대 전류 설정 (단위: A, 기본값: 1, 범위: 0 ~ 3)
run_current, ruc	182/ 0~1	I8	모터의 구동시 흐르는 최대 전류 설정 (단위: A, 기본값: 3, 범위: 0 ~ 3)
excitation_mode, exm	183/ 0~1	I8	모터의 마이크로 스테핑 모드 설정 (x2, x8, x16, x32, x64, x128, x10, x20)
decay_ratio, der	184/ 0~1	I8	모터의 전류 소진 모드 설정 (Fast, Mixed, Slow)

8.8.1 idle_current_delay - Idle Current Delay

'idle_current_delay' 파라미터는 모터의 구동 상태에서 정지 상태로 바뀔 때 걸리는 지연시간을 설정합니다. 즉, **'run_current'** 값이 적용된 모터의 구동 상태에서 정지하면 바로 **'idle_current'** 값으로 적용되는 것이 아니라 **'idle_current_delay'** 시간 이후에 **'idle_current'** 값으로 적용이 됩니다.

반대로 모터의 정지 상태에서 구동 상태로 바뀔 때는 즉각적으로 **'idle_current'**에서 **'run_current'** 값으로 적용됩니다.

8.8.2 idel_current - Idle Current

'idle_current' 파라미터는 모터가 정지했을 때 흐르는 최대 전류를 설정합니다.

이 파라미터 값이 크면 클수록 모터가 정지시 흐르는 전류가 커지기 때문에 모터와 제어기의 열이 올라갑니다. 최소한 **'run_current'** 파라미터 값보다 작은 값으로 설정해서 제어기를 구동해야 합니다.

8.8.3 run_current - Run Current

'run_current' 파라미터는 모터가 구동할 때 흐르는 최대 전류를 설정합니다.

이 파라미터는 모터가 구동할 수 있는 최소한의 전류 값으로 설정해야 합니다. 이는 각각의 모터마다 다르니 데이터시트를 확인하시기 바랍니다.

8.8.4 excitation_mode - Excitation Mode

'excitation_mode' 파라미터는 모터의 마이크로 스테핑 모드를 설정합니다. 파라미터의 값으로 다음 중 하나를 선택합니다.

- 0 - Half Step (기본값)
- 1 - 1/8 Step
- 2 - 1/16 Step
- 3 - 1/32 Step
- 4 - 1/64 Step
- 5 - 1/128 Step
- 6 - 1/10 Step
- 7 - 1/20 Step

8.8.5 decay_ratio - Decay Ratio

'decay_ratio' 파라미터는 모터의 전류 소진 모드를 설정합니다. 이 파라미터의 값으로 다음 중 하나를 선택합니다. 이 파라미터 설정을 통해서 모터의 정밀도를 개선합니다. 파라미터의 값으로 다음 중 하나를 선택합니다.

- 0 - Fast Decay (기본값)
- 1 - Mixed Decay
- 2 - Slow Decay

0 - Fast Decay:

1 - Mixed Decay

2 - Slow Decay

9 I/O 오브젝트

제어기는 외부 센서와 액츄에이터를 연결할 수 있는 디지털 입력, 디지털 출력, 아날로그 입력, 펄스 입력 포트를 가지고 있습니다. 이러한 입출력 포트들은 모터제어기의 특정 기능으로 매핑되어 모터제어기가 외부로 인터페이스 할 수 있도록 합니다.

이 장에서는 제어기의 입출력 채널에 관련된 오브젝트들에 대해 설명합니다.

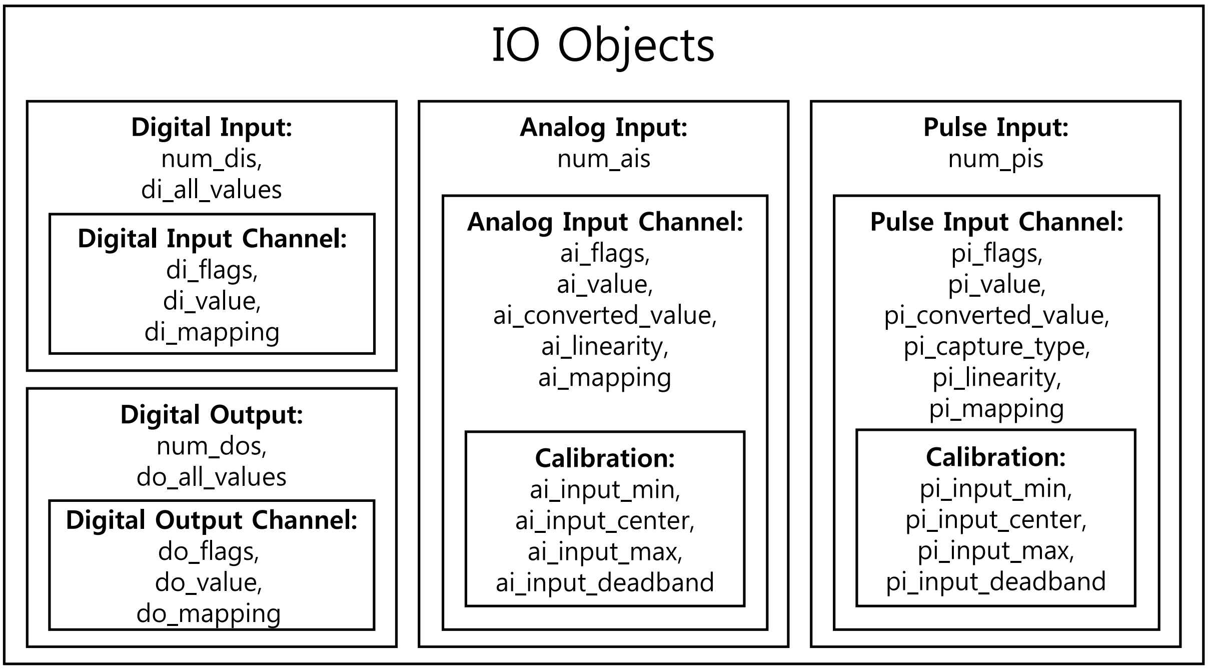


그림 9-1 I/O Objects

모터제어기 모델에 따라 I/O 채널의 수는 달라집니다. I/O의 각 채널을 구분하기 위해 Sub-index를 사용합니다.

9.1 디지털 입력

이 절에서는 디지털 입력 채널들에 대해 공통으로 사용되는 오브젝트들에 대해 다룹니다.

표 9-1 디지털 입력 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
num_dis, ndi	60/0	I8	디지털 입력 채널의 수
di_all_values, di	63/0	I32	디지털 입력 채널들을 하나의 32bit 값으로 모음

상기 표에서 Long Name과 Short Name은 텍스트 모드에서 오브젝트를 엑세스 할 수 있도록 부여된 이름입니다. 그리고 Index와 Sub-index는 오브젝트를 엑세스 하기 위한 주소입니다.

Type 열은 오브젝트의 형식을 나타냅니다:

- I8 - 부호를 가지는 8bit 정수형 수
- I16 - 부호를 가지는 16bit 정수형 수
- I32 - 부호를 가지는 32bit 정수형 수
- F32 - 부호를 가지는 32bit 실수형 수

9.1.1 num_dis - Number of DI

'num_di' 상수는 제어기가 보유한 디지털 입력 채널의 수를 나타냅니다.

9.1.2 di_all_values - DI All Values

'di_all_values' 상태의 각 비트는 해당 디지털 입력 채널의 값(**di_value**)을 나타냅니다:

- Bit0 - 디지털 입력 채널 1의 값
- Bit1 - 디지털 입력 채널 2의 값
- …

이 상태는 모든 디지털 입력 채널의 값을 한 번에 읽을 수 있도록 합니다.

9.2 디지털 입력 채널

다음은 디지털 입력의 각 채널 별로 설정되는 오브젝트입니다.

표 9-2 디지털 입력 채널 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
di_flags, dif	200/ 1~N	I8	디지털 입력 사용 여부, 신호 반전 설정
di_value, div	201/ 1~N	I8	디지털 입력 채널의 값 (범위: 0 or 1)
di_mapping dim	202/ 1~N	I16	디지털 입력 채널을 특정 모터의 동작으로 매핑

상기 표에서 Sub-index에 사용된 O는 디지털 입력 채널의 수(**num_di**)를 나타냅니다.

9.2.1 di_flags - DI Flags

'di_falgs' 파라미터는 아래와 같이 디지털 입력 사용 여부 및 신호 반전에 대해 설정합니다.

- 0x01 - Enable
- 0x02 - Invert Input Signal

9.2.2 di_value - DI Value

'di_value' 상태는 디지털 입력 채널의 값을 나타냅니다. 값으로 1bit의 0이나 1을 가집니다.

9.2.3 di_mapping - DI Mapping

'di_mapping' 파라미터는 제어기의 디지털 입력 채널을 특정 모터의 디지털 입력 버퍼의 동작으로 매핑합니다. 이 파라미터의 상위 8bit로는 대상 모터를 선정합니다:

- 0x0000 - (채널 1번에) 연결된 모터 선택

하위 8bit로는 동작을 선택합니다:

- 0 - None
- 1 - Emergency Stop (Disable Motor)
- 2 - Quick Stop
- 3 - Slowdown Stop
- 4 - Forward Limit Switch
- 5 - Reverse Limit Switch
- 6 - Home Sensor
- 7 - Run Script

9.3 디지털 출력

이 절에서는 디지털 출력 채널들에 공통으로 사용되는 오브젝트들에 대해 다룹니다.

표 9-3 디지털 출력 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description

num_dos, ndo	65/0	I8	디지털 출력 채널의 수
do_all_values, do	68/0	I32	디지털 출력 채널들을 하나의 32bit 값으로 모음

9.3.1 num_do - Number of DO

'num_do' 상수는 제어기가 가진 디지털 출력 채널의 수를 나타냅니다.

9.3.2 do_all_values - DO All Values

'do_all_values' 변수의 각 비트는 해당 디지털 출력 채널의 값(**do_value**)을 나타냅니다:

- Bit0 - 디지털 출력 채널 1의 값
- Bit1 - 디지털 출력 채널 2의 값
- ...

이 변수는 모든 디지털 출력 채널의 값을 한 번에 내보낼 수 있도록 합니다.

9.4 디지털 출력 채널

다음은 디지털 출력의 각 채널 별로 설정되는 오브젝트입니다.

표 9-4 디지털 출력 채널 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
do_flags, dof	210/ 1~M	I8	디지털 출력 사용 여부, 신호 반전 설정
do_value, dov	211/ 1~M	I32	디지털 출력 채널의 값 (범위: 0 or 1)
do_mapping, dom	212/ 1~M	I32	디지털 출력 채널을 특정 모터의 상태로 매핑

상기 표에서 Sub-index에 사용된 M은 디지털 출력 채널의 수(**num_do**)를 나타냅니다.

9.4.1 do_flags - DO Flags

'do_falgs' 파라미터는 아래와 같이 디지털 출력 사용 여부 및 신호 반전에 대해 설정합니다.

- 0x01 - Enable
- 0x02 - Invert Input Signal

9.4.2 do_value - DO Value

'do_value' 변수는 디지털 출력 채널에 값을 읽고 씁니다. 값으로 1bit의 0이나 1을 가집니다.

디지털 출력 채널에 **'do_mapping'**으로 기능이 매핑되지 않은 경우, 사용자가 직접 **'do_value'**에 값을 씀으로 디지털 출력 채널로 원하는 값을 내보낼 수 있습니다.

9.4.3 do_mapping - DO Mapping

do_mapping 파라미터는 제어기의 디지털 출력 채널을 특정 모터의 디지털 출력 버퍼의 상태로 매핑합니다. 이 파라미터의 상위 8bit로는 대상 모터를 선정합니다:

- 0x0000 - (채널 1번에) 연결된 모터 선택

하위 8bit로는 상태를 선택합니다:

- 0 - None
- 1 - Motor Enable (Typical Use: Motor brake release)
- 2 - Motor Running (Typical Use: Turn ON door lock)
- 3 - Motor Fault (Typical Use: Turn ON alarm lamp)

9.5 아날로그 입력

이 절에서는 아날로그 입력 채널에 공통으로 사용되는 오브젝트들에 대해 다룹니다.

표 9-5 아날로그 입력 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
num_ais, nai	70/0	I8	아날로그 입력 채널의 수

9.5.1 num_ai - Number of AI

num_ai 상수는 제거기가 보유한 아날로그 입력 채널의 수를 나타냅니다.

9.6 아날로그 입력 채널

다음은 아날로그 입력의 각 채널 별로 설정되는 오브젝트입니다.

표 9-6 아날로그 입력 채널 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
ai_flags, aif	220/ 1~O	I8	아날로그 입력 사용 여부, 신호 반전, Min/Max Safety, Senter Safety 설정
ai_value, aiv	221/ 1~O	I32	아날로그 입력 채널의 원시 값(raw value) (범위: 0 ~ 4095)
ai_converted_value, aicv	222/ 1~O	F32	변환 과정을 거쳐 정규화된 값(normalized value) (범위: -1 ~ 1)
ai_linearity, ail	223/ 1~O	I8	지수/로그 변환 형식 지정
ai_mapping, aim	224/ 1~O	I16	아날로그 입력을 특정 모터의 명령이나 피드백과 매핑
ai_input_min, ain	225/ 1~O	I32	캘리브레이션: 입력 최소값
ai_input_center, aic	226/ 1~O	I32	캘리브레이션: 입력 중앙값
ai_input_max, aix	227/ 1~O	I32	캘리브레이션: 입력 최대값
ai_input_deadband, aidb	228/ 1~O	I32	캘리브레이션: 입력의 데드밴드(deadband) 값

상기 표에서 Sub-index에 사용된 O는 아날로그 입력 채널의 수(**num_ai**)를 나타냅니다.

9.6.1 ai_flags - AI Flags

'ai_falgs' 파라미터는 아래와 같이 아날로그 입력 사용 여부, 신호 반전, Min/Max Safety 그리고 Senter Safety에 대해 설정합니다.

- 0x01 - Enable
- 0x02 - Invert Input Signal

- 0x04 - Min/Max Safety
- 0x08 - Center Safety

9.6.2 ai_value - AI Value

아날로그 입력 포트에는 0V 과 5V 사이의 전압이 가해집니다. 전압은 12bits AD 컨버터에 의해 디지털로 변환되어 마이크로컨트롤러가 읽게 됩니다.

'ai_value' 상태는 아날로그 입력 채널의 값을 나타냅니다. 값은 변환이 이루어지기 전의 0과 4095 사이의 12bits 디지털 값입니다.

9.6.3 ai_converted_value - AI Converted Value

'ai_converted_value' 상태는 아날로그 입력 채널에서 읽은 원시 값을 변환 과정을 거쳐 -1과 1 사이의 정규화 된 값으로 나타냅니다.

9.6.4 ai_linearity - AI Linearity

'ai_linearity' 파라미터는 아날로그 입력의 정규화 된 값을 지수/로그 함수로 변환합니다. 이 파라미터의 값으로 다음 중 하나를 선택합니다:

- 0 - linear(no change) (기본값)
- 1 - exp weak
- 2 - exp medium
- 3 - exp strong
- 4 - log weak
- 5 - log medium
- 6 - log strong

9.6.5 ai_mapping - AI Mapping

'ai_maping' 파라미터는 제어기의 아날로그 입력 채널을 특정 모터의 명령 버퍼나 피드백 버퍼, 디지털 입력 버퍼로 매핑 합니다. 이 파라미터의 상위 8bit로는 대상 모터를 선정합니다:

- 0x0000 - (채널 1번에) 연결된 모터 선택

하위 8bit로는 명령 버퍼, 디지털 입력 버퍼의 기능을 선택합니다:

- 0 - None
- 1 - Go Velocity
- 2 - Go Position
- 3 - Emergency Stop (Disable Motor)
- 4 - Quick Stop
- 5 - Slowdown Stop
- 6 - Forward Limit Switch
- 7 - Reverse Limit Switch
- 8 - Home Sensor
- 9 - Run Script

여기서 **'Go Velocity'**와 **'Go Position"** 기능은 명령 버퍼로 매핑되는 기능이고. 나머지 기능은 디지털 입력 버퍼로 매핑되는 기능입니다.

9.6.6 ai_input_min, ai_input_center, ai_input_max - AI Input Min, Center, Max

'ai_input_min', 'ai_input_center', 'ai_input_max'는 아날로그 입력을 정규화하는데 사용되는 캘리브레이션 파라미터들입니다.

'ai_input_min' 파라미터는 아날로그 입력의 최소값을 나타냅니다. 이 값이 정규화되면 -1이 됩니다. **'ai_input_center'** 파라미터는 아날로그 입력의 중앙값을 나타냅니다. 이 값이 정규화되면 0이 됩니다. **'ai_input_max'** 파라미터는 아날로그 입력의 최대값을 나타냅니다. 이 값이 정규화되면 1이 됩니다.

'ai_input_min' 보다 **'ai_input_center'** 값이 같거나 커야 하고, **'ai_input_center'** 보다 **'ai_input_max'** 값이 같거나 커야 합니다.

9.6.7 ai_input_deadband - AI Input Deadband

'ai_nput_deadband' 파라미터는 아날로그 입력의 중앙값에서 0으로 인식하는 입력 범위를 나타냅니다.

9.7 펄스 입력

이 절에서는 펄스 입력 채널에 공통으로 사용되는 오브젝트들에 대해 다룹니다.

표 9-7 펄스 입력 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
num_pis, npi	75/0	I8	펄스 입력 채널의 수

9.7.1 num_pi - Number of PI

'num_pi' 상수는 펄스 입력 채널의 수를 나타냅니다.

9.8 펄스 입력 채널

다음은 펄스 입력의 각 채널 별로 설정되는 오브젝트들입니다.

표 9-8 펄스 입력 채널 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
pi_flags, pif	230/ 1~P	I8	펄스 입력 사용 여부, 신호 반전, Min/Max Safety, Senter Safety 설정
pi_value, piv	231/ 1~P	I32	펄스 입력 채널의 원시 값(raw value)
pi_converted_value, picv	232/ 1~P	F32	변환 과정을 거쳐 정규화 된 값(normalized value) (범위: -1 ~ 1)
pi_capture_type, pit	233/ 1~P	I8	펄스 캡쳐의 종류
pi_linearity, pil	234/ 1~P	I8	지수/로그 변환 형식 지정
pi_mapping, pim	235/ 1~P	I16	펄스 입력을 특정 모터의 명령이나 피드백과 매핑
pi_input_min, pin	236/ 1~P	I32	캘리브레이션: 입력 최소값
pi_input_center, pic	237/ 1~P	I32	캘리브레이션: 입력 중앙값
pi_input_max, pix	238/ 1~p	I32 (CP)	캘리브레이션: 입력 최대값
pi_input_deadband, pidb	239/ 1~p	I32 (CP)	캘리브레이션: 입력의 데드밴드(deadband) 값

상기 표에서 Sub-index에 사용된 P는 펄스 입력 채널의 수(**num_pi**)를 나타냅니다.

9.8.1 pi_flags - PI Flags

'pi_falgs' 파라미터는 아래와 같이 펄스 입력 사용 여부, 신호 반전, Min/Max Safety 그리고 Senter Safety에 대해 설정합니다.

- 0x01 - Enable
- 0x02 - Invert Input Signal
- 0x04 - Min/Max Safety
- 0x08 - Center Safety

9.8.2 pi_value - PI Value

펄스 입력 포트는 펄스기반 입력 신호를 받아들입니다. 펄스 입력은 최소 20Hz에서 최대 20kHz 사이에서 동작하여야 합니다. 그리고 펄스의 ON 신호 폭은 최소 10μs 이상 되어야 합니다.

'pi_value' 상태는 설정된 **'pi_capture_type'**에 따라 Pulse Width, Frequency, Duty Cycle 중 하나의 값을 캡쳐 한 원시 값입니다.

9.8.3 pi_converted_value - PI Converted Value

'pi_converted_value' 상태는 펄스 입력 채널에서 읽은 원시 값을 변환 과정을 거쳐 -1과 1 사이의 정규화 된 값으로 나타냅니다.

9.8.4 capture_type - PI Capture Type

'capture_type' 파라미터는 펄스 입력의 데이터 수집 형식(Capture Type)을 나타냅니다. 이 파라미터의 값으로 다음 중 하나를 선택합니다:

- 0 - Pulse Width
- 1 - Frequency
- 2 - Duty Cycle

데이터 수집 형식으로 **'0 - Pulse Width'**가 설정되면 **'pi_value'**는 펄스의 폭을 측정합니다. **'1 - Frequency'**가 설정되면 펄스의 주파수를 측정합니다. **'2 - Duty Cycle'**로 설정되면 펄스의 듀티 사이클을 측정 하며 측정 값은 0 ~ 1000‰ 사이가 됩니다.

9.8.5 pi_linearity - PI Linearity

'pi_linearity' 파라미터는 펄스 입력의 정규화 된 값을 지수/로그 함수로 변환합니다. 이 파라미터의 값으로 다음 중 하나를 선택합니다:

- 0 - linear(no change) (기본값)
- 1 - exp weak
- 2 - exp medium
- 3 - exp strong
- 4 - log weak
- 5 - log medium
- 6 - log strong

9.8.6 pi_mapping - PI Mapping

'pi_mapping' 파라미터는 제어기의 펄스 입력 채널을 특정 모터의 특정 모터의 명령 버퍼나 피드백 버퍼, 디지털 입력 버퍼로 매핑합니다. 이 파라미터의 상위 8bit로는 대상 모터를 선정합니다:

- 0x0000 - (채널 1번에) 연결된 모터 선택

하위 8bit로는 명령 버퍼나 디지털 입력 버퍼의 기능을 선택합니다:

- 0 - None
- 1 - Go Velocity
- 2 - Go Position
- 3 - Emergency Stop (Disable Motor)
- 4 - Quick Stop
- 5 - Slowdown Stop
- 6 - Forward Limit Switch
- 7 - Reverse Limit Switch
- 8 - Home Sensor
- 9 - Run Script

여기서 '**Go Velocity**'와 **'Go Position'** 기능은 명령 버퍼로 매핑되는 기능이고. 나머지 기능은 디지털 입력 버퍼로 매핑되는 기능입니다.

9.8.7 pi_input_min, pi_input_center, pi_input_max - PI Input Min, Center, Max

'pi_input_min', 'pi_input_center', 'pi_input_max'는 펄스 입력을 정규화하는데 사용되는 캘리브레이션 파라미터들입니다.

'pi_input_min' 파라미터는 펄스 입력의 최소값을 나타냅니다. 이 값이 정규화되면 -1이 됩니다. **'pi_input_center'** 파라미터는 펄스 입력의 중앙값을 나타냅니다. 이 값이 정규화되면 0이 됩니다. **'pi_input_max'** 파라미터는 펄스 입력의 최대값을 나타냅니다. 이 값이 정규화되면 1이 됩니다.

'pi_input_min' 보다 **'pi_input_center'** 값이 같거나 커야 하고, **'pi_input_center'** 보다 **'pi_input_max '** 값이 같거나 커야 합니다.

9.8.8 pi_input_deadband - PI Input Deadband

'pi_input_deadband' 파라미터는 펄스 입력의 중앙값에서 0으로 인식하는 입력 범위를 나타냅니다.

10 통신 프로토콜

이 장에서는 PC나 마이크로컨트롤러에서 제어기의 오브젝트 값을 읽고 쓰기 위한 통신 프로토콜에 대해 설명합니다. 제어기에 동작을 명령하거나 제어기의 구성 파라미터를 설정하는 것은 해당 오브젝트에 특정 값을 쓰는 것을 의미하며, 제어기의 상태를 읽거나 제어기의 구성 파라미터를 읽는 것은 해당 오브젝트에서 특정 값을 읽는 것을 의미합니다.

10.1 용어의 정의

제어기의 객체를 통신 프로토콜로 읽고 쓰는데 사용되는 용어를 표 10-1에서 정의합니다.

표 10-1 통신 프로토콜에 사용되는 용어

Term	Description
Index	오브젝트의 참조 색인
Sub-index	오브젝트의 참조 부-색인
Short Name	Index와 호환되는 오브젝트의 짧은 이름
Long Name	Index와 호환되는 오브젝트의 긴 이름

상기 표에서 사용되는 용어 중 Index, Short Name, Long Name은 오브젝트를 표현하는 방식의 차이입니다. Index는 모터제어기 내부의 오브젝트를 참조하는 색인으로, 시리얼 바이너리 패킷에서 사용됩니다.

Sub-index는 오브젝트를 참조하는 부-색인으로, 통상 모터의 채널 번호를 의미합니다. (채널 1번에) 연결된 모터에 대한 명령이나 설정이라면 Sub-index로 1을 사용하면 됩니다. 연결된 모터가 아니라 모터제어기 자체에 대한 것이라면 통상 Sub-index는 0이 됩니다. Sub-index는 상황에 따라 외부 I/O의 채널 번호가 되기도 합니다.

Short Name과 Long Name은 Index와 호환되는 오브젝트의 짧은 이름과 긴 이름으로, 시리얼 텍스트 패킷에서 사용됩니다. 이 이름은 모터제어기 내부에서 Index 값으로 변환되어 Index와 동일하게 사용됩니다.

만약 사용자가 제어기에 인가된 주 전원의 전압을 체크하는 **'power_source_voltage'** 오브젝트의 내용을 읽고자 할 때, 시리얼 텍스트 기반으로 통신하는 경우, 해당 오브젝트의 Short Name인 **'psv'**나 Long Name인 **'power_source_voltage'**를 사용하면 됩니다. 시리얼 바이너리 패킷 통신을 사용하는 경우는 해당 오브젝트의 Index인 8을 사용하면 됩니다.

표 10-1에서는 오브젝트의 4가지 속성을 보여줍니다. 각각의 오브젝트 타입에 따라 패킷의 길이가 달라질 수 있습니다.

표 10-2 오브젝트의 타입

Object Type	Size	Description
INT8	1byte	부호를 가지는 8 bit 정수형
INT16	2bytes	부호를 가지는 16 bit 정수형
INT32	4bytes	부호를 가지는 32 bit 정수형
FLOAT	4bytes	부호를 가지는 32 bit 실수형

또한, 표 10-2에서는 오브젝트의 엑세스 속성을 나타냅니다. RO로 표시된 오브젝트는 읽기 전용으로 상수(Constant)와 상태(Status) 오브젝트가 여기에 해당하며, 값을 쓰려고 할 때 에러를 리턴할 것입니다. WO로 표시된 오브젝트는 쓰기 전용으로 명령(Command) 오브젝트가 여기에 해당하며, 값을 읽으려고 할 때 에러를 리턴할 것입니다. RW로 표시된 오브젝트는 구성 파라미터(Configuration Parameter)와 변수(Variable) 오브젝트가 여기에 해당합니다.

표 10-3 오브젝트의 엑세스 속성

Access	Object	Description
RO	Constant, Status	읽기 전용 (Read Only)
WO	Command	쓰기 전용 (Write Only)
RW	Configuration Parameter, Variable	읽고 쓰기 가능 (Read Writeable)

10.2 시리얼 바이너리 패킷

제어기의 오브젝트들은 시리얼 통신으로 읽고 쓰기가 가능합니다. 시리얼 통신은 텍스트 기반의 패킷과 함께 바이너리 기반의 패킷 통신을 함께 지원합니다.

이 절은 제어기에 존재하는 오브젝트들을 엑세스하기 위한 시리얼 통신의 바이너리 패킷 구조에 관해 설명합니다.

10.2.1 바이너리 패킷의 메시지 구조

마스터 PC와 모터제어기 간에 시리얼 통신으로 주고받는 패킷의 주요 구성 요소는 Node ID와 메시지의 길이(Length) 그리고 전송되는 메시지(Message)가 됩니다(아래 그림 참조).

Node ID	Length	Message
11bit	0~8	0 ~ 8byte

Node ID로는 최대 11bit를 설정할 수 있는데, 제어기의 **'Device ID'**가 여기에 사용됩니다. 마스터에서 장치로 메시지를 보내거나 장치가 마스터로 회신 할 때, Node ID에는 장치 ID가 동일하게 지정되어야 합니다. Length는 메시지(Message)의 길이를 나타내는데, 메시지의 최대 크기가 8byte이므로 0에서 8사이의 값이 됩니다. 메시지에는 전송되는 데이터를 담습니다.

다음 그림은 메시지(Message)의 기본 구조를 나타냅니다. 메시지는 최대 8byte 크기를 가질 수 있으며, 1byte의 Command와 2byte의 Index, 1byte의 Sub-index와 나머지는 상황에 맞는 오브젝트의 값(Value)으로 구성되어 있습니다. 상황에 따라 5th ~ 8th byte 지점의 값들은 존재할 수도 있고 아닐 수도 있습니다. 만약 5th byte 지점까지만 파라미터가 존재한다면 5th byte까지의 내용만 전송하면 됩니다.

1st byte	2nd byte	3rd byte	4th byte	5th byte	6th byte	7th byte	8th byte
Command	Index		Sub-index	Value			

여기서 Command는 표 10-4과 같은 액세스 형식(Access Code)과 오브젝트 형식(Object Type)의 조합으로 1byte를 구성하게 됩니다.

표 10-4 액세스 형식(Access Code)과 오브젝트 형식(Object Type)

구분	코드	내용
Access Code	0x10	오브젝트 쓰기 요청
	0x20	오브젝트 쓰기 요청에 대한 응답
	0x30	오브젝트 읽기 요청
	0x40	오브젝트 읽기 요청에 대한 응답
	0x80	오브젝트 읽기/쓰기 요청에 대한 에러 응답
Object Type	0x00	INT8 - 8 bit signed integer
	0x04	INT16 - 16 bit signed integer
	0x08	INT32 - 32 bit signed integer
	0x0C	FLOAT - 32 bit IEEE Standard 754 floating-point

※ Index나 Value와 같이 한 바이트 이상의 데이터가 메시지에 저장될 때는 데이터의 하위 바이트부터 왼쪽에 저장되는 리틀 인디안(Little-Endian)방식을 따릅니다.

10.2.2 바이너리 패킷의 기본 구조

다음은 바이너리 패킷의 기본 구조입니다. Value의 최대 길이를 4byte로 고정하고 패킷 전체 길이를 13byte로 고정합니다.

1st byte	2nd byte	3rd byte	4th ~ 11th byte	12th byte	13th byte
STX (0x02)	Length (=13)	Device ID	Message	Checksum	ETX (0x03)

상기 그림에서 처음과 끝의 STX, ETX는 시작과 끝을 의미하는 문자입니다. 그리고 Length는 13으로 고정되어 있습니다.

Checksum은 3rd ~ 11th byte 지점 까지를 바이트 단위로 더한 후 결과에서 1byte만 취한 값입니다. 다음은 C언어로 작성된 Checksum 계산 예제 코드입니다:

```
char Checksum (char *msg, int len)
```

```
{
        char cs = 0;

        for (int i=0; i<len; ++i)
                cs += msg[i];
        return cs;
}
```

4th ~ 11th byte 영역의 Message는 10.2.1 바이너리 패킷의 메시지 구조를 확인하시기 바랍니다. 참고로 8byte의 메시지는 모두 채워야 합니다. 만약 메시지의 크기가 8byte가 되지 않는다면, 남는 공간에 0을 채우면 됩니다.

10.2.3 오브젝트 읽기 요청

오브젝트의 값을 읽을 때는, 패킷의 4th ~ 11th byte 지점에 위치하는 Message를 아래와 같이 구성하면 됩니다. Command는 표 10-4의 Access Code 0x30과 읽고자 하는 Object Type을 조합해서 Command를 먼저 구성해야 합니다.

<table>
<tr><th>4th byte</th><th>5th byte</th><th>6th byte</th><th>7th byte</th><th>8th byte</th><th>9th byte</th><th>10th byte</th><th>11th byte</th></tr>
<tr><td>Command</td><td colspan="2">Index</td><td>Sub-index</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>

10.2.4 오브젝트 쓰기 요청

오브젝트에 값을 쓸 때는, 7th byte 지점 까지는 오브젝트 읽기 요청과 동일하게 구성합니다. 단 Command는 표 10-4의 Access Code에 의해 0x10과 오브젝트 형에 맞는 조합으로 작성되어야 하며, 그 형식에 맞춰 오브젝트의 값(Value)을 작성합니다.

<table>
<tr><th>4th byte</th><th>5th byte</th><th>6th byte</th><th>7th byte</th><th>8th byte</th><th>9th byte</th><th>10th byte</th><th>11th byte</th></tr>
<tr><td rowspan="4">Command</td><td rowspan="4" colspan="2">Index</td><td rowspan="4">Sub-index</td><td>Value(INT8)</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td colspan="2">Value(INT16)</td><td>0</td><td>0</td></tr>
<tr><td colspan="4">Value(INT32)</td></tr>
<tr><td colspan="4">Value(FLOAT)</td></tr>
</table>

10.2.5 오브젝트 읽기/쓰기 요청에 대한 성공 응답

오브젝트의 값을 읽거나 쓰기가 성공한 경우에는 오브젝트의 값(Value)이 응답으로 돌아옵니다. 패킷의 구성은 오브젝트 쓰기 요청에서와 같습니다. 단, Command는 표 10-4의 Access Code에 제시된 것처럼 0x20이나 0x40중 하나와 Object Type에 맞는 조합으로 작성됩니다.

<table>
<tr><th>4th byte</th><th>5th byte</th><th>6th byte</th><th>7th byte</th><th>8th byte</th><th>9th byte</th><th>10th byte</th><th>11th byte</th></tr>
<tr><td rowspan="2">Command</td><td rowspan="2" colspan="2">Index</td><td rowspan="2">Sub-index</td><td>Value(INT8)</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td colspan="2">Value(INT16)</td><td>0</td><td>0</td></tr>
</table>

			Value(INT32)
			Value(FLOAT)

제어기는 사용자가 쓴 값이 적용되지 않을 수 있기 때문에 사용자의 오브젝트를 읽는 요청뿐만 아니라 오브젝트에 쓰기 요청에 대해서도 해당 오브젝트의 값을 응답합니다.

10.2.6 오브젝트 읽기/쓰기 요청에 대한 실패 응답

오브젝트의 내용을 읽거나 쓰기 위한 패킷을 제어기에 요청했을 때, 오류가 발생하는 경우는 다음 그림과 같은 형식의 오류 패킷이 돌아옵니다.

4th byte	5th byte	6th byte	7th byte	8th byte	9th byte	10th byte	11th byte
Command	Error Code		0	0	0	0	0

오류 메시지는 표 10-5에 나타나 있습니다. 시리얼 바이너리 패킷에서 오류 메시지 내용은 전달되지 않으며 Error Code만 반환됩니다. 그리고 Command는 0x80이 반환됩니다.

표 10-5 오브젝트의 읽기/쓰기 과정에서 발생하는 오류

Error Code	Error Message	내용
1	undefined index	Index나 Sub-index로 지정한 오브젝트가 존재하지 않음
2	packet format error	패킷의 구성이 잘못 되었음
3	variable access error	읽기 전용 오브젝트에 쓰거나, 쓰기 전용 오브젝트의 읽기를 시도함

※ 오류 메시지는 패킷의 구성에 관련된 오류로 제어기의 오류 상황이나 오작동에 대한 부분이 아닙니다.

10.3 시리얼 텍스트 패킷

시리얼 통신에서는 텍스트(text) 기반으로 제어기의 오브젝트를 읽고 쓸 수 있도록 합니다. 이는 사용자가 통신을 위한 전용 프로그램을 사용하지 않더라도 Hyperterminal과 같은 유틸리티를 사용하여 제어기의 오브젝트들을 쉽게 엑세스 할 수 있도록 합니다.

시리얼 텍스트 기반 패킷에서는 Index를 직접적으로 사용하지 않고 표 10-1의 Short Name이나 Long Name과 Sub-index를 사용합니다.

10.3.1 오브젝트 읽기 요청

오브젝트 값을 읽을 때는 텍스트 패킷을 다음과 같이 구성합니다.

Short/Long Name	Sub-index	CR/LF

Short/Long Name은 오브젝트의 Index에 해당하는 이름이며, Sub-index는 해당 Index에 따라 필요하면 모터 채널 혹은 외부 I/O의 채널을 의미합니다. Sub-index가 0인 경우 생략 가능합니다.

CR/LF는 Carriage Return/Line Feed의 약자로 ASCII 코드상 각각 0x0D, 0x0A입니다. 일반적으로는 키보드의 Enter 키에 해당하며 하이퍼터미널과 같은 유틸리티로 PC에서 연결했다면 명령 입력 후 Enter 키를 입력하는 것입니다. (이후 CR/LF를 ↲ 기호로 대체)

연결된 모터에 내려진 속도 명령의 값을 알고자 할 때는 아래와 같이 입력합니다.

```
go_velocity↲
gv1↲
```

모터의 속도 명령은 **'Go Velocity'**로, 이에 대한 Long Name은 **'go_velocity'**이고 Short Name은 **'gv'**입니다. 그러므로 상기 두 명령은 동일한 결과를 가져옵니다.

10.3.2 오브젝트 쓰기 요청

오브젝트 값을 쓸 때는 텍스트 패킷을 다음과 같이 구성합니다.

Short/Long Name	Sub-index	=	Value	CR/LF

Short/Long Name은 오브젝트의 Index에 해당하는 이름이며, Sub-index는 해당 Index에 따라 필요하면 모터 채널 혹은 외부 I/O의 채널을 의미합니다. Sub-index가 0인 경우 생략 가능합니다.

연결된 모터를 100 pulse/sec 속도로 구동시키고자 할 때 아래와 같이 입력합니다.

```
gv=100↲
go_velocity1=100↲
```

모터의 속도 명령은 **'Go Velocity'**로, 이에 대한 Long Name은 **'go_velocity'**이고 Short Name은 **'gv'**입니다. 그러므로 상기 두 명령은 동일한 결과를 가져옵니다.

10.3.3 오브젝트 읽기/쓰기 요청에 대한 성공 응답

오브젝트의 값을 읽거나 쓰기가 성공한 경우에는 오브젝트의 값(Value)이 응답으로 돌아옵니다. 제어기가 응답하는 형식은 다음과 같습니다.

Short/Long Name	Sub-index	=	Value	CR/LF

사용자가 Short/Long Name 형식 중 하나로 질의를 하면, 같은 형식으로 응답합니다. 응답하는 문장의 끝에는 CR 문자와 LF 문자가 함께 붙습니다.

두 채널을 동시에 다루는 명령에 대해 다음과 같이 응답합니다.

Short/Long Name	=	Value1	,	Value2	CR/LF

제어기는 사용자가 쓴 값이 적용되지 않을 수 있기 때문에 사용자의 오브젝트를 읽는 요청뿐만 아니라 오브젝트에 쓰기 요청에 대해서도 해당 오브젝트의 값을 응답합니다.

10.3.4 오브젝트 읽기/쓰기 요청에 대한 실패 응답

오브젝트의 내용을 읽거나 쓰기 위한 패킷을 제어기에 요청했을 때, 오류가 발생하는 경우는 다음 그림과 같은 형식의 오류 패킷이 돌아옵니다.

!	Error Code	,	Error Message	CR/LF

오류 메시지는 표 10-5에 나타나 있습니다.

※ 오류 메시지는 패킷의 구성에 관련된 오류로 제어기의 오류 상황이나 오작동에 대한 부분이 아닙니다.

10.3.5 Device ID 의 부여

만일 둘 이상의 제어기가 RS-422이나 RS-485 버스에 연결된 경우, 시리얼 텍스트 패킷에는 각각의 제어기를 구분하는 장치의 ID를 부여해야 합니다.

이 때는 다음 그림과 같이 패킷을 수신할 장치의 ID를 패킷 앞에 붙이면 됩니다.

Device ID	;	Serial Text Packet

만약 **'Device ID'** 1번을 가지는 제어기의 모터를 100 pulse/sec 속도로 구동시키고자 할 때 아래와 같이 입력합니다.

```
1;gv1=100↲
1;go_velocity1=100↲
```

제어기가 장치 ID를 가지는 패킷을 수신하였을 때는, 응답 패킷에도 장치 ID를 붙여 응답합니다. 응답 패킷도 다음과 같은 형태가 됩니다.

Device ID	;	Serial Text Packet

※주의※ RS-422이나 RS-485 버스에 둘 이상의 제어기를 연결할 때는, 각각의 모터제어기 간에 Device ID가 서로 충돌하지 않도록 설정하여야 합니다.

11 Mini-C 스크립트 언어

Mini-C 스크립트 언어는 C언어에서와 유사하게 제어기의 프로그래밍을 하도록 하기 위한 C언어의 서브셋으로 설계된 언어입니다. C언어의 제어문과 수식 연산구조를 일부 따오면서 배열이나 함수포인트 등 복잡한 부분을 제거하여 스크립트 언어를 처음 접하는 사용자가 쉽게 배우고 사용할 수 있습니다. 만일 C언어를 알고 있는 사용자라면 **"11.2 C언어와의 차이"**만 살펴보더라도 바로 사용 가능합니다.

이 장은 스크립트의 기본 구성과 특징에 대하여 설명하고 사용자가 직접 프로그램을 작성할 수 있도록 연산자, 제어문, 내장 함수들에 대하여 예제와 함께 설명합니다.

11.1 스크립트 관련 구성

Mini-C 스크립트 언어를 이용하여 작성한 제어기 프로그램은 Mini-C 스크립트 컴파일러를 사용하여 바이트코드로 변환 되고, 이 바이트코드는 제어기로 다운로드 되어 가상머신에 의해 해석되어 실행됩니다.

Mini-C 스크립트 언어:
Mini-C 스크립트 언어는 C언어의 서브셋으로 제어기의 프로그래밍에 사용되는 언어입니다.

Mini-C 스크립트 컴파일러:
Mini-C 스크립트 언어로 작성된 프로그램을 제어기의 가상머신에서 수행 가능한 바이트코드로 컴파일 합니다. 컴파일러는 Motor Control UI 유틸리티에 내장되어 있습니다.

바이트코드(Bytecode):
바이트코드는 제어기의 가상머신에서 실행될 수 있도록 정의된 중간코드입니다. 스크립트 언어로 작성된 프로그램은 바이트코드로 해석된 다음 제어기에 다운로드 되고 실행됩니다. 이러한 두 단계 구조는 바이트코드에 실행 시 필요한 정보만 담게 되어 스크립트 코드를 직접적으로 인터프리팅 하는 시스템에 비해 수행 성능을 높일 수 있습니다.

가상머신(Virtual Machine):
가상머신은 모터제어기의 마이크로컨트롤러에 포팅되어 있습니다. 가상머신은 바이트코드로 컴파일 된 프로그램을 해석하여 실행합니다.

11.2 C언어와의 차이

Mini-C 스크립트 언어는 C언어의 문법을 참조하여 설계되었습니다. 스크립트 언어는 C언어에서 다음과 같은 것들을 제공하지 않는 서브셋 언어 입니다:

- #define, #ifdef, #if 등의 전처리 문
- 함수의 정의와 선언 (main 함수 포함)
- return 키워드
- int, float, long 등 자료형 키워드
- 문자열 상수 (Ex: "abc", "ABC")와 문자 상수 (Ex: 'a', 'A')
- 배열 연산자([]), 포인터 연산자(*), 주소 참조 연산자(&), 멤버 연산자(., ->)
- sizeof 연산자
- ? : 연산자
- 캐스트 연산자 (Ex: (int), (float))
- typedef, struct, union, enum 등 자료구조 관련 키워드
- switch, case 분기문 키워드

변수의 선언에 자료형은 사용되지 않으며, 변수는 정수형이나 실수형으로 초기화 하면서 선언됩니다. 다음 변수 선언 예를 참조하기 바랍니다:

- a=123 - 정수형 변수 a를 선언하면서 값 할당
- b=4.567 - 실수형 변수 b를 선언하면서 값 할당

C언어에서 사용되는 다음 분기문과 반복문을 사용 가능합니다:

- if, else - 조건에 따라 정해진 영역의 프로그램을 실행
- goto, label - 특정 프로그램 코드 위치로 무조건 점프
- for - 특정 조건이 완료 될 때까지 특정 영역을 반복 실행
- while - 특정 조건이 만족할 동안 특정 영역을 반복 실행
- do, while - 먼저 특정 영역을 실행하고 조건이 맞으면 이를 반복
- break - 현재 실행하고 있는 반복문 블록을 빠져 나옴
- continue - 현재 실행하고 있는 반복문 블록의 초기로 이동

C언어에서 사용되는 다음 수학 연산자는 사용 가능합니다:

- 산술 연산자: +, -, *, /, %, ++, --
- 관계 연산자: ==, !=, >, <, >=, <=
- 논리 연산자: !, &&, ||
- 비트 연산자: ~, &, |, ^, <<, >>
- 대입 연산자: =, +=, -=, *=, /=, %=, &=, |=, ^=, <<=, >>=

Mini-C 스크립트 언어에서 사용하는 키워드는 다음과 같습니다:

- if, else, do, for, while, goto, break, continue

Mini-C 스크립트 언어에서는 다음 상수들이 정의되어 있습니다:

- M_E, M_PI, M_EULER, FALSE, TRUE, SS_NONE, SS_SCRIPT_RUN, …

11.3 문장

스크립트로 작성된 프로그램은 문장(Statement)들로 구성됩니다. 문장은 프로그램을 실행하기 위한 기본적인 실행 단위라고 볼 수 있습니다. 다음 그림 11-1은 스크립트 프로그램의 구조를 보여주고 있습니다.

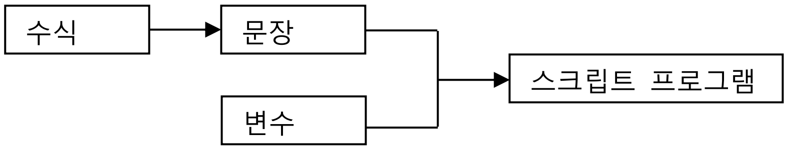


그림 11-1 스크립트 프로그램의 구조

11.3.1 수식과 문장

프로그램을 이루는 가장 기본적인 것은 수식입니다. 연산자는 "`a = 10`", "`a < b`", "`a + b`", "`a << 4`" 등과 같이 하나의 연산을 행하는 가장 단순한 것입니다. 그리고 단순한 수식이 여러 개가 모여 "`a+b*10`", "`(a>b) && (b>c)`" 등과 같이 복잡한 수식을 이루게 됩니다. 이러한 수식은 하나 또는 그 이상이 모여 하나의 문장을 구성합니다.

스크립트 언어에서는 ';'로 문장을 구분합니다. 이렇게 ';'로 구분된 각각의 문장을 단일문이라고 합니다. 다음 예제를 참고하십시오.

```
a = 5;  10*a + 3;
```

11.3.2 복합문

여러 개의 문장들을 '{'와 '}'를 이용하여 하나의 실행단위로 묶어 줄 수 있는데, 이를 블록 또는 복합문(Compound Statement)이라 하고 이 복합문을 하나의 문장으로 다시 고려할 수 있습니다. 다음과 같이 사용하면 복합문이 됩니다.

```
{
    a = 5;
    b = 10*a + 3;
}
```

11.3.3 제어문

스크립트 언어에서는 일정한 형식을 가지고 프로그램의 흐름을 제어하는 문장들도 있습니다. 예를 들자면, 조건을 판단하여 분기하는 `if-else` 문장 같은 것입니다.

제어문에 대해서는 **"11.8 제어문"**에서 상세하게 설명합니다.

11.4 주석

스크립트 언어에서 주석문(Comment)은 '/*' 와 '*/'사이 또는 '//'뒤에 기술하며, 컴파일 대상에서 제외됩니다. 주석문은 주로 프로그램의 이해를 증진시키기 위해 사용합니다. 스크립트 프로그램 내에 주석문을 나타내기 위해, C언어에서 사용하는 것과 동일한 두 가지 방법을 제공합니다.

11.4.1 블록 주석문

블록 주석은 '/*'과 '*/'로 둘러 싸인 주석문을 말합니다. 이러한 블록 주석문은 한 라인 또는 두 라인 이상을 주석으로 처리할 수 있으며 다음과 같이 사용할 수 있습니다.

```
/*
    블록 주석 내용1
    블록 주석 내용2
    ...
 */
```

11.4.2 라인 주석문

라인 주석은 '//'를 사용하여 나타내며, '//'가 한 라인의 어디에 나와도 상관없고, '//'이 나온 후부터 그 라인의 끝까지 주석으로 처리하게 됩니다. 라인 주석문은 다음과 같이 사용할 수 있습니다.

```
// 라인 주석 내용1
// 라인 주석 내용2
```

11.5 상수

상수는 한번 값이 결정되면 프로그램이 실행되는 동안 새로운 값으로 변경할 수 없는 정보를 말합니다. 스크립트 언어에서 사용되는 상수(리터럴 혹은 리터럴 상수)는 정수형과 실수형(부동 소수; floating point), 미리 정의된 상수로 구분됩니다.

스크립트 언어는 문자 상수(`'A', 'a'`), 문자열 상수(`"abc", "DEF"`)를 지원하지 않습니다.

11.5.1 정수형 상수

정수형 상수는 스크립트 언어에서 사용되는 가장 기본적인 형태의 상수입니다. 메모리에 저장될 때는 32bit 크기를 가지며 범위는 -2^{31} 과 $2^{31}-1$ 사이가 됩니다.

다음과 같이 사용되면 정수형 상수로 인식됩니다.

```
10, -10, 999              // 10진수로 표현된 정수형 상수
05, 011, -077             // 8진수로 표현된 정수형 상수
0x11, -0x11, 0xFF         // 16진수로 표현된 정수형 상수
```

상기 예에서와 같이 정수형 상수는 8진수, 10진수, 그리고 16진수 등 세가지 진법으로 표현할 수 있습니다:

- 8진수 : '0'으로 시작하는 정수형 상수, '0' 다음에는 '0'에서 '7'까지의 8진수 숫자들이 옴
- 10진수: 일반적으로 사용하는 정수형 상수, '0'에서 '9'까지의 10진수 숫자들로 표현
- 16진수: '0x' 또는 '0X'로 시작하는 정수형 상수, '0x' 또는 '0X' 다음에는 '0'에서 '9'와 'A' 또는 'a'에서 'F' 또는 'f' 등 16진수 숫자들이 옴

10진수는 양수, 0, 음수가 있습니다. 스크립트에서 10진수를 사용할 때 0으로 시작해서는 안됩니다. 예를 들면 01234라고 쓰면 이것은 10진수가 아닙니다. 10진수는 0을 제외한 나머지 숫자로 시작하고 그 뒤로는 0부터 9까지의 숫자가 올 수 있습니다. 0으로 시작되는 숫자는 스크립트에서는 8진수나 16진수를 의미합니다.

11.5.2 실수형 상수

실수형 상수는 부동소수 값을 표현할 수 있습니다. 메모리에 저장될 때는 IEEE 754 표준에 따라 64bit 크기를 가지며 범위는 음수일 때 -1.79769313486231E+308 과 -4.9406564584124E-308 사이고 양수일 때 4.9406564584124E-308 과 1.79769313486231E+308 사이가 됩니다.

다음과 같이 사용되면 실수형 상수가 됩니다.

```
1.234, -1.234, 123.       // 고정 소수점으로 표기된 실수형 상수
1.234e3, 1.234e-3         // 부동 소수점으로 표기된 실수형 상수
```

상기 예에서와 같이 실수형 상수의 가장 간단한 표기법으로는 소수점을 기준으로 왼쪽에 정수부 오른쪽에 소수부를 적는 고정 소수점 표기법을 사용하는 것입니다. 예로 숫자 5를 쓸 때, 5라고 쓰면 정수형 상수가 됩니다. 실수형 상수임을 나타내려면 5.0이라고 쓰거나 0을 생략하고 5.라고 써야 합니다.

또 다른 표기법으로 e를 기준으로 왼쪽에 가수 오른쪽에 지수를 적는 부동 소수점 표기법이 있습니다. 숫자 중간에 나오는 e(또는 E)는 부동 소수의 지수부분을 의미합니다. 실수는 내부적으로 모두 부동 소수점 방식으로 기억되지만 상수를 표현할 때는 고정 소수점과 부동 소수점 표기법을 모두 사용할 수 있습니다.

11.5.3 미리 정의된 상수

다음은 스크립트 언어에서 미리 정의된 수학 상수입니다. 수학 상수는 프로그램이 컴파일 될 때, 실수형 상수로 바뀌게 됩니다.

표 11-1 정의된 수학 상수

상수	값	설명
M_E	2.7182818284590452354	자연상수
M_PI	3.14159265358979323846	원주율
M_EULER	0.57721566490153286061	오일러-마스케로니 상수

수학 상수는 다음과 같이 사용할 수 있습니다.

```
a = sin(M_PI/2);          // a는 1을 가짐
b = cos(M_PI/2);          // b는 0을 가짐
```

또한 사용자가 제어기의 오브젝트에 할당되는 값들에 대한 상수를 정의하고 있습니다. 이 상수들은 주로 특정 오브젝트에 한정되어 사용되어야 합니다.

표 11-2 스크립트에서 사용 가능한 상수

Definition	value	Description
Boolean 값 정의		
FALSE	0	
TRUE	1	
system_command Object와 함께 사용되는 상수 정의		
SC_SAVE_CONFIG_FLASH	1	Save properties in Flash Memory
SC_LOAD_CONFIG_FLASH	2	Load properties from Flash Memory
SC_SCRIPT_RUN	8	Script Run
SC_SCRIPT_STOP	9	Script Stop
SC_FACTORY_DEFAULT	98	Reset Configuration Parameters to Factory Default value
SC_RESET_CONTROLLER	99	Software Reset Controller
system_status Object와 함께 사용되는 상수 정의		
SS_NONE	0	(None)
SS_SCRIPT_RUN	0x00010000	Script 수행 중인 상태
command Object와 함께 사용되는 상수 정의		
MC_DISABLE	0	Motor Disable, 모터에 공급되는 전원 차단
MC_ENABLE	1	Motor Enable, 모터는 구동 가능한 상태가 됨
MC_CLEAR_FAULT	2	제어기에 발생한 폴트 해제

MC_HOME_SEARCH	3	Home search 시작
MC_STOP	6	구동중인 모터의 감속 정지
MC_QUICK_STOP	7	구동중인 모터의 긴급 정지
homing_method Object와 함께 사용되는 상수 정의		
HS_HOME_SENSOR	0	Home search 시에 Home Sensor 사용
HS_FORWARD_LIMIT	1	Home search 시에 Forward limit 센서 사용
HS_REVERSE_LIMIT	2	Home search 시에 Reverse limit 센서 사용
status Object와 함께 사용되는 상수 정의		
ST_ENABLE	0x0001	모터에 전력이 공급되고 있는 상황
ST_MOVING	0x0002	모터가 회전하고 있음
ST_FAULT	0x0004	제어기에서 폴트 발생
ST_HOMING	0x0008	모터가 homing 중임
ST_DI_DISABLE	0x0010	Emergency stop 버튼이 눌림
ST_AI_NOT_CENTER	0x0100	모터 Enable 전환 후 analog input 값이 center(0)에 있지 않음
ST_PI_NOT_CENTER	0x0200	모터 Enable 전환 후 pulse input 값이 center(0)에 있지 않음
ST_AI_LOSS	0x0400	Analog input 이 min~max 범위를 넘어갔거나 적절한 값이 들어오지 않음
ST_PI_LOSS	0x0800	Pulse input 이 min~max 범위를 넘어갔거나 적절한 값이 들어오지 않음
ST_DI_STOP	0x1000	Digital Input에서 stop, quick stop가 작동
ST_AT_LIMIT	0x2000	Position 값이 위치 리미트 범위를 벗어남
ST_SERIAL_GO	0x4000	Serial Port (RS-232, RS-485, Ethernet)으로부터 구동 명령이 입력 중
ST_ANALOG_GO	0x8000	Analog, Pulse input port로부터 구동 명령이 입력 중
ST_SCRIPT_RUN	0x00010000	Script가 실행 중
di_flags Object와 함께 사용되는 상수 정의		
DI_ENABLE	0x01	사용(활성화)
DI_INVERT	0x02	입력 신호 반전
do_flags Object와 함께 사용되는 상수 정의		
DO_ENABLE	0x01	사용(활성화)
DO_INVERT	0x02	출력 신호 반전
ai_flags Object와 함께 사용되는 상수 정의		
AI_ENABLE	0x01	사용(활성화)
AI_INVERT	0x02	입력 신호 반전
AI_MINMAX_SAFETY	0x04	입력 신호로부터 Min/Max safety 체크

AI_CENTER_SAFETY	0x08	입력 신호로부터 Center safety 체크
pi_flags Object와 함께 사용되는 상수 정의		
PI_ENABLE	0x01	사용(활성화)
PI_INVERT	0x02	입력 신호 반전
PI_MINMAX_SAFETY	0x04	입력 신호로부터 Min/Max safety 체크
PI_CENTER_SAFETY	0x08	입력 신호로부터 Center safety 체크
ai_linearity, pi_linearity Object와 함께 사용되는 상수 정의		
LT_LINEAR	0	입력 값의 변화 없음
LT_EXP_WEAK	1	입력 값을 약한 exp 그래프로 변환
LT_EXP_MEDIUM	2	입력 값을 중간 exp 그래프로 변환
LT_EXP_STRONG	3	입력 값을 강한 exp 그래프로 변환
LT_LOG_WEAK	4	입력 값을 약한 log 그래프로 변환
LT_LOG_MEDIUM	5	입력 값을 중간 log 그래프로 변환
LT_LOG_STRONG	6	입력 값을 강한 log 그래프로 변환
pi_capture_type Object와 함께 사용되는 상수 정의		
CT_PULSE_WIDTH	0	Pulse input의 캡쳐 타입을 Pulse Width로 선택
CT_FREQUENCY	1	Pulse input의 캡쳐 타입을 Frequency로 선택
CT_DUTY_CYCLE	2	Pulse input의 캡쳐 타입을 Duty Cycle로 선택
ai_mapping, pi_mapping Object와 함께 사용되는 상수 정의		
AI_NONE	0	매핑하지 않음
AI_MC_VELOCITY	1	Motor Command (Velocity)
AI_MC_POSITION	2	Motor Command (Position)
di_mapping Object와 함께 사용되는 상수 정의		
DI_NONE	0	매핑하지 않음
DI_DISABLE	1	Disable Motor (Emergency Stop)
DI_QUICK_STOP	2	Quick Stop (Deadman Switch)
DI_STOP	3	Slowdown Stop
DI_FWD_LIMIT_SW	4	Forward Limit Switch
DI_REV_LIMIT_SW	5	Reverse Limit Switch
DI_HOME_SENSOR	6	Load Home Counter
DI_RUN_SCRIPT	7	Run Script
do_mapping Object와 함께 사용되는 상수 정의		
DO_NONE	0	매핑하지 않음
DO_MOTOR_ENABLE	1	Motor Enable (Typical Use: Motor brake release)
DO_MOTOR_RUNNING	2	Motor Running (Typical Use: Turn ON door lock)

상기의 상수들은 제어기 오브젝트와 함께 다음과 같이 사용할 수 있습니다.

```
_command = MC_ENABLE;

if (_status & ST_ENABLE) {
        ...
}
```

11.6 변수

변수는 프로그램이 실행되는 동안 처리되는 데이터를 임시 저장하는 메모리 공간입니다. 변수는 프로그램이 시작될 때 모두 0으로 초기화 됩니다.

11.6.1 변수명

변수명을 만들 때 사용 가능한 문자는 대문자, 소문자, 숫자, 밑줄문자('_')입니다. 이 중 숫자는 변수명의 처음 문자로 올 수 없습니다. 변수명은 다음과 같은 규칙에 의해 사용자가 정합니다:

- 영문자('a'~'z', 'A'~'Z'), 숫자('0'~'9'), 밑줄문자 '_'로 구성되며, 첫 글자는 반드시 영문자 또는 밑줄문자 '_'가 되어야 함
- 대문자와 소문자는 서로 다른 문자로 인식함
- 예약어(`for`, `while`, `do`, …)와 미리 정의된 상수(`M_PI`, `M_E`, `TRUE`, `FALSE`, …)는 변수명으로 사용할 수 없음

변수 명은 다음과 다음과 같이 만들 수 있습니다.

```
abc, abc123, _123, ABC, ABCdef
```

11.6.2 변수의 선언과 초기화

변수의 선언에 자료형은 사용되지 않으며, 변수에 상수가 대입될 때와 같이 정수형이나 실수형으로 초기화 되면서 자료형이 결정됩니다. 변수는 자료형을 결정하는 내부 파라미터를 가지며, 변수의 사용처에 따라 혹은 변수에 저장되는 값의 자료형에 따라 변수의 자료형이 결정됩니다.

변수를 선언할 때 초기화 하지 않으면 컴파일러는 변수를 사용하는 것으로 인식합니다. 그래서 선언되지 않은 변수가 사용되었다는 에러를 발생합니다. 다음 예제를 참고하십시오.

```
a = 123;            // 변수 a는 정수형 변수가 됨
b = 1.23;           // 변수 b는 실수형 변수가 됨
c;                  // c는 에러 발생, 변수가 선언될 때는 항상 초기화 되어야 함
```

변수의 범위는 프로그램 전체에 연장되어있습니다. 변수는 한 번 이상 프로그램에서 선언 할 수 없습니다.

11.6.3 시스템 변수

시스템 변수는 제어기의 모든 오브젝트와 연결된 변수로, 제어기에서 기본적으로 선언되어 있는 변수들입니다. 제어기의 오브젝트 값을 읽거나 쓰기 위해 사용되는 getv(), setv() 함수를 대체하여 사용될 수 있습니다. 또한 표현식의 구조를 간단하게 만들어 주며 사용자가 직관적으로 프로그래밍 할 수 있도록 합니다. 다음 예제를 참고하십시오.

```
a = getv (125, 0);          // 모터의 현재 위치를 읽어옴
setv (111, 0, 5000);        // 모터에 5000 펄스 위치로 이동 명령을 내림
```

상기 예제는 시스템 변수를 사용하여 다음과 같이 작성됩니다.

```
a = _position;              // 모터의 현재 위치를 읽어옴
_go_position = 5000;        // 모터에 5000 펄스 위치로 이동 명령을 내림
```

시스템 변수는 제어기의 오브젝트 이름(Long name or Short name) 앞에 '_' 문자를 붙여 만들어 집니다. 만일 오브젝트의 sub-index가 0이 아니라면, 오브젝트 이름 뒤에 sub-index 값을 붙이면 됩니다. 다음 예제를 참고하십시오.

```
_device_id;                 // 제어기의 장치 ID, sub-index = 0
_serial_baudrate1;          // RS-232 포트의 통신 속도, sub-index = 1
_serial_baudrate2;          // RS-485 포트의 통신 속도, sub-index = 2
```

제어기에서 제공하는 시스템 변수는 다음 표와 같습니다.

표 11-3 시스템 변수 목록

Index	Sub-index	System Variable (Short Name)	System Variable (Long Name)
1	0	_vid	_vendor_id
2	0	_pid	_product_id
3	0	_swv	_software_version
4	0	_hwv	_hardware_version
5	0	_sst	_system_status
7	0	_sco	_system_command
8	0	_psv	_power_source_voltage
11	0	_id	_device_id
13	1~2	_sb1~2	_serial_baudrate1~2
14	0	_sw	_serial_watchdog
16	0	_suc	_startup_command
17	0	_ssa	_script_stop_action

31	0	_ip	_ip_address
32	0	_sm	_subnet_mask
33	0	_gi	_gateway_ip
34	1~6	_mac1~6	_mac_address1~6
36	0	_cm	_connection_mode
37	0	_cip	_connection_ip
38	0	_cpt	_connection_port
56	1~32	_uv1~32	_user_variable1~32
57	1~32	_tv1~32	_temp_variable1~32
60	0	_ndi	_num_dis
63	0	_di	_di_all_values
65	0	_ndo	_num_dos
68	0	_do	_do_all_values
70	0	_nai	_num_ais
75	0	_npi	_num_pis
80	0	_nm	_num_motors
101	0 or 1	_co	_command
102	0 or 1	_s	_status
103	0 or 1	_f	_fault
111	0 or 1	_gp	_go_position
112	0 or 1	_gv	_go_velocity
121	0 or 1	_tp	_temperature
124	0 or 1	_v	_velocity
125	0 or 1	_p	_position
141	0 or 1	_np	_min_position
142	0 or 1	_xp	_max_position
143	0 or 1	_hp	_home_position
146	0 or 1	_usl	_use_soft_limit
147	0 or 1	_hm	_homing_method
148	0 or 1	_hv	_homing_velocity
153	0 or 1	_xv	_max_velocity
154	0 or 1	_ac	_acceleration
155	0 or 1	_dc	_deceleration
161	0 or 1	_ohl	_overheat_limit
165	0 or 1	_ovl	_overvoltage_limit
166	0 or 1	_uvl	_undervoltage_limit
174	0 or 1	_dir	_direction
179	0 or 1	_icd	_idle_current_delay
181	0 or 1	_idc	_idle_current
182	0 or 1	_ruc	_run_current

183	0 or 1	_exm	_excitation_mode
184	0 or 1	_der	_decay_ratio
200	1~N	_dif1~N	_di_flags1~N
201	1~N	_div1~N	_di_value1~N
202	1~N	_dim1~N	_di_mapping1~N
210	1~M	_dof1~M	_do_flags1~M
211	1~M	_dov1~M	_do_value1~M
212	1~M	_dom1~M	_do_mapping1~M
220	1~O	_aif1~O	_ai_flags1~O
221	1~O	_aiv1~O	_ai_value1~O
222	1~O	_aicv1~O	_ai_converted_value1~O
223	1~O	_ail1~O	_ai_linearity1~O
224	1~O	_aim1~O	_ai_mapping1~O
225	1~O	_ain1~O	_ai_input_min1~O
226	1~O	_aic1~O	_ai_input_center1~O
227	1~O	_aix1~O	_ai_input_max1~O
228	1~O	_aidb1~O	_ai_input_deadband1~O
230	1~ P	_pif1~ P	_pi_flags1~ P
231	1~ P	_piv1~ P	_pi_value1~ P
232	1~ P	_picv1~ P	_pi_converted_value1~ P
233	1~ P	_pit1~ P	_pi_capture_type1~ P
234	1~ P	_pil1~ P	_pi_linearity1~ P
235	1~ P	_pim1~ P	_pi_mapping1~ P
236	1~ P	_pin1~ P	_pi_input_min1~ P
237	1~ P	_pic1~ P	_pi_input_center1~ P
238	1~ P	_pix1~ P	_pi_input_max1~ P
239	1~ P	_pidb1~ P	_pi_input_deadband1~ P

* N – Digital Input 체널의 수, M – Digital Output 체널의 수,

* O – Analog Input 체널의 수, P – Pulse Input 체널의 수

11.6.4 자료형 변환

스크립트에서 사용되는 값들에 대해 필요에 따라 서로 형변환이 가능하도록 합니다. 스크립트에서는 내부적으로 형변환은 자동으로 수행하는 묵시적 형변환(Implicit conversion)을 사용합니다.

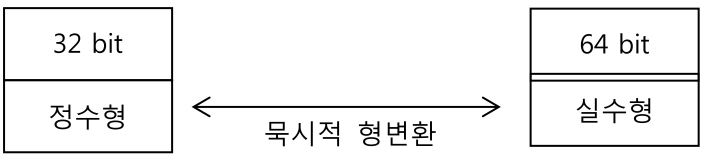


그림 11-2 스크립트에서의 자료 형변환

C언어에서는 데이터의 손실이 없이 안정적으로 형변환이 가능할 경우에만 내부적으로 자동으로 묵시적 형변환을 수행하고, 데이터의 손실을 야기하는 소지가 있는 경우에는 자동으로 형변환하는 것을 방지합니다. 하지만 스크립트에서는 언어를 간단히 하기 위해 실수형에서 정수형으로의 변환도 묵시적으로 이루어집니다. 실수형에서 정수형으로의 형변환 될 때는 실수형 숫자에 가장 가까운 정수형 숫자를 선택하여 변환 됩니다.

다음 예는 실수에서 정수로 묵시적 형변환이 발생하는 경우입니다. 나머지 연산자(%)는 피연산자로 정수형을 취하기 때문에, 수식 내의 실수형 리터럴은 정수형으로 묵시적 형변환되어 연산됩니다.

```
a = 10 % 3;          // 결과는 1
b = 10.33 % 3;       // 10%3과 동일
```

실수에서 정수로 명시적 형변환이 필요한 경우에는 내장함수 `int()`를 사용할 수 있습니다. 다음 예를 참고하십시오.

```
a = int(1.44);       // a는 1
b = int(1.55);       // b는 2
```

11.7 연산자

연산자는 피연산자와의 조합으로 수식을 만들어냅니다. 스크립트 언어에서는 C언어에서 지원하는 모든 수학 연산자를 사용할 수 있습니다.

11.7.1 산술, 부호 연산자

스크립트 언어에서는 산술 연산을 표현할 수 있도록 산술 연산자를 제공합니다. 더하기는 '+', 빼기는 '-', 곱하기는 '*', 나누기는 '/', 나머지 연산은 '%' 기호를 사용하여 각각의 연산을 표현합니다. '%' 연산자의 피연산자로 실수형을 취할 수 없는데, 이때는 실수형을 정수형으로 묵시적 형변환 한 후 연산이 수행됩니다.

수식에 대한 부호를 나타내기 위해 '+' 및 '-' 등과 같은 부호 연산자를 사용합니다. 부호 연산자는 피연산자를 하나만 취하는 단항 연산자이고, 산술연산자는 두 개의 피연산자를 취하는 이항 연산자입니다.

산술 연산자와 부호 연산자에 대해 정리하면, 다음과 같습니다.

분류	연산자	연산식	예제		설명
			연산식	결과	

이항 연산자	+	a + b	7 + 5	13	a와 b를 더하기
	-	a - b	7 – 5	2	a에서 b를 빼기
	*	a * b	7 * 5	35	a와 b를 곱하기
	/	a / b	7 / 5	1	a를 b로 나누기
	%	a % b	7 % 5	2	a를 b로 나눈 나머지
단항 연산자	+	+a	+7	7	a가 양의 수임을 나타냄
	-	-a	-7	-7	a의 부호를 바꿈

11.7.2 증감 연산자

증감 연산자는 변수의 값을 1씩 증가 또는 감소시킨 후, 변화된 값을 다시 그 변수에 저장하는 연산자이고, 하나의 피연산자를 취하는 단항 연산자입니다. 증가 연산자는 1씩 증가시키고, 감소 연산자는 1씩 감소시킨다는 것 외에 동일합니다.

이러한 증감 연산자는 사용되는 위치에 따라 전위형과 후위형 두 가지로 나눌 수 있습니다. 전위형으로 나타내면 변수의 값에 대해 먼저 증감 연산을 수행한 후, 변화된 변수의 값을 참조하여 그 변수가 포함된 연산식에 적용됩니다. 후위형으로 나타내면 변수의 값을 참조하여 연산식에 먼저 적용을 한 후, 변수의 값에 대해 증감 연산을 수행합니다.

증감 연산자에 대해 정리하면 다음과 같습니다.

분류	연산자	연산식	예제	설명
단항 연산자	++	a++	num++	a값을 참조 후 1증가
		++a	++num	a값을 1증가 후 참조
	--	a--	num--	a값을 참조 후 1감소
		--a	--num	a값을 1감소 후 참조

증감 연산자의 사용시 다음 사항에 주의해야 합니다. 한 변수가 수식 내에 두 번 이상 사용될 경우 증감 연산자를 사용하지 않는 것이 바람직합니다. 물론, 값의 변화를 모두 정확히 추정하여 미리 계산하고 나서 사용한다면 상관 없겠지만, 그렇지 않은 경우에는 사용자의 의도와는 다른 결과를 야기할 수 있으므로 조심해야 합니다.

11.7.3 관계 연산자

관계 연산자는 관계를 따져보기 위한 연산자입니다. 이러한 관계 연산자는 두 개의 피연산자를 필요로 합니다. 두 개의 피연산자에 대해 관계 연산자가 의미하는 관계를 따져보고, 그 결과는 0 또는 1 값을 가집니다. 따라서 이러한 관계 연산자로 표현된 조건식을 'boolean-식'이라 할 수 있습니다.

스크립트 언어에서 다음과 같은 관계 연산자들을 제공합니다.

연산자	연산식	예제		설명
		연산식	결과	
>	a > b	3 > 7	0	a가 b보다 크면 참
>=	a >= b	3 >= 7	0	a가 b보다 크거나 같으면 참
<	a < b	3 < 7	1	a가 b보다 작으면 참
<=	a <= b	3 <= 7	1	a가 b보다 작거나 같으면 참
==	a == b	3 == 7	0	a와 b가 같으면 참
!=	a != b	3 != 7	1	a와 b가 다르면 참

11.7.4 논리 연산자

논리 연산자는 논리값에 대한 논리적인 연산을 수행하도록 해 주는 연산자를 말하고, 이러한 논리 연산자를 사용하는 식을 논리식이라 합니다. 논리 연산자는 TRUE(1) 또는 FALSE(0)와 같은 논리값을 사용하여 논리 연산을 수행하고, 그 결과 역시 마찬가지로 TRUE 또는 FALSE의 논리값이 됩니다. 이러한 논리식 역시 조건식과 마찬가지로 boolean-식이라 할 수 있습니다.

스크립트 언어에서 사용 가능한 논리 연산들은 다음과 같습니다.

<table>
<tr><th rowspan="2">연산자</th><th rowspan="2">연산식</th><th colspan="2">예제</th><th rowspan="2">설명</th></tr>
<tr><th>연산식</th><th>결과</th></tr>
<tr><td>!</td><td>!a</td><td>!1</td><td>0</td><td>a가 거짓(false; 0)이면 참(true; 1)</td></tr>
<tr><td>&&</td><td>a && b</td><td>0&&1</td><td>0</td><td>a와 b가 모두 참이면 참</td></tr>
<tr><td>||</td><td>a || b</td><td>1||0</td><td>1</td><td>a가나 b 둘 중 하나라도 참이면 참</td></tr>
</table>

논리합 연산자('&&')에서 a가거짓이면 b를 평가하지 않습니다. 그리고 논리곱 연산자('||')에서 a가 참이면 b를 평가하지 않습니다.

11.7.5 비트 연산자

비트 연산자는 비트 단위의 연산을 수행합니다. 피연산자로 실수형을 제외한 정수형 데이터를 취합니다. 이러한 비트 연산자에는 1의 보수 연산, AND, OR, XOR 등과 같은 비트 연산과 왼쪽/오른쪽 비트 이동을 위한 연산자가 있습니다:

- 1의 보수 연산자('~')는 각 비트에 대해 0은 1로 1은 0으로 바꿈
- AND 연산자('&')는 주로 데이터의 특정 비트를 0으로 만들기(mask off) 위해 사용되고, 두 개의 피연산자에 대해 대응하는 두 비트 중 하나라도 0이면 결과도 0이 됨
- OR 연산자('|')는 주로 어떤 데이터의 특정 비트를 1로 만들기(mask on) 위해서 사용하며, 이때 두 개의 피연산자에 대해 대응하는 두 비트 중 어느 하나라도 1이면 결과 비트는 1이 됨

- XOR 연산자('^')는 두 개의 피연산자에 대해 대응하는 두 비트가 서로 다르면 결과 비트가 1이 됨
- 쉬프트 연산자('<<', '>>')는 주어진 자릿수 만큼 왼쪽 또는 오른쪽으로 비트열을 쉬프트 함

이러한 비트 연산자를 정리하면 다음과 같습니다.

연산자	연산식	예제		설명
		연산식	결과	
>>	a>>b	10001100 >> 1	11000110	a를 b만큼우측으로 비트 이동
<<	a<<b	10110000 << 1	01100000	a를 b만큼좌측으로 비트 이동
&	a&b	11000110 & 00001111	00000110	a와 b의 비트 단위의 논리곱
|	a|b	11000110 | 00001111	11001111	a와 b의 비트 단위의 논리합
^	a^b	10111011 ^ 10101000	00010011	a와 b의 비트단위의 XOR (배타적 논리합)
~	~a	~10110110	01001001	a에 대하여 비트 단위의 보수

'<<' 연산자를 쓸 경우 쉬프트 연산의 결과로 오른쪽에 생긴 빈자리 부분은 0으로 채웁니다.

'>>' 연산자를 사용하여 쉬프트 연산을 수행한 후 왼쪽에 생긴 빈자리에는 MSB 즉, 최상위 비트인 부호비트가 복사됩니다.

11.7.6 대입 연산자

대입연산자는 우변에 있는 수식의 값을 좌변이 가리키고 있는 메모리의 위치에 저장(대입)하기 위해 사용하는 연산자입니다. 이때, 좌변에 오는 것을 좌변값(l-value)이라고 하는데, 좌변값에 올 수 있는 것으로는 변수와 같이 메모리에 저장 공간을 가지는 것들만 가능합니다. 스크립트 언어에서의 '='은 수학에서의 "같음"의 의미가 아닌 "**대입**"의 의미를 가집니다.

대입을 허용하는 경우와 그렇지 않은 경우에 대해 다음 예제를 참고하시기 바랍니다.

```
a + 1 = b;          // 허용하지 않음, 컴파일시 에러 발생
2 = b;              // 허용하지 않음, 컴파일시 에러 발생
a = 100;            // 100을 변수 a에 대입
b = a + 20;         // 변수 a에 20을 더하여 변수 b에 대입
a = b = c = 100;    // 100을 a, b, c에 대입
```

스크립트 언어에서 사용 가능한 대입식과 단축 대입식은 다음과 같습니다.

연산자	연산식	예제	설명
=	a = b	n = 1;	일반 대입식

<table>
<tr><td>+=</td><td>a += b</td><td>n = n + 1; → n += 1;</td><td>a = a + b</td></tr>
<tr><td>-=</td><td>a -= b</td><td>n = n - 1; → n -= 1;</td><td>a = a – b</td></tr>
<tr><td>*=</td><td>a *= b</td><td>n = n * 1; → n *= 1;</td><td>a = a * b</td></tr>
<tr><td>/=</td><td>a /= b</td><td>n = n / 1; → n /= 1;</td><td>a = a / b</td></tr>
<tr><td>%=</td><td>a %= b</td><td>n = n % 1; → n %= 1;</td><td>a = a % b</td></tr>
<tr><td>&=</td><td>a &= b</td><td>n = n & 1; → n &= 1;</td><td>a = a & b</td></tr>
<tr><td>|=</td><td>a |= b</td><td>n = n | 1; → n |= 1;</td><td>a = a | b</td></tr>
<tr><td>^=</td><td>a ^= b</td><td>n = n ^ 1; → n ^= 1;</td><td>a = a ^ b</td></tr>
<tr><td><<=</td><td>a <<= b</td><td>n = n << 1; → n <<= 1;</td><td>a = a << b</td></tr>
<tr><td>>>=</td><td>a >>= b</td><td>n = n >> 1; → n >>= 1;</td><td>a = a >> b</td></tr>
</table>

11.7.7 연산자 우선순위

연산자 우선순위는 서로 다른 연산자들 사이의 수행 순서를 결정합니다. 예를 들자면, "a+b*c"와 같은 수식에서 '*' 연산을 '+' 연산보다 먼저 수행하는 것입니다.

연산자 결합성은 같은 우선 순위의 연산자가 두 개 이상 연속적으로 나올 경우, 이들 간의 우선 순위를 결정합니다. 예를 들자면, "a+b+c"와 같이 '+' 연산자와 '+' 연산자가 순서대로 두 개 이상 나올 경우 왼쪽에 있는 "a+b" 연산을 "b+c" 연산보다 먼저 수행합니다.

다음 표는 스크립트 언어에서 사용하는 연산자들에 대해 연산자 우선순위와 연산자 결합성을 보여주고 있습니다.

<table>
<tr><th>우선순위</th><th>연산자</th><th>결합법칙</th></tr>
<tr><td>기본연산자</td><td>[ ] . ( ) ++ --</td><td>←</td></tr>
<tr><td>전치</td><td>+ - ++ --~!</td><td>→</td></tr>
<tr><td>승제</td><td>* / %</td><td>→</td></tr>
<tr><td>가감</td><td>+ -</td><td>→</td></tr>
<tr><td>비트 이동</td><td><< >></td><td>→</td></tr>
<tr><td>관계</td><td>< <= > >=</td><td>→</td></tr>
<tr><td>등식</td><td>== !=</td><td>→</td></tr>
<tr><td>비트 AND</td><td>&</td><td>→</td></tr>
<tr><td>비트 XOR</td><td>^</td><td>→</td></tr>
<tr><td>비트 OR</td><td>|</td><td>→</td></tr>
<tr><td>논리곱</td><td>&&</td><td>→</td></tr>
<tr><td>논리합</td><td>||</td><td>→</td></tr>
<tr><td>대입</td><td>= *= /= %= += -= <<= >>= >>>= &= ^= |=</td><td>←</td></tr>
</table>

스크립트에서 정의하고 있는 연산자 우선순위와 연산자 결합성은 C언어에서 정의하고 있는 것과 동일하며, 스크립트에서는 포인터를 사용하지 않으므로 C언어에서 포인터 연산을 위해 사용하는 구조체 포인터의 항목 참조 연산자(->), 주소 연산자(&), 간접연산자(*) 등은 제공되지 않습니다.

11.8 제어문

스크립트 프로그램은 나열된 문장을 순차적으로 실행합니다. 이렇게 순차적으로만 수행하는 것은 작업이 매우 단순할 때지만, 특정 작업을 반복적으로 수행해야 할 경우에는 매우 비효율적인 프로그래밍이 됩니다. 그래서 좀더 효과적인 문장 표현을 위해 다음과 같이 세 가지 형태의 제어문을 제공합니다.

첫 번째, 조건문은 특정 조건에 대해 그 조건이 만족하면 해당 문장 또는 블록을 실행합니다. 스크립트 언어에서는 조건문으로 `if`문, `if-else`문을 제공합니다.

두 번째, 반복문은 어떤 문장 또는 블록을 반복 실행할 수 있도록 합니다. 스크립트 언어에서 사용 가능한 반복문은 `while`문, `for`문, `do-while`문 세 가지입니다.

세 번째, 분기문은 문장을 순서대로 실행해 나가다가 프로그램의 실행흐름을 특정 위치로 옮기고자 할 때 사용합니다. 이를 위해 스크립트에서는 `break` 문, `continue` 문, `goto` 문을 제공합니다.

11.8.1 if 문

`if`문은 가장 간단한 형태의 조건문으로 조건식이 만족할 경우에만 다음 문장을 실행하고자 할 때 사용합니다. 조건식이 참이면 `if`문 다음의 문장을 실행하고, 거짓이면 `if`문 다음의 문장을 실행하지 않고 지나갑니다.

If 문
if (조건식**)** 문장 또는 블록

다음은 입력이 90 이상일 때 해당 문장이 실행되도록 하는 예입니다.

```
if (input >= 90) {
  …
}
```

11.8.2 if-else 문

`if-else` 문은 조건식이 참 혹은 거짓에 따라 문장을 양자택일할 경우 사용합니다. 조건식이 참

이면 if문 다음의 문장만을 실행하고, 거짓이면 else문 다음의 문장만을 실행합니다. 여기서 else 문은 if문에 종속적입니다. if문 없이 else 만 독립적으로 사용될 수는 없습니다.

If-else 문
if (조건식) 문장 또는 블록 **else** 문장 또는 블록

다음은 홀수와 짝수를 구분하여 동작을 결정하는 예입니다.

```
if ((n % 2) == 0) {
   …
} else {
   …
}
```

스크립트 언어에서는 if 문과 else 문을 짝짓기 위한 규칙이 있습니다. 가장 가까운 if와 else를 순서대로 짝지어 주는 것입니다. 이러한 혼동을 피하기 위하여 '{'와 '}'를 이용하여 if-else 문을 올바르게 묶어 주어야 합니다.

If-else 문

if (조건식**)** {
   if (조건식**)** 문장 또는 블록
   else 문장 또는 블록
}

11.8.3 if-else-if-else 문

다중택일을 위해서 if-else 문을 연결하여 if-else-if-else 과 같이 사용합니다.

If-else-if-else 문
if (조건식) 문장 또는 블록 **else if (조건식)** 문장 또는 블록 **else if (조건식)** 문장 또는 블록 …… **else** 문장 또는 블록

다음은 입력이 90 이상이면 A기능 실행, 80에서 89 사이이면 B기능 실행, 70에서 79 사이이면 C기능 실행, 60에서 69 사이이면 D기능 실행, 나머지는 E기능 실행하는 예입니다.

```
if(input >= 90) {
   // A statement
} else if((input >= 80)&&(input < 90)) {
   // B statement
} else if((input >= 70)&&(input < 80)) {
   // C statement
} else if((input >= 60)&&(input < 70)) {
   // D statement
} else {
   // E statement
}
```

11.8.4 while 문

while 문은 조건식이 참일 동안 문장 또는 블록을 반복적으로 실행하게 되고, 반복 실행 중에 조건식이 거짓으로 바뀌게 되면 반복 실행을 중단합니다.

while 문
while (조건식) 문장 또는 블록

다음은 1에서 100까지 더하는 연산을 while문으로 작성한 예입니다.

```
i = 1;
sum = 0;
while (i <= 100) {
       sum += i;
       i++;
}
```

11.8.5 for 문

반복문을 실행할 때, 다음과 같이 세 단계로 나누어 실행을 해야 할 경우에는 for 문을 사용하는 것이 유리합니다: 1)처음 시작할 때 하는 초기화 과정, 2)반복 여부를 검사하기 위한 조건식 검사, 3)반복할 때마다 실행할 문장 또는 블록.

for 문은 위에 나타난 세 가지 과정을 하나의 문장으로 표현 가능하도록 함으로, 반복 실행을 위한 문장을 작성하는데 드는 노력을 줄여줍니다.

for 문
for (초기화수식; 조건식; 증감수식) 문장 또는 블록

for 문의 초기화 수식에는 반복 실행을 하는데 필요한 초기화 문장들을 나열합니다. 이때 각 문장들은 ','로 구분합니다. 이 초기화수식은 필요에 따라 생략 가능합니다.

조건식은 문장 또는 블록의 반복 여부를 판단합니다. 이 조건식이 만족할 동안 문장 또는 블록이 반복 실행됩니다. 조건식은 생략될 수 없습니다.

증감 수식에는 주로 증감 연산식이 사용됩니다.

다음은 1에서 100까지 더하는 연산을 for문으로 작성한 예입니다.

```
sum = 0;
for (i=1; i <= 100; i++) {
        sum += i;
}
```

11.8.6 do-while 문

do-while 문은 조건식이 참일 동안 문장 또는 블록을 실행합니다. while 문에서는 반복 실행의 앞부분에서 조건식을 검사했지만, do-while 문에서는 문장 또는 블록을 실행한 후 조건식을 검사합니다. 그래서 조건식이 처음부터 거짓일 경우 연결된 문장 또는 블록을 아예 실행하지 않게 됩니다. 그러나 do-while 문은 문장 또는 블록을 먼저 실행 한 후 조건식을 검사하기 때문에 적어도 한번은 do-while 문에 연결된 문장 또는 블록을 실행을 하게 됩니다.

do-while 문
do {   문장 또는 블록 } while (조건식);

다음은 1에서 100까지 더하는 연산을 do-while문으로 작성한 예입니다.

```
i = 1;
sum = 0;
do {
        sum += i;
        i++;
} while (i <= 100);
```

11.8.7 break 문

break 문은 항상 for, while, do-while 문과 같이 쓰이며, 현재 실행중인 위치에서 break 문을 만나게 되면 프로그램의 실행이 반복문의 밖으로 빠져나가게 됩니다.

break문을 for, while, do-while 문과 상관 없이 사용하면 에러가 발생합니다.

break 문

while (i<10) {
    …
    break;
    …
}

다음은 1에서 100까지 더하는 연산을 while문과 break문으로 작성한 예입니다.

```
i = 1;
sum = 0;
while (1) {
       if (i <= 100) sum += i;
       else break;
       i++;
}
```

11.8.8 continue 문

continue 문도 break 문처럼 항상 for, while, do-while 문과 같이 쓰이며, 현재 실행중인 위치에서 continue 문을 만나게 되면 프로그램의 실행이 반복문의 다음 시작 위치로 이동하게 됩니다.

continue 문이 break 문과 다른 점은 제어흐름이 반복문의 밖으로 이동하는 것이 아니라 반복문의 시작 위치로 이동하는 것입니다.

continue 문

while (i<10) {
    …
    continue;
    …
}

다음은 1에서 100까지 더하는 연산을 do-while문과 continue문으로 작성한 예입니다.

```
i = 1;
sum = 0;
do {
       sum += i;
       i++;
```

```
        if (i <= 100) continue;
} while (0);
```

11.8.9 goto 문

현재 실행중인 위치에서 특정 위치로 이동하기 위하여 goto 문을 사용합니다. 스크립트에서 레이블을 지정하는 방법은 "레이블이름:"과 같이 합니다. 그리고 하나의 프로그램 내에서 사용된 레이블은 각각 구분 가능해야 합니다.

goto 문
레이블: ... goto 레이블;

다음은 1에서 100까지 더하는 연산을 goto문으로 작성한 예입니다.

```
i = 1;
sum = 0;

loop:
sum += i;
i++;
if (i <= 100) goto loop;
```

11.9 내장 함수

내장 함수는 스크립트 언어로 프로그램을 작성할 때 호출 가능한 미리 정의된 함수들입니다.

스크립트 언어에서는 다음과 같은 내장 함수들을 제공합니다:

- clock() - 현재 시간을 밀리초 단위로 반환
- rand() - 0과 32767 사이의 의사-난수를 반환
- sleep(x) - x 밀리초 동안 스크립트의 실행을 대기
- getv(index,sub_index) - index와 sub_index로 지정된 오브젝트의 값을 읽어옴
- setv(index,sub_index,x) - index와 sub_index로 지정된 오브젝트에 값을 기록함
- int(x) - x를 소숫점 첫째 자리에서 반올림하여 정수를 반환
- sin(x) - 라디안으로 주어진 x의 sine 값을 반환
- cos(x) - 라디안으로 주어진 x의 cosine 값을 반환
- tan(x) - 라디안으로 주어진 x의 tangent 값을 반환
- asin(x) - x(sine x의 결과 값)의 arc sine의 값을 반환

- acos(x) - x(cosine x의 결과 값)의 arc cosine의 값을 반환
- atan(x) - x(tangent x 의 결과 값)의 arc tangent의 값을 반환
- sinh(x) - 수학적으로 exp(x) - exp(-x)/2 로 정의된, x의 쌍곡선 sine을 반환
- cosh(x) - 수학적으로 exp(x)-exp(-x)/2 로 정의된, x의 쌍곡선 cosine을 반환
- tanh(x) - sinh(x)/cosh(x)이라는 수학적 정의를 가진, x 의 쌍곡선 tangent x를 반환
- fabs(x) - x의 절대값을 반환
- floor(x) - x보다 작거나 같은 정수 중에서 최대 값을 반환
- ceil(x) - x보다 크거나 같은 정수 중에서 최소 값을 반환
- sqrt(x) - x의 음이 아닌 루트 값을 반환
- exp(x) - 자연대수 e의 x 승 값을 반환
- log(x) - x의 자연로그를 반환
- log10(x) - 10을 밑으로 하는 x의 로그 값을 반환
- atan2(x,y) - 두 개의 인수를 가진 arc tangent 함수
- pow(x,y) - x의 y승을 반환하는 일반적 지수 함수
- min(x,y) - 주어진 두 인자의 최소값을 반환
- max(x,y) - 주어진 두 인자의 최대값을 반환

11.9.1 clock

Declaration: clock()

제어기의 현재 시각을 밀리초 단위로 읽어 옵니다. 시각은 제어기가 시작된 후 밀리초 단위로 카운트 되는 시각입니다.

Examples:

```
// 코드 블록의 실행시간 측정
t1 = clock ();
{ ... }
t2 = clock ();
dt = t2 - t1;
```

11.9.2 rand

Declaration: rand()

0과 32767 사이의 의사-난수를 반환합니다.

Remarks: 제어기가 시작된 후 매번 난수를 다르게 발생시키기 위하여 시드(seed) 값을 설정합니다. 시드 값은 제어기의 특정 아날로그 입력 포트의 값을 읽어 지정하게 됩니다.

Examples:

```
// 0에서 99 사이의 난수 발생
a = rand()%100;
```

11.9.3 sleep

Declaration: sleep(x)

스크립트 프로그램의 실행을 x 밀리초 동안 대기합니다.

Remarks: 스크립트 프로그램의 실행이 `sleep()` 함수에 의해 중단된 상황에서도 제어기의 다른 모든 기능은 정상적으로 수행됩니다.

Examples:

```
// 채널 1번의 모터에 1초 마다 위치 -5000과 5000 사이를 이동합니다.
while (1) {
       _go_position = -5000;
       sleep (1000);
       _go_position = 50000;
       sleep (1000);
}
```

11.9.4 getv

Declaration: getv (index, sub_index)

인수 index와 sub_index로 지정된 제어기의 오브젝트 값을 읽어옵니다. `getv()` 함수로 제어기의 모든 오브젝트 값을 읽을 수 있습니다.

index는 제어기 오브젝트명의 참조 값이고 sub_index는 I/O나 모터의 채널 번호입니다.

Remarks: 유효하지 않은 index와 sub_index에 대해 읽기를 시도하는 경우, `getv()` 함수는 아무런 경고 없이 0을 돌려줍니다. 읽은 값이 0일 때, 실제 오브젝트의 값이 0인지 혹은 유효하지 않은 오브젝트를 엑세스 하였는지 알 수 없기 때문에 사용자는 `getv()` 함수로 넘기는 index와 sub_index가 유효한지 미리 판단하여야 합니다.

Examples:

```
// 모터의 현재 위치를 읽어옴
a = getv (125, 0);        // a = _position; 과 동일
```

11.9.5 setv

Declaration: setv (index, sub_index, x)

인수 index와 sub_index로 지정된 제어기의 오브젝트에 x를 씁니다. `setv()` 함수로 제어기의 모든 오브젝트에 값을 변경할 수 있습니다. 변경된 값은 제어기에 즉시 적용됩니다. 하지만 변경된 값이 제어기의 플래시 메모리에 저장되지는 않습니다.

index와 sub_index에 대한 설명은 `getv()` 함수의 설명을 참조하십시오.

Remarks: `setv()` 함수는 오브젝트에 쓰는 값 x가 유효한지 검사하지 않습니다. 사용자는 오브젝트의 값을 변경할 때 주의하여야 합니다.

Examples:

```
// 모터에 5000 위치로 이동 명령을 내림
setv (111, 0, 5000);        // _go_position = 50000; 과 동일
```

11.9.6 int

Declaration: int(x)

주어진 실수에서 가장 가까운 정수를 반환합니다.

인수 x에는 모든 값을 사용할 수 있습니다.

반환 값은 정수입니다.

Examples:

```
a = int(1.4);               // a는 1
b = int(1.6);               // b는 2
```

11.9.7 sin

Declaration: sin(x)

라디안 각 x의 sine 값을 계산합니다.

인수 x에는 모든 값을 사용할 수 있습니다.

반환 값은 -1과 1 사이의 값입니다.

Examples:

```
// 크기가 5000[pulse/s]인 정현파 속도를 가해 모터를 구동
for (i=0; i<100000000; i++) {
      v = 5000*sin(i*0.01);
      _go_velocity = v;
      sleep (10);
}
```

11.9.8 cos

Declaration: cos(x)

라디안 각 x의 cosine 값을 계산합니다.

인수 x에는 모든 값을 사용할 수 있습니다.

반환 값은 -1과 1 사이의 값입니다.

11.9.9 tan

Declaration: tan(x)

라디안 각 x 의 tangent 값을 계산합니다.

인수 x에는 모든 값을 사용할 수 있습니다.

반환 값은 모든 값이 될 수 있습니다.

11.9.10 asin

Declaration: asin(x)

x의 arc sine 값을 계산합니다. `sin()` 함수의 역함수입니다.

인수 x의 값은 -1과 1 사이의 값입니다.

반환 값은 -π/2와 π/2사이의 라디안 각입니다.

11.9.11 acos

Declaration: acos(x)

x의 arc cosine 값을 계산합니다. 이 함수는 `cos()` 함수의 역함수입니다.

인수 x의 값은 -1과 1 사이의 값입니다.

반환 값은 0과 π사이의 라디안 각입니다.

11.9.12 atan

Declaration: atan(x)

x의 arc tangent 값을 계산합니다. 이 함수는 `tan()` 함수의 역함수입니다.

인수 x에는 모든 값을 사용할 수 있습니다.

반환 값은 -π/2와 π/2사이의 라디안 각입니다.

11.9.13 sinh

Declaration: sinh(x)

x의 쌍곡선 sine을 계산합니다.

인수 x에는 모든 값을 사용할 수 있습니다.

반환 값은 모든 값이 될 수 있습니다.

11.9.14 cosh

Declaration: cosh(x)

x의 쌍곡선 cosine을 계산합니다.

인수 x에는 모든 값을 사용할 수 있습니다.

반환 값은 모든 값이 될 수 있습니다.

11.9.15 tanh

Declaration: tanh(x)

x 의 쌍곡선 tangent를 계산합니다.

인수 x에는 모든 값을 사용할 수 있습니다.

반환 값은 모든 값이 될 수 있습니다.

11.9.16 fabs

Declaration: fabs(x)

x의 절대값을 계산합니다.

인수 x에는 모든 값을 사용할 수 있습니다.

반환 값은 항상 양수입니다.

Examples:

```
a = fabs(5.5);      // a는 5.5
b = fabs(-5.5);     // b는 5.5
```

11.9.17 floor

Declaration: floor(x)

x보다 작거나 같은 정수 중에서 최대 값을 계산합니다.

인수 x에는 모든 값을 사용할 수 있습니다.

반환 값은 정수입니다.

Examples:

```
a = floor(5.5);     // a는 5
b = floor(-5.5);    // b는 -6
```

11.9.18 ceil

Declaration: ceil(x)

x보다 크거나 같은 정수 중에서 최소 값을 계산합니다.

인수 x에는 모든 값을 사용할 수 있습니다.

반환 값은 정수입니다.

Examples:

```
a = ceil(5.5);      // a는 6
b = ceil(-5.5);     // b는 -5
```

11.9.19 sqrt

Declaration: sqrt(x)

x의 음이 아닌 루트의 값을 계산합니다.

인수 x의 값은 음수가 될 수 없습니다.

반환 값은 항상 양수입니다.

Examples:

```
a = sqrt(4);        // a는 2
```

11.9.20 exp

Declaration:exp(x)

자연상수 e의 x 승 값을 계산합니다.

인수 x에는 모든 값을 사용할 수 있습니다.

반환 값은 모든 값이 될 수 있습니다.

Examples:

```
a = exp(2);         // a는 7.389...
b = pow(M_E, 2);    // b는 7.389...
```

11.9.21 log

Declaration: log(x)

x 의 자연로그를 계산합니다.

인수 x에는 모든 값을 사용할 수 있습니다.

반환 값은 모든 값이 될 수 있습니다.

Examples:

```
a = log(M_E);              // a는 1
b = log(M_E*M_E);          // b는 2
```

11.9.22 log10

Declaration: log10(x)

10을 밑으로 하는 x의 로그를 계산합니다.

인수 x에는 모든 값을 사용할 수 있습니다.

반환 값은 모든 값이 될 수 있습니다.

Examples:

```
a = log10(10);      // a는 1
b = log10(1000);    // b는 3
```

11.9.23 atan2

Declaration: atan2 (y, x)

두 개의 인수 y, x로 arc tangent를 계산합니다. y/x에 대한 arc tangent 연산을 수행하고, y와 x의 부호로 4분원(quadrant)을 결정합니다.

인수 y, x는 0이 될 수 없습니다.

반환 값은 -π와 π사이의 라디안 각입니다.

Examples:

```
a = atan ( 1/  1);  //  1/4*M_PI
b = atan (-1/  1);  // -1/4*M_PI
c = atan ( 1/ -1);  // -1/4*M_PI
d = atan (-1/ -1);  //  1/4*M_PI
e = atan2( 1,  1);  //  1/4*M_PI
f = atan2(-1,  1);  // -1/4*M_PI
g = atan2( 1, -1);  //  3/4*M_PI
h = atan2(-1, -1);  // -3/4*M_PI
```

11.9.24 pow

Declaration: pow (x, y)

x의 y승을 계산합니다.

만일 인수 y가 분수 값이면 인수 x는 음수가 될 수 없습니다. 그리고 y가 0보다 작거나 같으면 x는 0이 될 수 없습니다.

Examples:

```
a = pow( 2,  1.5);  // 2.828...
b = pow(-2,  1.5);  // 계산 불가
c = pow(1.5,  2);   // 2.25
d = pow(1.5, -2);   // 0.444...
```

11.9.25 min

Declaration: min (x, y)

주어진 두 인수 x, y의 최소값을 선택합니다.

인수 x, y에는 모든 값을 사용할 수 있습니다.

11.9.26 max

Declaration: max (x, y)

주어진 두 인수 x, y의 최대값을 선택합니다.

인수 x, y에는 모든 값을 사용할 수 있습니다.

12 프로그램의 작성과 실행

제어기의 강력한 기능 중 하나는 사용자가 프로그램을 작성하여 제어기에 다운로드하고 실행하는 기능입니다. 이 기능은 제어기의 모터 제어 기능에 PC를 결합하는 것과 동일합니다. PC에서 수행하던 일부 혹은 모든 기능을 제어기에서 구현할 수 있기 때문에, 전체 시스템 구성을 단순화 할 수 있습니다.

12.1 프로그램의 작성

Mini-C 스크립트 언어로 프로그램을 작성하는 것은 제어기의 기능을 확장하여 다양한 용도로 사용할 수 있도록 합니다.

Mini-C 스크립트 언어는 C언어와 유사합니다. C언어의 구조는 장비를 제어하는 낮은 수준의 프로그래밍에 유리하고 배우기 쉽습니다. 만일 C언어에 친숙한 사용자라면, **"11.2 C언어와의 차이"**만 살펴보더라도 바로 프로그램을 작성할 수 있습니다.

12.1.1 소스코드 작성

사용자는 Mini-C 스크립트 언어의 문법을 사용하여 제어기에서 실행되는 프로그램을 작성합니다. 프로그램의 소스코드는 텍스트로 작성되며, *.scr 확장자를 가지는 텍스트 파일에 저장하거나 다시 읽어올 수 있습니다.

소스코드 작성에 Motor Control UI 유틸리티를 사용할 수 있습니다. 또한 Windows 운영체제에서 제공하는 notepad와 같은 일반적인 텍스트 편집 유틸리티도 소스코드 작성에 사용할 수 있습니다.

소스코드는 문장의 길이, 들여쓰기 등에 제약 받지 않고 자유로운 형태의 텍스트로 작성하면 됩니다.

12.1.2 소스코드 구조

스크립트 언어로 프로그램을 작성할 때 소스코드의 구조는 다른 종류의 프로그램을 작성하는 것과 비슷합니다. 단순하게, 프로그램이 한 번 실행되고 끝나도록(단일 실행구조) 프로그램을 작성할 수 있습니다. 이와 달리, 반복적으로 실행되도록(반복 실행구조) 프로그램을 작성할 수도 있습니다.

단일 실행구조는 제어기가 시작될 때 혹은 외부의 명령에 의해 정해진 기능을 수행하고 종료되는 형태입니다. 다음 예제와 같이, 제어기가 시작될 때 제어기의 구성 파라미터를 설정하는 것입니다.

```
// 모터에 대하여 최대 속도, 가속도, 감속도 설정
_max_velocity = 3000;
_acceleration = 1500;
_deceleration = 1500;
```

반복 실행구조는 제어기 프로그램을 작성하는 좀 더 일반적인 방법입니다. 프로그램은 아날로그/디지털 입력 또는 모터 상태(전류, 전압, 온도, 속도, …)를 읽어 모터 전원 및 디지털 출력을 조절하는 과정을 무한히 반복하도록 작성될 수 있습니다.

```
// 모터에 대하여 최대 속도, 가속도, 감속도 설정
_max_velocity = 1000;
_acceleration = 500;
_deceleration = 500;

// 아날로그 입력 채널 1의 값을 읽어 모터의 속도를 조절
while (1) {
	speed = _ai_converted_value1;
	_go_velocity = speed * 1000;
	sleep (100);
}
```

반복 실행구조는 보통 `while(1) { ... }` 과 같이 만듭니다. `while` 문의 조건은 항상 참이기 때문에, 이 프로그램은 사용자가 스크립트의 실행을 중단하거나 제어기가 꺼지기 전까지 무한히 반복합니다.

반복 실행되는 블록 끝에는 `sleep()` 함수로 일정 대기 시간을 삽입하는 것이 좋습니다. 불필요하게 프로세서 자원을 사용하지 않으면서 원하는 기능을 달성할 수 있도록, 대기 시간은 필요한 만큼 짧게 설정 합니다. 예를 들어, 배터리를 모니터링하고 배터리의 낮은 전압에 대해 디지털 출력을 트리거 하는 스크립트가 밀리 초 마다 실행될 필요는 없습니다. 100ms의 대기 시간이면 충분하고 제어기가 스크립트의 실행에 불필요한 시간을 할당하는 것을 방지해야 합니다.

12.1.3 스크립트 예제 (다음 적절한 예제 코드를 하나 편집 부탁합니다.)

~~다음 예제 소스코드는 듀얼 채널 모터제어기를 차동 구동형 이동로봇에 적용하여, 사용자의 명령에 따라 로봇을 움직이고 움직인 위치를 추정하는 프로그램입니다.~~

```
/*
This script reads the velocity command from variable _user_value 1, 2
and control velocity of left and right wheels of the mobile robot.
While moving, left and right wheels' encoder values are used to calculate
the mobile robot's position x, y and heading theta.
*/

// get the robot's properties
r = getv (_wheel_radius, 0);
b = getv (_axle_length, 0);
g = getv (_gear_ratio, 0);
```

```
enc = getv (_encoder_ppr, 1);

enc1_p = getv (_position, 1);
enc2_p = getv (_position, 2);

while (1) {
        // reads the user command
        v = getv (_user_value, 1);
        w = getv (_user_value, 2);

        // calculate the velocity of left and right wheels
        rps2rpm = 60/(2*M_PI);

        vl = g/r*(v-w*b/2);
        vr = g/r*(v+w*b/2);

        // set the velocity to left and right wheels' motor
        setv (_velocity, 1, vl*rps2rpm);
        setv (_velocity, 2, vr*rps2rpm);

        // get the encoder value and estimate the position that robot moved
        enc1 = getv (_position, 1);
        enc2 = getv (_position, 2);
        de1 = enc1 - enc1_p;
        de2 = enc2 - enc2_p;
        enc1_p = enc1;
        enc2_p = enc2;

        de1 *= 2*M_PI/enc;
        de2 *= 2*M_PI/enc;

        // delay 10ms
        sleep (10);
}
```

12.2 빌드

사용자가 작성한 소스코드는 제어기에서 직접 실행될 수 없습니다. 따라서 소스코드는 빌드 과정을 거쳐 제어기의 가상머신에서 실행 가능한 바이트코드로 변환되어야 합니다.

일반적으로, 프로그램의 빌드는 컴파일과 링크 과정을 의미합니다. Mini-C 스크립트 언어로 작성된 프로그램은 하나의 소스코드만으로 하나의 프로그램을 만들어 냅니다. 그래서 컴파일러만으로 빌드 과정을 끝낼 수 있으며 링커는 사용하지 않습니다.

빌드는 Motor Control UI 유틸리티 상에서만 가능합니다. 유틸리티의 [Build] 버튼을 눌러 빌드 과정을 진행합니다. 현재 Mini-C 스크립트를 독립적으로 컴파일 하는 컴파일러는 제공되지 않습니다.

12.2.1 컴파일

컴파일러는 스크립트 소스코드를 제어기의 가상머신에 의해 해석되는 바이트코드로 변환(컴파일) 합니다. 소스코드의 양에 따라 다르겠지만, 컴파일 과정은 보통 순식간에 진행됩니다.

컴파일 하는 동안 소스코드의 문법 오류를 검사하고 오류가 발견되면 오류 메시지를 출력합니다. 컴파일 도중 오류가 발생하더라도 컴파일러는 전체 소스코드를 스캔 합니다. 그렇기 때문에, 오류 메시지는 여러 개가 출력될 수 있습니다.

다음과 같이 간단한 예제 소스코드를 컴파일 하면,

```
// File: scr\new.scr
a = 1;
b = 2;
c = a + bb;  // bb가 선언되지 않았다고 컴파일 오류 발생
```

컴파일러는 다음과 같은 오류 메시지를 출력합니다.

```
scr\new.scr(4) : Error : "bb" : undeclared identifier.
>>> scr\new.scr - 1 error(s), build failed
```

사용자는 오류가 발생한 파일의 메시지를 참조하여 소스코드에서 오류를 수정해야 합니다. 컴파일러의 모든 오류 메시지는 **"12.2.2 컴파일 오류 메시지"**를 참조하시기 바랍니다.

소스코드가 오류 없이 컴파일 되면 다음과 같은 메시지를 출력합니다.

```
>>> scr\new.scr - 0 error(s), build succeeded
```

오류 없이 컴파일이 끝난 경우, 어셈블리(확장자 *.asm) 파일과 바이트코드(확장자 *.bin) 파일이 생성됩니다. 바이트코드 파일은 바이트코드를 바이너리 형태로 저장한 파일입니다. 이 파일이 제어기로 다운로드 되고 가상머신에서 읽혀 실행됩니다.

12.2.2 컴파일 오류 메시지

다음 오류 메시지는 사용자가 작성한 소스코드의 컴파일 과정에서 문법 오류에 의해 발생하는 메시지입니다. 이 메시지를 참조하여 소스코드의 오류를 수정할 수 있습니다.

- 'xxx' : undeclared identifier
- 'xxx' : function not found
- 'xxx' : function does not take x arguments
- 'xxx' : left operand must be l-value
- 'xxx' : right operand must be l-value
- label 'xxx' was undefined
- label redefined 'xxx'
- undeclared identifier 'x'
- unexpected end of file found in comment
- missing expression
- missing '(' after 'if'
- missing '(' after 'for'
- missing '(' after 'while'
- missing ')'
- missing '}'
- missing ';'
- missing ';' after 'break'
- missing ';' after 'continue'
- missing ';' after 'goto'
- missing 'while' after 'do'
- missing label after 'goto'
- syntax error 'x'
- syntax error : ')'
- syntax error : ','
- syntax error : 'xxx'
- syntax error : missing 'x'
- syntax error : missing ')'
- syntax error : function call missing ')'
- illegal 'break'
- illegal 'continue'
- illegal 'else' without matching 'if'

12.3 다운로드 및 실행

12.3.1 다운로드

빌드가 성공하면 바이트코드 파일(확장자 *.bin)이 만들어집니다. 바이트코드 파일을 제어기로 다운로드 하는데 Motor Control UI 유틸리티가 사용됩니다. 유틸리티의 [Download] 버튼은 바이너리 파일을 읽어 제어기의 플래시 메모리로 전송합니다. 그리고 플래시 메모리에 저장된 바이트코드는 새로운 파일을 다운로드 하지 않는 한 영구적으로 유지됩니다.

제어기의 플래시 메모리에는 최대 64Kbyte의 바이트코드를 저장할 수 있는 공간이 예약되어 있습니다. 이를 소스코드로 환산하면 약 10만 라인 이상입니다. 이는 사용자가 스크립트로 필요한 기능을 구현하는데 충분한 양이 될 것입니다.

만일 제어기의 가상머신에서 프로그램이 실행 중일 때 새로운 파일을 다운로드 하면, 현재 실행 중인 프로그램은 중단되고 제어기는 새로운 바이트코드를 수신하여 플래시 메모리에 저장하게 됩니다.

※ 제어기에는 하나의 프로그램만 저장되고 실행됩니다. 만일 새로운 파일을 다운로드 하면, 기존의 저장된 프로그램에 덮어쓰게 됩니다.

12.3.2 실행

스크립트 프로그램을 실행하는 방법은 두 가지가 있습니다. 하나는, 제어기 시작 시 자동으로 실행되도록 설정하는 것입니다. 다른 하나는, 수동으로 실행되도록 설정하고 사용자가 제어기에 스크립트의 시작/종료 명령을 보내는 것입니다.

제어기 시작 시 스크립트의 실행 여부를 결정하는 것은 Motor Control UI 유틸리티에서 설정할 수 있습니다. Configuration 탭에서 Script 그룹의 Run Script at Startup 항목을 Enable 혹은 Disable 하는 것입니다.

수동으로 실행되도록 설정되었다면, RS-232, RS-485, Ethernet 포트를 통해 명령을 전송하여 실행합니다. Motor control UI 유틸리티에서 Script 탭의 [Run/Stop] 버튼으로 스크립트 프로그램을 실행하거나 중단할 수 있습니다.

만일 Hyperterminal과 같은 유틸리티가 RS-232 포트를 통해 연결되어 있다면, 다음과 같이 텍스트 기반으로 스크립트의 시작/종료 명령을 내리고 실행 상태를 읽어올 수 있습니다.

```
system_command=8     - 스크립트 시작 명령
system_status        - 스크립트의 실행 상태 읽기
system_command=9     - 스크립트 종료 명령
system_status
```

자세한 내용은 "7.2.2 system_command - System Command"을 참조하기 바랍니다.

스크립트가 실행되면서 가상머신에 할당된 스택과 데이터 메모리를 초과하여 사용할 수가 있습니다. 가상머신은 수행 효율을 위해 이러한 스택과 데이터 메모리의 오버플로우를 검사하지 않습니다. 이러한 상황은 시스템 충돌로 이어져, 실행되는 스크립트를 중지하고 새로운 스크립트를 로드할 수 없게 되거나 PC와 통신할 수 없게 됩니다. 만일 이러한 상황이 발생하면 제어기의 상태를 리셋을 통해서 **"제품 초기 설정 값"** 상태로 되돌려야 합니다.

상기와 같은 상황이 발생하는 것을 방지하기 위해, 제어기 시작 시 스크립트가 수동으로 실행되도록 설정하고 사용자가 작성한 스크립트를 충분히 테스트해야 합니다.

13 Motor Control UI 유틸리티

PC 기반의 Motor Control UI 유틸리티는 무료로 다운로드 하여 사용 가능합니다. 이 프로그램은 직관적인 GUI를 제공하며, 사용자는 버튼 및 슬라이더를 사용하여 제어기를 설정하고 운영합니다.

13.1 소프트웨어 다운로드 및 실행

13.1.1 시스템 요구사항

이 유틸리티를 실행하기 위해서는 다음과 같은 PC 환경이 필요합니다:

- Window XP/7/8 32bit/64bit OS
- 10MByte의 HDD 여유공간
- 1GByte 이상의 RAM
- USB 또는 시리얼 COM(RS-232) 포트
- USB to RS-485 Converter

Motor Control UI 유틸리티는 제품에 동봉되어 있지 않습니다. 이 유틸리티를 다운로드하기 위해서 PC는 인터넷에 연결되어 있어야 합니다.

13.1.2 다운로드

Motor Control UI 유틸리티 프로그램은 디바이스마트 문워커 제품 판매페이지에서 다운로드 받을 수 있습니다:

- 다운로드 경로: 판매페이지 주소

13.1.3 실행

Motor Control UI 유틸리티는 설치 과정이 필요 없습니다. 압축을 푼 폴더 안에 있는 i-Step UI.exe 파일을 실행하면 됩니다. 이 유틸리티의 첫 실행 화면을

그림 13-1에서 보여주고 있습니다.

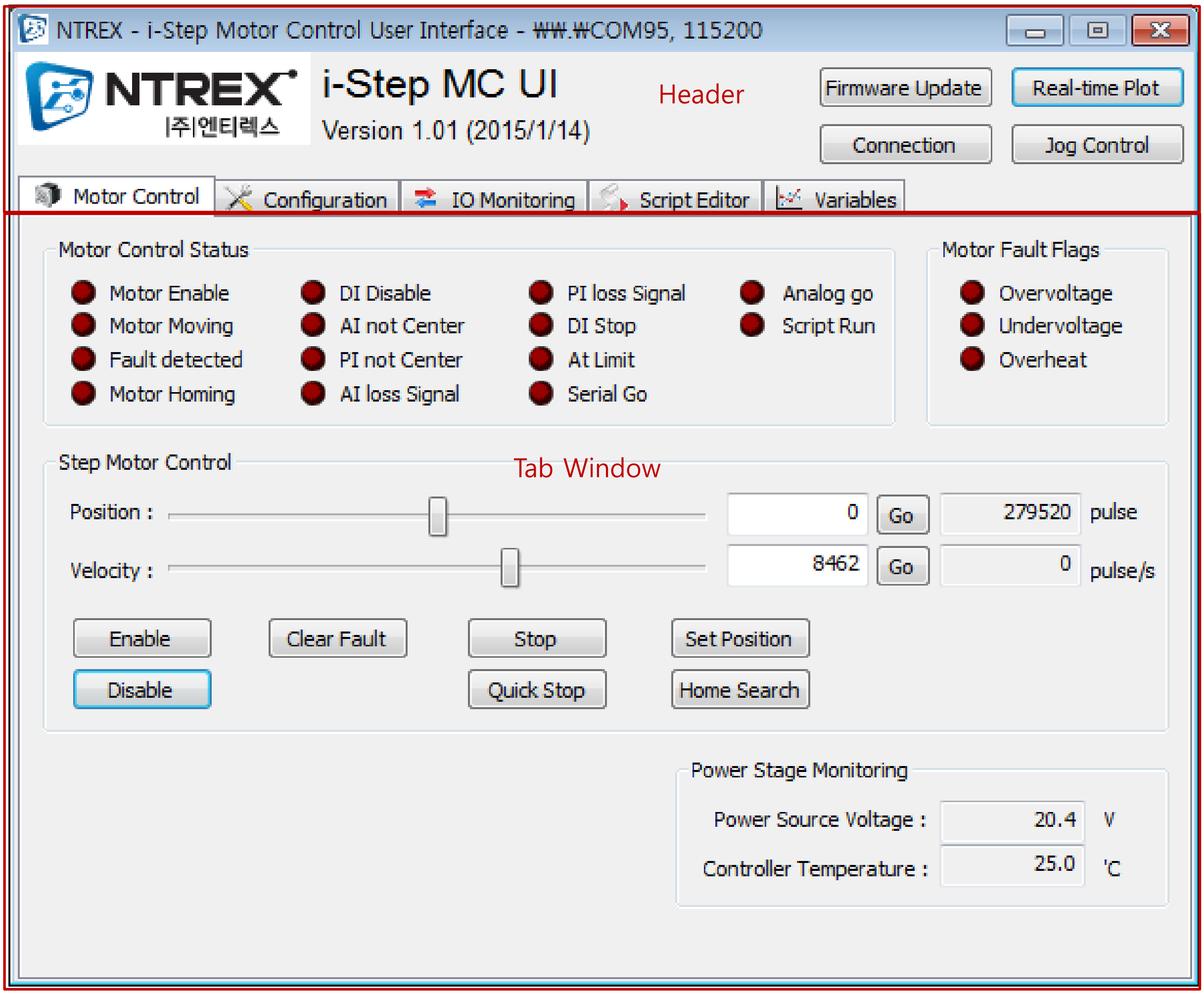


그림 13-1 Motor Control UI 유틸리티의 실행화면

이 유틸리티를 올바르게 사용하려면 하나 이상의 제어기가 PC에 연결되어 있어야 합니다. 그렇지 않으면 유틸리티의 모든 기능이 활성화 되지 않습니다.

※ Motor Control UI 유틸리티는 수시로 업데이트될 수 있습니다. 따라서 사용자는 최신 버전을 확인하고 사용하기 바랍니다.

13.2 메인 화면 구성

Motor Control UI 유틸리티의 메인 화면은 헤더와 탭 윈도우 부분으로 구성됩니다.

13.2.1 헤더

헤더에는 Motor Control UI 유틸리티를 제어기에 연결하기 위한 통신 설정, 장치의 검색, 검색된

장치 목록 표시 등의 기능이 있습니다.

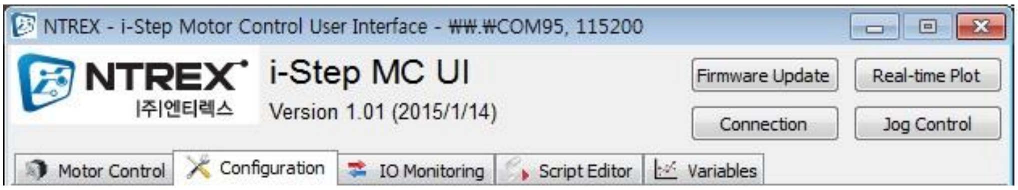


그림 13-2 헤더 파트

헤더의 제일 왼쪽에는 ㈜엔티렉스 상호와 프로그램 이름(Motor Control UI)과 버전이 표시됩니다.

[Connection] 버튼을 누르면 PC에 연결된 제어기를 검색하고 연결하기 위한 설정 대화 상자를 표시합니다.

[Real-time Plot] 버튼은 Real-time Plot 창의 표시를 토글합니다. 현재 창이 표시되지 않는 상태에서 버튼을 누르면 창이 표시되고, 반대로 창이 표시되는 상태에서 버튼을 누르면 창이 표시되지 않습니다.

[Firmware Update] 버튼을 누르면 제어기를 최신 펌웨어로 업데이트하기 위한 설정을 하는 대화 상자를 표시합니다.

[Joy Control] 버튼은 사용자가 수동으로 동작하기 위한 설정 대화 상자를 표시합니다.

13.2.2 탭 윈도우

탭 윈도우는 기본적으로 Motor Control, Configuration, I/O Monitoring, Script Editor, Variables의 5 가지 탭으로 구성되어 있습니다.

- Motor Control Tab - 제어기의 채널 별로 모터 제어 명령 인가 및 상태 모니터링
- Configuration Tab - 모터제어기, 모터, I/O 구성 설정
- I/O Monitoring Tab - 아날로그, 디지털, 펄스 입출력 채널 제어 및 모니터링
- Script Tab - 스크립트 작성과 다운로드 및 실행
- Variables - 스크립트에서 값을 주고받기 위해 사용하는 32개 변수 설정

사용자는 필요에 따라 해당 탭 선택하여 사용하면 됩니다.

13.3 Firmware Update

제어기의 버그를 수정하거나 기능이 향상에 의해 제어기의 펌웨어는 수시로 업데이트 됩니다. 사용자는 펌웨어 업데이트로 제어기에 항상 최신 펌웨어를 설치하고 사용할 수 있습니다.

13.3.1 펌웨어 다운로드

펌웨어를 업데이트하기 위해서는 먼저 디바이스마트 홈페이지에서 최신 펌웨어를 다운로드 받아야 합니다.

다운로드 파일은 zip으로 압축되어 있습니다. 압축 파일을 풀면 **"VSCB24D3S_버전명"** 폴더 안에 모터제어기 모델별로 펌웨어 파일들이 있습니다. 이 중 사용자가 보유한 모터제어기 모델에 맞는 펌웨어를 설치하면 됩니다.

13.3.2 펌웨어 업데이트

Firmware Update 대화상자는 Motor Control UI 유틸리티 헤더의 [Firmware Update] 버튼을 눌러 실행합니다. 만일 현재 제어기와 연결되어 있다면, 그림 13-3과 같이 연결된 제어기의 Device ID, Vender ID, Product ID, H/W Version, S/W Version 정보를 표시합니다. 모터제어기 정보를 확인하고 [Next] 버튼을 눌러 다음 스텝으로 이동하거나 [Cancel] 버튼을 눌러 펌웨어 업데이트를 취소할 수 있습니다.

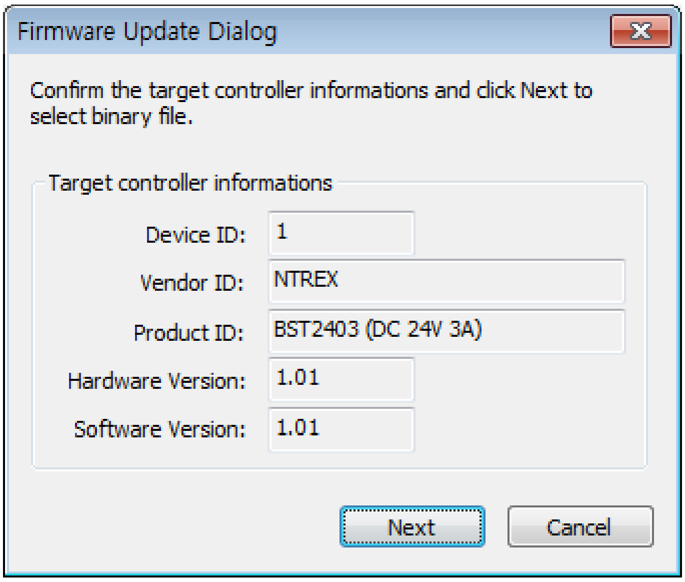


그림 13-3 Firmware Update - Target controller information 창 화면

현재 모터제어기 버전을 확인한 후 [Next] 버튼을 누르면 그림 13-4와 같이 업데이트할 펌웨어를 선택할 수 있는 대화상자가 표시됩니다. 만일 현재 Motor Control UI 유틸리티가 제어기와 연결되어 있지 않다면, 그림 13-3의 대화상자는 표시되지 않고 바로 아래 그림의 대화상자가 제일 먼저 표시됩니다.

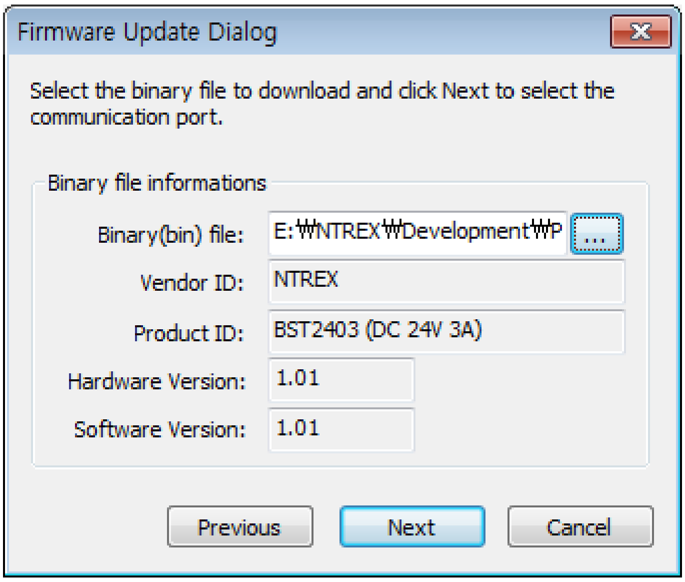


그림 13-4 Firmware Update - Binary file information 창 화면

이 대화상자에서 Binary(bin) file 옆의 [...]버튼을 눌러서 엔티렉스 연구소 홈페이지에서 다운받은 펌웨어 파일을 선택합니다. 그러면 펌웨어 파일의 Vender ID, Product ID, H/W Version, S/W Version을 확인할 수 있습니다. 이 정보와 대상 제어기의 모델 정보가 일치하는지 꼭 확인하기 바랍니다. 그리고 [Next] 버튼을 누르면 그림 13-5과 같이 펌웨어를 다운로드 할 포트 정보가 표시됩니다.

※주의※ 대상 제어기의 모델과 펌웨어 모델이 일치하는지 확인합니다. 만일 다른 모델의 펌웨어를 업데이트 하면 제어기가 올바르게 동작하지 않습니다.

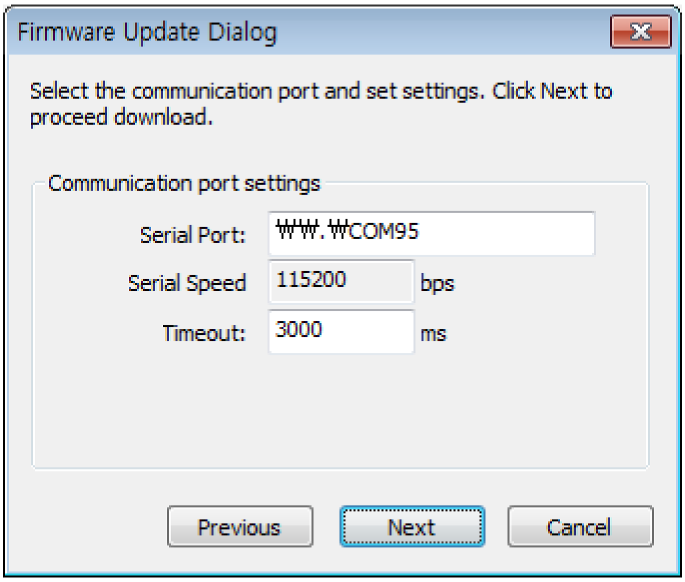


그림 13-5 Firmware Update – Communication port settings 창 화면

여기서는 제어기에 펌웨어를 다운로드 할 시리얼 포트와 와 통신속도, 패킷 전송 Timeout을 설정합니다. 그리고 [Next] 버튼을 누르면 그림 13-6와 같이 펌웨어를 다운로드 하기위한 대화상자가 표시됩니다.

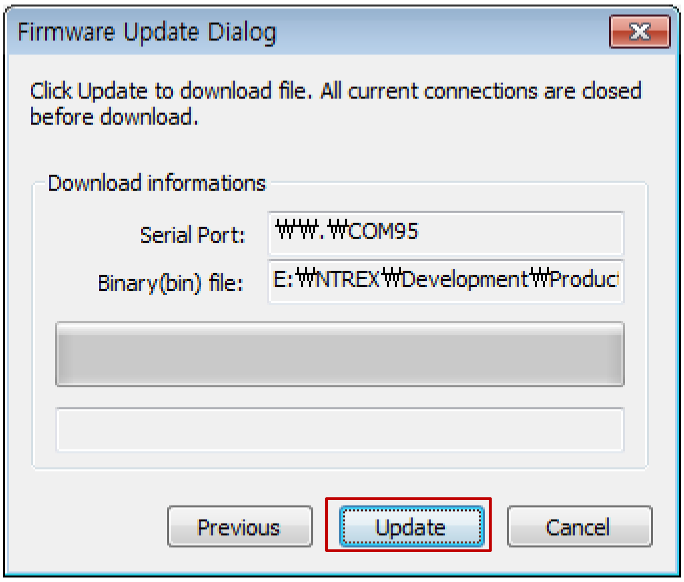


그림 13-6 Firmware Update – Download information 창 화면

여기서 다시한번 펌웨어를 다운로드 하기위한 시리얼 포트와 펌웨어 파일을 확인합니다. 마지막으로 [Update] 버튼을 누르면 기존의 모든 모터제어기 연결이 종료되고 그림 13-7과 같이 최신 펌웨어 다운로드가 시작됩니다.

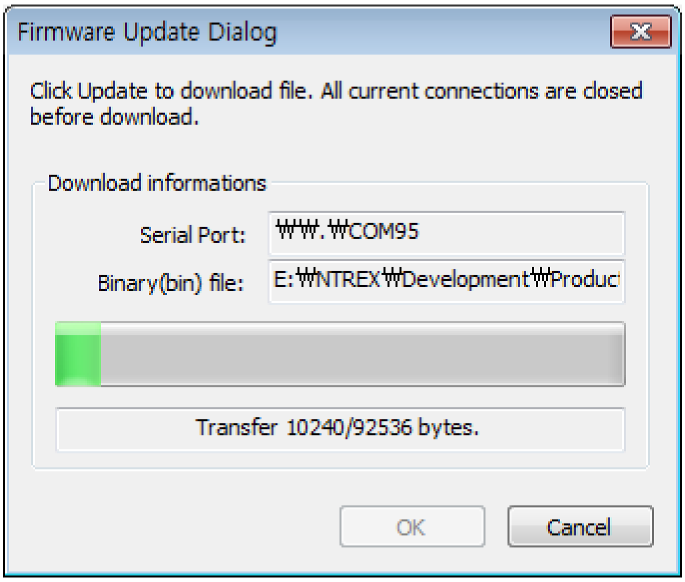


그림 13-7 펌웨어 다운로드 화면

펌웨어 다운로드 중 [Cancel] 버튼을 누르면 다운로드 과정을 취소하고 펌웨어 업데이트 대화상자를 닫습니다. 펌웨어 다운로드가 완료가 되면 [OK] 버튼이 표시되고, [OK] 버튼을 눌러 펌웨어 업데이트 과정을 종료합니다.

13.3.3 펌에어 업데이트 확인

펌웨어 업데이트 과정에서 Motor control UI 유틸리티와 연결된 모든 제어기는 연결이 끊어집니다. 유틸리티 헤더의 [Scan Devices] 버튼을 눌러 다시 제어기를 검색합니다.

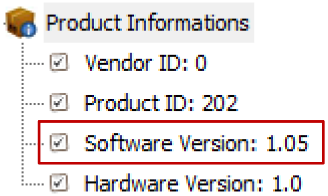


그림 13-8 업데이트 된 버전 화인

그리고 Configuration 탭으로 이동하여 [Read Configurations] 버튼을 클릭합니다. 마지막으로 Product Informations 그룹에서 Software version이 업데이트되었는지 확인합니다.

13.4 Connection

PC의 RS-232, RS-485, Ethernet 네트워크를 검색하고 검색된 제어기에 연결하기에 앞서, 네트워크를 검색하기 위한 기본 설정을 해야 합니다. Port Config 대화상자는 메인 화면의 헤더에서 [Port Config] 버튼을 클릭해서 화면에 표시합니다.

Connection 대화상자는 다음 그림과 같습니다.

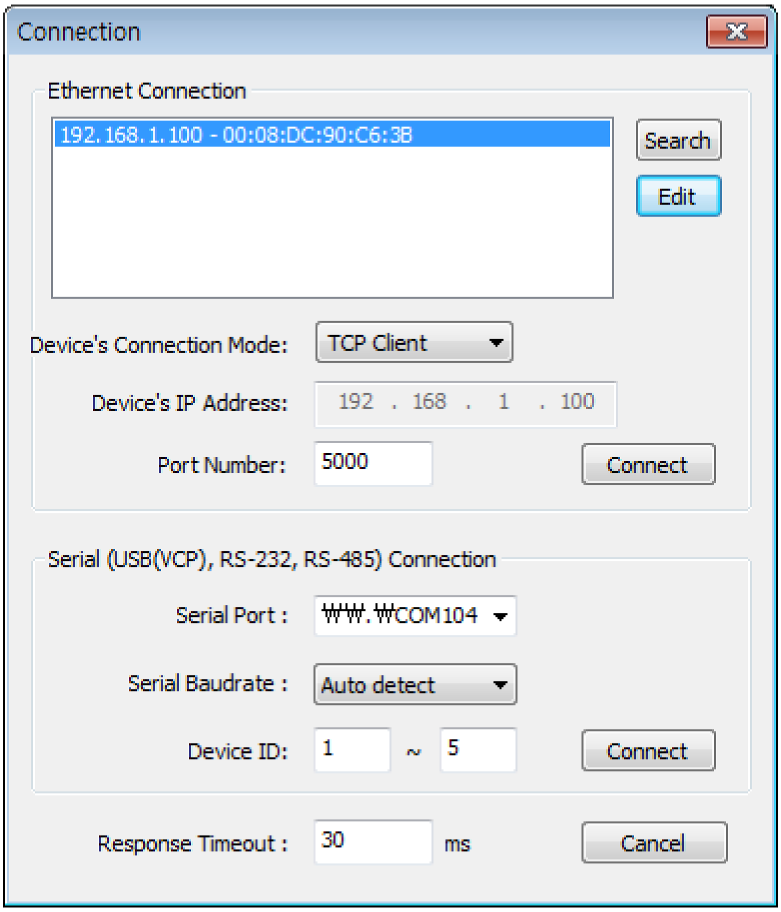


그림 13-9 Connection 창 화면

'Ethernet Connection'은 Ethernet 통신을 사용하기 위한 설정 창입니다.

[Search] 버튼을 누르면 제어기가 PC와 연결 가능한 IP 주소를 검색됩니다.

'Device's Connection Mode'에서 어떤 방법(TCP Server, TCP Client, UDP)으로 연결할지 선택하고 'Port Number'에서 제어기가 접속해야 할 PC의 포트 번호를 입력합니다. 만약 TCP Client로 연결할 경우 'Device's IP Address'에 원격 PC의 IP 주소를 입력합니다.

모든 설정이 끝나면 [Connet] 버튼을 눌러 제어기에 연결합니다.

[Edit] 버튼을 누르면 아래와 같이 Device's Ethernet Configuration 대화상자가 화면에 표시됩니다.

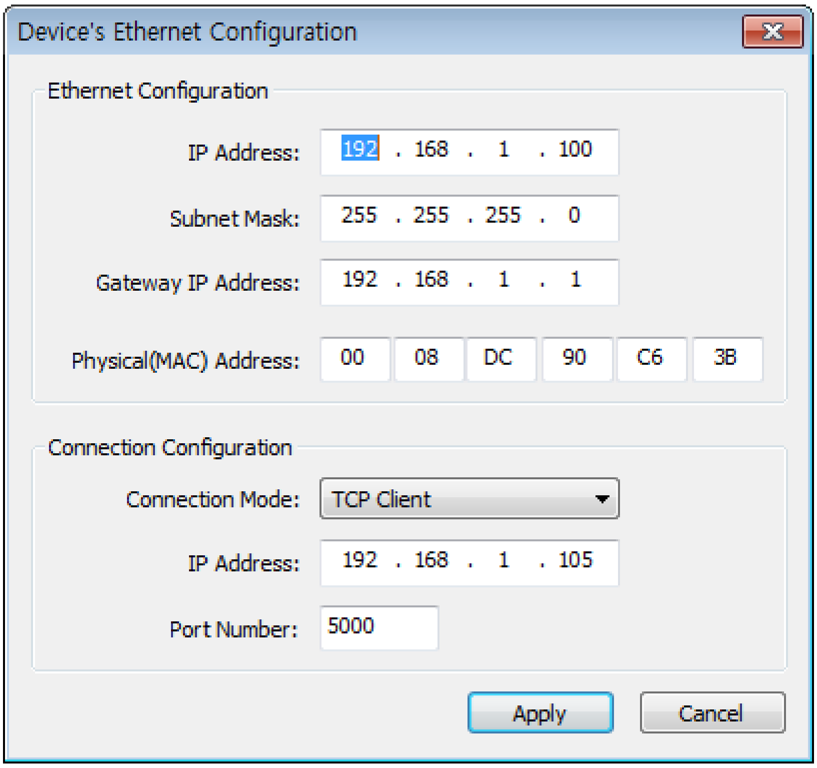


그림 13-10 Device's Ethernet Configuration 창 화면

만약 사용자가 이더넷 연결 환경을 재변경하거나 혹은 다른 방법으로 이더넷을 연결하고 싶을 때 Device's Ethernet Configuration 대화상자에서 설정 변경이 가능합니다.

참고로 'Connection Configuration'의 설정과 값을 변경하면 Connection 대화상자의 'Device's Connection Mode', 'Device's IP Address', 'Port Number'도 변경됩니다.

이에 대한 자세한 내용은 **"7.5 이더넷 통신 설정"**을 참조하시기 바랍니다.

'Serial (USB(VCP), RS-232, RS-485) Connection'은 Serial 통신을 사용하기 위한 설정 창입니다.

'Serial Baudrate'에서는 COM 포트의 통신 속도(9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600 bits/s)를 선택합니다. 만일 'Auto detect'를 선택하면, 모든 통신 속도를 검색하게 됩니다.

'Device ID'는 모터제어기(장치) ID의 검색 범위를 설정합니다. 상기 그림과 같이 "1~5"로 설정하면 모터제어기 ID를 1부터 5까지 검색하겠다는 의미입니다. 이때 Motor Control UI 유틸리티는 많은 모터제어기 ID를 검색해야 하기 때문에 검색 시간이 오래 걸립니다. 만약 모터제어기 ID 값이 1인 경우만 검색하고 싶다면 1 ~ 1로 설정하면 됩니다.

마지막으로 'Response Timeout'에는 제어기에 패킷을 보내고 되돌아올 때까지 기다리는 응답 시간을 밀리세컨드 단위로 설정합니다.

모든 설정이 끝나면 [Connet] 버튼을 눌러 제어기에 연결합니다. 만일 [cancel] 버튼을 누르면 변경된 설정을 저장하지 않고 대화상자를 닫습니다.

13.5 Real-time Plot

Real-time Plot 창은 배터리 전압, 온도, 모터에 내려지는 위치, 속도 명령과 센서 측정 그리고 입출력 채널 중 선택된 항목의 값을 실시간 그래프로 모니터링 합니다.

이 창은 메인 화면의 헤더에서 [Real-time Plot] 버튼을 눌러 표시하거나 닫습니다. 화면에 표시되는 창은 그림 13-11과 같습니다.

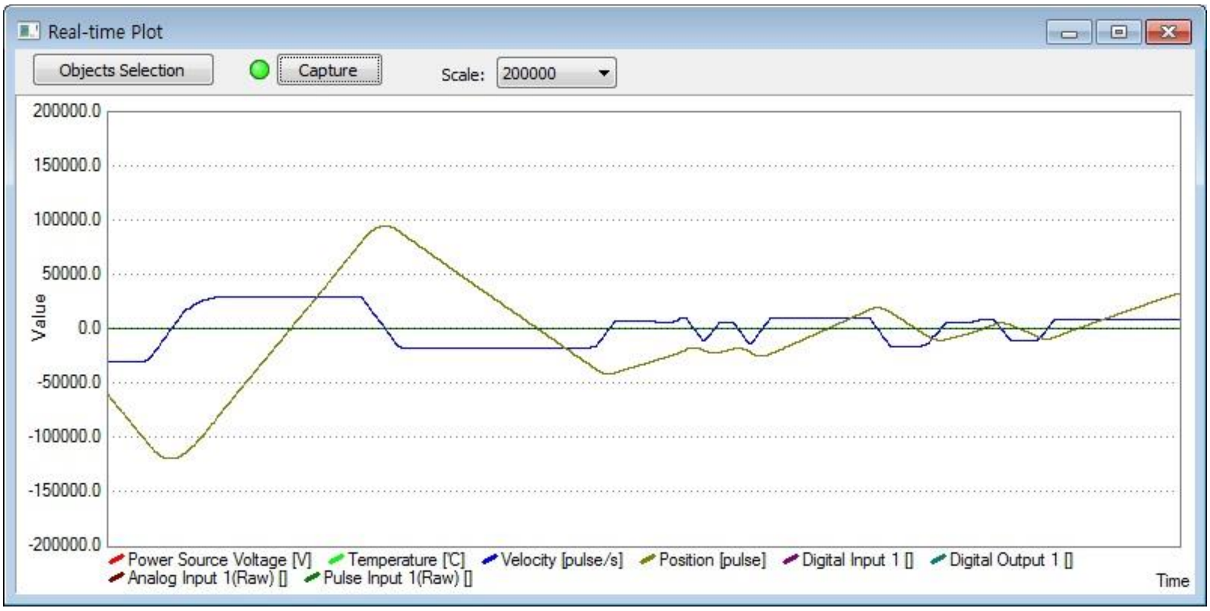


그림 13-11 Real-time Plot 창 화면

그래프의 왼쪽에 표시되는 스케일(Scale)은 마우스 휠을 위아래로 굴리거나 상단에 Scale 크기를 선택해서 확대하거나 축소할 수 있습니다. 또한, 그래프의 아래에는 그래프로 표시되는 항목들에 대한 세부 정보가 표시됩니다.

[Objects Selection] 버튼을 클릭하면 그림 13-12과 같이 Objects Selection Dialog가 표시됩니다. 사용자는 최대 8개의 오브젝트를 선택할 수 있으며, Object에서 모니터링하고 싶은 오브젝트를, Channel에서는 모터 채널 혹은 I/O 채널을 그리고 Scale에서는 해당 오브젝트에 곱해질 값과 색을 선택합니다.

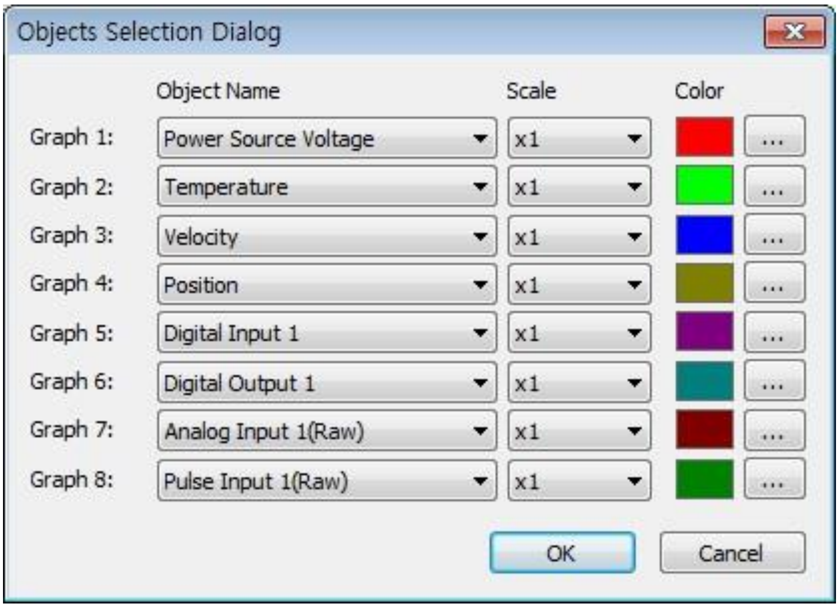
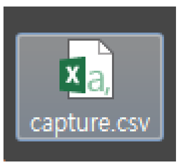


그림 13-12 Objects Selection Dialog 창 화면

오브젝트(Object)는 아래와 같은 항목을 선택할 수 있습니다.

- Power Source Voltage
- Temperature
- Velocity
- Position
- Digital Input 1~4
- Digital Output 1~2
- Analog Input 1 (Raw), (Adj)
- Pulse Input 1 (Raw), (Adj)

[Capture] 버튼을 클릭하면 그림 13-11와 같이 녹색 LED 활성화가 되면서 실시간으로 측정한 데이터를 csv 확장자 파일로 출력 및 저장할 수 있습니다.



13.6 Jog Control

Jog Control은 스텝 모터를 간단하게 수동으로 구동 테스트할 수 있는 컨트롤러입니다. [Jog Control] 버튼을 클릭하면 그림 13-13과 같이 Jog Control가 표시됩니다.

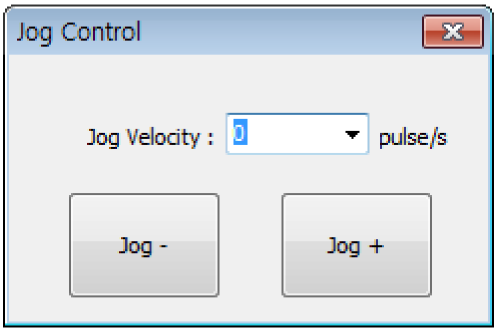


그림 13-13 Joy Control 창 화면

Jog Velocity는 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10000 Pulse/s를 선택할 수 있으며 [Joy+] 버튼을 클릭하면 정방향으로 [Joy-] 버튼을 클릭하면 역방향으로 모터가 설정된 속도 값으로 구동됩니다.

13.7 Motor Control 탭

Motor Control 탭은 제어기의 모터 상태와 모터 폴트를 LED로 표시하고, 모터에 위치, 속도, 전류, 전압 명령을 내릴 수 있습니다. 또한, 제어기의 전압, 전류, 온도를 모니터링하고 위치와 속도 피드백 값을 모니터링 합니다.

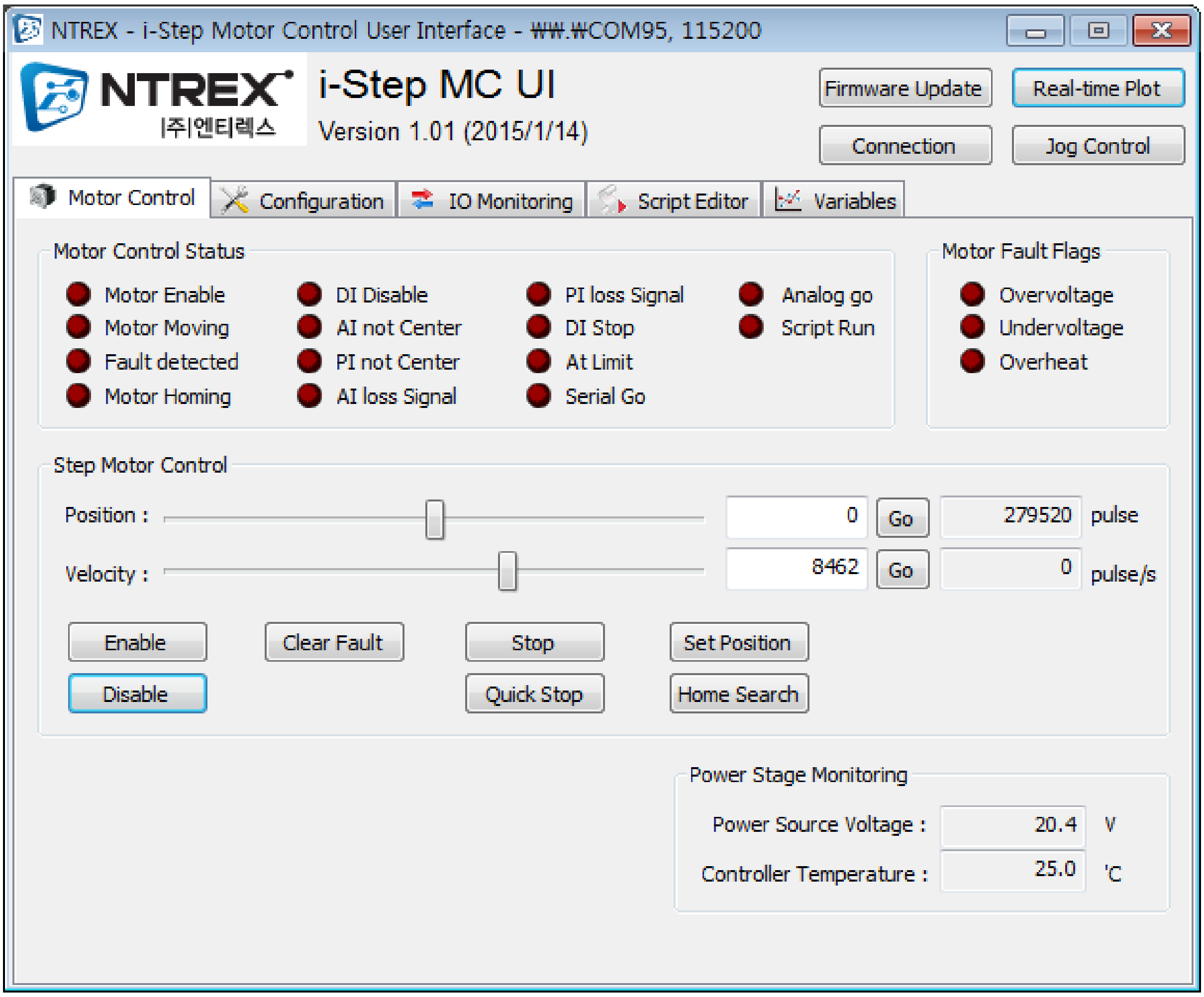


그림 13-14 Motor Control 탭 화면

13.7.1 Motor Control Status

Motor Control Status 그룹에서는 제어기의 모터 구동 상태 및 제어기의 I/O 상태 등을 확인할 수 있습니다. 표시되는 상태를 정리하면 다음과 같습니다:

- Motor Enable - 모터 구동을 위한 활성화 상태
- Motor Moving - 모터가 회전하고 있는 상태
- Fault Detected - 모터에 폴트가 발생한 상황
- Motor Homing - Home Search가 동작중인 상태
- DI Disable - 디지털 입력에서 Emergency Stop (Disable Motor)이 실행
- AI not Center - 아날로그 입력 값이 센터에서 시작되지 않음
- PI not Center - 펄스 입력 값이 센터에서 시작되지 않음
- AI loss Signal - 아날로그 입력에서 적절한 값이 들어오지 않음
- PI loss Signal - 펄스 입력에서 적절한 값이 들어오지 않음
- DI Stop - 디지털 입력에서 Quick Stop, Slowdown Stop 이 실행

- At Limit - 모터의 위치가 리미트 범위를 벗어난 상태
- Serial go - RS-232, RS-485 명령어가 입력되는 상태
- Analog go - 아날로그 및 펄스 입력에서 명령어가 입력되는 상태
- Script Running - 스크립트가 실행 중인 상태

제어기의 모터 상태와 모터 폴트를 표시하는 LED는 다음과 같습니다:

-  - 활성 상태 표시
-  - 비활성 상태 표시
-  - 제어기와 연결이 끊긴 상태 표시

13.7.2 Motor Fault Flags

Motor Control Status는 모터에서 폴트가 발생한 세부 내역을 확인할 수 있습니다. 모터의 폴트 플래그 목록은 다음과 같습니다:

- Overvoltage - 제어기에 과전압이 걸림
- Undervoltage - 제어기에 저전압이 걸림
- Overheat - 방열판이 과열 됨

제어기에서 폴트가 발생하면 모터는 Disable 되며, 폴트 조건이 해제된 이후에 모터 사용이 가능합니다.

13.7.3 Step Motor Control

Motor Control 그룹에서는 모터에 위치, 속도 명령을 내리고 모터의 현재 위치, 속도를 읽을 수 있습니다.

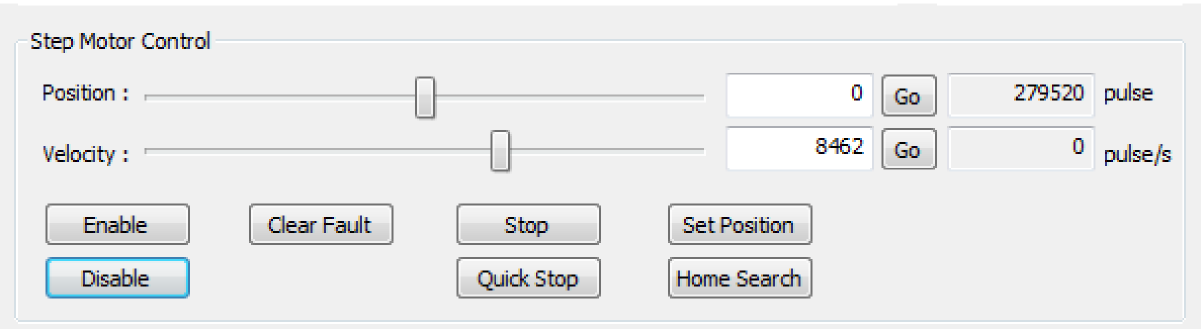


그림 13-15 Motor Control 창

위치, 속도에 대한 각각의 슬라이드 바를 움직이면 슬라이드 바 오른쪽 편집 박스에 바뀐 값이 표시되고 이 값이 제어기에 명령으로 내려갑니다. 이는 편집 박스에 값을 직접 입력하고 [Go] 버튼을 누르는 것과 같은 효과를 냅니다. 제일 오른쪽 편집 불가능한 박스에는 모터의 현재 상태(위

치, 속도, 전류, 전압)가 표시됩니다. 이 값은 주기적으로 업데이트 됩니다.

[Enable]과 [Disable] 버튼은 모터에 Enable이나 Disable 명령을 내립니다. [Clear Fault] 버튼은 제어기의 모터에 발생한 모든 폴트 플래그를 리셋 합니다.

[Stop]과 [Quick Stop] 버튼은 선택된 제어기의 모터를 감속하여 멈추거나 긴급히 멈추게 됩니다.

[Set Positon] 버튼을 클릭하면 그림 13-16 Set Position Dialog 창그림 13-16과 같이 Set Position Dialog가 표시됩니다. 'Current position'은 현재 위치값이고 'New Position'에 새로운 위치값을 입력후 OK 버튼을 클릭하면 새로운 위치로 값이 바뀌게 됩니다.

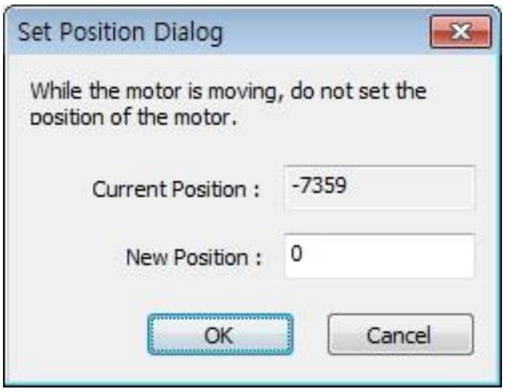


그림 13-16 Set Position Dialog 창

[Home Search] 버튼은 신호(Home Signal)을 받을때까지 모터가 이동합니다. 이는 Configuration 탭에서 설정된 Homing Position, Homing Method, Homing Velocity 설정 값에 영향을 받습니다.

자세한 내용은 **"8.4.2 home_position - Home Position"**을 참조하시기 바랍니다.

13.7.4 Power Stage Monitoring

Power Stage Monitoring에서는 배터리 전압과 방열판의 온도를 모니터링 합니다.

13.8 Configuration 탭

Configuration 탭은 제어기의 구성 파라미터를 설정할 수 있습니다. 또한 설정 값들을 플래시 메모리에 저장하고 제품 초기 설정 값을 불러올 수 있으며, PC의 파일로 저장하고 불러올 수 있습니다.

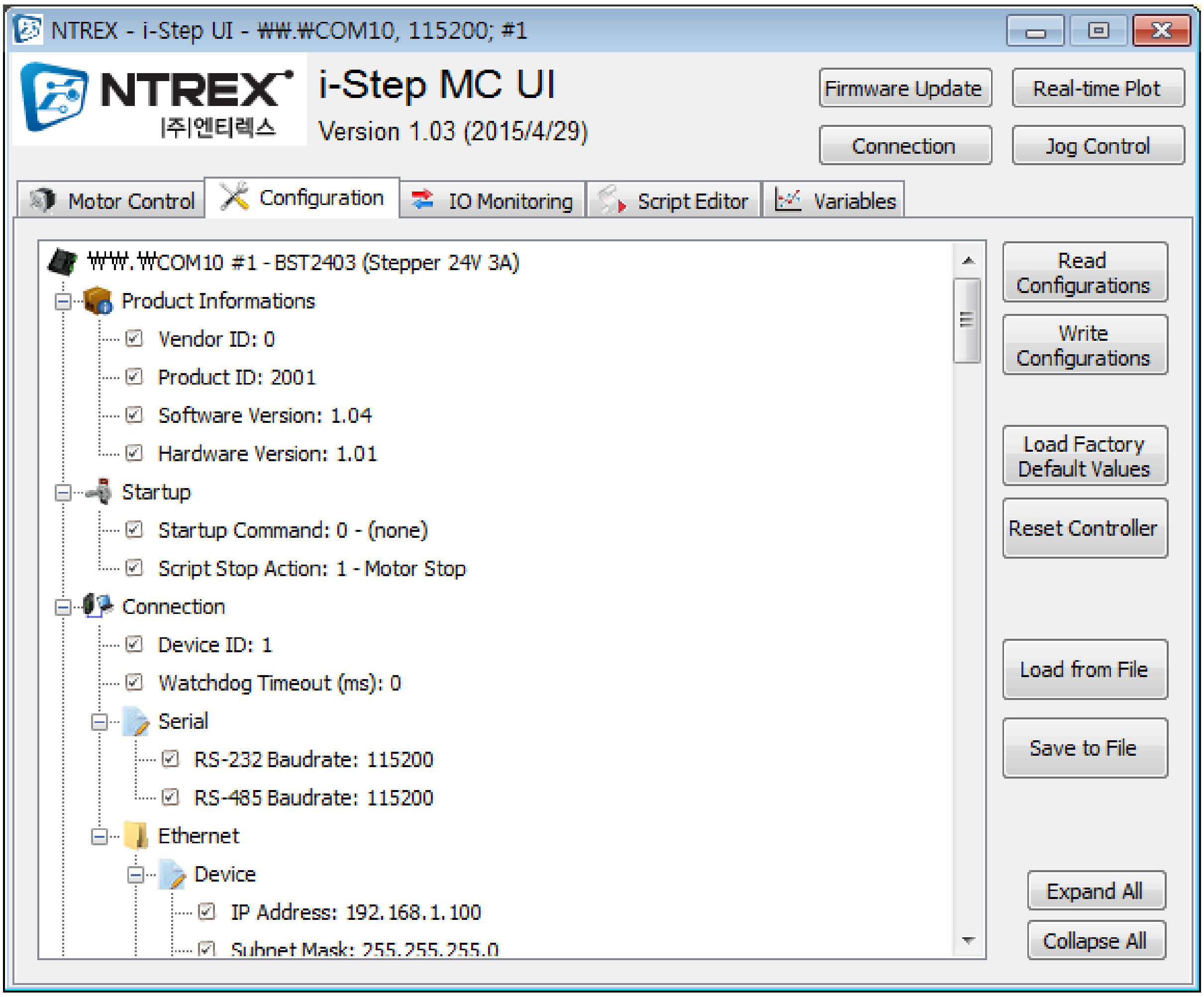


그림 13-17 Configuration 탭 화면

탭의 오른쪽 버튼들에 대한 기능을 요약하면 다음과 같습니다:

- Read Configurations - 제어기의 구성 파라미터 값을 UI 유틸리티로 읽기
- Write Configurations - UI 유틸리티의 구성 파라미터 값을 제어기로 쓰기
- Load Factory Default Values - 플래시 메모리 저장된 구성 파라미터 값을 불러오기
- Reset Controller - 제어기의 소프트웨어 리셋
- Load from File - 외부에 저장된 설정값 불러오기 (cfg 파일)
- Save to File - 외부에 설정값 저장하기 (cfg 파일)
- Expand All - 트리 메뉴를 모두 펼침
- Collapse All - 트리 메뉴를 모두 접기

13.8.1 Product Information

Product Information은 현재 연결된 제품 공급자 ID, 제품 ID, 제어기와 펌웨어 버전을 확인할 수 있습니다. 이 설정은 사용자가 임의로 변경할 수 없습니다.

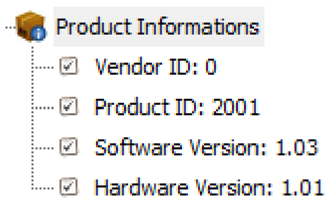


그림 13-18 Product Information

이 그룹의 항목에 대해서는 다음 설명을 참조하십시오:

- Vendor ID - 제품 공급자 ID
- Product ID - 제품 ID
- Software Version - 모터제어기 펌웨어 버전
- Hardware Version - 모터제어기 하드웨어 버전

13.8.2 Startup

Startup 그룹에서는 모터제어기 시작 시 명령을 설정합니다.

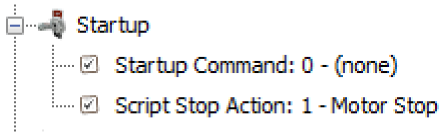


그림 13-19 Startup

이 그룹의 항목에 대해서는 다음 설명을 참조하십시오:

- Startup Command - 모터제어기 시작 시 한번 수행되는 명령
- Script Stop Action - 스크립트 정지 후 수행되는 동작 명령

(1) Startup Command

Startup Command는 제어기에 전원이 투입되고 제어기가 기동되었을 때 한번 수행되는 명령을 지정합니다. 지정 가능한 명령은 다음과 같습니다.

- 0 - (none) - 명령 없음
- 1 - Motor Enable - 모터 활성화
- 2 - Run Script - 제어기에 저장한 스크립트 실행
- 3 - Motor Enable and Run Script - 1번과 2번 명령 실행

(2) Script Stop Action

Script Stop Action은 스크립트가 중간에 멈추거나 혹은 SW 적으로 정지했을 때 수행되는 동작 명령을 지정합니다. 지정 가능한 명령은 다음과 같습니다.

- 0 - (none)
- 1 - Motor Stop - 모터 감속하면서 정지
- 2 - Motor Quick Stop - 모터 급격히 정지
- 3 - Motor Disable - 모터 비활성화

13.8.3 Connection

Connection은 모터제어기 ID, 와치독, 현재 연결된 Serial 통신 속도, Ehternet 통신 속도 등을 설정하고 확인할 수 있습니다.

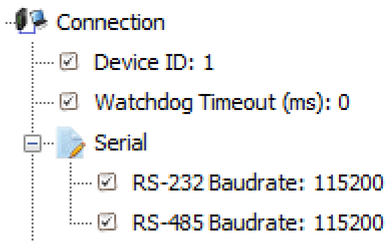


그림 13-20 Connection - Serial

이 그룹의 항목에 대해서는 다음 설명을 참조하십시오:

- Device ID - 장치 ID
- Watchdog Timeout - 통신 포트로 수신되는 명령어 패킷에 대한 와치독 타이머 타임아웃 값
- RS-232 Baudrate - RS-232 통신 속도
- RS-485 Baudrate - RS-485 통신 속도

(1) RS-232 Baudrate

RS-232 Baudrate의 값으로 다음 중 하나를 설정하여 시리얼 통신 속도를 변경합니다..

- 9600 - 9600bps
- 19200 - 19200bps
- 38400 - 38400bps
- 57600 - 57600bps
- 115200 - 115200bps (기본값)
- 230400 - 230400bps
- 460800 - 460800bps
- 921600 - 921600bps

※ RS-232 Baudrate 값을 변경한 경우에는 제어기를 재시작해야 변경된 값이 반영됩니다.

(2) RS-485 Baudrate

RS-485 Baudrate의 값으로 다음 중 하나를 설정하여 시리얼 통신 속도를 변경합니다..

- 9600 - 9600bps
- 19200 - 19200bps
- 38400 - 38400bps
- 57600 - 57600bps
- 115200 - 115200bps (기본값)
- 230400 - 230400bps
- 460800 - 460800bps
- 921600 - 921600bps

※ RS-485 Baudrate 값을 변경한 경우에는 제어기를 재시작해야 변경된 값이 반영됩니다.

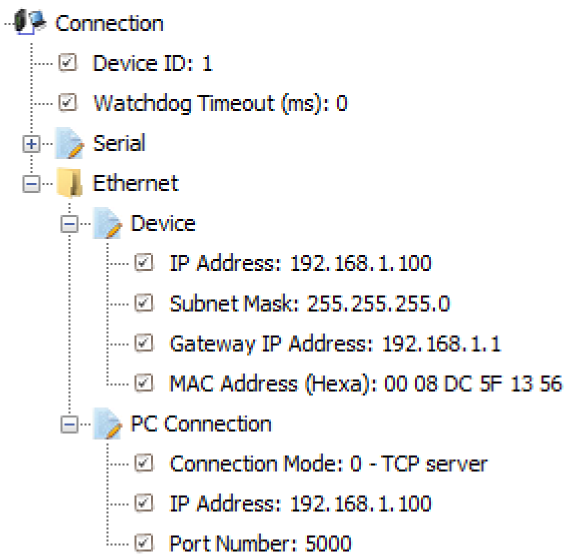


그림 13-21 Connection - Ethernet

이 그룹의 항목에 대해서는 다음 설명을 참조하십시오:

- Device - 제어기의 IP 주소와 MAC 주소를 설정
- PC Connection - 제어기와 PC 연결 방법 및 IP 주소 설정

(1) Device

Device는 모터제어기 자체의 IP 주소와 MAC 주소를 설정하는 부분입니다.

- IP Address - IP 주소 설정

- Subnet Mask - 서브넷 마스크 설정
- Gateway IP Address - 게이트웨이 IP 주소 설정
- MAC Address - MAC 주소 설정

자세한 내용은 "7.5 이더넷 통신 설정"을 참조하시기 바랍니다.

(2) PC Connection

PC Connection은 제어기와 호스트 PC간의 연결에 관련되어 설정하는 부분입니다.

- Connrection Mode - 제어기와 PC간 연결 방법 설정
- IP Address - PC의 IP 주소 설정
- Port Number - PC의 포트 번호 설정

'Connrection Mode'에서는 아래와 같이 3가지 연결 방법을 선택할 수 있습니다.

- 0 - TCP Server
- 1 - TCP Client
- 2 - UDP

자세한 내용은 "7.5 이더넷 통신 설정"을 참조하시기 바랍니다.

※주의※ TCP 서버 모드나 TCP 클라이언트 모드에서 호스트 PC로 연결되어있지 않은 경우 시리얼 포트로부터 수신된 데이터는 폐기됩니다.

※주의※ 제어기가 TCP Client 모드로 설정되어 호스트 PC로 연결을 시도하는 경우, 호스트 PC에 방화벽(firewall)이 설정되어 있다면 호스트 PC로의 접속 시도가 거절될 수 있습니다.

13.8.4 Motor

Motor 그룹은 Position Limits, Homing, Velocity Profile, Fault Condition, Stepper Properties 등 5개의 설정으로 구분되어 있습니다.

13.8.5 Motor - Position Limits

Position Limits은 위치에 대한 설정입니다. 위치 제어를 하기 위해 최소/최대 위치 값, Soft Limit 사용 여부를 설정할 수 있습니다.

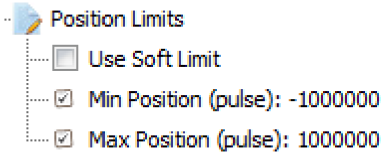


그림 13-22 Position Limits

'Use Soft Limit'은 최대/최소 위치 범위를 벗어나면 Forward/Reverse Limit Switch를 작동한 효과를 보여주는 기능을 가지고 있습니다. 만약 이 기능이 활성화된 상태에서 설정한 최대/최소 위치 값이 이상으로 벗어나면 Motor Control 탭에서 At Limit이 활성화 됩니다.

13.8.6 Motor - Homing

Homing은 사용자가 원하는 홈 포지션과 방법 그리고 속도 값을 설정할 수 있습니다.

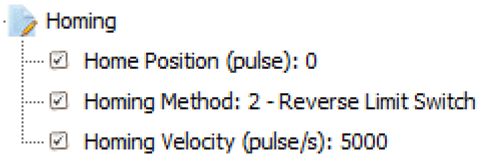


그림 13-23 Homing

(1) Homing Method

Status는 디지털 출력 채널을 특정 모터의 동작으로 매핑합니다. 지정 가능한 동작 명령은 다음과 같습니다.

- 0 - Home Sensor
- 1 - Forward Limit Switch
- 2 - Reverse Limit Switch

자세한 내용은 "8.4 모터의 위치 및 Homing 설정"을 참조하시기 바랍니다.

13.8.7 Motor - Velocity Profile

Velocity Profile은 사용자가 최소/최대 속도와 모터의 회전 가감속도를 설정할 수 있습니다.

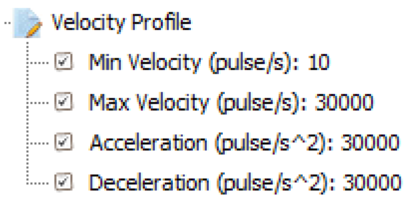


그림 13-24 Velocity Profile

13.8.8 Motor - Fault Conditions

Fault Condition은 사용자 그리고 제품의 안전을 위해 Fault 감지 조건을 설정할 수 있습니다.

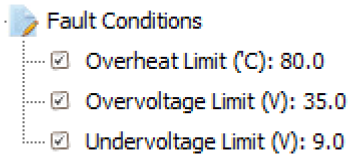


그림 13-25 Fault Condition

만약 위와 같이 설정되었을 때 제어기의 온도가 80도 이상, 과전압 35V 이상, 저전압 9V 이하가 한 번이라도 감지되면 폴트(Fault)가 발생하게 됩니다.

※주의※ 사용자는 사용하고자 하는 제어기의 매뉴얼와 스펙을 잘 숙지한 후 과열 온도, 과전압, 저전압 한계 값을 적용해 주시기 바랍니다. 만약 스펙 이상의 값이 설정되면 모터제어기가 파손될 위험이 있으니 주의하기 바랍니다.

13.8.9 Motor - Stepper Properties

Stepper Properties은 스텝 모터를 어떻게 구동할지를 설정할 수 있습니다.

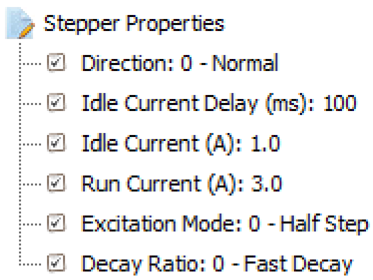


그림 13-26 Operations

이 그룹의 항목에 대해서는 다음 설명을 참조하십시오:

- Direction - 모터의 방향 설정
- Idle Current Delay - 대기 전류 지연 시간 설정
- Idle Current - 대기 전류 설정
- Run Current - 구동 전류 설정
- Excitation Mode - 마이크로 스텝 설정 (분해능 설정)
- Decay Ratio - 감쇄 비율 설정

(1) Idle Current

Idle Current의 값으로 대기 전류를 설정합니다.

- 0 ~ 3 - 0A에서 3A까지 설정할 수 있습니다.

(2) Run Current

Run Current의 값으로 구동 전류를 설정합니다.

- 0 ~ 3 - 0A에서 3A까지 설정할 수 있습니다.

(3) Excitation Mode

Excitation Mode의 값으로 다음 중 하나를 설정하여 Micro Step을 설정합니다.

- 0 - Half Step - 400 Pulse/Rev (1.8°)
- 1 - 1/8 phase - 1600 Pulse/Rev (1.8°)
- 2 - 1/16 phase - 3200 Pulse/Rev (1.8°)
- 3 - 1/32 phase - 6400 Pulse/Rev (1.8°)
- 3 - 1/64 phase - 12800 Pulse/Rev (1.8°)
- 3 - 1/128 phase - 25600 Pulse/Rev (1.8°)
- 3 - 1/10 phase - 2000 Pulse/Rev (1.8°)
- 3 - 1/20 phase - 4000 Pulse/Rev (1.8°)

(4) Decay Ratio

Decay Ratio는 마이크로 스텝 구동 시 감쇄 속도를 조절함으로써 마이크로 스텝 정밀도를 개선할 수 있습니다. Decay Ratio의 값으로 다음 중 하나를 설정하여 감쇄 비율을 설정합니다.

- 0 - Fast Drcay
- 1 - Mixed Decay
- 2 - Slow Decay

13.8.10 Digital Inputs

Digital Input은 모터의 구동에 필요한 정보 및 명령을 받아들이기 위해 사용합니다. 디지털 입력 활성화, 디지털 입력 극성 반전 활성화, Mapping 기능을 설정할 수 있습니다.

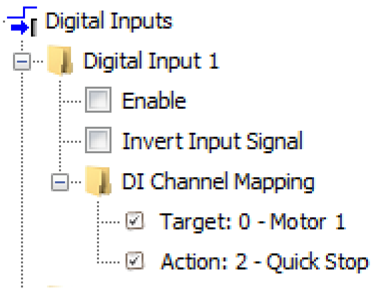


그림 13-27 Digital Inputs

이 그룹의 항목에 대해서는 다음 설명을 참조하십시오:

- Enable - 디지털 입력 채널 사용 여부
- Invert Input Signal - 디지털 입력 채널 반전 여부
- Target, Action - 디지털 입력 채널을 특정 모터의 동작으로 매핑

(1) Action

Action은 디지털 입력 채널을 특정 모터의 동작으로 매핑합니다. 지정 가능한 동작 명령은 다음과 같습니다.

- 0 - None
- 1 - Emergency Stop (Disable Motor) - 모터에 공급되는 전원 차단
- 2 - Quick Stop - 모터를 급격히 정지
- 3 - Slowdown Stop - 모터를 감속하여 정지
- 4 - Forward Limit Switch - 정방향 리미트 센서가 감지
- 5 - Reverse Limit Switch - 역방향 리미트 센서가 감지
- 6 - Home Sensor - 모터의 홈 위치 센서가 감지
- 7 - Run Script - 스크립트 실행

13.8.11 Digital Outputs

Digital Output은 모터의 현재 구동 상태를 외부에 알리기 위해 사용합니다. 디지털 출력 활성화, 디지털 출력 극성 반전 활성화, Mapping 기능을 설정할 수 있습니다.

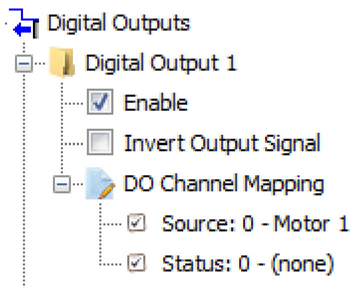


그림 13-28 Digital Outputs

이 그룹의 항목에 대해서는 다음 설명을 참조하십시오:

- Enable - 디지털 출력 채널 사용 여부
- Invert Output Signal - 디지털 출력 채널 반전 여부
- Source, Status - 디지털 출력 채널을 특정 모터의 동작으로 매핑

(1) Status

Status는 디지털 출력 채널을 특정 모터의 동작으로 매핑합니다. 지정 가능한 동작 명령은 다음과 같습니다.

- 0 - None
- 1 - Motor Enable - 모터 활성화
- 2 - Motor Running - 모터 구동
- 3 - Motor Fault - 모터 폴트 발생

13.8.12 Analog Inputs

Analog Input은 모터의 구동에 필요한 정보 및 명령을 아날로그 신호로 받아들이기 위해 사용합니다. 아날로그 입력 활성화, 아날로그 입력 극성 반전 활성화, 모터에 내려지는 속도, 명령의 Safety 설정, Linearity 설정, Mapping 기능, Calibration을 설정할 수 있습니다.

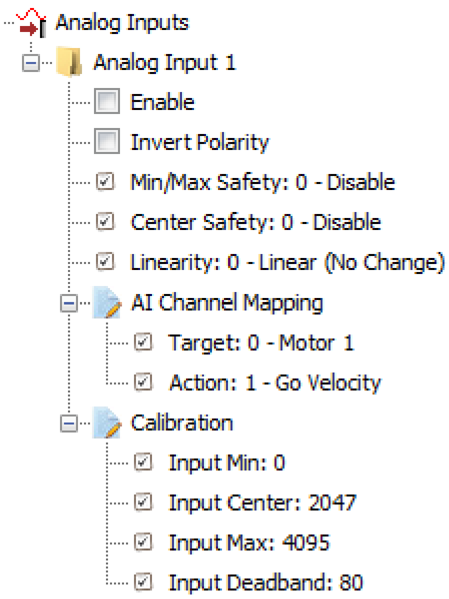


그림 13-29 Analog Input

이 그룹의 항목에 대해서는 다음 설명을 참조하십시오:

- Enable - 아날로그 입력 채널 사용 여부
- Invert Polarity - 아날로그 입력 채널 반전 여부
- Min/Max Safety - 잘못된 아날로그 신호 출력 감지
- Center Safety - 아날로그 입력 값이 센터 위치에 있는지 확인
- Linearity - 아날로그 입력 값을 지수/로그 함수로 변환
- Target, Action - 아날로그 입력 채널을 특정 모터의 동작으로 매핑
- Calibration - 아날로그 입력 캘리브레이션

(1) Linearity

Linearity는 아날로그 입력의 정규화 된 값을 지수/로그 함수로 변환합니다. 지정 가능한 지수/로그 함수는 다음과 같습니다.

- 0 - linear(no change) (기본값)
- 1 - exp weak
- 2 - exp medium
- 3 - exp strong
- 4 - log weak
- 5 - log medium
- 6 - log strong

(2) Action

Action은 아날로그 입력 채널을 특정 모터의 동작으로 매핑합니다. 지정 가능한 동작 명령은 다음과 같습니다.

- 0 - None
- 1 - Go Velocity - 속도 제어 명령
- 2 – Go Position - 위치 제어 명령
- 3 - Emergency Stop (Disable Motor) - 모터에 공급되는 전원 차단
- 4 - Quick Stop - 모터를 급격히 정지
- 5 - Slowdown Stop - 모터를 감속하여 정지
- 6 - Forward Limit Switch - 정방향 리미트 센서가 감지
- 7 - Reverse Limit Switch - 역방향 리미트 센서가 감지
- 8 - Home Sensor - 모터의 홈 위치 센서가 감지
- 9 - Run Script - 스크립트 실행

13.8.13 Pulse Inputs

Pulse Input은 RC 조종기 및 디지털 신호를 받아들이기 위해 사용합니다. 펄스 입력 활성화, 펄스 입력 극성 반전 활성화, 모터에 내려지는 속도, 명령의 Safety 설정, Linearity 설정, Capture Type 설정, Linearity 설정, Mapping 기능, Calibration을 설정할 수 있습니다.

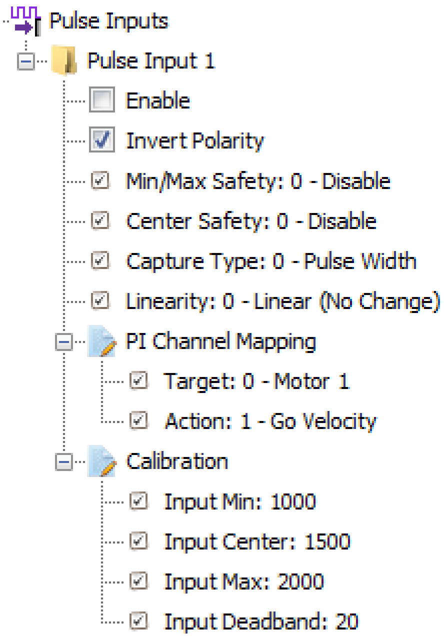


그림 13-30 Pulse Input

이 그룹의 항목에 대해서는 다음 설명을 참조하십시오:

- Enable - 펄스 입력 채널 사용 여부
- Invert Polarity - 펄스 입력 채널 반전 여부
- Min/Max Safety - 잘못된 펄스 신호 출력 감지
- Center Safety - 아날로그 입력 값이 센터 위치에 있는지 확인
- Capture Type - 펄스 입력 데이터 수집 형식 타입 설정
- Linearity - 펄스 입력 값을 지수/로그 함수로 변환
- Target, Action - 펄스 입력 채널을 특정 모터의 동작으로 매핑
- Calibration - 펄스 입력 캘리브레이션

(1) Capture Type

Capture Type은 펄스 입력 데이터 수집 형식을 선택할 수 있습니다. 지정 가능한 Capture Type은 다음과 같으며, 측정 값은 0 ~ 1000‰ 사이가 됩니다.

- 0 - Pulse Width - 펄스 폭 측정
- 1 – Frequency - 펄스 주파수 측정

- 2 - Duty Cycle - 펄스 듀티 사이클 측정

(2) Linearity

Linearity는 펄스 입력의 정규화 된 값을 지수/로그 함수로 변환합니다. 지정 가능한 지수/로그 함수는 다음과 같습니다.

- 0 - linear(no change) (기본값)
- 1 - exp weak
- 2 - exp medium
- 3 - exp strong
- 4 - log weak
- 5 - log medium
- 6 - log strong

(3) Action

Action은 펄스 입력 채널을 특정 모터의 동작으로 매핑합니다. 지정 가능한 동작 명령은 다음과 같습니다.

- 0 - None
- 1 - Go Velocity - 속도 제어 명령
- 2 – Go Position - 위치 제어 명령
- 3 - Emergency Stop (Disable Motor) - 모터에 공급되는 전원 차단
- 4 - Quick Stop - 모터를 급격히 정지
- 5 - Slowdown Stop - 모터를 감속하여 정지
- 6 - Forward Limit Switch - 정방향 리미트 센서가 감지
- 7 - Reverse Limit Switch - 역방향 리미트 센서가 감지
- 8 - Home Sensor - 모터의 홈 위치 센서가 감지
- 9 - Run Script - 스크립트 실행

13.8.14 Calibration

아날로그 입력이나 펄스 입력 채널로부터의 원시 값은 캘리브레이션 과정을 거쳐 -1과 1 사이의 정규화 된 값으로 변환됩니다. 캘리브레이션 과정에는 min, max, center, deadband 파라미터의 설정이 필요합니다. 자동으로 캘리브레이션 파라미터의 설정을 위해서는 아날로그 입력이나 펄스 입력에서 Calibration을 선택해서 [...] 버튼을 클릭합니다.

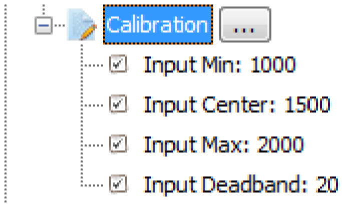


그림 13-31 Pulse Input - Calibration

버튼을 누르면 다음과 같은 Calibration 대화상자가 화면에 표시됩니다. 캘리브레이션을 진행하기에 앞서 해당 아날로그 입력이나 펄스 입력 포트는 Enable 상태로 설정되어 있어야 합니다. 여기서는 설명을 쉽게 하기 위해 펄스 입력 채널 1에 조이스틱이 연결되었다고 가정하겠습니다.

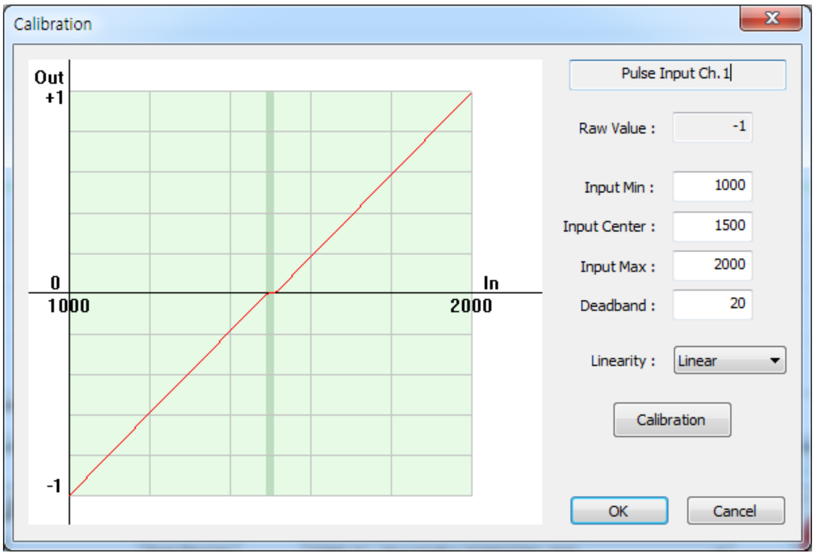


그림 13-32 Calibration - Dead-band

상기 창에서 자동 캘리브레이션을 시작하려면 [Calibration] 버튼을 누릅니다. 이후 조이스틱을 최대 최소 범위로 움직여 'Input Min'과 'Input Max' 값을 찾습니다. 그리고 조이스틱을 중앙에 두고 'Input Center' 값을 고정한 후 [Calibration] 버튼을 한 번 더 눌러 캘리브레이션 과정을 종료합니다. 마지막으로 [OK] 버튼을 누르면 캘리브레이션 과정에서 찾은 파라미터 들이 Configuration 탭의 파라미터 값들로 복사됩니다.

※주의※ 사용자는 캘리브레이션을 하기 전에 먼저 사용하고자 하는 아날로그 입력 포트와 펄스 입력 포트를 활성화(Enable)한 후 설정을 플래시 메모리에 저장([Write Configuration] 버튼 클릭 후 모터가 2~3표 정지)해야 합니다. 그리고 난 후 캘리브레이션을 진행해야 합니다.

※주의※ 캘리브레이션을 진행하기에 앞서 해당 채널에 연결된 모터는 Disable 상태에 있어야 합니다. 만일 모터가 Enable 상태에 있다면 캘리브레이션 도중 모터가 갑자기 회전하여 위험한 상황을 초래할 수 있습니다.

13.9 I/O Monitoring 탭

I/O Monitoring 탭에서는 디지털 입력, 디지털 출력, 아날로그 입력, 펄스 입력 채널의 값을 모니터링하고 디지털 출력 채널을 제어할 수 있습니다.

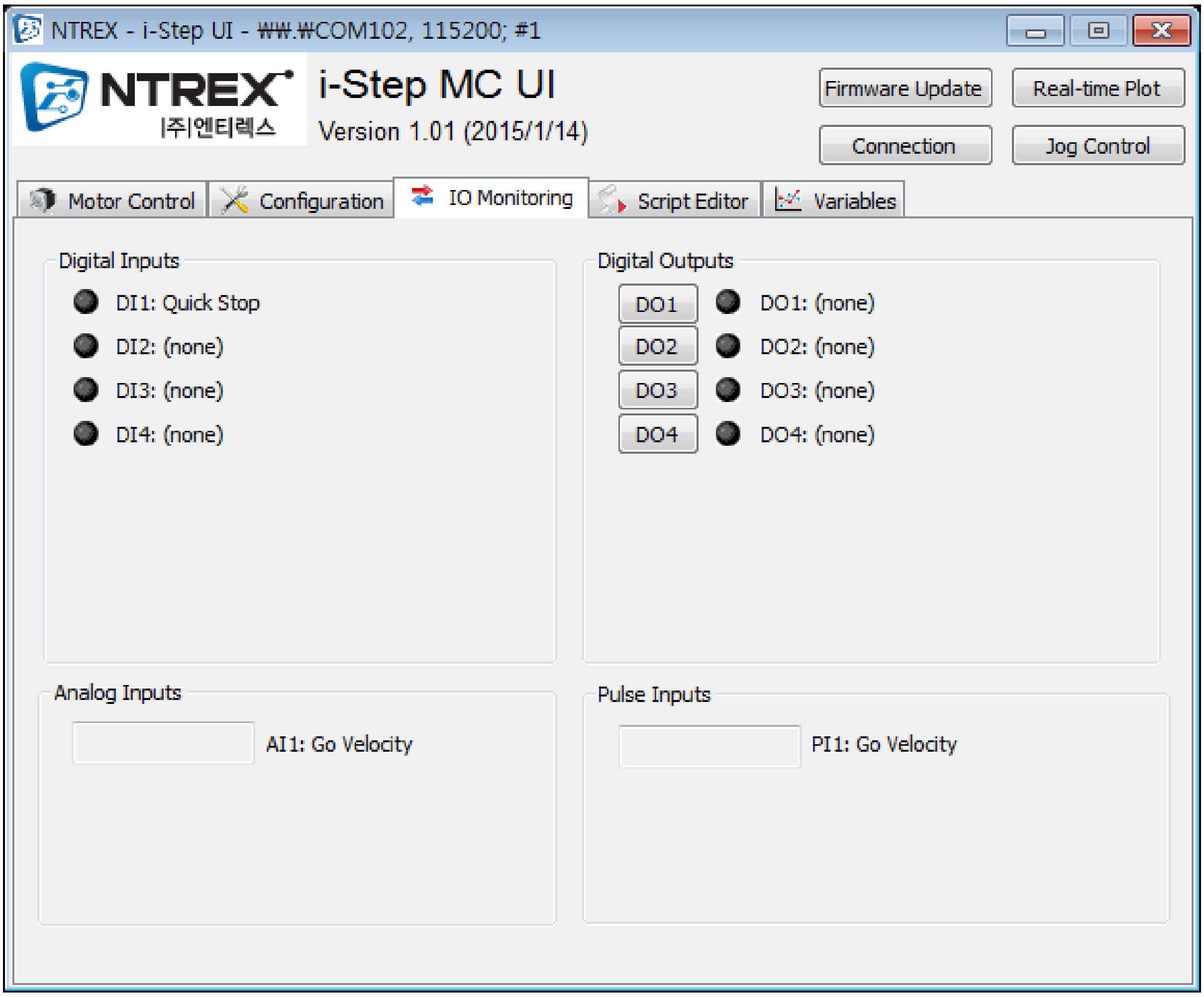


그림 13-33 I/O Monitoring 탭 화면

13.9.1 Digital Inputs

디지털 입력 그룹은 Configuration 탭에서 Enable 된 디지털 입력 채널의 값을 모니터링하고 매핑된 모터와 기능을 읽을 수 있습니다.

디지털 입력 값의 상태를 표시하는 LED는 다음과 같습니다:

-  - 입력이 ON(1)인 상태 표시
-  - 입력이 OFF(0)인 상태 표시
-  - 제어기와 연결이 끊겼거나 입력이 사용 불가(disable) 상태

13.9.2 Digital Outputs

디지털 출력 그룹은 Configuration 탭에서 Enable 된 디지털 출력 채널의 값을 모니터링하고 매핑된 모터와 기능을 읽을 수 있습니다. 디지털 출력 값의 상태를 표시하는 LED는 디지털 입력 값의 상태를 표시하는 LED와 동일하게 표시됩니다.

디지털 출력 값은 [DOx] 버튼을 눌러 켜거나 끌 수 있습니다. 사용자가 버튼으로 디지털 출력 값을 직접 제어하고자 할 때는 디지털 출력 채널에 매핑된 기능이 없어야 합니다.

13.9.3 Analog Inputs

아날로그 입력 그룹은 Configuration 탭에서 Enable 된 아날로그 입력 채널의 원래(raw) 값과 변환된(converted) 값을 모니터링하고 매핑된 모터와 기능을 읽을 수 있습니다. 괄호 바깥의 숫자는 변환된 값이고, 괄호 안의 숫자는 원래 값입니다.

아날로그 입력 채널의 원래 값은 아날로그 입력 포트의 12bit A/D 변환기로부터 직접 읽은 값입니다. 이 값의 범위는 0 ~ 4095 범위를 가집니다. 이 값은 변환 과정을 거쳐 -1 ~ 1 사이의 값으로 변환됩니다.

13.9.4 Pulse Inputs

펄스 입력 그룹은 Configuration 탭에서 Enable 된 펄스 입력 채널의 원래(raw) 값과 변환된(converted) 값을 모니터링하고 매핑된 모터와 기능을 읽을 수 있습니다. 괄호 바깥의 숫자는 변환된 값이고, 괄호 안의 숫자는 원래 값입니다.

펄스 입력 채널의 원래 값은 펄스 입력 포트로 들어오는 신호의 Pulse Width, Frequency, Duty Cycle을 읽은 값입니다. 이 값의 범위는 캡쳐 타입에 따라 달라집니다. 이 값은 변환 과정을 거쳐 -1 ~ 1 사이의 값으로 변환됩니다.

13.10 Script 탭

스크립트를 작성하고 실행하는 것은 제어기의 강력한 기능 중 하나입니다. 스크립트는 제어기의 구성 파라미터 설정만으로 불가능한 다양한 작업을 수행할 수 있도록 합니다.

스크립트 탭은 다음 그림과 같이 몇 개의 버튼과 스크립트 편집 창, 변수 모니터링 창, 메시지 창으로 구성됩니다.

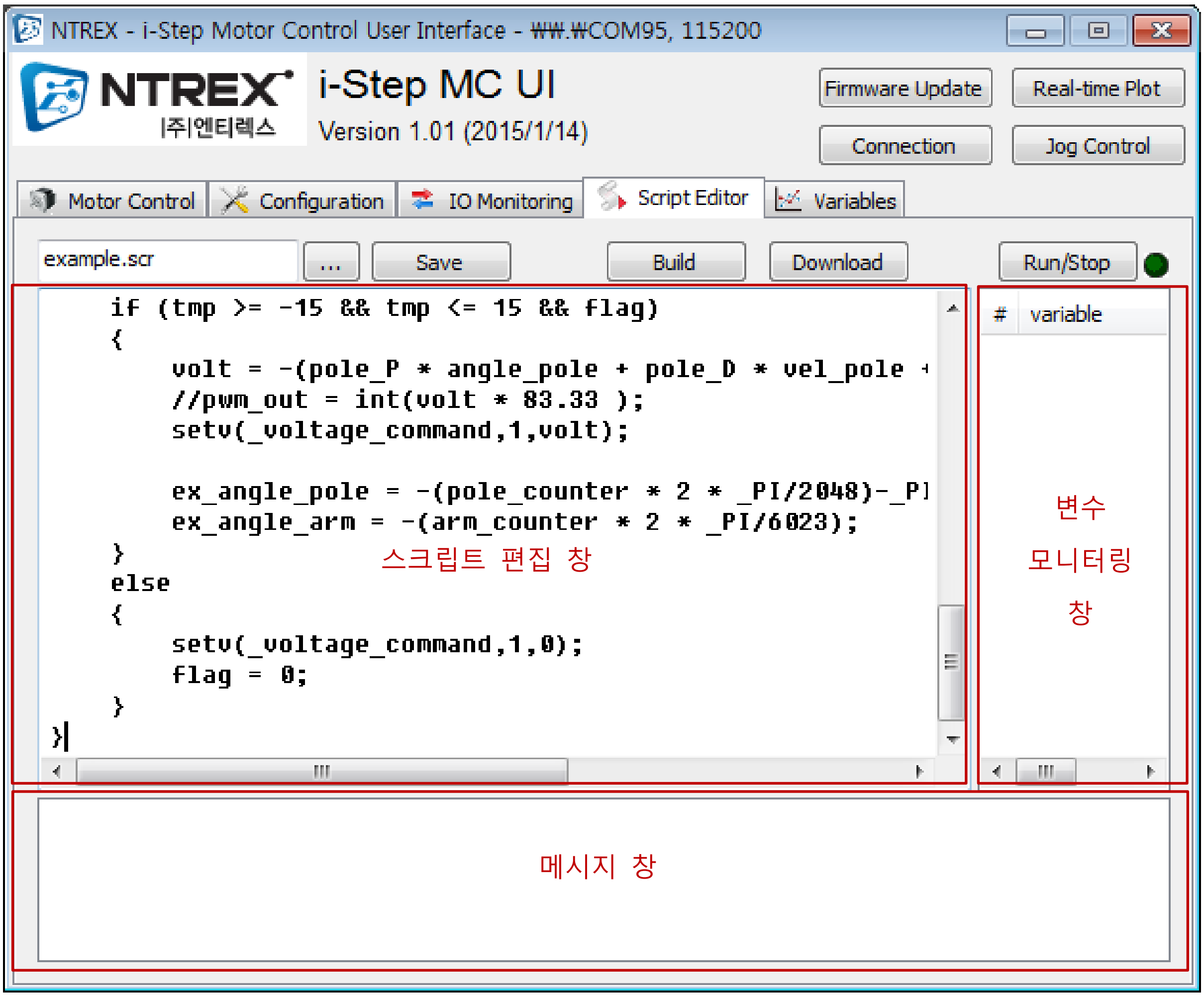


그림 13-34 Script 탭 화면

다음은 스크립트 탭에서 사용하는 버튼에 대한 간단한 설명입니다:

- ... - 불러올 스크립트 파일 이름을 선택
- Save - 스크립트 편집 창의 코드를 파일로 저장
- Build - 스크립트 편집 창의 코드를 빌드 하여 바이트코드 파일 생성
- Download - 제어기에 바이트코드 파일을 다운로드
- Script Run/Stop - 제어기에서 스크립트 프로그램 실행/중단

13.10.1 Script 작성

스크립트 편집 창은 스크립트를 입력하는 데 사용됩니다. 편집기는 소스 코드를 작성하는 데 필요한 기본 텍스트 편집 기능을 가지고 있습니다.

작성한 코드를 저장하고 싶다면 [...] 버튼을 눌러 PC에 저장할 위치와 파일 이름을 선택한 후 [Save] 버튼을 누르면 됩니다.

13.10.2 빌드 및 다운로드

스크립트 작성이 완료되면 [Build] 버튼을 클릭하여 소스코드를 바이트코드로 컴파일합니다. 메시지 창에는 컴파일 메시지와 컴파일 결과가 나타납니다.

빌드가 성공적으로 수행되면, bin 파일(바이트코드 파일)과 asm 파일(어셈블리 파일)이 만들어집니다. 그리고 [Download] 버튼에 의해 바이트코드 파일은 제어기로 다운로드 됩니다. 제어기에 다운로드 완료된 바이트코드는 플래시 메모리로 저장되어 영구적으로 실행될 수 있습니다.

하나 이상의 에러가 발생하면 빌드가 실패합니다. 그리고 asm 파일과 bin 파일은 생성되지 않습니다. 이때는 메시지 창의 에러 메시지를 참고하여 소스코드를 올바르게 수정합니다. 메시지 창의 에러 메시지를 더블 클릭하면 스크립트 편집 창에서 커서가 에러가 발생한 행으로 이동합니다.

13.10.3 실행

제어기에 다운로드 된 스크립트를 실행하기 위해서는 [Script Run/Stop] 버튼을 누릅니다. 스크립트가 실행 중이라면 버튼 옆의 녹색 LED가 켜지면서 실행 중인 상태를 표시합니다. 이때 [Script Run/Stop] 버튼을 한 번 더 누르면 스크립트의 실행이 중단됩니다.

실행 중 스크립트에서 사용된 전역 변수들의 값은 오른쪽 변수 모니터링 창에 실시간으로 업데이트됩니다. 이 기능은 스크립트 개발 및 디버깅을 편리하게 합니다.

13.10.4 Variables

Variables 그룹은 스크립트에서 값을 주고받기 위해 사용하는 32개의 변수를 읽고 쓸 수 있습니다. 이 변수는 플래시 메모리에 저장되지 않기 때문에, 제어기에 전원이 꺼지면 소실되는 값입니다.

사용자는 읽거나 쓰고자 하는 변수를 체크박스에서 선택합니다. 그러면 Read 칼럼에 값이 표시됩니다. 이 변수의 값을 바꾸고자 하면 Write 칼럼에 값을 적고 [Write] 버튼을 누르면 됩니다.

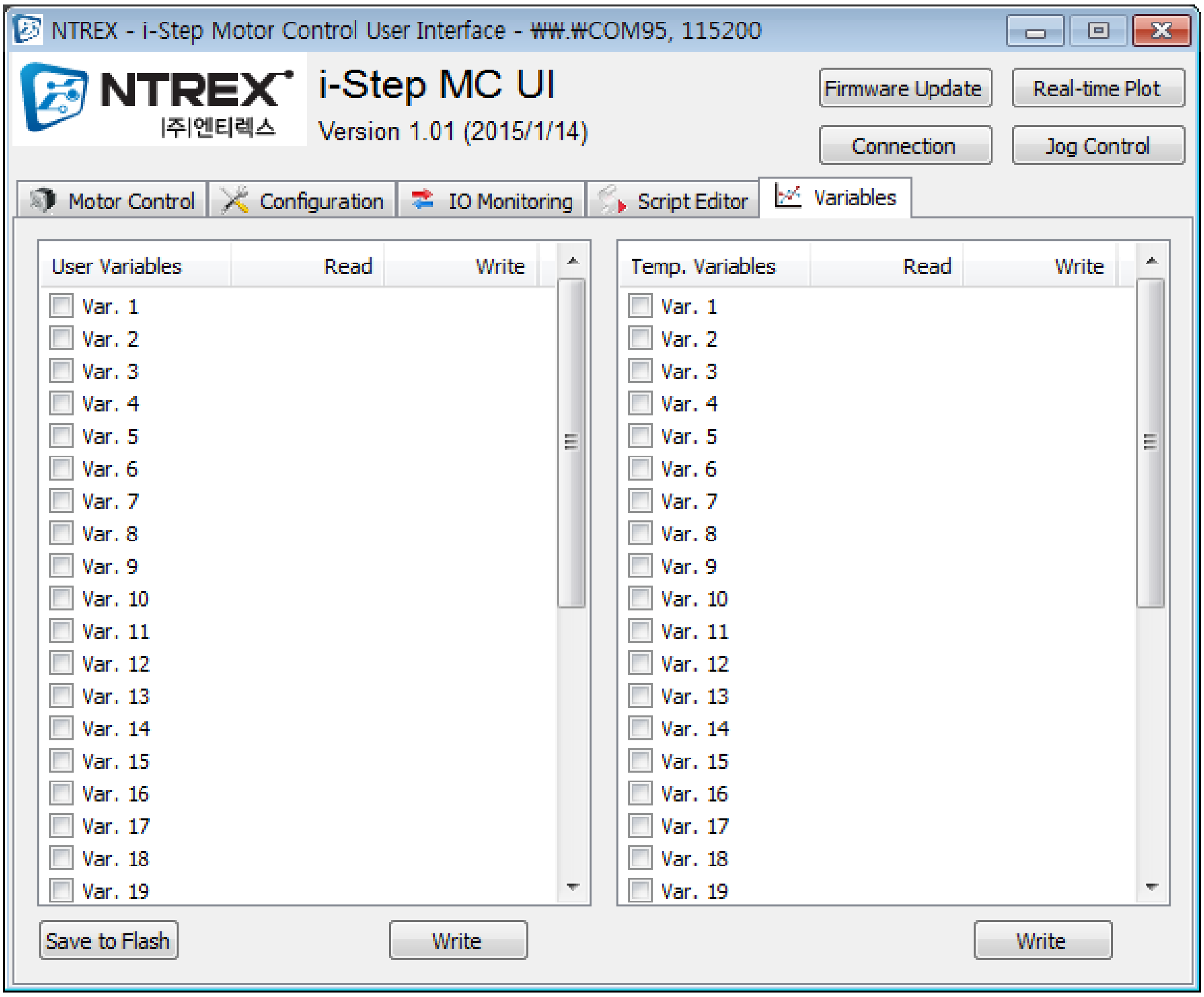


그림 13-35 Variables 탭 화면

14 관련 자료

아래 홈페이지에서 MoonWalker 시리즈 관련 문서와 예제 그리고 동영상 자료를 확인하실 수 있습니다.

- 엔티렉스:
  http://www.ntrexgo.com/
- 디바이스마트:
  http://www.devicemart.co.kr/
- MoonWalker 소개:
  http://www.ntrexgo.com/moonwalker-%EC%86%8C%EA%B0%9C
- MoonWalker 판매페이지:
  http://www.devicemart.co.kr/goods/list.php?category=006011008

문서 변경 이력

Data	Version	Charges
2015. 05. 01	1.00	- 첫 출시

제품의 보증

1. 본 제품은 엄정한 품질관리 및 검사과정을 거쳐서 만들어 진 제품입니다.
2. 제품 구입 후 6개월 이내에 제품 고장 발생 시에 무상으로 A/S를 해드립니다.
3. 정상적인 사용 상태에서 고장이 발생하였을 경우 보증기간 동안은 무상으로 A/S를 해드립니다.
4. 제품 보증기간이 경과한 후에 고장이 발생할 경우 유상으로 A/S를 해드립니다.
5. 보증기간 이내라 하더라도 본 보증 이내의 유상 서비스 안내에 해당되는 경우 서비스 따라 유상으로 A/S를 해드립니다.
6. 오용, 남용 및 인가되지 않은 인력에 의한 수리, 부적절한 보관상태 자연 재해로 인한 파손은 유상으로 A/S를 해드립니다.
7. 고객 변심 또는 구매 후 7일 이후에는 반품이 되지 않습니다.

회 사 명	(주)엔티렉스
본 사 주 소	인천 남구 주안동 5-38 (주)엔티렉스
전 화 번 호	070 - 7019 - 8887
팩 스 번 호	02 - 6008 - 4953
E – Mail	기술문의 - lab@ntrex.co.kr 영업문의 - stock@ntrex.co.kr