MoonWalker Series

Motor Controllers

MW DCS01 / MW DCS02

MW DCM01 / MW DCM02

MW DCL01 / MW DCL02

User's Manual

http://ntrexgo.com

(주)엔티렉스
NTREX
Nano Technology Revolution eXecute

※ 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.

(주)엔티렉스
NTREX
Nano Technology Revolution eXecute

머리말

저희 엔티렉스는 여러분께 MoonWalker 시리즈라고 이름 붙여진 이 강력한 모터제어기를 소개할 수 있게 된 것이 정말 기쁩니다. 아마 귀하가 1개 혹은 2개의 DC 모터를 좀 더 강력하게 구동하기 위해 저희 제품을 선택하였다면 아주 올바른 선택이라고 자신합니다. MoonWalker 시리즈는 기본적으로 전류/속도/위치 제어기를 탑재하고 있어서 로봇 암이나 주행로봇, 산업용 기기 등을 연구부터 실제 응용까지 아주 광범위한 적용이 가능합니다.

저희가 제공하는 Motor Control UI 유틸리티를 사용하여 PC에서 손쉽게 제어기의 각 종 설정을 조절할 수 있으며 제어기의 상태 확인을 그래프로도 확인할 수 있습니다. 특히 직관적이고 손쉬운 다양한 구동 방식을 제공하므로 본 제품의 구매 후 간단히 제품의 연결에 관련된 내용을 숙지하고, 일반적인 시리얼 통신 포트나 혹은 Mini-USB 단자를 이용하여 PC와 연결한 후 Motor Control UI 유틸리티를 통해 내용을 확인하면 됩니다. 이렇게 괜찮은 가격에 좋은 프로그램을 번들로 제공할 수 있는 것이 저희도 역시 즐겁습니다.

기존의 제어기라는 개념이 아닌 단순한 DC 모터 구동용 드라이버만 가지고 무선 RC 조종기나 조이스틱을 이용하여 로봇을 구동하면서 부족한 기능에 실망하셨다면 역시 저희 MoonWalker 시리즈가 제공하는 솔루션에서 그 해답을 얻을 수 있을 겁니다. 간편히 PC와의 시리얼 통신을 통해서 원하는 제어를 설정한 다음 RC 조종기나 조이스틱을 이용하여 구동할 수 있게 되어 있습니다. 특히 조종기 레버의 감도 조절이나 다양한 안전상의 설정들을 확인할 수 있습니다.

또한, MoonWalker 시리즈는 단순히 Incremental형 엔코더뿐만 아니라 Absolute형 엔코더와 Tachometer, Potentiometer 등의 다양한 센서를 적용할 수 있습니다. 특히 Motor Control UI 유틸리티가 제공하는 Mini-C 컴파일러로 구현된 스크립트 기능을 Digital input/output과 Analog/Pulse input 단자와 함께 응용하면 별도의 중계용 보드와 같은 장치 없이 직접 구현이 가능합니다.

마지막으로 저희 MoonWalker 시리즈를 선택해 주셔서 감사합니다. 그리고 그 선택에 절대 부족하지 않은 제품이라고 저희 엔티렉스는 자신합니다. 저희 제품에 대한 많은 응용 예제와 구동방법 및 기술 대응을 원하시면 엔티렉스 연구소 홈페이지(www.ntrexgo.com)에 있는 MoonWalker 카테고리를 방문하시기 바랍니다.

1 소개

본 사용자 매뉴얼는 ㈜엔티렉스의 MoonWalker 제어기를 올바르게 사용하기 위해 사용자가 알아야 할 내용들을 담고 있습니다.

이번 장에서는 사용자 매뉴얼에 대해 간단히 요약하고 제어기의 리셋 스위치와 표시등에 대해 설명합니다.

1.1 제품별 데이터시트

사용자 매뉴얼는 제어기의 데이터시트와 함께 제공됩니다. 데이터시트는 제어기 모델에 따른 정보를 가지고 있습니다. 데이터시트는 엔티렉스 연구소 홈페이지 (www.ntrexgo.com)에서 다운로드를 받을 수 있습니다.

데이터시트는 다음과 같은 정보를 제공합니다.

- 제어기의 제원
- 제어기의 기능
- 제어기의 배선도
- 제어기의 커넥터 정보
- 기구의 도면 및 치수

1.2 사용자 매뉴얼

사용자 매뉴얼는 MoonWalker 제어기가 공통적으로 가지는 내용을 제공합니다. 사용자 매뉴얼는 다음과 같은 순서로 구성되어 있습니다.

2장. 전원 및 모터 연결:
제어기에 전원(Power Source, 배터리 또는 파워서플라이)과 모터를 연결하는 방법과 연결 시 주의사항에 대해 설명합니다.

3장. 통신 포트 연결:
PC나 마이크로컨트롤러를 제어기의 통신 포트(USB, RS-232, CAN)에 연결하는 방법에 대해 설명합니다.

4장. 센서 및 액츄에이터 연결:
제어기의 입출력 포트에 센서(Sensor)와 액츄에이터(Actuator), 기타 액세서리를 연결하는 방법에 대해 설명합니다.

5장. 모터의 안전한 사용을 위한 기본설정:
모터의 제어 앞서 모터를 정격 범위 내에서 안전하게 구동하기 위한 구성 파라미터들을 올바르게 설정하는 것에 대해 설명합니다.

6장. 제어기의 구조:
제어기의 내부 구조에 대해 설명합니다. 제어기의 내부 구조를 파악하는 것은 제어기를 올바르게 운용하는데 꼭 필요한 내용이므로, 사용자는 본 장의 내용을 숙지하기 바랍니다

7장. 모터제어기:
모터제어기의 폐루프 위치/속도/전류 제어기와 프로파일 생성기, 각종 오브젝트, 외부와 데이터 교환을 위한 입출력 버퍼에 대해 설명합니다.

8장. 모터제어기 인터페이스:
모터제어기에 전달되는 명령과 피드백 신호의 처리에 대해 설명합니다.

9장. I/O 신호처리:
제어기의 I/O 포트 기능과 사용 용도 그리고 I/O 신호처리 방법에 대해 설명합니다.

10장. 제어기 오브젝트:
제어기의 제품 정보와 버전, 통신, 스크립트에 관련된 오브젝트들(상수와 명령, 상태, 구성 파라미터)에 대해 설명합니다.

11장. 모터제어기 오브젝트:
모터 제어부의 구성 파라미터 설정 및 명령과 상태에 관련된 오브젝트들에 대해 설명합니다.

12장. I/O 오브젝트:
제어기의 입출력 채널에 관련된 오브젝트들에 대해 설명합니다.

13장. 통신 프로토콜:
제어기의 오브젝트 값을 읽고 쓰기 위한 통신 프로토콜에 대해 설명합니다.

14장. Mini-C 스크립트 언어:
제어기에 내장된 Mini-C 스크립트 언어에 대해 설명합니다 언어의 특징과 능력 그리고 어떻게 사용자 스크립트를 사용하는지에 대한 정보를 제공합니다.

15장. 프로그램의 작성과 실행:
Mini-C 스크립트 언어를 이용해서 UI 유틸리티에 프로그램을 작성하여 제어기에 다운로드하고 실행하는 방법에 대해 설명합니다.

16장. Motor Control UI 유틸리티:
Motor Control UI 유틸리티 사용법에 대해 설명합니다. UI 유틸리티를 이용해서 제어기 기능 설정 및 변경 방법, 모터와 I/O 포트 모니터링 방법, Mini-C 스크립트 실행 방법 등 대한 정보를 제공합니다.

1.3 제어기 공통 사항

MoonWalker 제어기는 모델에 관계없이 리셋 스위치와 3개의 LED 표시등을 가지고 있습니다. 본 절에서는 제어기의 리셋 스위치와 LED 표시등에 대해 설명합니다.

1.3.1 리셋 스위치

리셋 스위치는 보통 제어기 전면이나 상판에 배치되어 있으며, 케이스 안에 숨어있기 때문에 가늘고 긴 송곳 같은 도구를 사용하여 누를 수 있습니다.

리셋 스위치는 다음 기능을 가집니다:

- Reset to Factory Default Configurations
- Software Reset
- Motor Power ON/OFF

제어기 전원을 켜면서 리셋 스위치를 5초간 누르고 있으면 제어기의 모든 설정이 제품 초기 설정값(Factory Default Value)으로 초기화됩니다. 이 기능은 제어기의 연결 설정을 잘못하였거나 제어기에서 스크립트의 실행 도중 오버플로우가 발생하여 PC에서 더는 제어기로 연결할 수 없을 때 사용할 수 있습니다.

제어기가 실행되고 있을 때 리셋 스위치를 4초간 누르면 제어기는 소프트웨어 리셋 됩니다. 이는 제어기의 전원을 끄고 켜는 것과 같습니다.

제어기가 실행되고 있을 때 리셋 스위치를 0.5초간 누르면 모터에 전력을 공급하는 상황과 차단하는 상황을 토글합니다(Motor Power ON/OFF 기능 토글). 듀얼 채널 제어기인 경우, 현재 Power ON 된 채널이 하나라도 있으면 모든 채널을 Power OFF하고, 모든 채널이 Power OFF 상태라면 모두 Power ON 합니다.

1.3.2 표시등

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

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

제어기에 전원을 투입 후, 3개의 LED가 모두 꺼져있거나 켜져 있는 경우는 모터제어기가 정상적으로 초기화되지 않은 상태입니다. 만일 모터제어기가 정상적으로 초기화 되었다면 동작상태를 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 다운로드 하고 실행

제어기에 새로운 펌웨어를 업데이트하다가 중단하고 제어기를 재 시작한 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 1-1 LED 상태 표시

Status	LED Color	Pattern	Description
동작 (Run)	청색	1000000000	모터에 전력이 공급되지 않는(Power OFF) 상태
		1111100000	모터에 전력이 공급되고(Power ON) 동작 가능한 상태 (듀얼 채널 제어기에서 두 채널 모두 전력이 공급될 때 이 상태로 표시 됨)
		1111111110	모터에 전력이 공급되고 0V가 아닌 전압으로 구동 중인 상태 (듀얼 채널 제어기일 경우 두 채널 중 하나만 구동 중이라도 이 상태로 표시됨)
폴트 (Fault)	적색	0000000000	제어기가 정상적으로 동작 중인 상태, 폴트가 발생하지 않은 상태
		1111100000	제어기에 폴트가 발생한 상태 (듀얼 채널 제어기일 경우 두 채널 중 하나라도 폴트 상태이면 이 상태로 표시됨)
통신 (Comm)	녹색 or 주황색	0	패킷을 주고받지 않는 상태
		1	마스터 PC가 CAN이나 RS-232, USB 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 전원 및 모터 연결

이 장에서는 제어기에 전원(Power Source, 배터리 또는 파워서플라이)과 모터를 연결하는 방법에 대해 설명합니다. 매뉴얼에서는 전원과 모터의 상세한 배선도, 커넥터 핀 배치도 정보를 제공하지 않습니다. 세부적인 내용은 해당 제어기의 데이터시트를 참조하십시오.

2.1 제어기의 연결 단자 설명.

전원과 모터는 스크류 터미널 블록(Screw terminal block)이나 전선을 통해 제어기에 연결됩니다. 다음 그림 2-1은 싱글 채널과 듀얼 채널 제어기의 연결 단자를 보여줍니다.

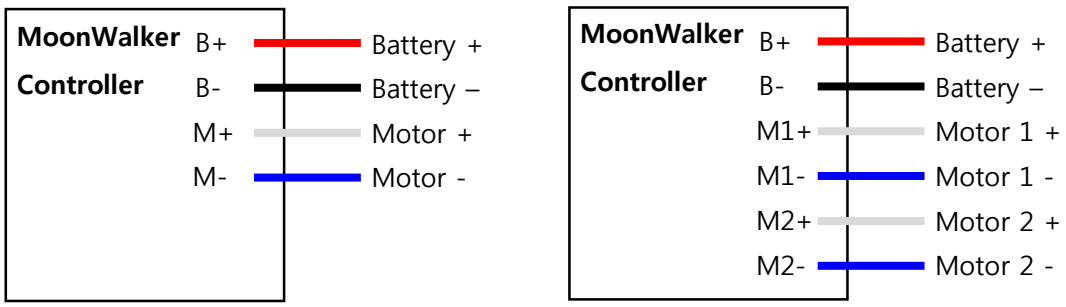

그림 2-1 싱글 채널과 듀얼 채널 제어기 연결 단자

제어기에 전원을 공급하는 단자는 케이스에 B+와 B-로 표시되어 있습니다. 배터리의 양극(Battery +)을 제어기의 B+ 단자(적색 전선)에 배터리의 음극(Battery -)을 제어기의 B- 단자(흑색 전선)에 연결합니다. 모터를 연결하는 단자는 M+와 M-로 표시되어 있습니다. 듀얼 채널 제어기의 경우 M1과 M2로 나뉘어 있습니다. 모터의 양극(Motor +) 단자를 제어기의 M1+(또는 M2+, 흰색 전선)에 모터의 음극(Motor -) 단자를 제어기의 M1-(또는 M2-, 청색 전선)에 연결합니다.

※주의※ 제어기는 높은 전력을 사용하는 전자 제품(장치)입니다. 전원의 극성을 잘못 연결하거나 잘못된 주변 회로 설계로 인해 제어기 및 주변 회로에 심각한 손상이나 화재가 발생할 수 있습니다. 특히 배선 오류로 인한 문제는 매우 심각한 결과를 초래할 수 있으며 제품 보증이 적용되지 않습니다. 따라서 사용자께서는 본 장을 숙지하신 후 구매한 제품을 사용하시기 바랍니다.

2.1.1 듀얼 채널의 동기화 구동

제어기 모델 중 MW DCL01은 MW DCL02과 외형이 같습니다. 하지만 MW DCL01 모델은 싱글 채널 제어기로 단일 부하를 두 배의 전력으로 구동할 수 있습니다. 다음 그림 2-2와 같이 M1+ 와 M2+ 단자의 출력을 묶어서 Motor +에 연결하고 M1-와 M2- 단자의 출력을 묶어서 Motor – 단자에 연결합니다.

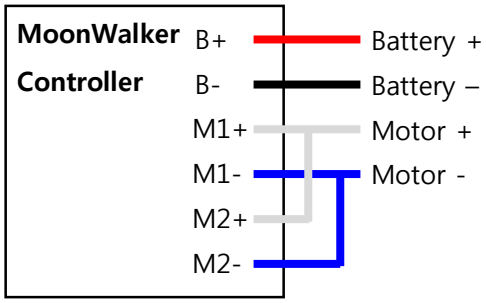

그림 2-2 싱글 채널 제어기의 동기 연결

모터로의 출력은 듀얼 채널이지만 싱글 채널 제어기인 경우, 각 채널의 출력은 서로 일치해야 합니다. 이러한 방식으로 동작하려면, 제어기의 각 채널을 스위칭 하는 MOSFET 소자가 완벽하게 동기화되어야 합니다. 만일 두 채널이 동기화되지 않았다면, 한 채널의 전류가 다른 채널로 흘러 들어가 제어기를 파괴하게 됩니다.

MW DCL01와 같은 싱글 채널 버전의 제어기는 두 채널이 동기화되어 스위칭 되도록 제어기 내부의 하드웨어가 구성되어 있으며 1번 채널의 모터에만 구동 명령을 내릴 수 있습니다.

※주의※ 모터의 구동 전력을 두 배로 증폭하기 위해, MW DCL01 이외의 듀얼 채널 제어기를 절대로 그림 2-2와 같이 연결하면 안됩니다. MW DCL01 모델만 제어기 내부에 두 채널이 동기화되어 스위칭 되도록 설계되어 있습니다.

2.2 전원 연결

이 절은 제어기와 전원을 보호하기 위한 회로 설계와 배선 방법 그리고 안전하게 전원을 ON/OFF 하는 방법에 대해 설명합니다.

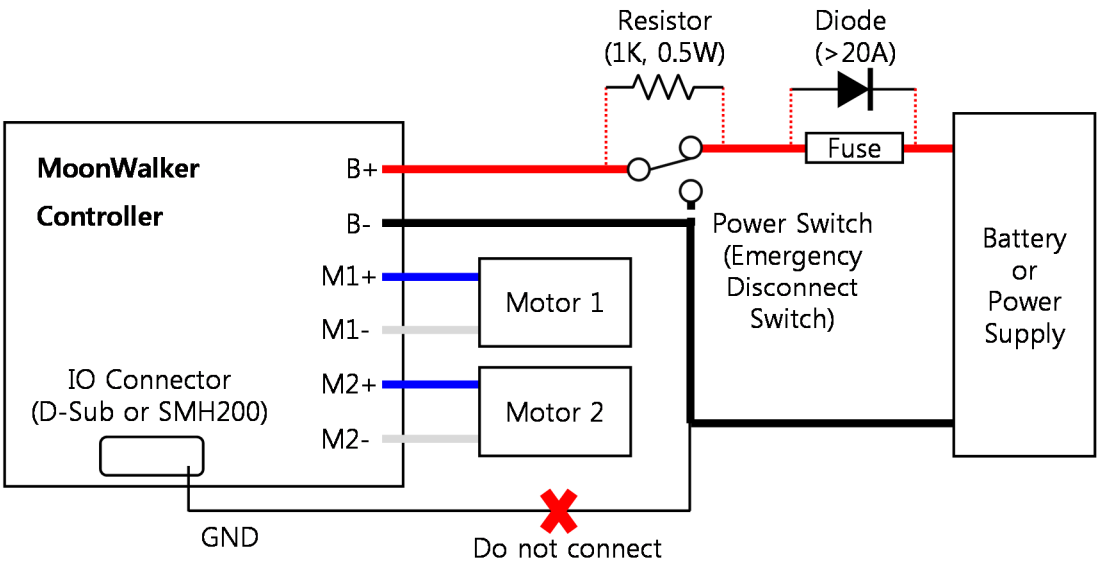

그림 2-3 제어기 및 전원 보호 회로 구성

그림 2-3 제어기 및 전원 보호 회로 구성에서 실선으로 표시된 연결은 필수사항이고 점선으로 표시된 연결은 권장사항입니다. I/O 커넥터는 제어기 모델에 따라 D-Sub과 SMH200 커넥터로 구분됩니다. 제어기의 데이터시트를 참조하기 바랍니다.

※주의※ 그림 2-3은 제어기를 사용하기 위한 기본적인 전원 연결에 대한 회로도 입니다. 이는 단순한 참조 예시일 뿐이며, 사용자는 제품을 사용할 환경에 맞게 보호 회로를 설계하여야 합니다. 다시 한번 강조하자면, 그림 2-2와 같은 보호 회로 구성이 모든 환경에 대해 정상적으로 대응 또는 동작하는 것은 아닙니다. 사용자의 환경에 맞게 회로를 구성하기 바랍니다. 사용자의 잘못된 연결로 인한 제어기의 손상 및 파손은 제품 보증이 적용되지 않습니다.

2.2.1 I/O 커넥터 그라운드 처리

D-Sub 또는 SMH200 커넥터의 입출력 핀에 스위치 또는 액츄에이터를 연결하는 경우, 커넥터 GND와 전원의 음극(-)을 연결하면 안됩니다. 만약 연결되어 있다면 모터 구동에 의한 노이즈가 I/O 포트의 신호를 교란하고 I/O 커넥터에 연결된 장치로 흘러 들어 갈 수 있습니다. 극단적으로는 연결된 장치를 파손 또는 손상시킬 수 있습니다.

※주의※ 사용자는 반드시 D-Sub 또는 SMH200 커넥터의 GND와 전원의 음극(-)을 분리해야 합니다. 그리고 주변회로 설계에서 숨은 연결을 찾아 제거해야 합니다. 또한, 전원의 음극을 통해 스위치나 센서 같은 소자들이 I/O 커넥터에 연결되지 않도록 주의해야 합니다.

2.2.2 퓨즈와 다이오드 삽입

퓨즈는 주로 과전압과 과전류로 인한 전원 배선 및 배터리의 손상을 보호합니다. 손상을 방지하기 위해 안전 조치로 ATO 또는 MAXI 시리즈보다 높은 전류의 퓨즈(Fuse) 사용을 권장합니다. 퓨즈를 선정할 때는 안전을 위해 제어기에 연결된 모터의 용량 합에서 80~90% 사이의 용량을 선택하는 것을 권장합니다.

참고로 40A 이상의 퓨즈는 일반적으로 끊어지는 속도가 느리기 때문에 제어기를 보호하는데 한계가 있습니다. 퓨즈가 끊어진 경우에도 배터리로의 반환 경로를 보장하기 위해서 고전류 다이오드를 퓨즈와 병렬로 연결해야 합니다.

※주의※ 일반적으로 퓨즈를 전원에 연결하면 제품을 안전하게 보호할 수 있다고 생각합니다. 하지만 이는 잘못된 생각입니다. 퓨즈는 일반적으로 탈 때까지 약간의 시간이 걸리기 때문에 일시적으로 퓨즈를 통해 높은 전류가 흐를 수 있습니다. 이러한 경우, 모터(구동 제품)의 가속이나 동작 시 일시적으로 높은 전류를 끌어다 쓸 수 있는 장점도 있지만, 반대로 퓨즈가 제어기를 제대로 보호하지 못하기도 합니다.

2.2.3 전원 스위치 및 비상 정비 버튼의 사용

파워서플라이와 같이 전원 스위치가 없는 배터리를 전원으로 사용할 때는, 그림 2-3에서와 같이 제어기의 전원을 ON/OFF 할 수 있는 전원 스위치의 연결을 권장합니다. 또는 비상시 전원을 차단하기 위한 비상 정지 스위치(Emergency Disconnect Switch)의 연결을 권장합니다.

그림 2-4 비상 정지 버튼

전원 스위치나 비상 정지 스위치 양단에는 1KΩ, 0.5W 저항을 연결하기를 권장합니다. 이 저항은 스위치가 켜질 때 스위치 내부에서 발생하는 전기적 아크(electric arc)를 방지하는 역할을 합니다.

※주의※ 모터가 고속으로 회전하는 동안에는 전원 스위치를 ON/OFF하지 마십시오. 제어기가 손상될 수도 있습니다.

2.3 모터 연결

제어기 모델에 따라 최대 두 개의 모터를 연결할 수 있습니다. 모터는 제어기의 M+와 M- 단자(듀얼 채널 제어기인 경우는 M1+와 M1- 그리고 M2+와 M2- 단자)에 연결합니다. 모터를 연결한 후에는 Motor Control UI 유틸리티를 사용해서 최소한의 전압을 인가해 봅니다. 이때 모터가 원하는 방향으로 회전하는지 확인합니다. 만약, 모터가 반대로 회전할 경우에는 구동을 정지하고 모터의 양(Motor +)극과 음(Motor -)극 단자를 교차하여 연결합니다.

※주의※ 모터의 극성이 반대로 연결되면 모터가 역방향으로 회전하게 됩니다. 이런 상황에서 폐루프 위치 또는 속도 제어기가 동작하면 모터는 전원을 끌 때까지 최고속도로 회전하며 제어가 불가능한 상황이 됩니다.

※주의※ 제어기보다 높은 정격 전류의 모터를 사용하지 마십시오.

2.3.1 전선 길이 제한

제어기는 모터에 공급되는 전력을 높은 주파수로 ON/OFF 스위칭하여 조절합니다. 이러한 주파수는 전선의 인덕턴스 성분에 의해 의도치 않는 기생 RF 방출(Parasitic RF emission), 진동(Ringing), 과전압 피크(Overvoltage peak) 등을 만들어 냅니다. 만일 전선이 길어지면 전선의 인덕턴스가 높아지고 더 많은 노이즈를 만들어냅니다. 따라서 제어기와 모터를 연결하는 전선은 최대한 짧게 배선하는 것이 좋습니다.

2.4 전원 및 모터 연결 시 주의사항

2.4.1 회생 전류에 의한 손상

모터에 가해지는 전압에서의 속도보다 더 빨리 회전하거나(내리막길에서 구동), 감속하는 경우에는 모터는 발전기 역할을 합니다. 이때 회생 전류가 전원으로 역류해서 제품의 손상을 가져올 수 있습니다.

따라서 제어기의 전원으로는 회생 전류를 다시 저장할 수 있는 충전 가능한 배터리 사용을 권장합니다. 만약 배터리가 아닌 파워서플라이를 사용할 경우 회생 전류는 파워서플라이로 역류하여 전원공급회로에 손상을 줄 수 있기 때문에 사용에 주의해야 합니다.

2.4.2 파워서플라이 사용시 주의사항

스위칭 방식의 파워서플라이(Switching power supply)를 사용할 경우, 모터에서 발전되는 전력에 의해 전원으로 전류가 역류해서 손상을 일으킬 수 있기 때문에 주의해야 하며, 다음과 같은 보호 단계들이 고려되어야 합니다:

1. 출력 전압보다 높은 전압이 역으로 가해질 때에도 파손되지 않는 전원공급장치를 사용해야 합니다.
2. 전원공급장치와 병렬로 저항성 부하(Load Resistor)를 설치하고 모터에서 발전되는 전력에 따라 부하를 켜고 끌 수 있는 회로를 설계합니다(아래 그림 2-5 참조).

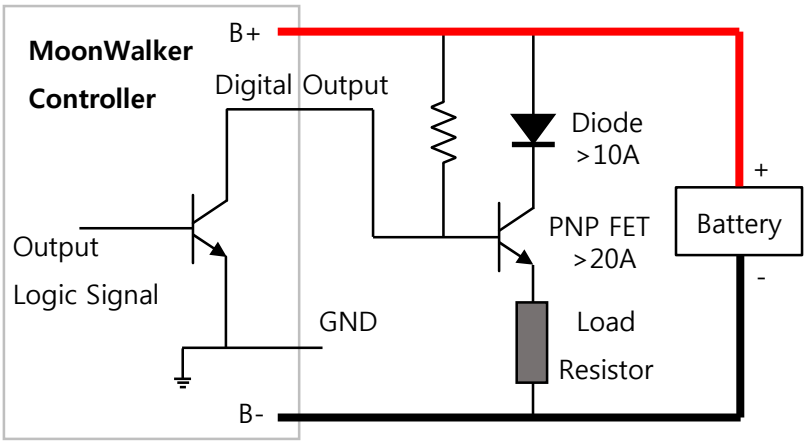

그림 2-5 Shunt load 회로

3. 전원공급장치의 출력과 병렬로 배터리를 연결합니다. 배터리는 회생 전류를 담을 수 있는 저수조 역할을 합니다. 배터리를 처음 연결할 때는 완전히 충전된 배터리를 사용해야 하며, 전원공급장치로 전류의 역류를 방지하기 위해 출력 단에 디커플링 다이오드를 연결합니다 (아래 그림 2-6 참조).

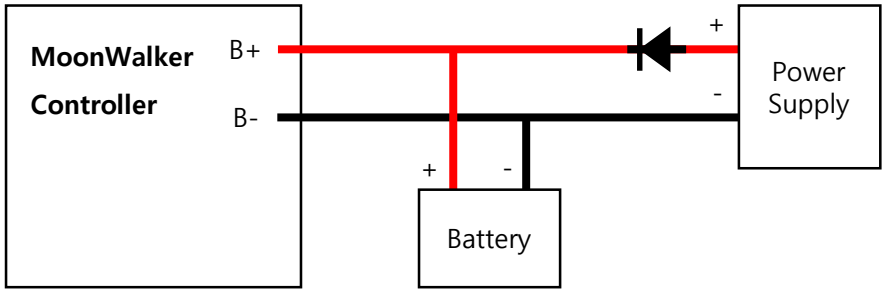

그림 2-6 파워서플라이와 배터리 병렬 연결

2.4.3 배터리 충전시 주의사항

충전식 배터리를 사용할 경우 그림 2-2와 같이 제어기 및 전원 보호 회로를 설계한 후, 전원 스위치를 끈 상태에서 배터리를 충전해야 합니다. 만약 제어기와 연결된 상태에서 충전할 경우, 제어기 및 주변 회로에 배터리 전압 이상의 전압이 인가될 수 있습니다.

2.4.4 전기 노이즈 감소 방법

전기적 노이즈를 감소시키기 위해 제어기 내에 여러 회로들이 설계되어 있지만, 제어기를 모터와 전원에 연결하여 사용할 때는 노이즈를 저감하기 위한 추가적인 작업이 필요합니다. 다음은 전기적 노이즈를 감소시키는 방법들 입니다:

- 전선은 가능한 짧게
- 전선을 페라이트 코어(Ferrite cores)에 감기
- 모터 단자에 스너버(Snubber) RC 회로 추가
- 제어기와 전선, 배터리를 외부와 접촉이 없는 금속 프레임에 설치

3 통신 포트 연결

이 장에서는 PC(Personal Computer)나 마이크로컨트롤러(Microcontroller)를 제어기의 통신 포트에 연결하는 방법에 대해 설명합니다. 제어기는 모델에 따라 시리얼과 CAN 통신을 사용할 수 있습니다. 여기서 시리얼 통신은 USB(VCP; Virtual COM Port)와 RS-232를 말합니다. 제어기 모델에 따라 지원되는 통신 포트와 통신선 결선 정보는 데이터시트를 참조하기 바랍니다.

통신 포트와 관련 있는 제어기의 구성 파라미터(Configuration Parameter) 오브젝트는 다음과 같습니다:

- **device_id - Device ID**
- **can_br - CAN Bitrate**
- **serial_bps - Serial Baudrate**
- **serial_witchdog - Serial Watchdog**

제어기의 구성 파라미터 오브젝트에 대해서는 "6.4 오브젝트"를 참고하기 바랍니다.

3.1 USB(VCP) 연결

사용자가 PC를 사용하여 제어기의 구성(Configuration)을 설정하고 운용 하는 가장 간단한 방법은 제어기와 PC 간에 USB 연결을 구성하고 PC에서 Motor Control UI 유틸리티를 사용하는 것입니다.

제어기의 Mini-USB (B type) 단자에 USB 케이블을 연결합니다. 그러면 PC는 새로운 하드웨어를 감지하고 USB 장치 드라이버를 설치합니다. 그리고 PC에 가상 시리얼 포트(VCP)가 자동으로 설치됩니다. 정상적으로 설치가 끝나면 PC에서 Motor Control UI 유틸리티를 실행하여 제어기를 설정하고 운용 할 수 있습니다.

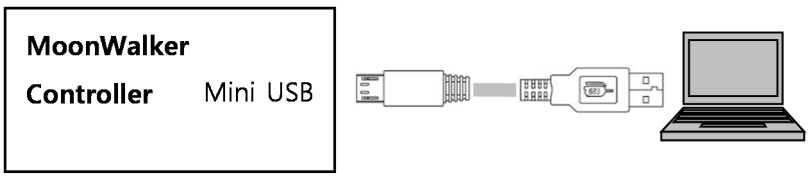

그림 3-1 제어기와 USB(VCP) 연결

USB의 VCP는 PC의 장치관리자에서 다른 시리얼 COM 포트와 동일하게 표시됩니다. VCP를 통해 시리얼 통신을 기반으로 하는 다양한 응용프로그램(예, HyperTerminal)을 사용할 수 있습니다. 또한 VCP는 사용자가 응용 소프트웨어를 쉽게 작성할 수 있도록 합니다. COM 포트를 열고 시리얼 데이터를 주고받는 것은 여러 프로그래밍 언어에서 잘 문서화 되어있으며, 전통적으로 제어기와 통신하기 위한 가장 기본적인 방법입니다.

제어기가 USB를 통해 PC에 처음으로 연결되면 Window는 무작위로 COM 포트 번호를 할당하게 됩니다. 하지만 Motor Control UI 유틸리티에서 장치를 검색하는 기능을 사용하면, 열려있는 모든

COM 포트를 검색하고 제어기를 감지해서 자동으로 연결해줍니다. 사용자가 제어기와 통신하는 소프트웨어를 직접 작성하는 경우, COM 포트 할당 번호가 바뀔 수 있다는 것을 고려해야 합니다.

3.1.1 시리얼 포트 설정

시리얼 포트 설정은 USB(VCP)와 RS-232에 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 USB(VCP)와 RS-232 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 변경이 불가능합니다. 사용자는 시리얼 통신을 사용하는 PC 또는 마이크로컨트롤러의 통신 파라미터를 제어기와 일치하도록 설정해야 합니다.

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

3.2 RS-232 연결

제어기는 RS-232 시리얼 통신을 통해서 PC에 접속할 수 있습니다. 이 연결은 제어기에 명령을 전송하고 제어기에서 다양한 상태 정보를 읽어오는데 사용됩니다. USB 연결에 비해 노이즈에 강하고 선의 길이 제한이나 통신 오류 발생시 복구 가능성이 높기 때문에, 제어기가 실제 현장에 배치되는 경우 RS-232 연결을 권장합니다.

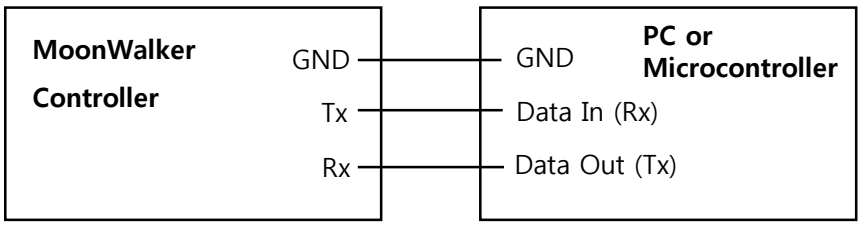

그림 3-2 RS-232 연결

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.

제어기 모델에 따라 커넥터가 D-Sub 또는 SMH200 커넥터로 구성되어 있습니다. 자세한 내용은 제품의 데이터시트를 참조하기 바랍니다.

※주의※ 전기적인 외란(electrical disturbance)이 발생하였을 때 USB 프로토콜은 RS-232보다 복구될 가능성이 적습니다. 따라서 제어기 구성 설정 및 모니터링에는 USB를 사용하고, 현장 배치에서는 RS-232를 사용하는 것이 좋습니다.

3.2.1 RS-232 케이블 연결

RS-232 연결 케이블은 일부 제어기의 액세서리로 판매가 되고 있습니다. 케이블이 액세서리로 제공되지 않는 제어기 모델의 경우 또는 다른 이유로 케이블을 직접 제작하여 사용하여야 할 경우, DB9 암 커넥터와 3선 케이블을 사용하여 쉽게 제작할 수 있습니다.

PC의 RS-232 포트는 보통 DB9 수(male) 커넥터로 구성됩니다. 그래서 제어기로부터 연결되는 신호 선들은 DB9 암(female) 커넥터에 결선 됩니다. 다음 그림을 참조하여 제어기의 Rx, Tx, GND 핀을 DB9 암 커넥터의 핀으로 연결합니다.

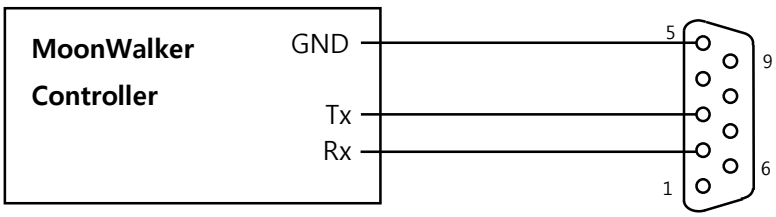

그림 3-3 제어기와 DB9 Female 커넥터 연결

여기서 커넥터의 핀 번호를 혼동하지 않아야 합니다. 상기 그림의 핀 번호는 전면에서 커넥터를 보는 것을 기준으로 합니다. 대부분의 커넥터 브랜드는 플라스틱 성형 부분에 핀 번호가 적혀 있습니다.

3.2.2 RS-232 케이블 확장

RS-232 연장 케이블은 대부분의 전자부품 취급점에서 구할 수 있습니다. 하지만, 쉽게 DB9 수 커넥터와 DB9 암 커넥터, 3선 케이블을 사용하여 제작할 수 있습니다. 제어기의 25핀 핀 배치와 일치하지 않는 상용 9-PIN TO 25-PIN 컨버터를 사용하지 마십시오.

이러한 구성 요소는 전자부품 취급점에서 구할 수 있습니다. CAT5 네트워크 케이블이 권장되고, 케이블 길이는 30m까지 연장할 수 있습니다.

RS-232 케이블을 확장할 때는 다음 두 케이블을 구분하여야 합니다:

- 연장 케이블 - 케이블 양단이 DB9 수 커넥터와 DB9 암 커넥터로 구성
- 크로스 케이블 - 케이블 양단이 DB9 암 커넥터와 DB9 암 커넥터로 구성

연장 케이블의 한쪽 끝은 PC에 연결하고 다른 한쪽 끝에는 제어기로부터의 RS-232 케이블 커넥터에 연결하게 됩니다. 그러므로 연장 케이블은 PC의 커넥터와 동일한 모양의 커넥터를 제공하기 때문에 PC에 연결하는 것과 동일하게 연결하면 됩니다.

하지만 크로스 케이블은 PC와 PC를 직접 연결하는데 사용되기 때문에 케이블 내부에서 Rx핀과 Tx핀이 교차 연결되어 있습니다. 이러한 케이블의 양쪽 끝은 동일한 DB9 암 커넥터로 구성됩니다. 다음 그림을 참고하여 DB9 수 커넥터와 DB9 암 커넥터의 연결을 혼동하지 마시기 바랍니다.

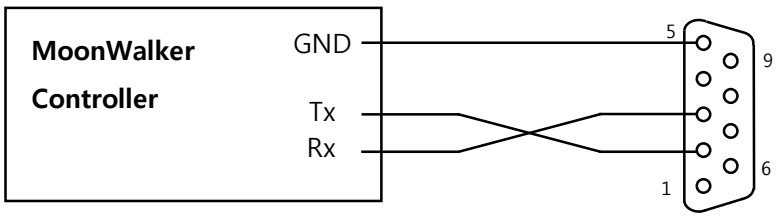

그림 3-4 제어기와 DB9 Male 커넥터 연결

3.3 CAN 연결

CAN(Control Area Network)은 차량용 네트워크 시스템으로 개발된 통신입니다. 주로 마이크로컨트롤러 간의 통신을 위해 설계된 시리얼 네트워크 통신 방식입니다. CAN은 1:1 통신뿐 아니라 멀티 마스터/슬레이브 통신이 가능합니다. 특히 장거리 통신이 가능하며 하나의 마스터 PC를 통해 여러 개의 슬레이브 제어기들을 제어할 수 있습니다. 단일 CAN 버스에서 최대 127개의 제어기를 최대 1Mbit/s의 속도로 함께 사용할 수 있습니다.

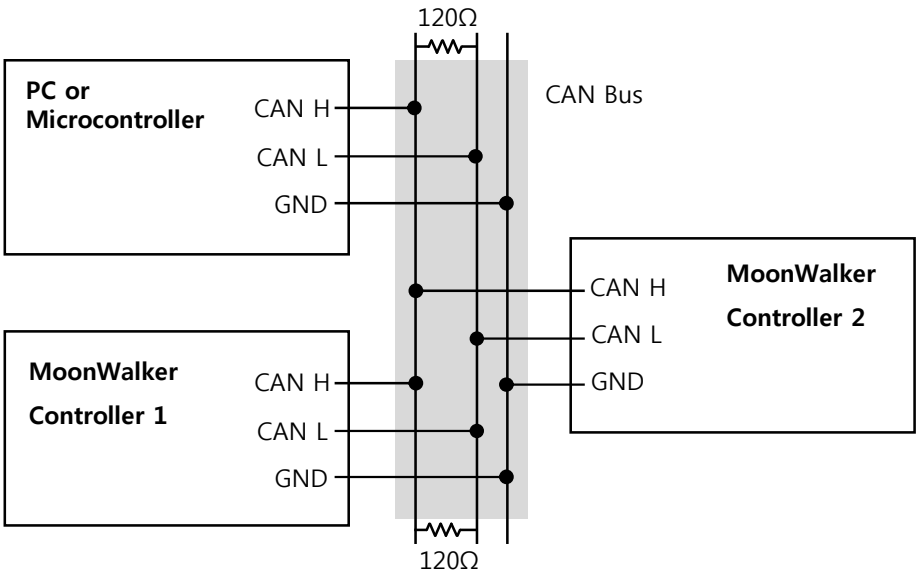

그림 3-5 CAN 통신을 위한 연결

CAN 통신은 2가닥의 CAN H(CAN Bus High)와 CAN L(CAN Bus Low) 권선으로 이루어진 버스 형태로 되어 있고 이 버스를 통해서 통신합니다. 상기 그림 3-5와 같이 제어기의 CAN H는 버스의 CAN H에 제어기의 CAN L은 버스의 CAN L에 연결합니다.

이때 CAN 버스의 종단에 120Ω 저항을 연결해야 합니다. 이유는 CAN H와 CAN L 사이의 전압을 2V로 유지해 줌으로써 버스라인 종단에서 데이터가 중첩될 가능성이 있는 메아리 현상을 제거해 줍니다.

3.3.1 CAN 통신속도 설정

CAN 버스를 공유하는 모든 제어기(장치)의 통신 속도는 동일하게 설정되어야 합니다. 제어기는 제품 초기 설정 값(Factory Default Value)으로 CAN 통신 속도(Bitrate)가 1Mbps로 설정되어 있습니다. 사용자는 **'CAN Bitrate'** 파라미터 설정을 통해 제어기의 통신 속도를 변경할 수 있습니다. 파라미터 설정 작업을 쉽게 하기 위해 Motor Control UI 유틸리티를 사용하십시오.

만일 CAN 연결을 사용하고자 할 때, 먼저 제어기에 USB 연결을 통해 UI 유틸리티로 CAN 통신 속도를 서로 동일하게 변경해야 합니다.

3.3.2 CAN 장치 ID 설정

CAN 버스에는 하나 이상의 제어기 및 다른 장치가 연결될 수 있습니다. 연결된 장치들은 자신의 ID를 메시지에 담아 네트워크에 전송합니다. 만일 둘 이상의 장치가 동일한 ID를 사용한다면 메시지를 수신하는 장치는 메시지의 발신자를 구분할 수 없게 되어, 장치간 메시지의 교환에 혼선을 초래합니다.

제어기의 장치 ID(Device ID)는 공장 출하시 모두 1로 설정되어 있습니다. 사용자는 **'Device ID'** 파라미터 설정을 통해 제어기의 장치 ID를 변경할 수 있습니다.

CAN 연결을 하고자 할 때, 먼저 제어기에 USB 연결을 통해 Motor Control UI 유틸리티로 장치 ID를 서로 중복되지 않게 변경해야 합니다. Motor Control UI 유틸리티는 CAN 통신으로도 제어기에 연결이 가능합니다. CAN 네트워크에 장치 ID를 변경하고자 하는 제어기를 하나만 연결한 상태에서 UI 유틸리티로 장치 ID를 변경합니다.

3.4 통신 포트의 기능

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

사용자가 제어기를 설정하고 제어기를 간단하게 운용 및 모니터링 하기 위해 PC에서 제어기로 USB 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(공급 전압, 소비 전류 등) 읽기
- 모터제어기의 구성 파라미터를 설정
- 제어기의 I/O 입력을 읽고 출력을 쓰기

그리고 제어기의 RS-232, CAN 연결을 통해 제어기를 운용하는 마스터 PC나 PLC, 마이크로컨트롤러가 연결될 수 있습니다.

3.4.1 명령의 충돌

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

다중 통신의 예로 다음과 같은 상황을 고려해 볼 수 있습니다: USB 포트로는 PC가 연결되어 사용자가 Motor Control UI 유틸리티를 통해 제어기를 설정하고 간단한 구동 명령을 내리면서 모니터링하고 있습니다. 그리고 RS-232나 CAN 포트로는 마스터 PLC가 연결되어 제어기에 구동 명령을 내리고 상태를 모니터링하고 있습니다.

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

만일 서로 다른 통신 포트들로부터 수신되는 명령이 서로 다른 모터 채널에 대한 명령일 때는 문제가 없습니다. 예를 들자면, RS-232 포트에서는 채널 1의 모터를 구동하는 명령이 수신되고 CAN 포트에서는 채널 2의 모터를 구동하는 명령이 수신되는 경우입니다.

※ 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.

3.4.2 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, USB 또는 RS-232, CAN 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.

USB 또는 RS-232, CAN 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.

4 센서 및 액츄에이터 연결

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

4.1 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. 그리고 포텐셔미터로부터 모터의 절대 위치를 피드백 받는데도 사용합니다.

펄스 입력(Pulse Input):
RC 조종기와 수신기로부터 모터에 내려지는 구동 명령을 읽어 들입니다. 그리고 PWM 출력이 가능한 절대 엔코더를 연결해서 사용할 수 있습니다.

제어기의 입출력 포트는 D-Sub과 SMH200 커넥터로 구성되어 있습니다. 주로 저전압, 저전류 감지를 위한 연결에 사용됩니다. 사용자는 환경과 목적에 맞게 입출력 포트를 구성하여 사용해야 합니다.

입출력 포트의 전압과 전류 사양, 입력과 출력 포트 수, 커넥터 위치 등은 제어기의 데이터시트를 참조하기 바랍니다. 제어기 모델에 따라 입출력 포트가 지원되지 않는 제품이 있습니다. 사용자는 구매하신 제어기의 데이터시트를 확인해 주시기 바랍니다.

입출력 포트가 지원되지 않는 제품 모델은 다음과 같습니다:

- MoonWalker MW DCS01
- MoonWalker MW DCS02

4.2 디지털 출력 포트

디지털 출력(Digital Output) 포트는 모터의 구동 상태(Power ON/OFF 상태, 회전 방향, 과전압 상태) 및 FET의 과열 상태를 외부에 알리는데 사용합니다.

제어기 모델에 따라 0~2개의 디지털 출력 포트를 가지고 있습니다. 디지털 출력 포트 말단에는 마이크로프로세서의 출력 신호에 비해 높은 전력의 부하를 구동하기 위해 MOSFET가 사용됩니다. 다음은 MOSFET 소자 정보입니다:

- Continuous drain source voltage : 60V
- On-state resistance : 500mΩ
- Nominal load current (VIN = 5V) : 1.3A

MOSFET는 60V에 1.3A에서 구동할 수 있는 Open Drain MOSFET 출력으로, 활성화된 경우 출력은 GND로 연결됩니다. 따라서 부하의 한쪽 끝은 출력에 연결하고 다른 한쪽 끝은 24V 배터리 같은 양의 전압 출력에 연결해야 합니다.

※ ZXMS6004FF MOSFET 데이터시트에는 60V, 1.3A까지 구동할 수 있다고 기재되어 있지만, 사용자와 제어기의 안전을 위해 50V, 1A까지 사용하시길 권장합니다.

4.2.1 비유도성 부하 연결

저항이나 LED 같이 내부에 인덕턴스 성분을 가지지 않는 비유도성 부하(Resistive loads = Non-inductive loads)는 다음 그림 4-1과 같이 연결합니다.

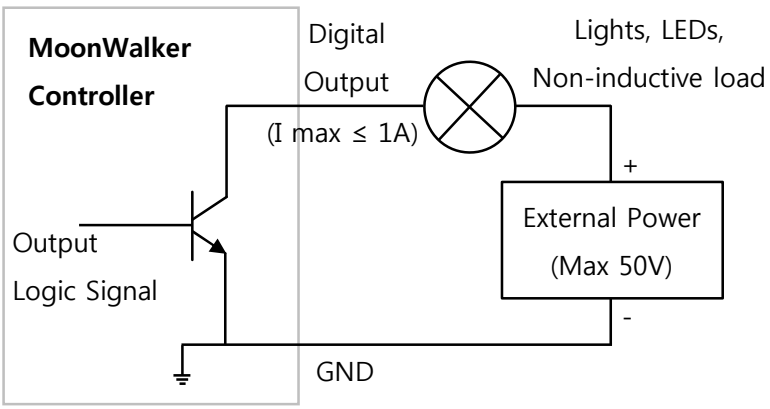

그림 4-1 비유도성 부하 연결

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

4.2.2 유도성 부하 연결

릴레이, 솔레노이드 밸브, 소형 모터와 같이 내부에 인덕턴스 성분을 가지는 유도성 부하(Inductive loads)는 다음 그림 4-2와 같이 연결합니다. 이때 유도성 부하 소자가 스위칭 할 때 유기된 고전압 스파크에 의해 트랜지스터가 파괴될 수 있으니 보호 다이오드를 사용하시기 바랍니다.

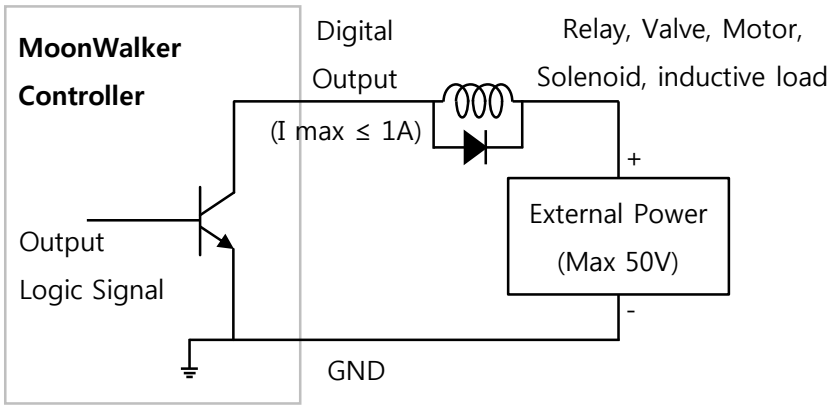

그림 4-2 유도성 부하 연결

4.3 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 입력 버퍼(Input Buffer)를 가지고 있습니다. 그리고 디지털 입력 단자와 입력 버퍼 사이에 직렬로 22Ω 저항이 연결되어 있습니다.

다양한 연결 방법이 있지만 다음과 같이 풀업(Pull-Up)과 풀다운(Pull-Down) 스위치를 연결하는 방법에 대해 설명합니다.

4.3.1 풀업 스위치 연결

다음 그림 4-3에서와 같이 디지털 입력 포트에 풀업(Pull-Up) 스위치를 사용하는 경우, 외부에 1K~10KΩ의 풀다운(Pull-Down) 저항을 연결합니다. 풀다운 저항을 사용하지 않고 스위치만 연결된 경우, 스위치가 꺼져 있을 때 디지털 입력은 플로팅 상태가 되며, 올바른 값을 읽을 수 없습니다. 풀다운 저항은 노이즈를 방지하기 위해서도 사용됩니다.

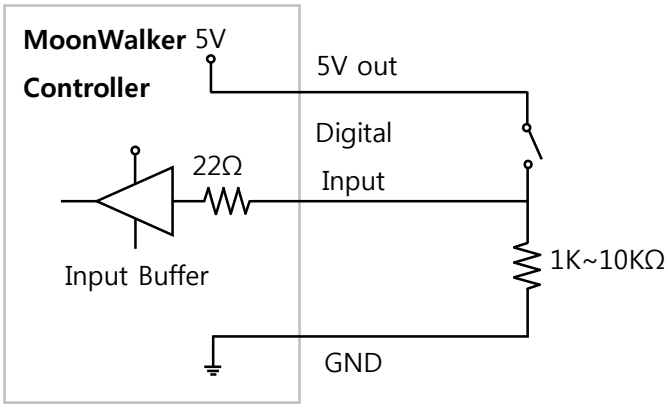

그림 4-3 풀업 스위치 연결

4.3.2 풀다운 스위치 연결

다음 그림 4-4에서와 같이 디지털 입력 포트에 풀다운(Pull-Down) 스위치를 사용하는 경우, 외부에 1K~10KΩ의 풀업(Pull-Up) 저항을 연결합니다. 풀업 저항은 "4.3.1 풀업 스위치 연결"에서 풀다운 저항과 같은 이유로 꼭 사용되어야 합니다.

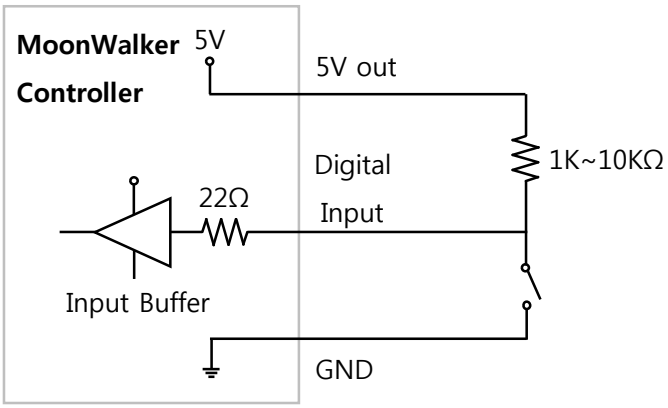

그림 4-4 풀다운 스위치 연결

4.4 아날로그 입력 포트

제어기의 아날로그 입력(Analog Input) 포트는 전압을 출력하는 아날로그 센서를 연결해서 사용하거나 포텐셔미터(Potentiometer)나 타코미터(Tachometer) 같은 아날로그 신호를 출력하는 센서를 연결해서 사용합니다. 또한, 써미스터를 연결해서 외부 온도를 감지할 수도 있습니다.

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.

4.4.1 전압을 출력하는 아날로그 센서 연결

제어기에 전압을 출력하는 아날로그 센서를 연결할 때는 그림 4-5와 같이 아날로그 입력에 직접 연결하면 됩니다.

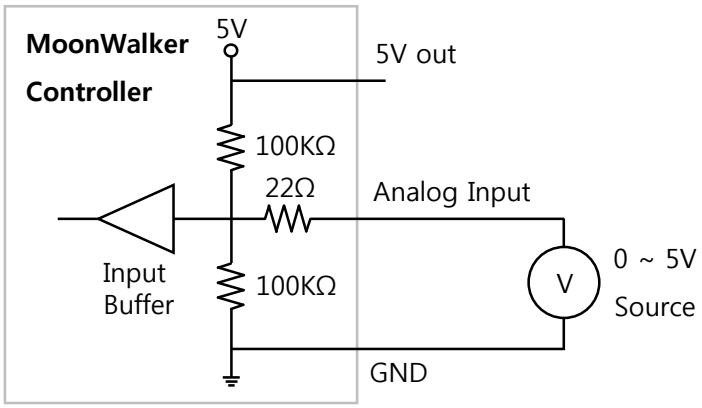

그림 4-5 전압을 출력하는 아날로그 센서 연결

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

4.4.2 포텐셔미터 연결

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

포텐셔미터는 그림 4-6과 같이 제어기의 아날로그 입력 포트에 연결되어, 일반적으로 속도제어나 위치제어 모드에서 제어기에 속도 명령을 내리거나 위치 피드백 정보를 제공하는데 사용합니다.

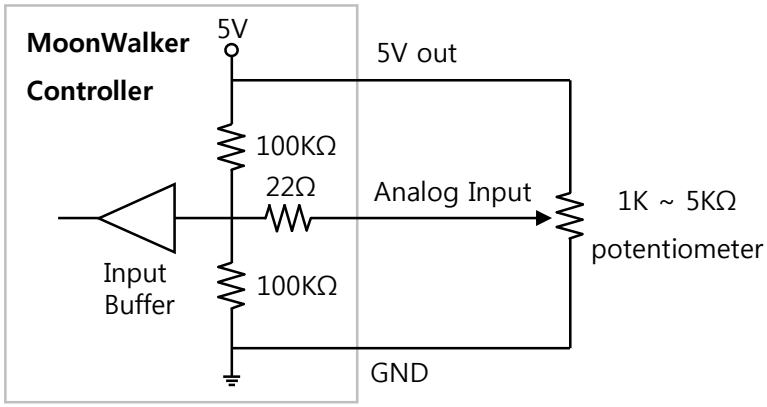

그림 4-6 포텐셔미터 연결

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 있기 때문에 주의해야 합니다.

4.4.3 안전 구간 사용

포텐셔미터로 모터의 속도를 제어할 때, 연결선이 단절되거나 하여 사용자가 모터에 대한 제어권을 상실할 수가 있습니다. 그림 4-7과 같이 포텐셔미터 양단에 저항을 연결함으로 이러한 상황을 감지할 수 있습니다. 양단의 저항은 포텐셔미터가 정상적인 동작 상황에서 전압의 최소 최대 출력이 0V나 5V가 될 수 없도록 합니다.

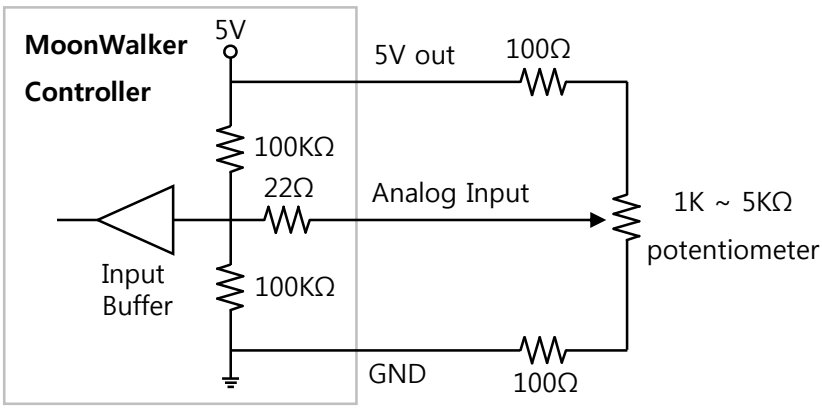

그림 4-7 포텐셔미터 양단에 저항 연결

상기 그림과 같이 설계했을 경우, 만일 5V 출력 선이 끊어진다면 아날로그 입력은 0V가 될 것입니다. 그리고 GND 선이 끊어진다면 5V가 될 것입니다. 또한 아날로그 입력 선이 끊어지면 2.5V가 될 것입니다. 정상 범위 내에서 작동한다면, 아래 그림의 저항 값으로 계산해 보면, 아날로그

입력 전압은 0.5V에서 4.5V까지 변하게 됩니다.

제어기는 아날로그 입력 값이 사용자가 설정한 최대 최소 범위를 벗어나는 경우 이상 상황으로 감지하고 모터를 안전하게 정지합니다. 이 기능을 사용하기 위해서는, Motor Control UI 유틸리티에서 캘리브레이션 과정을 통해 아날로그 입력의 최대, 최소 범위를 설정해야 합니다. 그리고 **'Min/Max safety'**를 Enable로 설정해야 합니다. "10.8.3 min_max_safety - Min/Max Safety"을 참조하기 바랍니다.

4.4.4 타코미터 연결

타코미터(Tachometer)는 모터의 회전 속도를 측정합니다. 일반 DC모터의 축을 강제로 회전하면 발전기가 되는데, 이러한 특성으로 인해 다음 그림 4-8와 같이 DC모터를 타코미터로 사용할 수 있습니다. 타코미터의 축은 감속기를 거치지 않고 모터에 바로 연결하는 것이 좋습니다.

타코미터 출력 전압을 역방향 최고속도인 -2.5V 그리고 정방향 최고 속도인 +2.5V로 조정하기 위해서 10K 포텐셔미터를 사용하고, 전압 분배기 역할을 하는 두 개의 1K 저항을 연결해 줍니다.

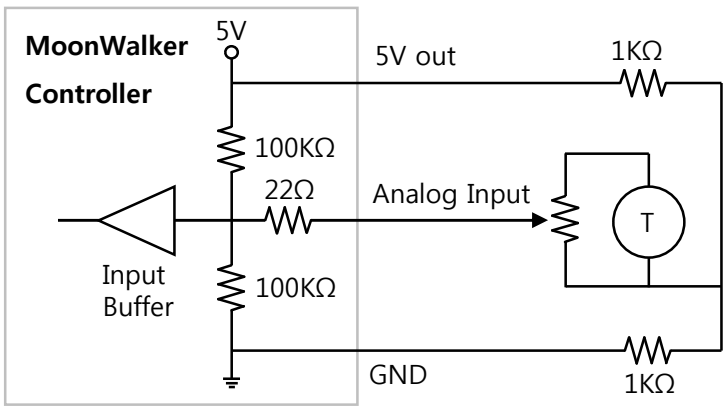

그림 4-8 타코미터 연결

상기 회로에서 타코미터가 멈추면 제어기의 아날로그 입력 포트에는 2.5V가 걸립니다. 그리고 역방향 최고속도로 회전할 때는 0V, 정방향 최고속도로 회전할 때는 +5V가 걸리게 됩니다.

타코미터가 최고 속도로 회전하면 2.5V를 초과하는 전압을 생성 할 수도 있습니다. 따라서 타코미터를 처음 설치할 때는 포텐셔미터가 최소 값을 가지도록 설정해야 합니다.

4.4.5 써미스터 연결

써미스터(Thermistor)는 온도에 따라 저항 수치가 변하는 소자입니다. 외부 써미스터를 사용할 경우 아래 그림 4-9과 같이 연결합니다.

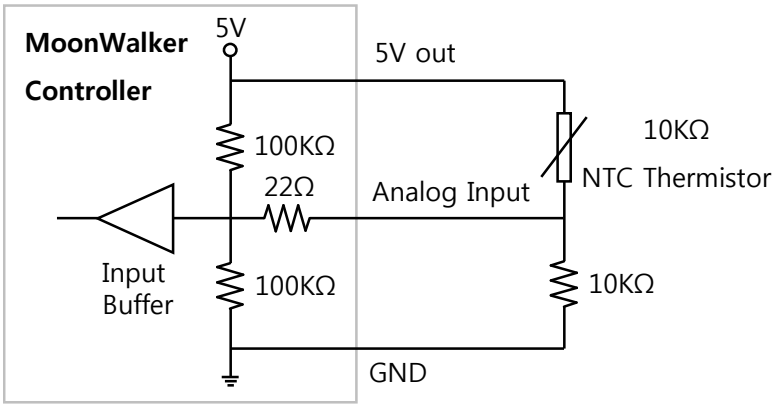

그림 4-9 써미스터 연결

외부 써미스터를 통해서 모터 온도를 감시하고 과열 시에 모터 전력 출력을 차단할 수 있습니다. 이 작업은 약간의 스크립트 작성을 필요로 합니다. 스크립트에 대한 자세한 내용은 "14 Mini-C 스크립트 언어"을 참고하기 바랍니다.

써미스터는 아래와 같은 특성을 가진 10KΩ NTC를 사용하길 권장합니다(NTC-103F397). 이때 외부에 10KΩ 풀다운(Pull-Down) 저항을 추가하여야 합니다. 풀다운 저항은 선정한 써미스터의 저항 수치에 따라 적절하게 선정하여야 합니다.

PART No.	저항(25℃)*1	B값(25℃/85℃)*2	열방산정수	열시정수	사용전력	사용온도
502F332F	5kΩ±1%	3324±1%				
502F347F	5kΩ±1%	3470±1%(25℃/50℃)				
502F397F	5kΩ±1%	3970±1%				
103F343F	10kΩ±1%	3435±1%				
103F345F	10kΩ±1%	3450±1%(25℃/50℃)				
103F397F	10kΩ±1%	3970±1%	3.5 mW/℃	15 sec max.	45 mW	-50~120℃
303F410F	30kΩ±1%	4100±1%				
403F400F	40kΩ±1%	4000±1%				
503F400F	50kΩ±1%	4000±1%				
503F408F	50kΩ±1%	4080±1%				
104F400F	100kΩ±1%	4000±1%				

※ 써미스터 모델, 브랜드, 저항 정밀도 그리고 사용자 환경에 따라서 아날로그 전압 값이 달라질 수 있습니다.

4.4.6 외부 전압 모니터링을 위한 연결

아날로그 입력은 배터리 잔량 또는 외부 DC 전원을 모니터링 하는데 사용할 수 있습니다. 측정 전압이 최대 5V인 경우 전압 출력을 아날로그 입력 핀에 직접 연결해서 사용하면 됩니다. 하지만 높은 전압을 측정할 경우에는 그림 4-10과 같이 전압을 분배하기 위해 두 개의 저항을 연결해야 합니다.

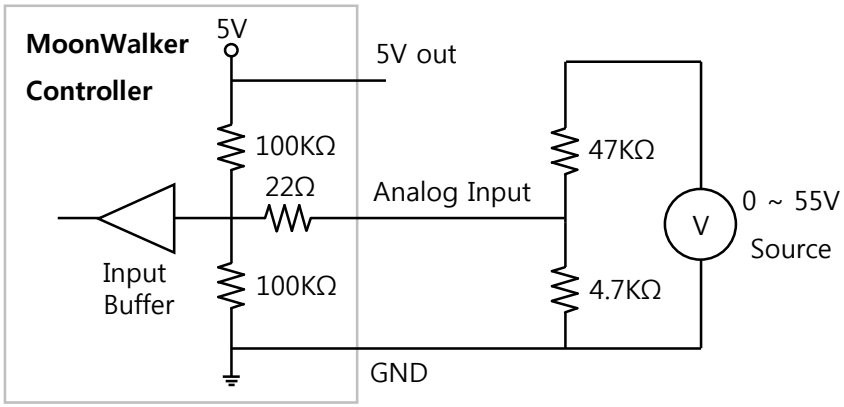

그림 4-10 외부 전압 모니터링을 위한 연결

상기 그림은 47KΩ 저항과 4.7KΩ 저항을 직렬로 연결하여 55V까지 측정할 수 있도록 전압을 1/11로 분배한 예입니다.

4.5 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.

4.5.1 RC 수신기 연결

펄스 입력은 그림 4-11와 같이 추가 구성 요소 없이 RC 수신기를 직접 연결해서 사용할 수 있습니다. 이때 센서는 항상 펄스를 출력해야 하고 일정한 DC 전압을 출력하면 안 됩니다. 펄스 입력 포트로 최소 20Hz에서 최대 20kHz 사이의 펄스가 입력되지 않으면 제어기는 신호가 비정상이라 판단하고 펄스 입력 값을 강제로 0으로 지정합니다.

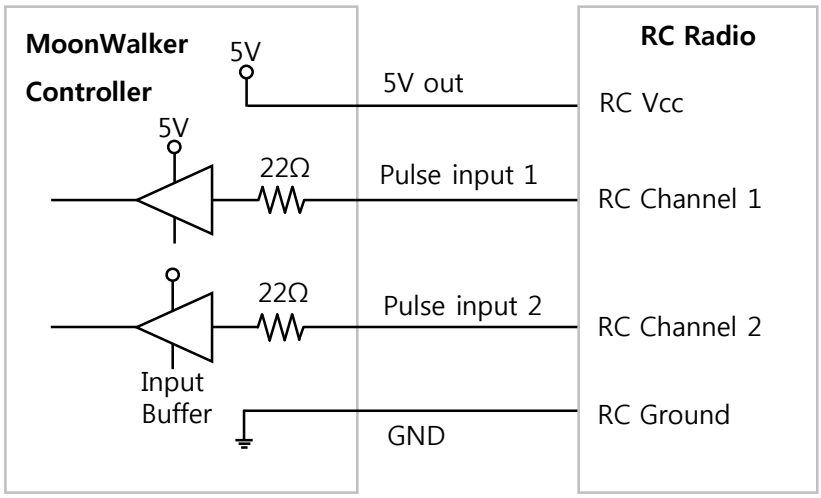

그림 4-11 RC 수신기 연결

4.6 광학식 증분 엔코더

광학식 증분 엔코더(Optical Incremental Encoder)는 모터의 회전각과 회전속도를 측정하는 가장 기본적인 센서입니다. 증분 엔코더는 절대 엔코더(Absolute Encoder)와는 달리 회전할 때 펄스 출력을 내보내며, 펄스를 카운트하면 모터가 얼마나 회전했는지 알 수 있습니다. 회전 속도는 정해진 시간 동안 카운트 되는 펄스 수로 계산되는데, 디지털 펄스를 사용하기 때문에 회전 각과 속도를 정확하게 측정할 수 있습니다.

구형파 엔코더는 전기적으로 90도 위상차를 가지는 두 채널 A와 B를 가지고 있는데, 두 채널 사이의 위상차로부터 모터의 회전 방향을 결정합니다. 또한, 2채널 엔코더는 각 채널의 상승 및 하강 에지를 카운트하여 4체배 해상도를 얻을 수 있습니다. 만약, 한 회전당 500 펄스를 생성하는 엔코더가 있다면 이 엔코더는 4체배 하여 2,000카운터를 만들어 냅니다.

엔코더와 관련 있는 제어기의 구성 파라미터 오브젝트는 다음과 같습니다:

- **encoder_ppr - Encoder PPR**

4.6.1 증분 엔코더 연결

증분 엔코더는 제어기의 엔코더 커넥터에 직접 연결됩니다. 커넥터는 엔코더에 5V 전원을 공급하고 엔코더에서 두 개의 구형파 신호 입력(A와 B)을 받습니다.

그림 4-12와 같이 제어기의 엔코더 포트(5V, GND, A상, B상)와 모터 배선을 잘 구별해서 연결하기 바랍니다. 만약 엔코더의 A와 B상이 반대로 연결되었다면, 모터가 정회전 할 때 엔코더 카운터가 다운 카운트 됩니다. 반대로 모터가 역회전 하면 엔코더 카운터는 업 카운트 됩니다. 이런 상황에서는 엔코더의 A와 B상을 바꿔 연결해야 합니다.

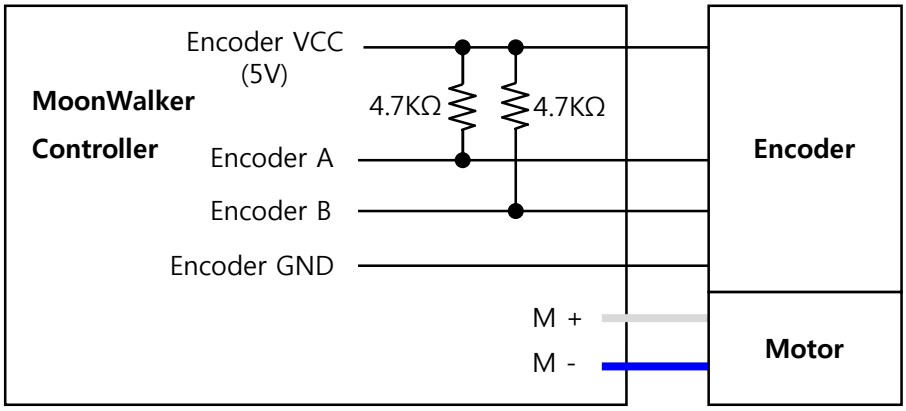

그림 4-12 Incremental형 엔코더 연결

그림 4-12와 같이 제어기 내부의 엔코더 신호선에 풀업(pull-up) 저항이 설계되어 있습니다. 사용자는 내부 풀업 저항을 고려하여 엔코더를 결선하기 바랍니다.

4.6.2 배선에 의한 전기적 노이즈 방지

엔코더 배선에 의한 전기적 노이즈를 방지하기 위해서는 케이블의 길이가 1m를 넘지 않도록 해야 합니다. 배선이 50cm를 넘어갈 때는 제어기 근처에 그림 4-13과 같은 페라이트 코어(Ferrite core)를 엔코더 선에 감아야 합니다. 그리고 전원 공급선과 신호선에서 깨끗한 신호가 전달되는지 오실로스코프를 사용하여 체크합니다.

그림 4-13 페라이트 코어(Ferrite Core)

※주의※ 케이블 길이가 길면 전기적 노이즈가 제어기 안으로 들어와 제어기의 오작동을 일으킬 수 있습니다. 이러한 상황에서는 즉시 제어기 동작 수행을 중지하기 바랍니다.

4.6.3 엔코더 입력 카운터

모터는 정방향 또는 역방향으로 움직일 수 있기 때문에 엔코더에서 입력되는 펄스로부터 위치 카운터를 증가 또는 감소할지를 구분해야 합니다. 구형파 엔코더는 전기적으로 90° 위상차를 가지는 두 채널 A와 B를 가지고 있습니다. 따라서 회전 방향은 두 채널 사이의 위상 관계를 모니터링하여 결정할 수 있습니다. 또한, A와 B의 2채널 엔코더의 4체배 해상도는 각 채널의 상승 및 하강 에지를 카운트하여 얻을 수 있습니다.

엔코더 카운터는 부호 있는 32bit 정수형입니다. 카운트 위치는 -2147483648(0x80000000)에서 2147483647(0x7FFFFFFF)까지입니다. 만일 카운트 위치가 오버플로우 되면 다음과 같은 상황이 됩니다:

- -2147483648(0x80000000)에서 1감소하면 2147483647(0x7FFFFFFF)이 됨
- 2147483647(0x7FFFFFFF)에서 1증가하면 -2147483648(0x80000000)이 됨

엔코더 카운터 값으로 모터의 회전 속도를 계산할 수 있습니다. 속도는 1ms 주기로 카운트 되는 펄스 수를 측정함으로 결정됩니다. 이 측정 방법은 저속에서 카운트 되는 펄스 수가 적어짐으로 정밀도가 떨어집니다. 하지만 제어기에는 이를 보완하는 알고리즘이 포함되어 있습니다.

엔코더의 한 회전당 펄스 수(PPR; Pulse Per Revolution)를 나타내는 **'Encoder PPR'** 파라미터는 정확한 RPM 값을 계산하기 위해 제어기에 올바른 값으로 설정해야 합니다. 속도 정보는 폐루프 속도제어기에서 피드백으로 사용됩니다.

절대 엔코더와 달리 증분 엔코더는 전원이 꺼졌을 때 카운트 위치를 잃어버립니다. 위치제어에 사용될 경우, 제어기는 모터를 홈 위치로 이동하고 카운트 위치를 리셋 해야 합니다. 이 위치를 기준으로 이후 움직이는 위치를 계산합니다.

5 모터의 안전한 사용을 위한 기본설정

이 장에서는 모터의 제어 앞서 모터를 정격 범위 내에서 안전하게 구동하기 위한 구성 파라미터들을 올바르게 설정하는 것에 대해 설명합니다.

5.1 모터 특성

모터의 주요 특성은 다음 그림 5-1에서와 같이 모터의 데이터시트로부터 얻을 수 있습니다.

모델명	전압	무부하		정격부하				기동	
		속도	전류	출력	속도	토크	전류	토크	전류
	(V)	(r/min)	(A)	(W)	(r/min)	(kgf-cm)	(A)	(kgf-cm)	(A)
	24	3300	1.8	300	3000	9.74	15.5	84	135
	90	3300	0.5	300	3000	9.74	4	125	58

그림 5-1 특정 모터의 주요 특성 표시 예

모터의 데이터시트에는 일반적으로 정격 전압, 정격 전류, 정격 토크, 정격 회전수 등이 표시됩니다. 또한, 무부하 전류와 무부하 회전수가 표시되며, 기동 시 전류와 토크도 표시됩니다.

모터를 정격에 따라 안전하게 구동하기 위해서는 모터의 특성들이 모터제어기에 적합한 범위 내에서 설정되어야 합니다. 다음 모터제어기 구성 파라미터들은 모터의 주요 특성을 설정하는데 사용됩니다:

- **max_current - Max Current**
- **max_voltage - Max Voltage**
- **max_velocity - Max Velocity**
- **acceleration - Acceleration**
- **deceleration - Deceleration**

5.1.1 최대 전류와 최대 전압

'Max Current'는 보통 정격 부하가 있을 때 허용되는 전류입니다. 모터제어기의 폐루프 전류제어기는 최대 허용 전류를 넘어가지 않도록 모터에 흐르는 전류를 제어합니다.

DC 모터에 흐르는 전류는 발생하는 토크와 비례 관계에 있습니다. 그리고 모터의 토크는 보통 회전 속도에 반비례합니다. 모터 기동 시 많은 전류가 흐르면서 큰 토크가 가해집니다. 모터가 최고 속도로 회전하면 모터에 가해지는 전압과 역기전력이 비슷해지고 모터에 흐르는 전류는 낮아집니다. 이때 모터가 낼 수 있는 토크는 작아집니다.

'Max Voltage'는 모터에 가장 높게 가해질 수 있는 정격 전압입니다. 부하가 일정할 때, 모터의 회전속도는 모터에 가해지는 전압에 비례합니다. 만일 모터에 정격 전압 이상을 가하게 되면 모터의 속도가 정격 속도 이상으로 증가합니다. 하지만 모터에서 열이 발생하고 수명을 단축하게 됩니다. 극단적으로는 모터가 탈 수도 있습니다.

모터의 최대 허용 전류와 전압은 Motor Control UI 유틸리티를 통해 제어기에 연결된 각각의 모터에 설정할 수 있습니다. 모터의 손상을 방지하기 위해, 모터의 데이터시트를 참고하여 올바른 값을 설정해야 합니다.

5.1.2 가속도와 감속도

모터 제어에 속도 프로파일을 사용하는 것은 모터의 급격한 속도 변화를 방지하고 모터의 서지 전류 및 기계적 피로를 최소화하는데 필요합니다.

그림 5-2와 같이 제어기는 모터를 일정한 토크로 구동하기 위해 사다리꼴 모양의 속도 프로파일을 사용합니다. 여기서 사용되는 **'Max Velocity', 'Acceleration', 'Deceleration'**은 속도 프로파일을 만드는 주요 요소입니다.

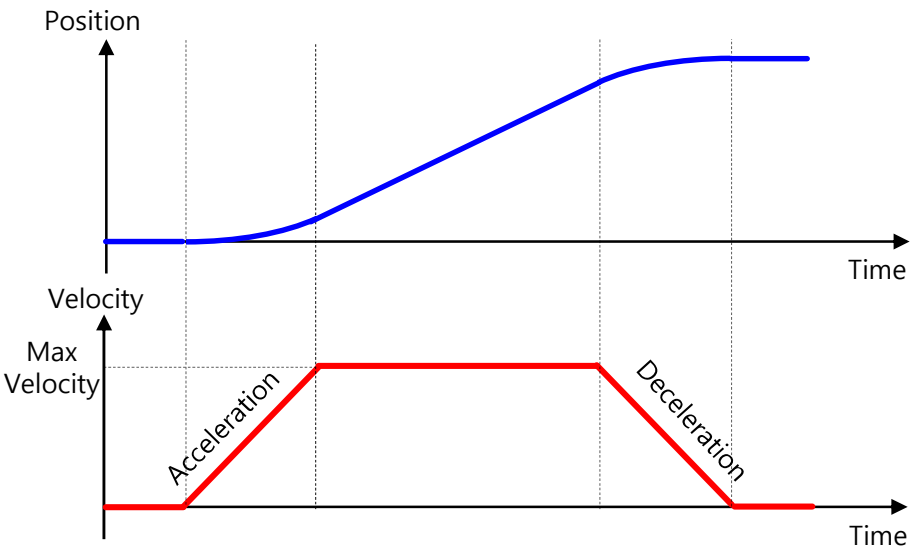

그림 5-2 속도 프로파일과 위치 변화

보통 모터의 최고 속도는 부하나 마찰에 따라 달라집니다. **'Max Velocity'**는 모터의 부하를 고려하여 설정해야 합니다. 만일 폐루프 위치제어나 속도제어에서 모터가 설정된 최고 속도에 도달하지 못하게 되면 PID 제어기의 적분기에 속도오차가 누적됩니다. 이 속도오차에 의해 제어기는 폴트 상황이 되고 모터가 Power OFF 될 수 있습니다. 이러한 모터의 구동 오류를 감지하는 것에 대해서는 "11.7.2 vel_error_detection - Velocity Error Detection"을 참고하기 바랍니다.

속도 명령을 내리면, 컨트롤러는 사용자가 설정한 **'Acceleration'**과 **'Deceleration'**으로 현재 속도에서 원하는 속도로 이동합니다. 위치 명령도 속도 명령과 마찬가지로 속도 프로파일을 생성하여

원하는 위치까지 이동하게 됩니다.

최고 속도, 가속도, 감속도는 Motor Control UI 유틸리티를 사용하여 변경할 수 있습니다. 가속도와 감속도는 각각 다른 값으로 설정할 수 있습니다. 최고 속도는 RPM 단위로, 가속도/감속도는 초당 RPM 단위로 입력합니다. 감속은 음의 값을 의미하지만, 감속도는 양의 값을 사용해야 합니다. 이 구성 파라미터들은 모터가 적용된 시스템과 부하를 고려하여 올바른 값을 설정해야 합니다.

5.2 홈 위치와 이동 범위

제어기에 위치 센서(엔코더, 홀 센서, 포텐셔미터)가 연결되어 있다면, 1ms마다 모터의 위치를 측정하고 속도를 계산합니다. 이 값은 모터의 초기위치를 찾고 구동 범위를 제한하는데 사용될 수 있습니다. 이와 관련되어 다음 구성 파라미터들이 사용됩니다:

- **home_position - Home Position**
- **min_position - Min Position**
- **max_postion - Max Position**
- **use_soft_limit - Use Soft Limit**

5.2.1 이동 범위 설정

제어기의 운용 환경으로 다음 그림 5-3과 같이 긴 레일 위를 바퀴에 연결된 모터가 이동하는 것을 생각해 볼 수 있습니다.

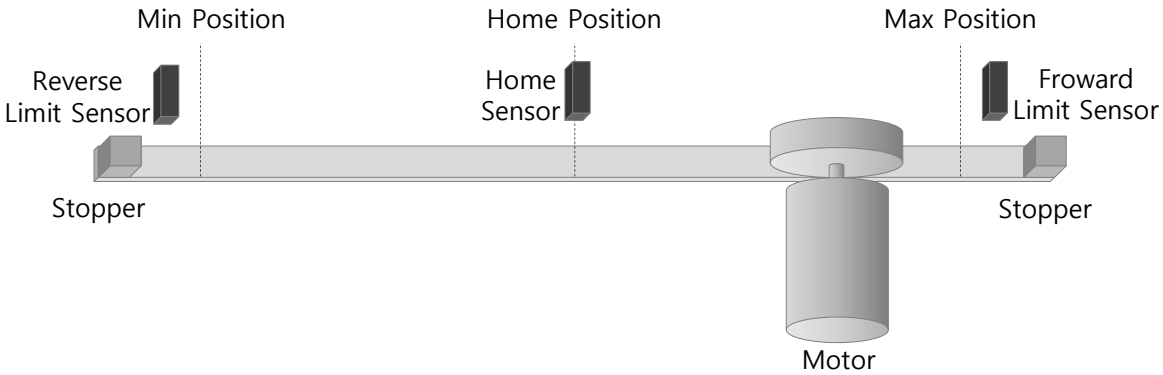

그림 5-3 홈 위치와 모터의 이동 범위 한계

상기 시스템에서 모터의 이동 범위를 제한하는 것은 다음과 같이 물리적, 전기적, 소프트웨어적인 3단계로 구성될 수 있습니다(이 중 하나 이상의 방법을 사용하여 모터의 이동 범위를 구속하게 됩니다):

1. 레일의 양 끝에 스토퍼(Stopper)를 설치하여 모터가 구동 범위를 벗어나지 못하도록 물리적으로 막습니다.
2. 모터가 스토퍼에 다다르기 전에 Forward/Reverse Limit Sensor가 이를 감지하고, 리미트 센서 위치를 벗어나지 못하도록 제어합니다.
3. 모터에 내려지는 위치 명령을 사용자가 설정한 **'Min Position'**과 **'Max Position'** 범위 내로 제한합니다. 그리고 모터의 위치가 **'Min Position'**과 **'Max Position'**을 벗어나면, 이 방향으로 움직이는 명령을 차단합니다.

모터가 어떠한 이유로 스토퍼에 막히게 되면 폐루프 위치/속도 제어기는 원하는 위치 혹은 속도에 도달하기 위해 모터에 공급하는 전력을 점점 증가시키게 됩니다. 하지만 모터의 위치는 변하지 않습니다. 모터제어기는 이러한 상황을 감지하고 모터를 Power OFF 할 수 있습니다. 이 설정을 위해서 "11.7.1 stall_detection - Stall Detection"을 참고합니다.

만일 모터가 Forward Limit Sensor나 Reverse Limit Sensor를 켜면, 모터제어기는 다음 그림 5-4와 같이 모터제어기에 내려지는 명령을 제한합니다.

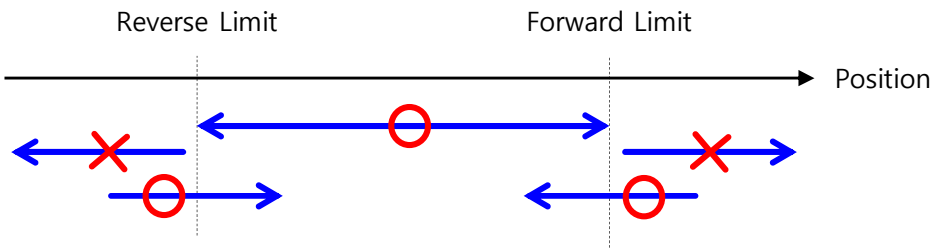

그림 5-4 리미트 범위와 이동 가능한 방향

모터가 Reverse Limit Sensor를 켜면 모터를 음(-)의 방향으로 이동하는 위치, 속도, 전류, 전압 명령은 실행되지 않습니다. 하지만 양(+)의 방향으로 이동하는 명령은 실행됩니다. 만일, Forward Limit Sensor를 켜면 모터를 양의 방향으로 이동하는 명령은 실행되지 않습니다. 하지만 음의 방향으로 이동하는 명령은 실행됩니다. 이러한 동작을 구성하기 위해서는 디지털 입력에 근접 센서나 리미트 스위치가 연결되어야 하고 적절한 I/O 설정이 필요합니다.

모터의 이동 범위를 **'Min Position'**과 **'Max Position'** 값으로 설정하고 소프트웨어적으로 이동 범위를 제한하는 것은 가장 쉽게 사용할 수 있는 이동범위 제한 방법입니다. 이는 Forward/Reverse Limit Switch를 사용하는 것과 같은 방법으로 운용됩니다. 이 방법을 사용하기 위해서는 **'Use Soft Limit'**의 설정이 켜져 있어야 합니다. "11.4.5 use_soft_limit - Use Soft Limit"를 참고하기 바랍니다.

'Min Position'과 **'Max Position'**은 모터가 이동 가능한 최소/최대 위치를 말합니다. 일반적으로 이 위치는 스토퍼와 Reverse/Forward Limit Switch에 도달하기 전에 여유를 두고 설정하는 것이 좋습니다.

5.2.2 홈 위치 설정

일반적으로 모터에는 증분 엔코더가 사용되기 때문에, 제어기 전원이 꺼졌다가 켜지면 카운트 위치를 잃어버립니다.

위치제어에 사용할 경우, 모터를 홈 위치로 이동하고 모터의 위치를 리셋 하는 과정이 우선되어야 합니다. 이 과정은 다음과 같이 구현될 수 있습니다:

1. **'Home Position'**에 모터의 홈 위치를 설정합니다.
2. 전압 명령으로 모터를 천천히 홈 센서가 설치된 방향으로 움직입니다.
3. 홈 센서가 트리거 되면 모터의 위치(**'Position'**)에 **'Home Position'** 값이 적재 됩니다.
4. 마지막으로 0V 전압 명령으로 모터를 정지합니다.

이후, 모터는 홈 위치를 기준으로 자신의 현재 위치를 파악하게 됩니다.

5.3 모터와 제어기 보호

모터제어기는 1ms마다 전원 전압, 방열판 온도를 읽습니다. 그리고 0.1ms마다 모터에 흐르는 전류를 읽습니다. 이 값들은 모터의 피크 전류와 과전류, 과전압, 저전압 등 제어기 보호와 관련된 상태를 체크하는데 사용됩니다.

모터제어기의 안전과 관련된 파라미터들은 다음과 같은 것들이 있습니다:

- **overvoltage_limit - Overvoltage Limit**
- **undervoltage_limit - Undervoltage Limit**
- **high_voltage - High Voltage**
- **overcurrent_limit - Overcurrent Limit**
- **overcurrent_delay - Overcurrent Delay**
- **peak_current_ratio - Peak Current Ratio**
- **overheat_limit - Overheat Limit**
- **high_temperature - High Temperature**

만일 측정된 모터 전류와 전원 전압, 방열판 온도 값들이 상기 리미트를 벗어나게 되면 폴트를 발생하고 모터를 Power OFF 합니다. 이때 측정값이 다시 리미트 범위 내로 들어오더라도 모터는 자동으로 Power ON 되지 않습니다.

5.3.1 과전압 보호

제어기에는 전원의 전압 모니터링 회로가 포함되어 있습니다. 전원의 전압이 설정된 **'Overvoltage Limit'** 값을 초과하면 과전압 폴트를 발생하고 모터를 Power OFF 합니다. 이 값은

제어기 모델에 따라 기본 값이 설정되어 있으며 사용자가 Motor Control UI 유틸리티를 사용하여 변경할 수 있습니다. 기본 값과 설정 가능한 범위는 제어기 모델의 데이터시트를 참조하기 바랍니다.

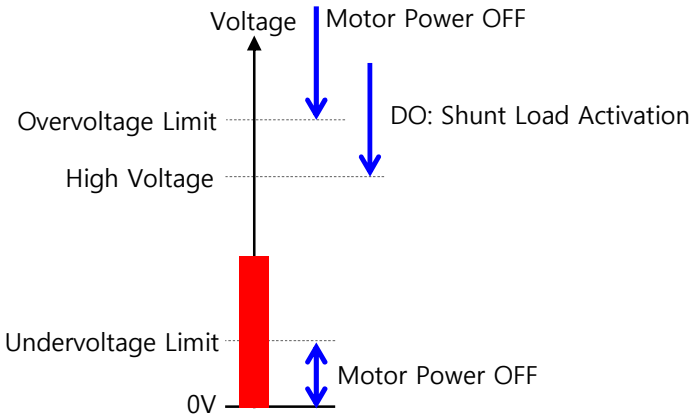

그림 5-5 전원의 전압에 따른 제어기 동작 트리거

모터에 공급되는 전원보다 전압이 높아지는 현상은 모터가 생성한 전압을 스위칭 회로가 증폭하여 발생합니다. 그리고 높아진 전압을 전원이 흡수하지 못하면 과전압 보호가 작동합니다.

과전압 보호로 모터가 Power OFF 되는 상황을 방지하기 위해, 전압이 사용자가 설정한 **'High Voltage'**에 다다르면 디지털 출력을 트리거 하도록 설정 할 수 있습니다. 이 디지털 출력은 회생 제동에 의해 생성된 에너지를 흡수하기 위해 전원의 +와 -간에 설치된 션트 저항을 켜게 됩니다. 과전압 보호는 전원으로 배터리 대신 전원공급장치를 사용할 때는 필수적으로 사용되어야 합니다.

5.3.2 저전압 보호

배터리 전압이 설정된 **'Undervoltage Limit'** 값 이하로 떨어질 경우 저전압 폴트를 발생하고 모터를 Power OFF 합니다. 이 값은 제어기 모델에 따라 기본 값이 설정되어 있으며, 제어기 하드웨어 회로의 특성을 반영하고 있기 때문에 변경하지 안는 것이 좋습니다.

모터에 공급되는 전원의 전압이 갑자기 떨어지는 현상은 보통 모터의 가감속 구간에서 전원의 정격 전류보다 높은 전류를 끌어다 씀으로 발생합니다. 측정 된 전압이 다시 **'Undervoltage Limit'** 위로 올라가더라도 모터는 Power ON 되지 않습니다.

저전압 보호로 모터가 Power OFF 되는 상황을 방지하기 위해, 제어기에 연결된 모터의 정격 파워보다 높은 전원의 배터리나 파워서플라이를 사용해야 합니다. 이것이 불가능할 때는, 모터의 **'Max Current'** 설정 값을 낮추거나 **'Acceleration', 'Deceleration'** 설정 값을 낮춰야 합니다.

"11.5.1max_current - Max Current"와 "11.5.4 acceleration, deceleration - Acceleration, Deceleration"을 참고하기 바랍니다.

5.3.3 과전류 보호

제어기는 모터에 흐르는 전류를 측정하기 위해 전력단 내부에 전류 센서를 가지고 있습니다. 전류센서는 매 0.1ms 마다 전류를 측정하고 지정된 시간('**Overcurrent Delay**') 동안 설정된 **'Overcurrent Limit'** 값을 초과하면 과전류 폴트를 발생하고 모터를 Power OFF 합니다. 이 파라미터는 제어기 모델에 따라 기본 값이 설정되어 있으며 사용자가 Motor Control UI 유틸리티를 사용하여 변경할 수 있습니다. 기본 값과 설정 가능한 범위는 제어기 모델의 데이터시트를 참조하기 바랍니다.

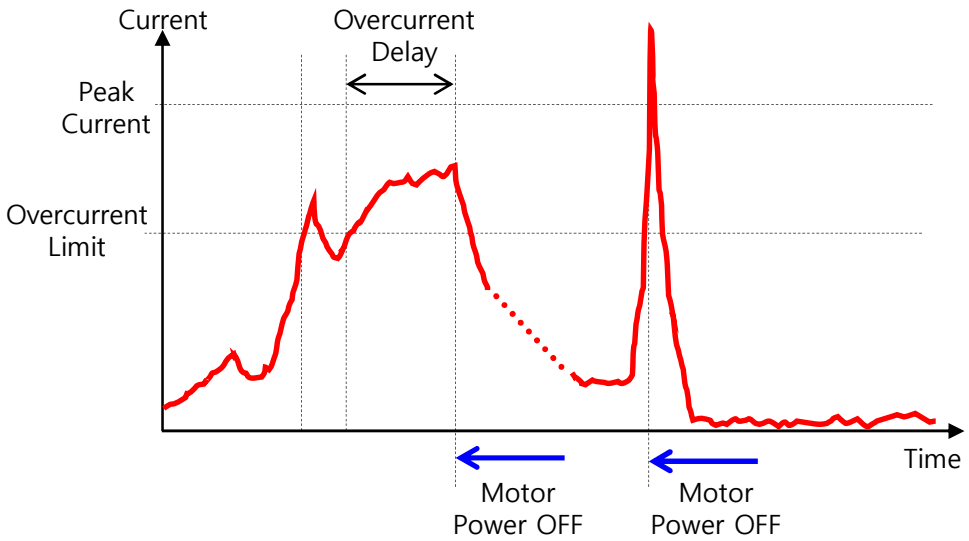

그림 5-6 모터의 전류에 따른 제어기 동작 트리거

'Overcurrent Limit' 값은 모터의 **'Max Current'**보다 120% 이상 높게 설정하는 것이 좋습니다. 모터의 전류 측정 값에는 정류자에서 발생하는 아크 및 MOSFET 스위칭시 발생하는 전압 스파이크에 의해 노이즈가 포함됩니다. 만일 이 두 값의 차이가 미소하다면 전류 노이즈로 인해 과전류 보호가 작동할 가능성이 높습니다.

또한, 제어기는 전력단 및 모터 결선의 단락 상황을 판단하는 매우 높은 서지 전류를 감지합니다. 모터에 흐르는 전류가 한 순간이라도 Peak Current 값을 넘어가면, 과전류 폴트를 발생하고 모터는 Power OFF 상태가 됩니다. Peak Current는 직접 설정되지 않고 **'Peak Current Ratio'** 설정에 의해 다음과 같이 계산됩니다:

Peak Current = **Peak Current Ratio** x **Overcurrent Limit**

'Peak Current Ratio'는 모든 제어기에 공통적으로 200%의 값이 설정되어 있으며 사용자가 Motor Control UI 유틸리티를 사용하여 변경할 수 있습니다.

5.3.4 과열 보호

제어기에는 방열판의 온도 모니터링 회로가 포함되어 있습니다. 매 1ms마다 온도를 측정하고 방열판의 온도가 설정된 **'Overheat Limit'** 값을 초과하면 과열 폴트를 발생하고 모터를 Power OFF 합니다. 온도가 다시 설정 값 아래로 떨어지더라도 모터는 자동으로 Power ON 되지 않습니다. 이 값은 제어기 모델에 따라 기본 값이 설정되어 있으며 사용자가 Motor Control UI 유틸리티를 사용하여 변경할 수 있습니다. 기본 값과 설정 가능한 범위는 제어기 모델의 데이터시트에 나와있습니다.

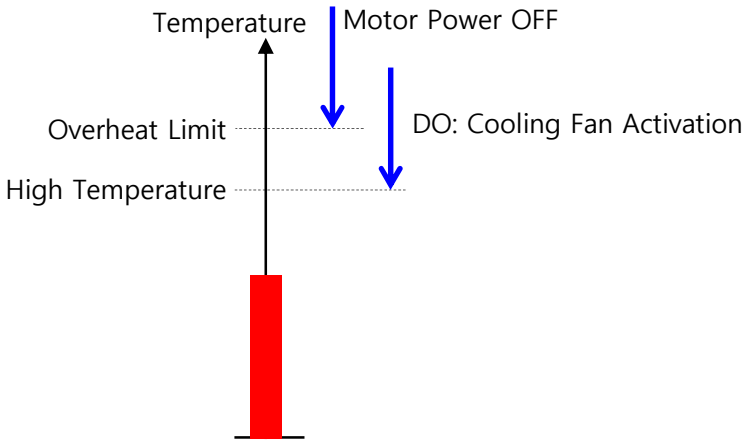

그림 5-7 방열판의 온도에 따른 제어기 동작 트리거

온도는 MOSFET 근처 방열판에서 측정되고 제어기 외부 표면보다 온도가 더 빨리 오르거나 떨어집니다. 방열판의 온도가 상승하는 데 걸리는 시간은 출력 전류와 주위 온도 및 냉각 팬의 사용 여부에 따라 달라집니다.

과열 보호로 모터가 Power OFF 되는 상황을 방지하기 위해, 온도가 사용자가 설정한 **'High Temperature'**에 다다르면 디지털 출력을 트리거 하도록 설정 할 수 있습니다. 이 디지털 출력은 제어기의 방열판을 냉각하도록 설치된 냉각 팬을 켜게 됩니다.

6 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 다음 그림 6-1과 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.

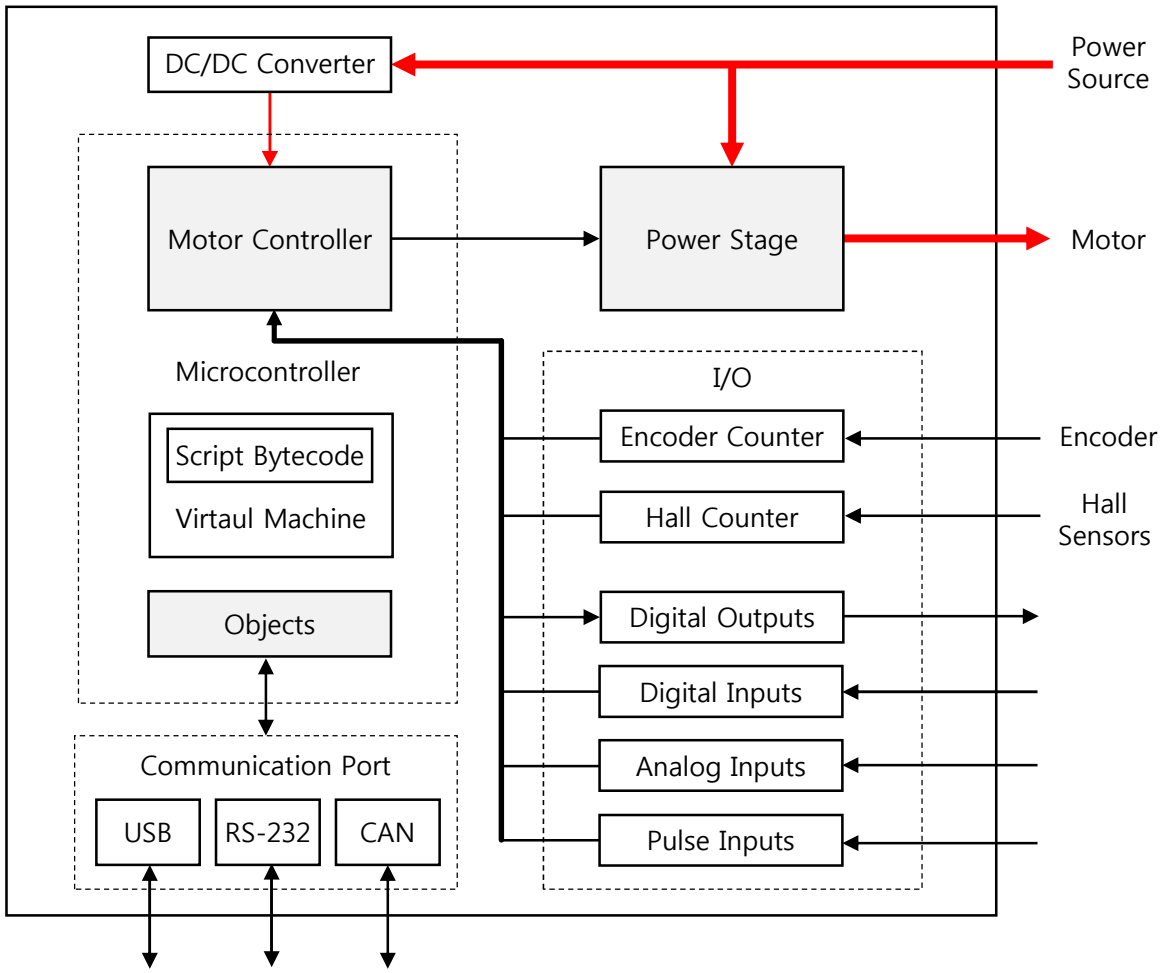

그림 6-1 제어기의 내부 구조

제어기의 마이크로컨트롤러에서 실행되는 프로그램은 모터제어기(Motor Controller)와 가상머신(Virtual Machine)으로 구성됩니다. 듀얼 채널 제어기는 모터제어기와 전력단이 각각 2개씩 존재합니다. 그리고 오브젝트들(Objects)을 저장하는 메모리 공간을 가집니다.

제어기에는 PC와 연결하기 위한 통신 포트가 있습니다. 그리고 각종 센서와 액츄에이터를 연결하는 I/O(Digital Outputs, Digital Inputs, Analog Inputs, Pulse Inputs, Encoder Counter)를 가집니다. I/O 포트는 제어기 모델에 따라 지원되거나 지원되지 않기도 합니다. 따라서 제어기 모델의 데이터시트를 참조하여 지원하는 I/O 포트를 확인해야 합니다.

6.1 모터제어기

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, 전류 명령을 내리고 모터제어기는 모터가 명령에 따르도록 속도나 전류, 전압을 출력합니다.

모터제어기에 대한 자세한 사항은 "7 모터제어기"를 참고하기 바랍니다.

모터제어기는 모터가 Power ON 상태에서만 동작합니다. 모터가 Power OFF 상태일 때는 먼저 모터를 Power ON 상태로 변경해야 합니다. "11.2.1 command – Command"를 참조하기 바랍니다.

6.2 전력단

전력단(Power Stage)은 DC모터의 구동을 위해 직류 전력을 가변할 수 있는 전력 변환 장치로서, 다음 그림 6-2과 같이 4개의 고전류 MOSFET로 구성되는 H-bridge 회로입니다. 이 회로는 높은 주파수의 펄스폭 변조(PWM; Pulse Width Modulation) 기술을 사용하여 MOSFET를 빠르게 ON/OFF 스위칭하여 모터에 공급되는 전력 출력을 조절합니다.

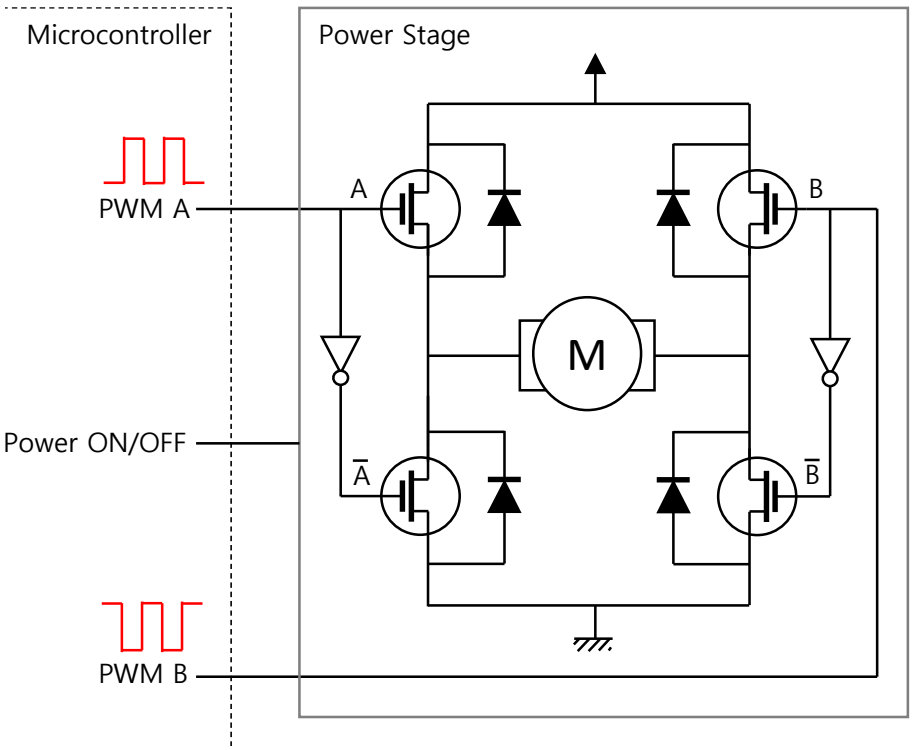

그림 6-2 DC모터 구동을 위한 전력단 구조

전력단은 Power ON/OFF 신호로 켜고 끌 수 있습니다. Power ON 상태에서는 PWM A와 B 채널로 입력되는 신호로 MOSFET를 스위칭합니다. 이때 모터에 전력이 공급되고 모터는 스위칭 신호에 따라 구동합니다. Power OFF 상태에서는 PWM A와 B 채널로 입력되는 신호는 무시되고 전력단의 모든 MOSFET는 꺼지게 됩니다.

전력단과 관계있는 구성 파라미터는 다음과 같습니다:

- **pwm_switching - PWM Switching**
- **pwm_frequency - PWM Frequency**

6.2.1 PWM 주파수

전력 MOSFET는 기본적으로 18kHz에서 스위칭합니다. 사용자는 **'PWM Frequency'** 파라미터를 설정하여 18kHz에서 40kHz 사이의 다른 주파수 값으로 변경할 수 있습니다. 주파수를 증가시키면 스위칭 손실로 인한 효율이 감소됩니다. 반대로 주파수를 낮추면 가청 잡음을 발생시키고 낮은 인덕턴스 모터에 비효율적 일 수 있습니다.

제어기는 제품 초기 설정 값(Factory Default Value)으로 PWM 주파수가 18kHz로 설정되어 있습니다. 이 파라미터는 Motor Control UI 유틸리티에 의해 다른 주파수로 변경 가능합니다.

※ PWM 주파수를 변경하는 것은 모터의 동작에 있어 큰 차이를 만들지는 않습니다. 되도록 설정 기본값을 사용하는 것이 좋습니다.

6.2.2 유니폴라와 바이폴라 구동

유니폴라(Unipolar)와 바이폴라(Bipolar) 구동 방식은 H-bridge 형태로 스위칭 소자들을 구성하여 운용하는 대표적인 방법입니다.

제어기는 **'PWM Switching'** 파라미터 설정을 통해 두 구동 방식 중 하나를 선택적으로 사용할 수 있습니다. 제어기는 제품 초기 설정 값으로 유니폴라 구동 방식으로 설정되어 있습니다. 이 파라미터는 Motor Control UI 유틸리티에 의해 바이폴라 구동으로 변경 가능합니다.

유니폴라 구동에서 모터를 정회전 하고자 할 때는 H-Bridge를 다음과 같이 운용합니다: MOSFET A는 고속으로 ON/OFF 스위칭하여 PWM 듀티비에 따라 모터의 속도를 결정합니다. MOSFET B는 항상 OFF 상태입니다. 역회전하고자 할 때는 A와 B의 역할을 바꾸면 됩니다.

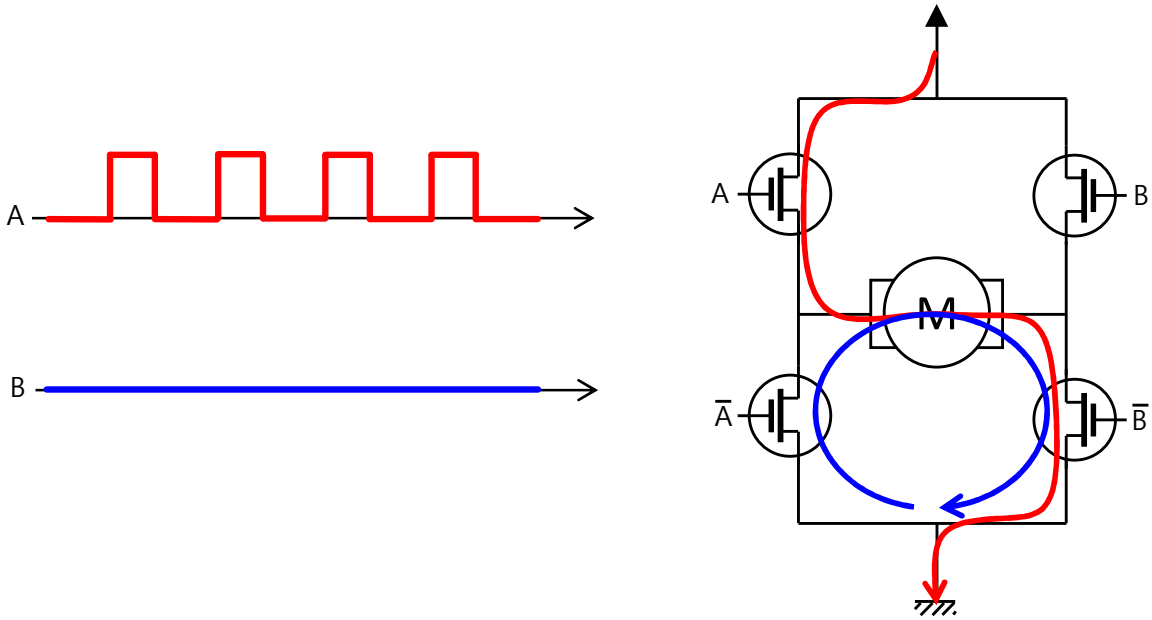

그림 6-3 DC모터의 유니폴라 구동

유니폴라 구동에서는 MOSFET A와 B 둘 중 하나에만 PWM 클럭을 가하게 되며, A와 B 모두 OFF 상태일 때 모터는 정지합니다. 만일 B가 OFF 상태고 A에 공급되는 PWM 듀티비가 50%라면 모터는 최고 속도의 50%로 정회전 합니다. 반대로 A는 OFF 상태고 B에 공급되는 PWM 듀티비가 50%라면 모터는 최고 속도의 50%로 역회전 합니다.

바이폴라 구동에서 모터를 정/역회전 하고자 할 때는 H-Bridge를 다음 그림과 같이 운용합니다: MOSFET A는 ON, B는 OFF 상태를 만듭니다. 다음에는 MOSFET B는 ON, A는 OFF 상태로 만듭니다. 즉, B가 A의 역상으로 동작하도록 합니다. 그리고 이 두 상황을 반복합니다.

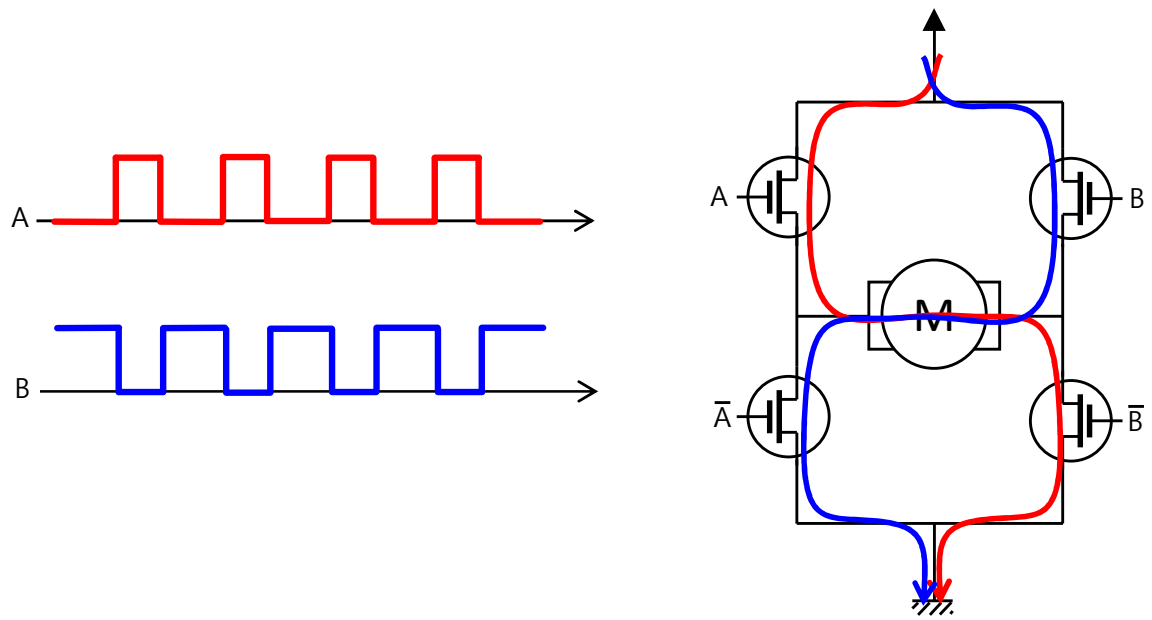

그림 6-4 DC모터의 바이폴라 구동

MOSFET B는 A의 역상으로 동작하며, A에 공급되는 PWM 듀티비가 50%일 때 모터는 정지합니다. 만일 A에 공급되는 PWM 듀티비가 75%가 되면 B에는 25%가 되면 모터는 최고 속도의 50%로 정회전 합니다. 반대로 A에 공급되는 PWM 듀티비가 25%가 되고 B에는 75%가 되면 모터는 최고 속도의 50%로 역회전 합니다.

또한 바이폴라 구동은 저속에서 속도제어 특성을 좋게 하지만 모터의 인덕턴스가 낮은 경우 대기 전류(Idle current)가 높아 전력을 낭비(모터는 돌지 않는데 전류는 계속해서 흐름)하고 모터 발열을 일으킬 수 있습니다. 이런 경우에는 PWM 주파수를 높이거나 PWM 스위칭 방법을 유니폴라로 바꾸거나 모터와 제어기 배선에 직렬로 인덕턴스가 큰 코일을 달아 해결할 수 있습니다.

6.2.3 전류 측정 방식

H-bridge를 통해 모터에 흐르는 전류는 다음과 같이 4가지 경로가 만들어집니다. 빨간색 실선은 스위치 A가 ON, B가 OFF 되었을 때 배터리에서 공급되는 전압에 의해 흐르는 전류의 경로입니다. 주황색 실선은 스위치 A가 OFF, B가 ON 되었을 때 배터리에서 공급되는 전압에 의해 흐르는 전류의 경로입니다. 파란색 실선은 스위치 B가 ON 되었을 때 모터의 역기전력에 의해 흐르는 전류의 경로입니다. 역기전력의 방향이 반대인 경우, A가 ON 되었을 때 흐르게 됩니다. 초록색 실선은 스위치 A가 OFF 되었을 때 모터의 역기전력에 의해 흐르는 전류의 경로입니다. 역기전력의 방향이 반대인 경우, B가 OFF 되었을 때 흐르게 됩니다.

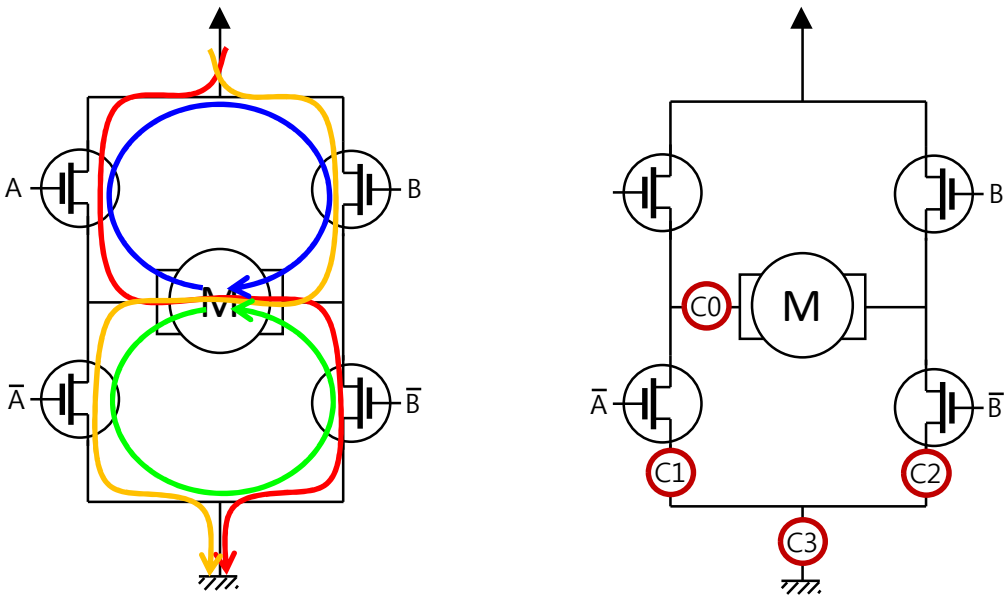

그림 6-5 H-bridge 회로에서 전류의 흐름과 전류 측정 포인트

상기 그림과 같이 모터에는 여러 전류 경로가 존재하기 때문에, 그림과 같이 C0 위치에서 측정이 가장 바람직합니다. 하지만 C0에서는 양방향 전류를 측정해야 하기에 전류 측정 회로가 복잡해집니다.

제어기는 C1과 C2 위치에서 각각 흐르는 단방향 전류를 측정합니다. 그러면 모터에 흐르는 전류

는 C2 – C1로 계산 가능합니다. 이 경우 바이폴라 구동에서 파란색 경로의 전류를 측정할 수 없는 문제점이 있습니다. 파란색 경로의 전류는 바이폴라 구동 방식에서 모터가 회전하다가 멈추고자(또는 역방향으로 회전하고자) 할 경우 흐르게 됩니다.

C3 위치에서 측정하는 전류는 파란색과 초록색 전류를 모두 측정하지 못하는 문제가 있습니다. 단지 배터리에서 공급되어 흐르는 전류만 측정하기 때문에 실제 파워서플라이에서 표시되는 전류와 C3에서 측정된 전류가 일치하게 됩니다. 하지만 모터에 실제 흐르는 전류가 아니므로 일정 범위 내로 전류제어를 하더라도 모터의 역기전력에 의한 전류로 MOSFET가 손상될 수 있습니다.

DC 모터제어기는 C3에서 측정하는 방식은 사용하지 않고 있습니다. 현재는 C1과 C2에서 측정하는 방식을 사용하고 있습니다.

6.2.4 배터리 전류와 모터 전류

제어기는 배터리에서 공급하는 전류가 아닌 모터를 통해 흐르는 전류를 측정하고 제어합니다. 일반적으로 모터를 통해 흐르는 전류는 배터리 전류보다 높습니다. 이러한 현상은 모터의 인덕턴스에 의한 플라이백(Flyback) 전류 때문이며, 배터리 전류가 낮게 흐르더라도 모터 전류는 극단적으로 높게 흘러 제어기의 잠재적인 손상을 가져올 수 있습니다.

그림 6-6과 같이 모터 전류는 플라이백 효과로 인해 PWM 스위칭에 의해 MOSFET가 OFF 되었을 때에도 ON 되었을 때와 같은 수준의 전류 양이 연속으로 흐릅니다.

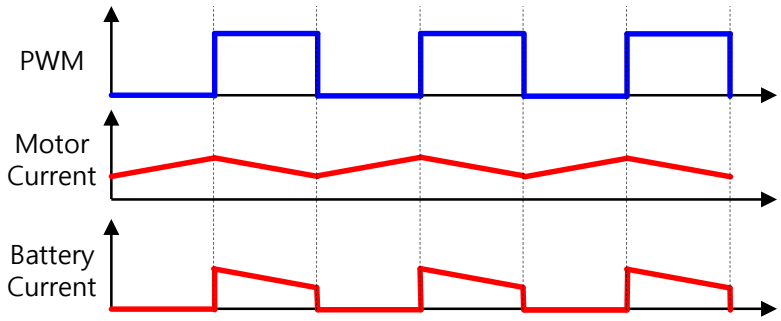

그림 6-6 PWM 스위칭에 따른 모터 전류와 배터리 전류

배터리 전류(Battery Current)와 모터 전류(Motor Current) 사이의 관계식은 다음과 같습니다:

Battery Current = Motor Current x PWM Duty Ratio

만약 제어기가 20% PWM 듀티비(Duty Ratio)에서 배터리 전류가 10A로 측정되면 모터에 흐르는 전류는 50A가 될 것으로 예상할 수 있습니다.

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

6.3.1 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.

6.3.2 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.

6.4 오브젝트

제어기에 명령(Command)을 내리거나 제어기의 구성 파라미터(Configuration Parameter)를 설정하는 것은 제어기 내부의 오브젝트(Object)에 값을 쓰는 것을 말하며, 제어기의 상수나 상태를 모니

터링하는 것은 제어기 내부의 오브젝트에서 값을 읽는 것을 말합니다. 이처럼 제어기가 가진 각종 오브젝트는 제어기와 통신으로 연결된 다른 장치와 데이터를 교환하는 기본 단위가 됩니다.

6.4.1 오브젝트의 종류

제어기 내부의 오브젝트들(Objects)은 다음과 같이 5가지 속성을 가지는 그룹으로 나뉘어 집니다:

- 상수(Constant) 오브젝트: 읽기 전용, 제어기 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 코드 값을 쓰거나 명령 인자(argument)를 쓰는 오브젝트입니다. 명령 코드를 쓰는 것은, 명령 오브젝트가 이미 개별 코드에 대해 '1-모터 정지, 2-모터 전원 차단, 3-모터 전원 공급'과 같이 기능이 부여된 상태에서 한 가지 기능을 선택하는 것과 같습니다. 예를 들면, 모터 정지 명령을 `'command = 1'`과 같이 사용하는 것입니다.

명령 인자를 쓰는 것은, 특정 기능을 수행하도록 구성된 명령 오브젝트에 인자와 함께 실행 명령을 내리는 것과 같습니다. 예를 들면, 모터를 50RPM의 속도로 구동하라는 명령을 `'velocity_command = 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.

6.4.2 오브젝트의 저장과 읽어오기

그림 6-7에서와 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.

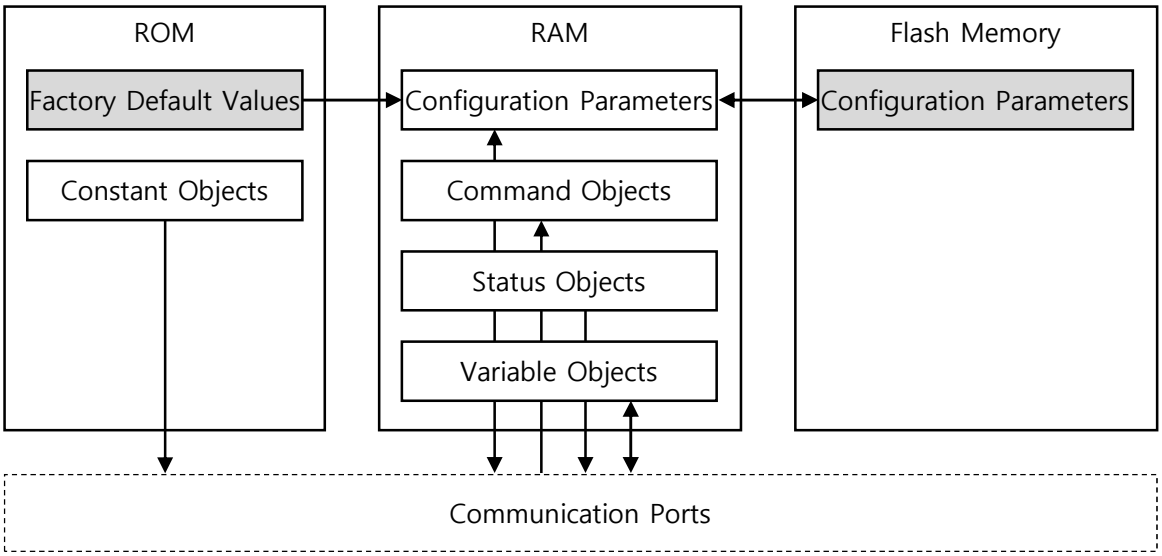

그림 6-7 오브젝트가 저장된 영역과 통신 포트를 통한 오브젝트의 엑세스

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. 자세한 설명은 "10.2.2 system_command - System Command"를 참고하시기 바랍니다.

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.

6.4.3 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 제어기

에 '98 - Reset to Factory Default' 명령을 내림으로 가능합니다.

또 다른 방법으로는 제어기의 전면부나 상판에 있는 리셋(Reset) 버튼을 사용하는 것입니다. 다음은 이를 사용하는 절차입니다:

1. 먼저 제어기의 전원을 끕니다.
2. 리셋 버튼을 누른 상태에서 제어기 전원을 켭니다.
3. 버튼을 5초 이상 누르고 대기합니다. 이때 제어기의 모든 LED는 켜져 있는 상태가 유지됩니다.
4. 5초 후, RAM과 플래시 메모리 영역에 있는 구성 파라미터 오브젝트에 제품 초기 설정값이 적용됩니다. 이때 제어기의 Run LED만 깜박이면서 정상적인 구동을 알리게 됩니다.

6.5 통신 포트

마스터 PC는 제어기의 USB, RS-232, CAN 포트로 연결이 가능합니다. 자세한 사항은 "3.4 통신 포트의 기능"을 참조하기 바랍니다.

6.6 입출력 포트

제어기는 각종 센서와 액츄에이터를 연결하는 입출력(Digital Outputs, Digital Inputs, Analog Inputs, Pulse Inputs) 포트를 가집니다. I/O에 대한 자세한 사항은 "9 I/O 신호 처리"를 참조하기 바랍니다.

그리고 엔코더와 홀센서 연결 포트를 가집니다. 이에 대한 세부 사항은 "4.6 광학식 증분 엔코더"를 참조하기 바랍니다.

7 모터제어기

이번 장에서는 설명하는 모터제어기는 제어기의 마이크로컨트롤러 상에서 실행되는 소프트웨어 알고리즘으로 하드웨어를 배제한 부분입니다. 모터제어기의 구성은 다음 그림 7-1과 같이 폐루프 위치, 속도, 전류 제어기와 프로파일 생성기(Profile Generator), 각종 오브젝트, 외부와 데이터 교환을 위한 입출력 버퍼로 구성됩니다.

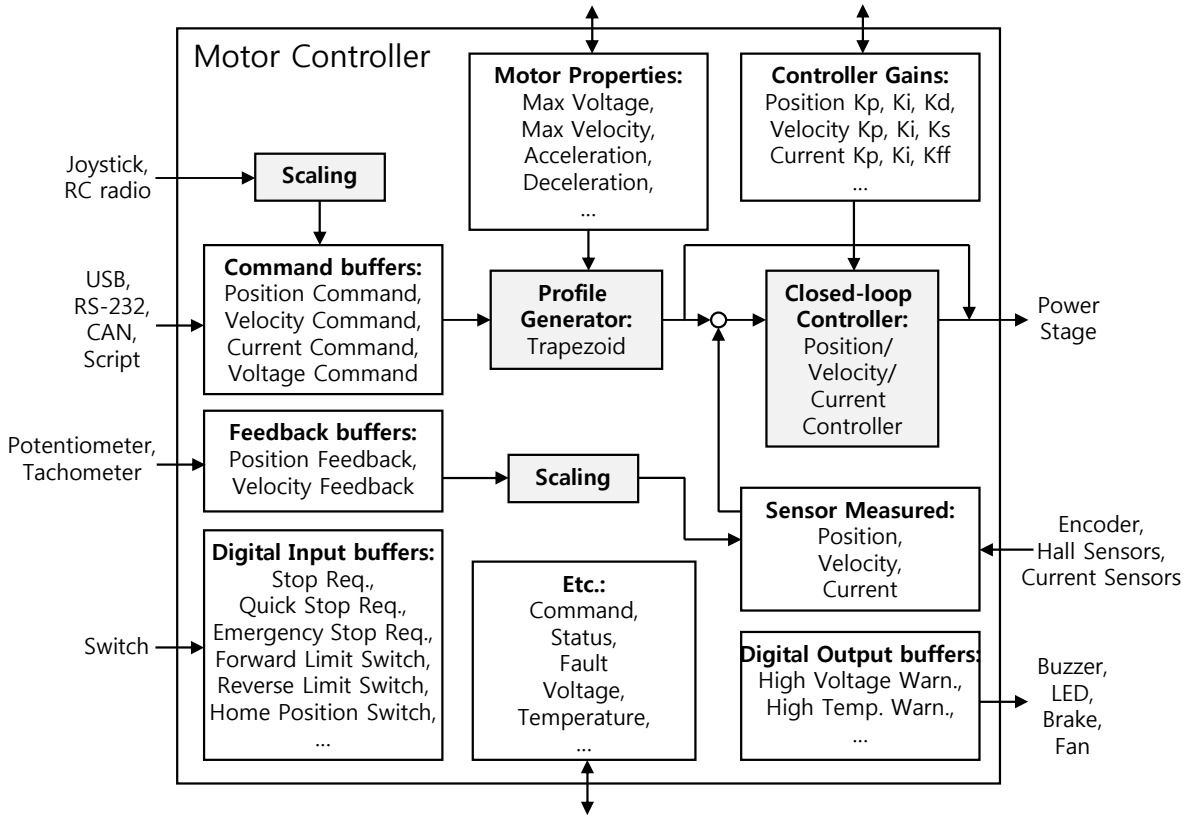

그림 7-1 모터제어기의 구조와 입출력 인터페이스

모터제어기의 오브젝트는 모터의 특성(Motor Properties), 제어기 이득(Controller Gains), 센서 측정(Sensor Measured)값 등 여러 오브젝트로 구성됩니다. 자세한 내용은 "11 모터제어기 오브젝트"을 참조하기 바랍니다.

모터제어기의 명령은 USB, RS-232, CAN, 스크립트를 통해 직접 명령 버퍼(Command buffers)에 들어오기도 하고 Joystick이나 RC radio에서 입력된 신호가 스케일 변환(Scaling) 과정을 거쳐 명령 버퍼에 들어오기도 합니다.

모터제어기의 피드백은 두 가지 경로를 사용하는데, 첫 번째 경로는 엔코더, 홀센서, 전류센서에서 측정한 값을 저장하는 오브젝트(Position, Velocity, Current)를 거치는 경로입니다. 두 번째 경로는 아날로그/펄스 입력 포트에 연결된 센서로부터 측정한 값을 저장하는 피드백 버퍼(Feedback buffers)를 거치는 경로입니다. 모터제어기는 사용자가 선택한 피드백 센서의 종류에 따라 어느 피드백을 사용할 지 결정하게 됩니다.

모터제어기는 모터가 Power ON 상태에서만 동작합니다. 모터가 Power OFF 상태일 때는 먼저 모터를 Power ON 상태로 변경해야 합니다. . "11.2.1 command – Command"를 참조하기 바랍니다.

7.1 모터제어기의 입출력 기능

소프트웨어 알고리즘으로 구현된 모터제어기가 제어기의 통신포트와 I/O 포트를 통해 센서, 액츄에이터, 스위치, 부저, LED 등과 데이터를 주고받기 위해 입출력 버퍼를 거치게 됩니다.

입출력 버퍼는 모터제어기가 가지고 있는 가상의 I/O입니다. 입출력 버퍼는 명령 버퍼(Command buffers)와 피드백 버퍼(Feedback buffers), 디지털 입력 버퍼(Digital Input buffers), 디지털 출력 버퍼(Digital Output buffers)로 구성됩니다.

7.1.1 명령 입력 기능

구동 명령은 명령 버퍼(Command buffers)를 통해 모터제어기에 전달됩니다. 명령 버퍼에는 시리얼 통신(USB, RS-232, CAN)이나 외부 아날로그/펄스 입력 포트, 스크립트에 의해 내려지는 위치, 속도, 전류, 전압 명령을 저장합니다. 명령의 종류는 다음 표를 참고하십시오.

표 7-1 모터제어기의 명령 버퍼의 기능

Buf. #	Function(Command)	Description
1	Voltage Command	모터제어기에 전압 명령을 내림, 모터제어기는 전압출력 모드가 됨
2	Current Command	모터제어기에 전류 명령을 내림, 모터제어기는 전류제어 모드가 됨
3	Velocity Command	모터제어기에 속도 명령을 내림, 모터제어기는 속도제어 모드가 됨
4	Position Command	모터제어기에 위치 명령을 내림, 모터제어기는 위치제어 모드가 됨

상기 표에서와 같이, 4종류의 명령이 있지만 둘 이상의 명령이 동시에 수행될 수는 없습니다. 모터제어기는 한 순간에 오직 한 가지 명령만 수행 가능합니다. 명령의 수행 도중 다른 종류의 명령이 들어오면, 모터제어기는 마지막으로 내려진 명령에 따라 동작모드를 결정합니다.

만일 시리얼 포트로 "`voltage_command = 12`"와 같은 명령이 내려졌다면, 명령 버퍼 1번에 12가 쓰여지고 제어기의 동작모드는 전압출력 모드가 됩니다. 그리고 다른 명령이 내려지기 전까지 모터에 12V 전압을 지속적으로 출력합니다.

다른 예로, "`velocity_command = 500`"과 같은 명령이 내려졌다면, 명령 버퍼 3번에 500이 쓰여지고 제어기의 동작모드는 속도제어 모드가 됩니다. 그리고 속도제어기와 전류제어기가 동작하

여 모터의 속도가 500RPM이 되도록 제어합니다.

7.1.2 피드백 입력 기능

센서 측정 값은 피드백 버퍼(Feedback buffers)를 통해 폐루프 위치 혹은 속도 제어기에 전달됩니다. 피드백 버퍼는 외부 아날로그 입력이나 펄스 입력 포트에 연결된 포텐셔미터/타코미터로부터 측정한 위치/속도 값을 저장합니다. 피드백의 종류와 기능은 다음 표를 참고하십시오.

표 7-2 모터제어기의 피드백 버퍼의 기능

Buf. #	Function(Feedback)	Description
1	Position Feedback	위치 센서의 피드백 값
2	Velocity Feedback	속도 센서의 피드백 값

'Buf. 1 - Position Feedback'은 포텐셔미터와 같은 모터의 회전각을 측정하는 센서로부터의 정규화된 위치 입력 값을 저장합니다.

'Buf. 2 - Velocity Feedback'은 타코미터와 같은 모터의 회전속도를 측정하는 센서로부터의 정규화된 속도 입력 값을 저장합니다.

피드백 기능을 사용하기 위해서는 **'Feedback Sensor'**로 해당 피드백 센서가 선택되어 있어야 합니다. "11.9.1 feedback_sensor - Feedback Sensor"를 참조하기 바랍니다.

7.1.3 디지털 입력 기능

외부 디지털 입력이나 아날로그 입력, 펄스 입력 포트에 연결된 스위치나 센서는 디지털 입력 버퍼(Digital Input buffers)를 통해 모터제어기의 기능을 실행합니다. 디지털 입력 버퍼는 모터제어기의 특정 동작(Action)과 연결되어, 입력 버퍼에 특정 값을 씀으로 연결된 동작을 트리거 할 수 있습니다. 동작의 종류와 기능은 다음 표를 참고하십시오.

표 7-3 모터제어기의 디지털 입력 버퍼의 동작

Buf. #	Function(Action)	Description
1	Emergency Stop	모터에 공급되는 전원을 차단함, 입력이 1인 동안 모터 Power ON 불가능
2	Quick Stop	모터를 빠르게 정지함, 입력이 1인 동안 모터의 구동은 불가능
3	Slowdown Stop	모터를 감속하여 정지함, 입력이 1인 동안 모터의 구동은 불가능
4	Run Scrip	스크립트의 실행과 중단

5	Forward Limit	정방향 리미트 센서가 감지됨, 입력이 1인 동안 정방향 구동 명령은 수행 불가능
6	Reverse Limit	역방향 리미트 센서가 감지됨, 입력이 1인 동안 역방향 구동 명령은 수행 불가능
7	Invert Direction	입력이 1인 동안 모터의 회전 방향을 반대로 바꿈
8	Load Home Counter	모터의 홈 위치 센서가 감지됨, `home_position`의 값을 현재 위치(`position`)로 로딩

액션들은 보통 입력이 0에서 1로 트리거 될 때 해당 동작을 수행하고 1이 유지되는 동안 활성화 상태를 유지합니다. 입력이 0이 되면 비활성화 됩니다.

'Buf. 1 - Emergency Stop'은 제어기가 가진 다른 동작들에 비해 우선 순위가 가장 높으며("7.1.5 입력 기능의 우선순위" 참조) 어떠한 상황에서도 우선적으로 실행됩니다. 이 동작이 트리거 되면 모터를 Power OFF 한 후, 입력이 1인 동안 모터 구동에 관련된 모든 명령과 동작의 수행을 차단합니다. 0이 될 때 모터는 자동으로 Power ON 되지 않습니다.

'Buf. 2 - Quick Stop'이 트리거 되면 모터를 가능한 빠르게 정지합니다. 그리고 입력이 1인 동안 위치, 속도, 전류, 전압 명령은 차단됩니다.

'Buf. 3 - Slowdown Stop'이 트리거 되면 모터를 감속하여 정지합니다. 그리고 입력이 1인 동안 위치, 속도, 전류, 전압 명령은 차단됩니다.

'Buf. 4 - Run Script'는 스크립트의 실행과 중단을 결정합니다. 입력이 0에서 1로 바뀔 때 스크립트의 실행을 시작하고 1인 동안 스크립트는 계속 실행됩니다. 이때 스크립트의 실행이 종료되더라도 스크립트를 다시 시작하지는 않습니다. 입력이 1에서 0으로 바뀔 때 스크립트가 실행 중이라면 실행을 중단합니다.

'Buf. 5/6 - Forward/Reverse Limit'은 모터의 구동 범위 양단에 설치된 리미트 센서의 입력을 모터 제어기에 알려줍니다. 입력이 트리거 되면 'Buf. 2 - Quick Stop'이 실행된 것과 같이 동작합니다. 그리고 입력이 1인 동안 리미트 센서를 벗어나는 방향으로의 위치, 속도, 전류, 전압 명령을 차단합니다.

'Buf. 7 - Invert Direction'은 모터의 회전 방향을 반대로 바꿉니다. 이는 모터에 내려지는 명령의 부호를 바꿈으로 실행됩니다. 즉, 입력이 1인 동안 모터에 내려지는 명령의 부호는 반대가 됩니다.

'Buf. 8 - Load Home Counter'는 홈 센서가 감지되었음을 모터제어기에 알려줍니다. 입력이 트리거 되면 모터의 위치에 홈 위치를 로드 합니다.

정지와 관련된 액션들('Buf. 2 - Quick Stop', 'Buf. 3 - Slowdown Stop', 'Buf. 5 - Forward Limit', 'Buf. 6 - Reverse Limit')은 AND 조건으로 적용됩니다. 예를 들자면, 'Buf. 5 - Forward Limit'과 'Buf. 6 - Reverse Limit' 액션이 동시에 ON 되면 모터는 'Buf. 2 - Quick Stop' 액션과 같이 어떠한 방향으로도 움직일 수 없게 됩니다.

7.1.4 디지털 출력 기능

모터제어기의 상태(Status)는 디지털 출력 버퍼(Digital Output buffers)를 통해 외부 디지털 출력 포트로 전달됩니다. 외부 디지털 출력 포트에는 Buzzer, LED, 냉각 팬 등이 연결되어 모터제어기의 상태를 물리적으로 표현할 수 있습니다. 상태의 종류와 기능은 다음 표를 참고하십시오.

표 7-4 모터제어기의 디지털 출력 버퍼의 상태

Buf. #	Function(Status)	Description
1	Motor Power ON	모터에 전원이 공급되는 상황
2	Motor is Reversed	모터가 역방향으로 회전하는 상황
3	High Voltage	모터제어기에 과전압이 걸린 상황
4	High Temperature	MOSFET와 방열판이 과열된 상황

디지털 출력 버퍼의 값은 0이나 1의 상태를 가집니다. 보통 1이면 제어기 외부에 연결된 장치를 enable 시키고 0이변 disable 시키도록 동작합니다.

'Buf. 1 - Motor Power ON' 기능은 모터에 전원이 공급되는 상황을 표시합니다. 이 기능은 모터의 브레이크를 해제하거나 모터의 Power ON/OFF 상황을 표시하는 램프를 켜고 끄는데 사용될 수 있습니다.

'Buf. 2 - Motor is Reversed' 기능은 모터가 역방향으로 회전하는 상황을 표시합니다. 이 기능은 차량에서 후진 경고 표시등이나 알람을 켜고 끄는데 사용될 수 있습니다.

'Buf. 3 - High Voltage' 기능은 모터의 역기전력으로 인해 전원의 전압이 높아진 상황을 표시합니다. 이 기능은 역기전력을 열로 소모하기 위한 제동저항(Brake Resistor)을 켜고 끄는데 사용될 수 있습니다.

'Buf. 4 - High Temperature' 기능은 제어기 방열판의 온도가 높아진 상황을 표시합니다. 이 기능은 방열판을 냉각하기 위한 냉각 팬을 켜고 끄는데 사용될 수 있습니다.

7.1.5 입력 기능의 우선순위

모터제어기에는 여러 명령이나 액션들이 동시에 내려질 수 있습니다. 이때 명령이나 액션이 적용되는 우선 순위는 표 7-5와 같습니다. 여기서 숫자가 적을수록 우선 순위가 높습니다. X로 표시된 것은 우선순위가 적용되지 않는 입력 입니다.

'Emergency stop' 액션이 우선 순위가 가장 높으며, 두 번째로는 모터의 정지와 관련된 액션들입니다. 위치, 속도, 전류, 전압 명령이 우선 순위가 가장 낮습니다. 즉, 긴급 정지 기능은 어떠한 경우에도 동작하며, 모터 속도 명령이 내려지는 중이라도 정지와 관련된 액션들로 언제든 모터를 정지할 수 있습니다.

표 7-5 명령 및 액션이 적용되는 우선 순위

Priority	Function
1	Emergency stop
2	Quick stop, Slowdown Stop, Forward Limit, Reverse Limit
3	Voltage Command, Current Command, Velocity Command, Position Command
X	Run Scrip, Invert Direction, Load Home Counter, Position Feedback, Velocity Feedback

7.2 명령과 피드백의 스케일 변환

스케일 변환(Scaling)은 조이스틱이나 RC수신기를 아날로그/펄스 입력 포트에 연결하여 모터제어기를 구동하거나 포텐셔미터나 타코미터를 아날로그/펄스 입력 포트에 연결하여 모터의 위치와 속도를 모터제어기에 피드백 할 때 필요합니다.

외부 아날로그 입력이나 펄스 입력 포트를 통해 캡쳐 되는 데이터는 변환 과정을 거쳐 -1과 1사이의 정규화 된 값입니다. 이 값을 모터제어기의 명령이나 피드백으로 사용하기 위해서는 모터제어기 내부에서 사용하는 변수(위치, 속도, 전류, 전압) 단위와 일치시켜야 합니다(표 7-6 참조).

표 7-6 모터제어기의 위치, 속도, 전류, 전압 단위

Object Class	Unit
Position	pulse
Velocity	RPM
Current	A
Voltage	V

스케일 변환에 사용되는 모터 속성과 위치센서 속성은 다음과 같습니다:

- **max_position - Max Position**
- **min_position - Min Position**
- **max_velocity - Max Velocity**

- **max_current - Max Current**
- **max_voltage - Max Voltage**
- **encoder_ppr - Encoder PPR**

7.2.1 위치 명령 스케일 변환

외부 아날로그 입력이나 펄스 입력 포트에 연결된 조이스틱이나 RC수신기가 모터제어기의 위치 명령 버퍼에 연결된 경우, 모터제어기에 전달되는 위치 입력 값은 -1과 1사이의 정규화 된 값입니다. 모터제어기의 위치 명령으로 사용하기 위해 다음과 같이 변환되어 위치 명령 버퍼에 저장됩니다.

Position Command =

0.5 x ((**Max Position** – **Min Position**) x Position Input + **Max Position** + **Min Position**)

여기서 'Position Input'은 위치 입력 값이고 **'Position Command'**는 위치 명령 버퍼에 저장되는 값입니다. 변환된 값의 단위는 [pulse]입니다.

7.2.2 속도 명령 스케일 변환

외부 아날로그 입력이나 펄스 입력 포트에 연결된 조이스틱이나 RC수신기가 모터제어기의 속도 명령 버퍼에 연결된 경우, 모터제어기에 전달되는 속도 입력 값은 -1과 1사이의 정규화 된 값입니다. 모터제어기의 속도 명령으로 사용하기 위해 다음과 같이 변환되어 속도 명령 버퍼에 저장됩니다.

Velocity Command = Max Velocity x Velocity Input

여기서 'Velocity Input'은 속도 입력 값이고 **'Velocity Command'**는 속도 명령 버퍼에 저장되는 값입니다. 변환된 값의 단위는 [RPM]입니다.

7.2.3 전류 명령 스케일 변환

외부 아날로그 입력이나 펄스 입력 포트에 연결된 조이스틱이나 RC수신기가 모터제어기의 전류 명령 버퍼에 연결된 경우, 모터제어기에 전달되는 전류 입력 값은 -1과 1사이의 정규화 된 값입니다. 모터제어기의 전류 명령으로 사용하기 위해 다음과 같이 변환되어 전류 명령 버퍼에 저장됩니다.

Current Command = **Max Current** x Current Input

여기서 'Current Input'은 전류 입력 값이고 **'Current Command'**는 전류 명령 버퍼에 저장되는 값

입니다. 변환된 값의 단위는 [A]입니다.

7.2.4 전압 명령 스케일 변환

외부 아날로그 입력이나 펄스 입력 포트에 연결된 조이스틱이나 RC수신기가 모터제어기의 전압 명령 버퍼에 연결된 경우, 모터제어기에 전달되는 전압 입력 값은 -1과 1사이의 정규화 된 값입니다. 모터제어기의 전압 명령으로 사용하기 위해 다음과 같이 변환되어 전압 명령 버퍼에 저장됩니다.

Voltage Command = **Max Voltage** x Voltage Input

여기서 'Voltage Input'은 전압 입력 값이고 **'Voltage Command'**는 전압 명령 버퍼에 저장되는 값입니다. 변환된 값의 단위는 [V]입니다.

7.2.5 위치 피드백 스케일 변환

외부 아날로그 입력이나 펄스 입력 포트에 연결된 포텐셔미터가 모터제어기의 위치 피드백 버퍼에 연결된 경우, 모터제어기의 위치 피드백 버퍼에 저장되는 값은 -1과 1사이의 정규화 된 값입니다. 모터제어기의 위치 피드백으로 사용하기 위해 다음과 같이 변환됩니다.

Position =

0.5 x ((**Max Position** – **Min Position**) x Position Feedback + **Max Position** + **Min Position**)

여기서 'Position Feedback'는 위치 피드백 버퍼의 값이고 **'Position'**는 변환된 위치 값입니다. 변환된 값의 단위는 [pulse]입니다.

위치 피드백 값으로부터 모터의 회전 속도는 다음과 같이 계산됩니다.

Velocity = 60 x Δ**Position** / (**Encoder PPR** x Δt)

여기서 **'Δ Position'**는 현재 시각에서의 위치 값과 Δt 이전 시각에서의 위치 값의 차입니다. Δt는 위치를 샘플링 하는 주기로 0.001[sec] 입니다. 계산된 회전 속도의 단위는 [RPM] 입니다.

7.2.6 속도 피드백 스케일 변환

외부 아날로그 입력이나 펄스 입력 포트에 연결된 타코미터가 모터제어기의 속도 피드백 버퍼에 연결된 경우, 모터제어기의 속도 피드백 버퍼에 저장되는 값은 -1과 1사이의 정규화 된 값입니다. 모터제어기의 속도 피드백으로 사용하기 위해 다음과 같이 변환됩니다.

Velocity = **Max Velocity** x Velocity Feedback

여기서 'Velocity Feedback'는 현재시각에 위치 피드백 버퍼의 값이고 **'Velocity'**는 변환된 속도 값입니다. 변환된 값의 단위는 [RPM]입니다.

속도 피드백 값으로부터 모터의 회전 위치는 다음과 같이 계산됩니다.

Δ**Position** = (**Velocity** x **Encoder PPR** x Δt) / 60

Position ← **Position** + Δ**Position**

여기서 **'Position'**는 현재 시각에서의 위치 값이고 Δt는 위치를 샘플링 하는 주기로 0.001[sec] 입니다. 계산된 회전 속도의 단위는 [RPM] 입니다.

7.3 모터제어기 구조

만일 모터가 Power ON 상태라면, 제어기는 다음 표와 같이 전압출력, 전류제어, 속도제어, 위치제어 모드 중 하나의 모드로 동작합니다.

표 7-7 제어기의 동작모드

동작모드	명령	제어 방식	전류 제어기	속도 제어기	위치 제어기
전압출력 모드	전압 명령	개루프 제어	X	X	X
전류제어 모드	전류 명령	폐루프 제어	O	X	X
속도제어 모드	속도 명령	폐루프 제어	O	O	X
위치제어 모드	위치 명령	폐루프 제어	O	O	O

* O-사용, X-사용 안함

모터가 Power OFF 상태에서는 전압출력 모드가 되지만, 모터에 공급되는 전력이 차단되기 때문에 아무런 의미가 없습니다. 모터가 Power ON 된 후에는 전압출력 모드가 되고 출력되는 전압은 0V로 설정됩니다.

동작모드는 모터제어기에 마지막으로 내려진 명령으로 결정됩니다. 모터제어기에 다른 명령이 내려지기 전까지 동작모드는 유지됩니다. 모터가 Power ON 상태라면, 전압/전류/속도/위치 명령에 의해 동작모드는 언제나 바뀔 수 있습니다. 하지만 모터가 구동중인 상황에서는 동작모드를 바꾸지 않는 것이 좋습니다. 제어기 구조가 갑자기 바뀜으로 인해 모터와 제어기에 전기적 기계적 충격을 유발할 수 있습니다.

개루프 제어(Open-loop Control)는 전압 명령을 PWM 듀티비로 변환하여 모터에 직접 인가하는 제어 방법입니다. 이러한 방법에서는 오차가 제어 입력 값에 반영되지 않기 때문에 모터의 출력이 원하는 목표와 달라질 수 있습니다.

폐루프 제어(Closed-loop Control)는 원하는 목표(위치, 속도, 전류 명령) 값과 센서의 피드백 값을

비교하여 오차가 작아지도록 모터의 출력을 제어하는 방법입니다. 이를 위해 위치/속도나 전류를 측정하는 센서가 필요한데, 전류 센서로는 제어기 내부의 션트(Shunt) 저항이 사용되고 위치/속도 센서로는 일반적으로 제어기 외부의 엔코더가 사용됩니다.

7.3.1 전체 제어기 구조

모터제어기의 전체 구성은 다음 그림 7-2에서 보이는 바와 같이 위치, 속도, 전류 제어를 위한 여러 제어기들이 직렬(Cascade)로 연결된 구조입니다.

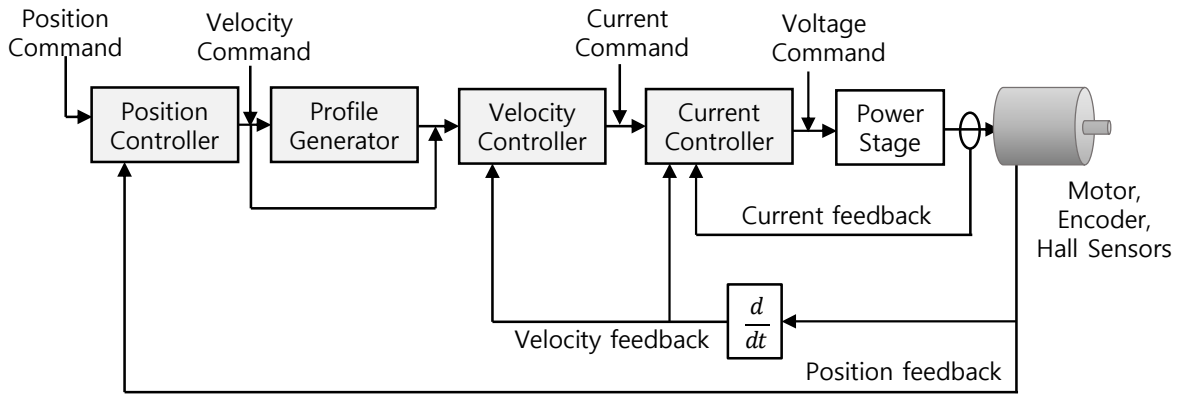

그림 7-2 내부에 설계된 제어기의 전체 구성도

이 제어기들 중 전류 제어기가 가장 내부 루프에 위치하고 있으며, 10KHz의 주파수로 동작합니다. 전류 제어를 통해 토크, 속도, 위치 등이 제어될 수 있기 때문에 전류 제어 응답이 가장 빠릅니다. 전류 제어기의 바깥에는 속도 제어기가 위치하며, 1KHz의 주파수로 동작합니다. 위치제어기는 루프의 가장 바깥에 위치하며, 100Hz의 주파수로 동작합니다. 위치제어기와 속도 제어기 사이에는 속도 프로파일을 생성하는 프로파일 생성기(Profile Generator)가 위치하고 있습니다.

7.3.2 개루프 전압 출력

전압출력 모드에서, 개루프 제어는 전압 명령에 비례하는 전압을 전력단(Power Stage)을 통해 모터에 공급합니다. 모터에 부하가 일정하다면 모터에 가해지는 전압은 모터의 회전속도와 비례합니다. 하지만 모터의 부하가 변할 때는 모터의 속도가 느려지거나 빨라집니다. 이 모드는 운전자가 모터의 구동 상황을 파악하면서 직접 제어하는 응용에 적합합니다.

제어기에 위치 센서(엔코더, 홀 센서, 포텐셔미터)가 연결되어 있다면, 실제 모터의 전류와 속도, 위치는 연결된 센서를 통해 측정됩니다. 하지만 측정된 값이 개루프 제어에 사용되지는 않습니다.

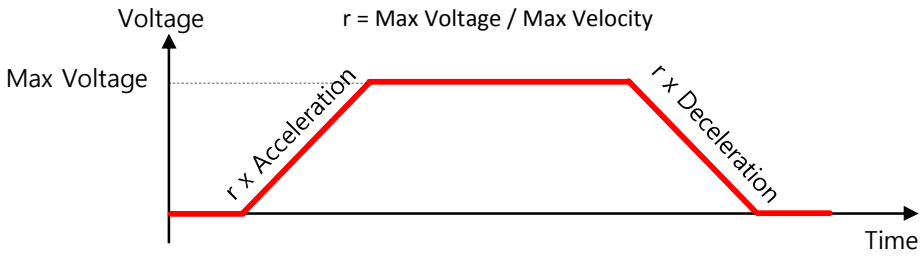

그림 7-3 개루프 전압 프로파일

프로파일 모드는 폐루프 위치제어와 속도제어 모드에서 그리고 개루프 전압출력 모드에서 사용가능 합니다. 만일 개루프 전압출력 모드에서 프로파일 모드가 사용되면, 다음 관계에 의해 사다리꼴 모양의 속도 프로파일이 전압 프로파일로 변환됩니다:

- 속도 프로파일의 **'Max Velocity'**는 전압 프로파일의 **'Max Voltage'**가 됨
- 이때 변환 비율 r은 **Max Voltage/Max Velocity**로 계산
- 전압 증가율은 r*__Acceleration__으로 계산
- 전압 감소율은 r*__Aeceleration__으로 계산

7.3.3 폐루프 전류 제어기

전기 모터에서 토크는 직접적으로 전류에 관련됩니다. 따라서 전류를 제어하는 것은 토크를 제어하는 것과 같습니다.

전류 제어기는 위치제어 모드, 속도제어 모드, 전류제어 모드에서 동작합니다.

그림 7-4에서 전류 제어기의 블록선도를 보여줍니다. 모터의 속도(Velocity Feedback), 모터에 흐르는 전류(Current Feedback), 속도 제어기 또는 사용자가 내린 전류 명령(Current Command)이 입력으로 인가되며, 동작 주파수는 10KHz입니다. 또한, PI 제어기로 구현되어 있으며 제어기의 출력(Voltage output)은 모터에 공급되는 전압이 됩니다.

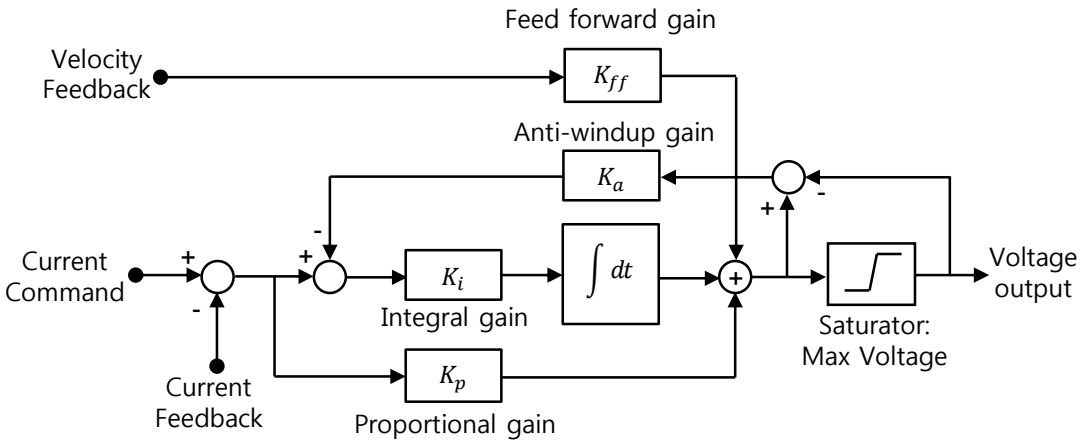

그림 7-4 제어기 내부에 설계된 전류 제어기

PI제어기의 비례 이득(Proportional gain)은 전류 명령과 모터에 흐르는 전류 간의 오차에 비례하여 전압을 조절합니다. 오차가 크다면 높은 전압이 공급되고 모터에 흐르는 전류가 전류 명령에 다가갈수록 전력은 점차 낮아지게 됩니다. 비례 이득을 높이게 되면, 측정 오차로부터 더 높은 전압을 공급하도록 함으로 모터가 명령 및 모터 부하의 변화에 더 신속하게 대응합니다. 하지만 너무 높으면 오버슈트와 진동이 발생합니다.

PI제어기의 적분 이득(Integral gain)은 시간에 따른 오차의 합에 비례하여 전압을 조절합니다. 오차가 0으로 접근할수록 제어기가 원하는 전류에 정확히 도착하고 유지하는데 도움을 줍니다.

전류 제어기에는 안티와인드업(Anti-Windup) 제어기가 설계되어 있습니다. 와인드업(Windup) 현상은 적분 제어기에 누적된 값이 제어기의 제한 폭을 넘어서 제어입력이 쌓이게 되는 경우를 말합니다. 와인드업 현상이 발생하면 제어기의 입력 부호가 바뀌어도 누적된 적분 제어기로 인해 실제 모터의 방향이 바뀌지 않는 현상이 생기게 됩니다. 그래서 적분기에 누적된 수치가 제어기 출력의 제한 값에 따라 포화되지 않도록 제한하는 기능이 필요하며 이것을 안티와인드업 제어기라고 합니다. 제어기의 안티와인드업 이득(Anti-windup gain)은 비례 이득의 역수를 사용하도록 설정되어 있으며 사용자가 설정할 수는 없습니다.

전류 제어기에서는 전향 보상 이득(Feed forward gain)을 이용해서 역기전력에 의한 모터 토크의 감쇠를 보상하고 있습니다. 모터는 회전속도에 비례하는 역기전력(Back EMF)을 생성합니다. 역기전력은 모터의 토크를 감소하게 만듭니다. 그래서 모터가 고속으로 회전하고 있을 때는 역기전력을 보상해 줌으로 모터 토크의 감쇠를 보상할 수 있게 됩니다.

표 7-8에는 전류 제어기의 오브젝트들이 정리되어 있습니다. 여기서 구성 파라미터(Configuration Parameter)는 사용자가 설정할 수 있는 값들입니다.

표 7-8 전류 제어기에서 사용되는 오브젝트들 정리

Name	Unit	Object Type	Object Name
Velocity Feedback	**rad/sec**	Status	**velocity**
Current Feedback	A	Status	**current**
Current Command	A	Command	**current_command**
Voltage output	V	Variable	**voltage**
Proportional gain(K_p)	-	Configuration Parameter	**cc_kp**
Integral gain(K_i)	-	Configuration Parameter	**cc_ki**
Anti-windup gain(K_a)	-	Internal Value, $K_a = \frac{1}{K_p}$	-
Feed forward gain(K_{ff})	V/(rad/sec)	Configuration Parameter	**cc_kff**
Max Velocity	V	Configuration Parameter	**max_velocity**

Motor Control UI 유틸리티에서 제어기의 속도관련 구성 파라미터를 설정할 때는 [RPM] 단위를 사용하지만 모터제어기 내부에는 [rad/sec] 단위로 변환되어 계산에 사용됩니다.

※ 구성 파라미터들 중 속도와 관련된 값들이 모터제어기 내부에서 사용될 때는 속도의 단위가 [RPM]에서 [rad/sec]로 변환되어 계산에 사용됩니다.

※주의※ 전류제어기의 전향보상 이득을 설정하려면 모터의 역기전력 상수를 알아야 합니다. 시중에서 쉽게 구할 수 있는 저렴한 모터들은 역기전력 상수가 표기되지 않은 제품들이 많습니다. 모터의 역기전력 상수를 모를 때는 전향보상 이득을 0으로 설정하기 바랍니다.

또한, 역기전력 상수는 모터마다 다양한 단위로 표기됩니다. 역기전력 상수가 전향보상 이득으로 사용되려면 역기전력 상수의 단위를 [V/(rad/sec)]로 변환해야 합니다. 그리고 제어기 내부에서 사용하는 속도의 단위가 [rad/sec] 임에 주의해야 합니다.

7.3.4 폐루프 속도 제어기

속도 제어기는 위치제어 모드, 속도제어 모드에서 동작합니다.

그림 7-5에서 속도 제어기의 블록선도를 보여줍니다. 모터의 속도(Velocity Feedback)와 위치 제어기 또는 사용자가 내린 속도 명령(Velocity Command)이 입력으로 인가되며, 동작 주파수는 1KHz입니다. 전류 제어기와 같이 속도 제어기도 PI 제어기로 구성되어 있으며 제어기의 출력(Current ref.)은 전류 값이고 전류 제어기의 전류 명령으로 입력됩니다.

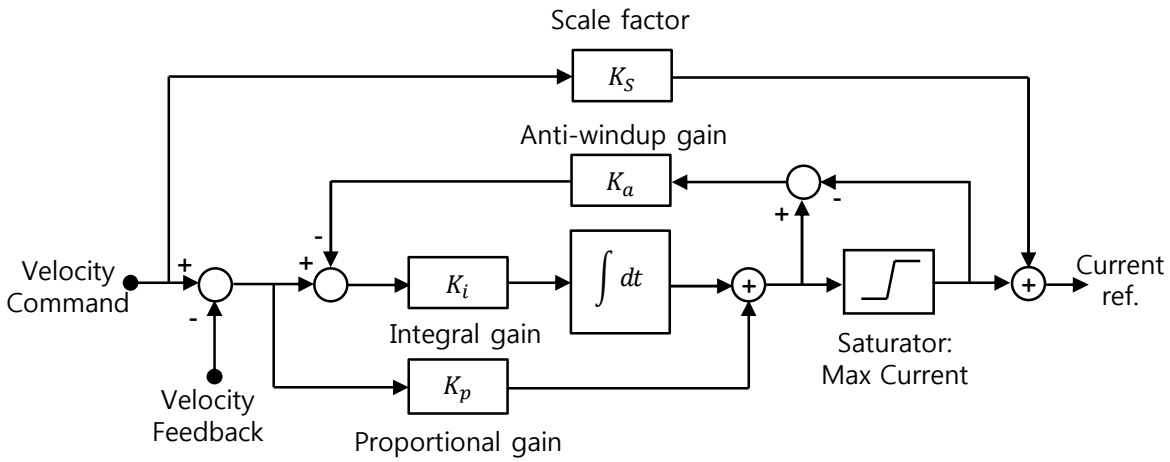

그림 7-5 제어기 내부에 설계된 속도 제어기

PI제어기의 비례 이득(Proportional gain)은 속도 명령과 모터의 속도 간의 오차에 비례하여 전류(Current ref.)를 조절합니다. 오차가 크다면 높은 전류가 출력되고 모터의 속도가 속도 명령에 다가갈수록 전류는 점차 낮아지게 됩니다.

PI제어기의 적분 이득(Integral gain)은 시간에 따른 오차의 합에 비례하여 전류를 조절합니다. 오차가 0으로 접근할수록 제어기가 속도 명령에 정확히 도착하고 유지하는데 도움을 줍니다.

속도 제어기에도 전류 제어기와 같이 안티와인드업 제어기가 설계되어 있습니다. 제어기의 안티

와인드업 이득(Anti-windup gain)은 비례 이득의 역수를 사용하도록 설정되어 있으며 사용자가 설정할 수는 없습니다.

제어기는 내부적으로 속도 명령을 적절한 단위변환을 거쳐 바로 전류 제어기의 전류 명령(Current ref.)으로 인가하는 일종의 스케일 팩터(Scale Factor) 역할을 수행하는 이득을 가지고 있습니다. 위치 센서로 사용하는 엔코더의 해상도가 너무 낮거나 위치 센서로 포텐셔미터가 사용될 경우, 위치 센서로부터 계산된 속도는 양자화되거나 노이즈를 포함하여 부정확합니다. 이러한 상황에서는 속도 제어기의 PI제어기를 바이패스 하는 것이 좋습니다. PI제어기의 비례 이득과 적분 이득을 '0'으로 두고, 속도 명령이 전류 명령 이 되도록 스케일(단위변환을 포함)만 변경하여 사용하면 됩니다.

표 7-9에는 속도 제어기의 오브젝트들이 정리되어 있습니다. 여기서 구성 파라미터(Configuration Parameter)는 사용자가 설정할 수 있는 값들입니다.

표 7-9 속도 제어기에서 사용하는 오브젝트들 정리

Name	Unit	Object Type	Object Name
Velocity Feedback	**rad/sec**	Status	**velocity**
Velocity Command	**rad/sec**	Command	**velocity_command**
Current ref.	A	Internal Value	-
Proportional gain(K_p)	-	Configuration Parameter	**vc_kp**
Integral gain(K_i)	-	Configuration Parameter	**vc_ki**
Anti-windup gain(K_a)	-	Internal Value, $K_a = \frac{1}{K_p}$	**vc_ka**
Scale factor(K_S)	A/(rad/sec)	Configuration Parameter	**vc_ks**
Max Current	[A]	Configuration Parameter	**max_current**

Motor Control UI 유틸리티에서 제어기의 속도관련 구성 파라미터를 설정할 때는 [RPM] 단위를 사용합니다. 하지만 제어기 내부에는 [rad/sec] 단위로 변환되어 계산에 사용됩니다.

※ 모터의 속도(Velocity Feedback)와 속도 명령(Velocity Command)이 제어기 내부에서 계산에 사용될 때는 단위가 [RPM]에서 [rad/sec]로 변환됩니다.

7.3.5 페루프 위치 제어기

위치 제어기는 위치제어 모드에서만 동작합니다.

그림 7-6에서 위치 제어기의 블록선도를 보여줍니다. 모터의 위치(Position Feedback)와 사용자가 내린 위치 명령(Position Command)이 입력으로 인가며, 동작 주파수는 100Hz 입니다. 위치 제어기는 PID 제어기로 구성되어 있으며, 제어기의 출력(Velocity ref.)은 속도 값이고 프로파일 모드의 설정 여부에 따라 프로파일 생성기(Profile Generator)로 입력되거나 속도 제어기의 속도 명령으로 입력됩니다.

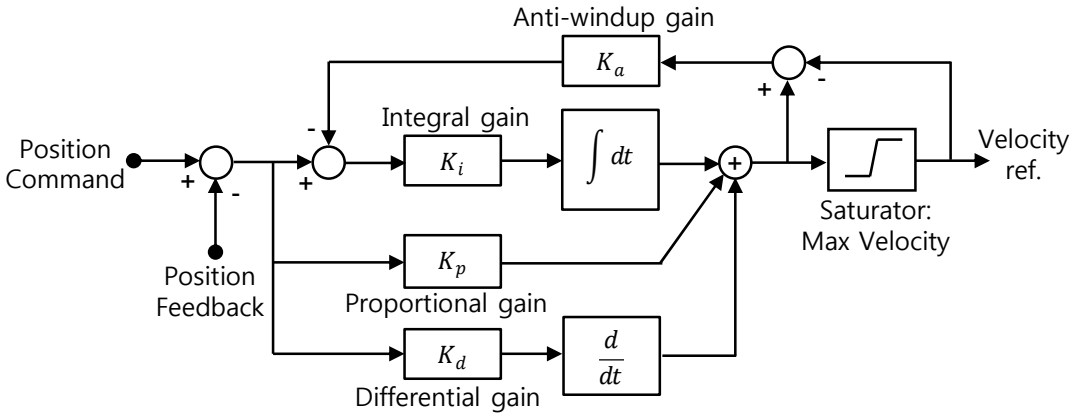

그림 7-6 제어기 내부에 설계된 위치 제어기

PID제어기의 비례 이득(Proportional gain)은 위치 명령과 모터의 위치 간의 오차에 비례하여 속도를 조절합니다. 오차가 크다면 높은 속도가 출력되고 모터의 위치가 위치 명령에 다가갈수록 속도는 점차 낮아지게 됩니다.

PID제어기의 적분 이득(Integral gain)은 시간에 따른 오차의 합에 비례하여 속도를 조절합니다. 오차가 0으로 접근할수록 제어기가 원하는 위치에 정확히 도착하고 유지하는데 도움을 줍니다.

PID제어기의 미분 이득(Differential gain)은 오차에 대한 변화를 계산합니다. 이러한 변화는 원하는 위치 또는 측정된 위치 값이 급격하게 변할 때마다 상대적으로 높은 수치가 됩니다. 이 부분의 효과는 원하는 위치 값의 변경으로 인해 모터를 구동하기 시작할 때 추가적인 속도를 더해줍니다. 미분 이득은 크게 오버슈트와 진동을 감소하는 데 도움이 됩니다.

위치 제어기에도 전류나 속도 제어기와 같이 안티와인드업 제어기가 설계되어 있으며, 제어기의 안티와인드업 이득(Anti-windup gain)은 비례 이득의 역수를 사용하도록 설정되어 있으며 사용자가 설정할 수는 없습니다.

표 7-10에는 위치 제어기의 오브젝트들이 정리되어 있습니다. 여기서 구성 파라미터(Configuration Parameter)는 사용자가 설정할 수 있는 값들입니다.

표 7-10 위치 제어기에서 사용하는 오브젝트들 정리

Name	Unit	Object Type	Object Name
Position Command	pulse	Status	**position_command**
Position Feedback	pulse	Command	**position**
Velocity ref.	**rad/sec**	-	**-**
Proportional gain(K_p)	-	Configuration Parameter	**pc_kp**
Integral gain(K_i)	-	Configuration Parameter	**pc_ki**
Differential gain(K_d)	-	Configuration Parameter	**pc_kd**
Anti-windup gain(K_a)	-	Internal Value, $K_a = \frac{1}{K_p}$	**pc_ka**
Max Velocity	**rad/sec**	Configuration Parameter	**max_velocity**

7.3.6 프로파일 생성기

프로파일 생성기(Profile Generator)는 사용자의 속도 명령이나 위치 제어기의 속도 출력을 받아 급가감속을 하지 않도록 사다리꼴로 속도 프로파일을 만드는 기능을 수행합니다. 그리고 속도 제어기와 같은 1KHz의 동작 주파수를 가집니다. 사용자는 **'Profile Mode'** 구성 파라미터 설정에 따라 프로파일 생성기를 사용 여부를 선택할 수 있습니다.

프로파일 생성기에 대한 자세한 내용은 "5.1.2 가속도와 감속도"를 참조하기 바랍니다.

7.4 모터제어기 이득 조정

제어기의 이득은 모터 전원, 기어 비율, 부하, 관성 등 많은 기계적인 파라미터를 모델링 하기 어렵기 때문에 기본적으로 실험을 반복하면서 수작업으로 동조하게 됩니다.

동조 과정은 먼저 전류 제어기의 PI 이득을 설정하고 다음으로 속도 제어기의 PI 이득을 설정합니다. 마지막으로 위치 제어기의 PID 이득을 설정합니다.

Motor Control UI 유틸리티는 위치/속도/전류 제어기의 Proportional(P), Integral(I), Differential(D) 이득을 설정하기 위한 구성 화면을 제공합니다. 또한, 모터를 구동하고 모니터링할 수 있는 또 다른 화면을 제공하여 이러한 실험적인 과정을 쉽게 할 수 있도록 합니다.

7.4.1 전류 제어기 PI 이득 조정

전류제어기는 직렬로 연결된 제어기 구조에서 모터와 직결된 제어기로 가장 먼저 동조되어야 합니다.

전류제어 모드를 사용하는 대부분의 응용(application)들은 제어기의 응답 특성이 너무 빠를 것을 요구하지 않습니다. 그래서 넓은 범위의 PI 이득 조정이 가능하고 좋은 결과도 얻을 수 있습니다. 전류 제어기에서는 P와 I 이득을 조정합니다. 제어기에 설정된 기본 이득(P=0.1, I=50)을 사용하여 첫 테스트를 수행하고, 필요에 따라 만족스러운 결과에 도달 할 때까지 I와 P 이득 조정을 반복합니다.

만일 제어기에 설정된 기본 이득을 사용하지 않고 제로 상태(P=0, I=0)에서 이득을 조정할 수 있습니다. 이와 같이 이득을 조정하려면 다음 절차를 따르도록 합니다:

1. P와 I 이득을 0으로 설정합니다.
2. I 이득을 50정도의 값에서 시작하여 20%정도의 오버슈트(overshoot)가 발생할 때까지 올립니다. 올릴 때는 50, 100, 200, 400, 800 과 같이 2배씩 증가시킵니다.
3. P 이득을 0.1에서 시작하여 오버슈트가 없어질 때까지 올립니다. 올릴 때는 0.2, 0.5, 1, 2, 5 와 같이 2배씩 증가시킵니다.

상기 조정 과정에서 적분 이득이 가장 중요하고 먼저 조정되어야 합니다. 다음으로 응답 시간 및 제어 안정성을 향상시키도록 비례 이득을 조정합니다.

※ 전류 제어기의 이득을 변경하고 응답성을 테스트할 때는 모터 회전축을 고정한 상태에서 진행하여야 합니다. 모터가 회전하는 상태에서는 모터의 역기전력으로 인해 전류의 흐름을 방해하게 됩니다.

7.4.2 속도 제어기 PI 이득 조정

속도 제어기를 동조하기에 앞서 전류 제어기의 이득이 동조되어 있어야 합니다. 그렇지 않은 경우, 앞 절을 참고하여 전류 제어기의 이득을 먼저 동조하기 바랍니다.

속도 제어기는 PI 제어기가 기본 구조이며 P와 I 이득을 조정합니다. 조정의 최종 목표는 모터가 오버슈트나 진동 없이 빠르게 원하는 속도에 도달하는 것입니다.

모터에 의해 움직이는 부하가 결정되지 않은 경우, PI를 예상되는 최소 부하에 대해 조정한 후 예상되는 최대 부하에서 다시 조정합니다. 그런 다음 두 조건에서 작동하는 값을 찾습니다. 최소 및 최대 부하간의 차이가 큰 경우에는 만족스러운 동조 값을 찾을 수 없는 경우도 있습니다.

속도 제어기를 동조할 때 최종적인 제어 대상이 위치인지(위치제어 모드) 혹은 속도인지(속도제어 모드)에 따라 방법이 조금 다릅니다.

위치제어 모드에서 사용되는 속도제어기를 동조해야 한다면, 속도제어기를 아주 세밀하게 동조할 필요는 없습니다. 속도제어기는 지정된 속도를 오버슈트 없이 빠르게 추종하는 정도로만 설정하고 위치제어기를 세밀하게 동조해야 합니다. 이런 경우 속도제어기에서는 P 이득만 설정하고 I 이득은 사용하지 않도록 합니다. I 이득을 사용하지 않을 경우 시스템의 응답성은 빨라지지만, 정상상태 오차가 발생합니다. 이 오차는 위치제어기를 통해 극복하면 됩니다.

먼저, 위치제어 모드에 사용되는 속도제어기의 동조 방법입니다:

1. I 이득은 0으로 설정합니다.
2. P 이득을 0.01부터 시작하여 오버슈트가 발생하기 전까지 올립니다. 올릴 때는 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2 와 같이 2배씩 증가시킵니다.

상기 과정에서, P 이득이 낮으면 모터의 속도가 속도 명령을 느리게 따라갑니다. P 이득을 올리게 되면 속도 명령에 도달하는 시간도 짧아지고 정상상태 오차도 줄어들게 됩니다. 계속해서 P 이득을 높여가면 오버슈트가 발생합니다. 하지만 정상상태 오차는 여전히 존재하게 됩니다. 정상상태 오차를 없애기 위해 P 이득을 더 올리면 결국 진동이 발생하고 제어기는 불안정해집니다.

※ 위치제어에 사용되는 속도제어기의 이득을 설정할 때는 오버슈트가 없는 상태에서 이득 조정을 중단합니다.

이전 단계에서 속도제어기의 P 이득을 조정하여 정상상태 오차는 있지만 오버슈트가 없는 상태로 이득을 조정해 둔 상태입니다.

다음으로 속도제어 모드에 사용되는 속도제어기의 동조 방법입니다:

3. I 이득을 0.01부터 시작하여 정상상태 오차가 없어질 때까지 올립니다. 올릴 때는 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2 와 같이 2배씩 증가시킵니다.
4. I 이득을 조정하여 정상상태 오차가 없어지지 않는다면 P 이득을 낮추고 과정을 반복해 봅니다.

정상상태 오차를 좀 더 빨리 극복하기 위해 I 이득을 더 높이게 되면 오버슈트가 커지게 되며, 여기서 I 이득을 더 높이게 되면 결국 진동이 발생하게 됩니다. 그러니 오버슈트와 정상상태 오차를 상황에 따라 절충하여 I 이득을 설정해야 합니다.

7.4.3 위치 제어기 PID 이득 조정

위치 제어기를 동조하기에 앞서 전류 제어기의 이득과 속도 제어기의 이득이 동조되어 있어야 합니다. 그렇지 않은 경우, 앞 절을 참고하여 속도 제어기와 전류 제어기의 이득을 먼저 동조하기 바랍니다.

위치 제어기에서는 P 이득을 기본적으로 조정하고 필요에 따라 D와 I 이득을 조정합니다. 조정의 최종 목표는 모터가 오버슈트나 진동 없이 빠르게 원하는 위치에 도달하는 것입니다. 만일 약간의 오버슈트나 정상상태 오차를 허용하는 응용이라면, P 이득을 세밀하게 조정하고 I와 D 이득은 작거나 0으로 설정합니다.

다음은 위치 제어기의 이득 동조 방법입니다:

1. 먼저 I 이득과 D 이득을 0으로 설정합니다.
2. P 이득을 0.01부터 시작하여 오버슈트가 발생하기 전까지 올립니다. 올릴 때는 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5와 같이 2배씩 증가시킵니다.

오버슈트를 없애거나 응답 속도를 개선하려면 다음 과정을 따릅니다:

3. 이전 과정에서 오버슈트 없이 목표 위치에 도달하였다면, 20% 정도의 오버슈트가 나타날 때까지 P 이득을 올립니다.
4. D 이득을 0부터 시작하여 조금씩 증가하면서 오버슈트가 없어지도록 합니다.

상기 과정에서 D 이득을 더 늘이게 되면 목표 위치에 도달하는 시간이 지연됩니다.

위치 제어기에서 일반적으로 I 이득은 잘 사용되지 않습니다. 위치제어에서는 P 이득만으로도 정상상태 오차가 나타나지 않으며, I 이득은 모터제어의 응답 특성을 전반적으로 느리게 하기 때문입니다. 하지만 특별한 이유로 I 이득을 설정해야 한다면, P 이득 조정 → D 이득 조정 → I 이득 조정 순서로 이득을 조정하면 됩니다.

7.5 모터제어기의 동작 이상 감지

이 절에서는 모터제어기 동작 중 발생할 수 있는 에러 상황을 감지하고 모터의 전원을 차단하는 모터제어기의 안전 기능에 대해 설명합니다.

7.5.1 스톨 상황 감지

스톨(stall)이란 모터에 전력이 공급되지만, 부하를 움직이기에는 힘이 부족하여 모터가 멈추는 상황을 의미합니다.

스톨 상황 감지는 모든 모드(전압출력, 전류제어, 속도제어, 위치제어 모드)에서 사용됩니다. 이 기능을 사용하기 위해서는 엔코더나 홀 센서, 포텐셔미터와 같은 위치 센서가 제어기에 연결되어야 합니다.

모터제어기는 일정 시간 동안 주어지는 전력에 대해 아무런 회전이 감지되지 않을 경우 모터의 전원을 차단하는 안전 기능을 설정할 수 있습니다. 스톨 상황 감지에서는 다음과 같은 전력과 시간의 조합을 사용할 수 있습니다:

- 100ms at 10% PWM duty ratio
- 200ms at 20% PWM duty ratio
- 400ms at 30% PWM duty ratio
- 700ms at 40% PWM duty ratio
- 1s at 50% PWM duty ratio

만일 지정된 시간 동안 지정된 값보다 높은 전력이 모터에 공급되는데도 움직임이 감지되지 않는다면, 폴트를 발생하고 모터를 Power OFF 합니다.

7.5.2 폐루프 속도제어 오차 감지

폐루프 속도제어 오차 감지는 위치제어와 속도제어 모드에서 사용 됩니다.

모터제어기는 기계나 센서 고장으로 인한 큰 속도 오차를 감지하고 문제가 발생한 경우 모터의 전원을 차단하는 안전 기능을 사용할 수 있습니다. 속도 오차란 폐루프 속도 제어기에서 명령 속도(원하는 속도)와 피드백 속도(실제 속도) 간의 오차를 말합니다. 속도제어 오차 감지에는 다음과 같은 시간과 에러의 크기 조합을 사용할 수 있습니다:

- 100ms and error > 100 RPM
- 200ms and error > 200 RPM
- 400ms and error > 500 RPM
- 700ms and error > 1500 RPM

- 1s and error > 3000 RPM

만일 지정된 시간 동안 속도 오차가 지정된 값 이상이라면, 폴트를 발생하고 모터를 Power OFF 합니다.

※ 만일 속도 명령이 드문드문 불연속적으로 내려지는 경우에는 Velocity Error Detection 기능을 사용하지 않는 것이 좋습니다. 명령이 내려질 때 큰 속도오차가 발생하여 폴트가 발생하게 됩니다.

7.5.3 폐루프 위치제어 오차 감지

폐루프 위치제어 오차 감지는 위치제어 모드에서 사용 됩니다.

모터제어기는 기계나 센서 고장으로 인한 큰 위치 오차를 감지하고 문제가 발생한 경우 모터의 전원을 차단하는 안전 기능을 사용할 수 있습니다. 위치 오차란 폐루프 위치 제어기에서 명령 위치(원하는 위치)와 피드백 위치(실제 위치) 간의 오차를 말합니다. 위치제어 오차 감지에는 다음과 같은 시간과 에러의 크기 조합을 사용할 수 있습니다:

- 100ms and error > 100 pulse
- 200ms and error > 500 pulse
- 400ms and error > 2000 pulse
- 700ms and error > 5000 pulse
- 1s and error > 20000 pulse

만일 지정된 시간 동안 위치 오차가 지정된 값 이상이라면, 폴트를 발생하고 모터를 Power OFF 합니다.

※ 만일 위치 명령이 드문드문 불연속적으로 내려지는 경우에는 Position Error Detection 기능을 사용하지 않는 것이 좋습니다. 명령이 내려질 때 큰 위치오차가 발생하여 폴트가 발생하게 됩니다.

8 모터제어기 인터페이스

이번 장에서는 모터제어기에 전달되는 명령과 피드백 신호의 처리를 다룹니다. 다음 그림 8-1은 모터제어기의 입출력과 관련된 내부와 외부 인터페이스 구조를 보여주고 있습니다.

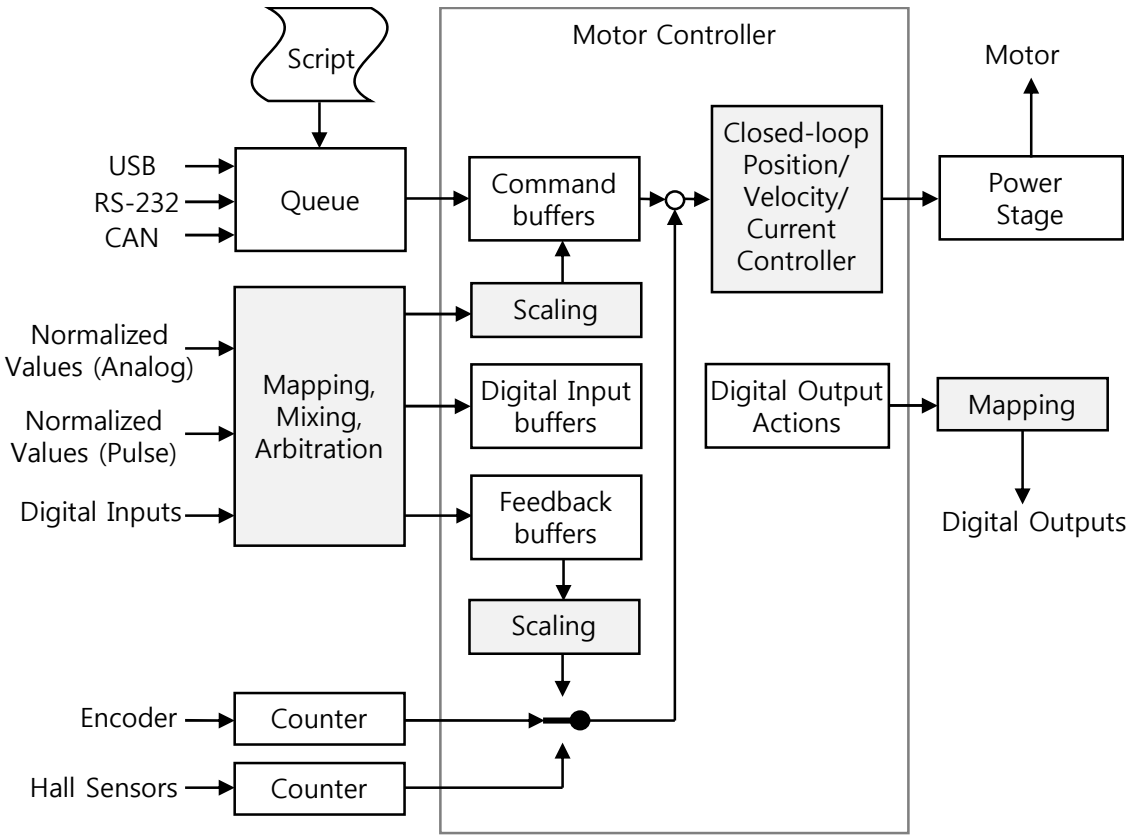

그림 8-1 모터제어기와 외부 I/O의 연결

8.1 모터 구동 명령

모터제어기는 다음 소스 중 하나에서 모터의 구동 명령을 받아들입니다:

- 통신 포트(RS-232, USB, CAN)
- 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.

8.1.1 통신으로 명령 전달

통신(USB 또는 RS-232, CAN) 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 2번 채널의 모터를 500rpm으로 구동하라는 명령을 내렸다면, 속도 명령 버퍼에 500이라는 값을 쓰고 모터제어기를 속도모드로 바꿔 주게 됩니다.

8.1.2 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.

만일 두 명령 소스가 각각 다른 채널의 모터를 향하고 있을 때는 문제가 되지 않습니다. 예로, 사용자가 USB 포트로 1번 채널의 모터에 위치 명령을 내리는 상황에서 스크립트가 2번 채널의 모터에 속도 명령을 내리는 상황입니다. 이러한 상황에서는 각각의 모터에서 명령이 올바르게 수행될 것입니다.

8.1.3 아날로그/펄스 입력 채널로부터 명령 전달

아날로그나 펄스 입력 채널 또한 명령을 수신하는 용도로 사용될 수 있습니다. 예를 들자면, 아날로그 입력 포트에 조이스틱을 연결하여 로봇을 조종하는 경우입니다. 이때도 아날로그 입력 채널로부터 입력되는 명령과 통신 포트에서 수신되는 명령 간에 충돌하지 않도록 주의해야 합니다. 만일 조이스틱으로 로봇을 조종하고 있을 때는 USB나 RS-232, CAN으로 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 연

속 신호로 봄).

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.

8.2 센서 피드백

제어기에 연결된 위치 혹은 속도 센서는 모터의 실제 위치나 속도를 측정하고 이를 원하는 위치나 속도와 비교합니다. 위치와 속도가 부하의 변화로 인해 바뀌는 경우, 모터제어기의 위치나 속도 제어기는 자동으로 모터에 공급되는 전력 출력을 보상합니다.

모터제어기는 다음 소스 중 하나에서 피드백을 받아들이게 됩니다:

- 엔코더 카운터
- 홀센서 카운터
- 펄스 입력 채널(PWM출력 위치센서, PWM출력 속도센서)
- 아날로그 입력 채널(포텐셔미터, 타코미터)

제어기는 내부적으로 전류를 측정하여 피드백으로 사용합니다. 그림 8-1에서는 전류 피드백에 관한 구성은 생략되어 있습니다.

8.2.1 엔코더 피드백

증분 엔코더는 모터의 위치와 속도를 측정하는데 사용될 수 있습니다. "4.6 광학식 증분 엔코더"를 참고하십시오.

8.2.2 홀센서 피드백

BLDC모터의 홀센서는 모터의 위치와 속도를 측정하는데 사용될 수 있습니다.

8.2.3 외부 속도, 위치 센서 피드백

아날로그 입력이나 펄스 입력 채널에 포텐셔미터나 타코미터를 연결하여 모터의 위치나 속도를 감지하는데 사용할 수 있습니다. 이 경우, 피드백 값은 명령의 전달과 같은 과정을 거쳐 모터제어기의 피드백 버퍼에 입력됩니다.

피드백 버퍼에 입력된 값은 -1과 1 사이의 정규화된 값으로 모터제어기의 명령 및 다른 운용 파라미터들과 단위의 일치가 필요합니다. 피드백 버퍼의 값이 스케일 변환(Scaling) 과정을 거쳐 모터제어기의 피드백 값으로 사용될 수 있습니다.

8.2.4 피드백 선택 스위치

여러 소스로부터의 피드백 값은 서로 믹싱 되지 않고 스위치에 의해 하나의 소스만 선택되어 폐루프 위치/속도/전류 제어기의 피드백 값으로 사용됩니다.

선택된 소스가 엔코더나 포텐셔미터와 같이 모터의 위치를 측정하는 센서라면, 모터의 속도는 위치를 차분하여 계산됩니다. 만일 타코미터와 같이 속도를 측정하는 센서라면, 모터의 위치는 속도를 증분하여 계산됩니다.

보통 포텐셔미터와 타코미터는 해상도가 높지 않고 측정값에 노이즈를 포함하고 있습니다. 오차가 조금이라도 있는 속도를 계속해서 위치에 증분하면 결과적으로 오차가 누적되어 모터의 실제 위치를 잃어버리게 됩니다. 또한, 작은 오차를 가지는 위치를 차분하여 속도를 계산할 때도 오차가 증폭되어 제대로 된 속도를 계산하지 못하게 됩니다.

※ 제어기는 다양한 종류의 센서를 피드백 센서로 사용할 수 있도록 하고 있습니다. 하지만 엔코더를 제외한 다른 센서(홀 센서, 포텐셔미터, 타코미터)를 모터의 위치와 속도 피드백 센서로 사용하는 것은 여러 가지 문제를 유발할 수 있기 때문에 권장하지 않습니다.

9 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. 제어기 모델별 포트의 개수, 전압 또는 전류 사양, I/O 커넥터의 위치 등은 데이터시트를 참조해야 합니다.

표 9-1 제어기에 사용되는 I/O의 기능과 장치 연결

I/O Type	Function	Sensor, Actuator
Digital Output	- Motor Power ON - Motor is Reversed - High Voltage Warning - High Temperature Warning	Buzzer, LED, Light, Brake, Shunt Load, Cooling Fan, Relay, Valve, Motor, Solenoid
Digital Input	- Emergency/Declaration/Quick Stop - Run Script - Forward/Reverse Limit Switch - Invert Direction - Load Home Counter	Switch류
Analog Input	- Motor Command: Voltage, Current, Velocity, Position - Motor Feedback: Position, Velocity	Joystick, Potentiometer, Tachometer, Thermistor, Voltage source
Pulse Input	- Motor Command: Voltage, Current, Velocity, Position - Motor Feedback: Position, Velocity	RC Radio, Absolute Encoder

아래 그림 9-1은 제어기의 I/O 입출력 신호의 전체 흐름을 단순화 된 블록 선도로 보여줍니다.

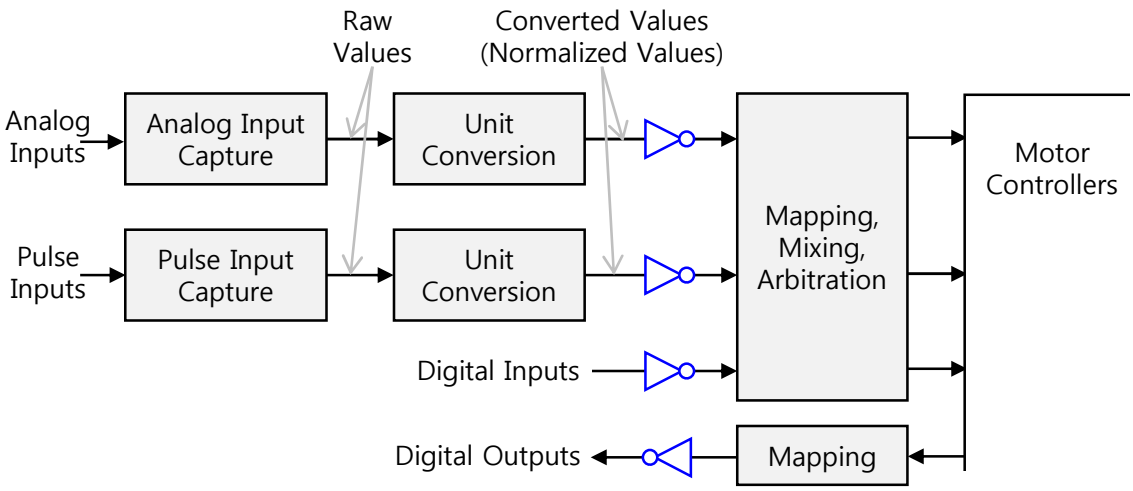

그림 9-1 I/O 입출력 신호의 전체 흐름을 보여주는 블록 선도

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, 믹싱, 중재(Mapping, Mixing, 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.

9.1 공통 설정

아날로그 입력, 펄스 입력, 디지털 입력, 디지털 출력의 사용 여부와 반전 여부는 다음 구성 파라미터들로부터 결정됩니다:

- **ai_enable - AI Enable**
- **ai_invert - AI Invert**
- **pi_enable - PI Enable**
- **pi_invert - PI Invert**
- **di_enable - DI Enable**
- **di_invert - DI Invert**
- **do_enable - DO Enable**
- **do_invert - DO Invert**

9.1.1 I/O 채널의 사용 여부 설정

아날로그 입력, 펄스 입력, 디지털 입력, 디지털 출력의 각 채널 별로 사용 여부(Enable 혹은 Disable)를 설정할 수 있습니다. 이는 4종류의 I/O에 대해 각각 32bit 구성 파라미터인 **'AI Enable', 'PI Enable', 'DI Enable', 'DO Enable'**를 사용하여 설정됩니다. 구성 파라미터의 각 비트는 해당 I/O 채널의 사용 여부를 나타냅니다:

- Bit0 - 채널 1의 사용 여부
- Bit1 - 채널 2의 사용 여부
- Bit2 - 채널 3의 사용 여부
- Bit3 - 채널 4의 사용 여부
- ...

해당 비트가 0인 경우 채널은 사용 불가능합니다. 1인 경우 채널은 사용 가능합니다.

디지털 출력 채널에 기능이 매핑되지 않은 경우, 스크립트 혹은 사용자가 직접 디지털 출력 채널을 제어할 수 있습니다.

아날로그 입력, 펄스 입력, 디지털 입력 채널에 센서가 연결되지 않으면 입력 포트가 플로팅(floating) 상태가 됩니다. 이러한 상태에서 채널을 활성화하고 값을 읽으면 쓰레기 값이 읽히게 됩니다.

9.1.2 I/O 채널의 반전 여부 설정

4종류의 I/O에 대해 각 채널의 반전 여부도 설정할 수 있습니다. 각각 32bit 구성 파라미터인 **'AI Invert', 'PI Invert', 'DI Invert', 'DO Invert'**를 사용하여 설정됩니다. 구성 파라미터의 각 비트는 해당 채널의 반전 여부를 나타냅니다:

- Bit0 - 채널 1의 반전 여부
- Bit1 - 채널 2의 반전 여부
- Bit2 - 채널 3의 반전 여부
- Bit3 - 채널 4의 반전 여부
- ...

해당 비트가 1인 경우, 입출력 값의 극성이 반전됩니다. 디지털 입출력의 경우 0은 1로 바뀌고 1은 0이 됩니다. 아날로그/펄스 입력의 경우 -1은 1로 바뀌고 1은 -1로 바뀝니다. 0의 값은 그대로 유지됩니다.

I/O의 종류에 따라 반전이 적용되는 시점이 다르다는 것에 주의해야 합니다(그림 9-1에서  기호가 반전 여부가 적용되는 위치를 표시함). 디지털 출력 채널의 경우, 반전 여부는 디지털 출력 포트로 신호가 나가기 직전에 적용됩니다. 디지털 입력 채널의 경우, 반전 여부는 디지털 입력 포트에서 신호를 읽고 나서 바로 적용됩니다. 아날로그와 펄스 입력의 경우, 포트에서 읽은 값이 정규화된 후 반전 여부가 적용됩니다.

9.2 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_function - DO Function**

사용자는 **'DO Function'** 파라미터를 설정하여, 제어기의 디지털 출력 채널을 모터제어기의 디지털 출력 버퍼의 기능으로 매핑할 수 있습니다. 아래의 그림 9-2는 사용 가능한 기능을 매핑 한 예를 보여줍니다. 여기서 검은색 화살표는 신호의 흐름이고 파란색 화살표는 매핑의 방향입니다.

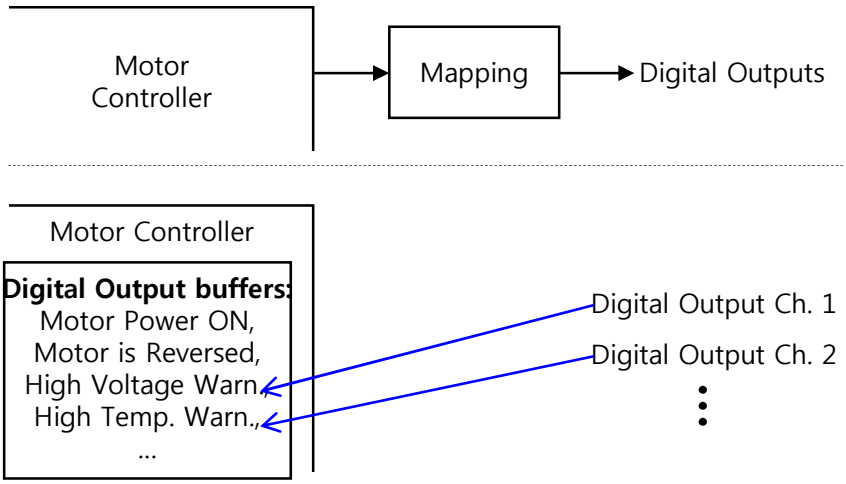

그림 9-2 디지털 출력 채널의 매핑 예

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.

9.3 아날로그 입력 캡쳐

아날로그 입력은 제어기에서 가장 간단하고 일반적인 입력 수단입니다. 보통 포텐셔미터나 포텐셔미터를 내장한 조이스틱을 연결하여 로봇을 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.

9.4 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.

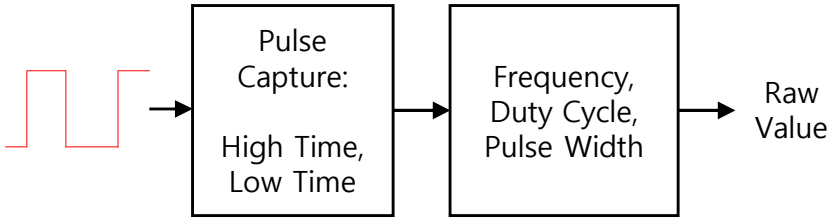

그림 9-3 펄스 신호 캡쳐

펄스 입력 캡쳐와 관련된 구성 파라미터는 다음과 같습니다:

- **PI Capture Type**

9.4.1 펄스 신호의 캡쳐

다음 그림 9-4과 같이, 펄스에서 ON 상태는 High time으로 OFF 상태는 Low Time으로 측정됩니다. Pulse Period는 High Time과 Low Time을 더한 시간입니다.

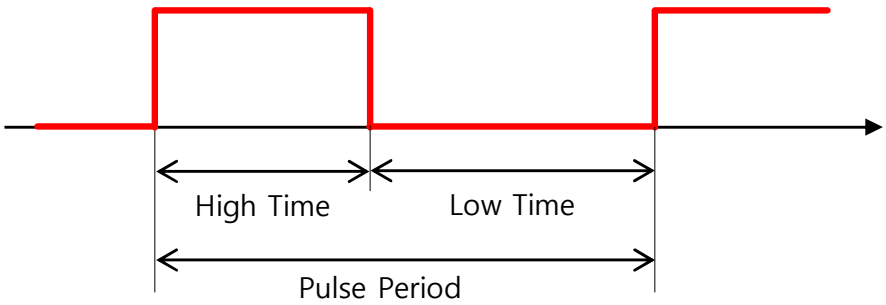

그림 9-4 펄스 신호의 구성

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‰

9.5 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 9-5와 같이 순차적으로 처리되어 변환된 값(Converted Value, Normalized Value)을 출력합니다.

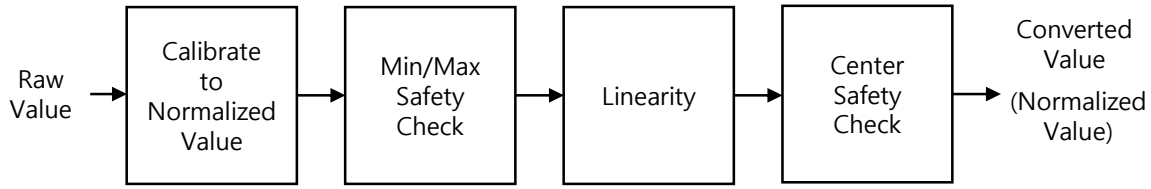

그림 9-5 아날로그/펄스 입력 값의 단위 변환

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

- **center_safety - Center Safety**
- **min_max_safety - Min/Max Safety**
- **ai_linearity - AI Linearity**
- **pi_linearity - PI 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_input_min - PI Input Min**
- **pi_input_max - PI Input Max**
- **pi_input_center - PI Input Center**
- **pi_input_deadband - PI Input Deadband**

9.5.1 정규화된 값으로의 캘리브레이션

센서로 측정하고자 하는 물리량의 값을 정확하게 측정하기 위해 센서를 연결한 후 반드시 캘리브

레이션 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 9-6과 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.

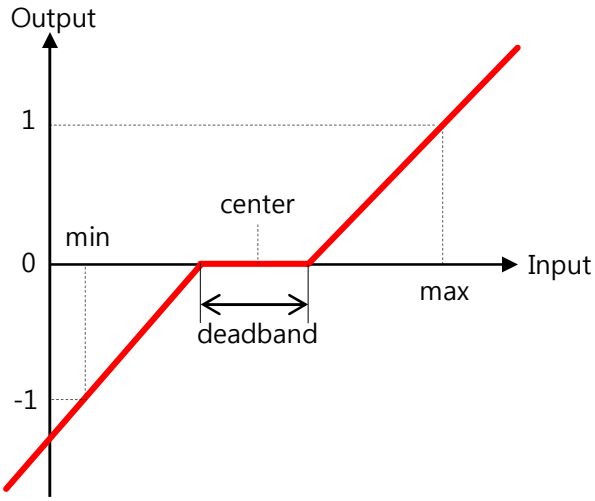

그림 9-6 아날로그 입력 값의 정규화

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.

9.5.2 Min/Max 안전 검사

사용자가 **'Min/Max Safety'** 파라미터의 설정을 통해, Min/Max 안전 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 9-7 참조).

Min/Max 안전 검사는 포텐셔미터의 결선이 끊어진 경우를 감지하는데 유용하게 사용할 수 있습

니다. 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.

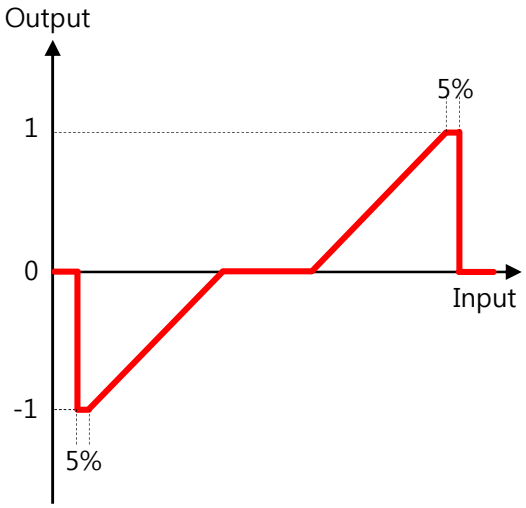

그림 9-7 Min/Max 범위 제한

※ Min/Max 안전검사 기능은 아날로그 입력 또는 펄스 입력이 모터의 속도명령이나 전류 명령, 전압 명령으로 매핑된 경우에만 사용 가능합니다.

9.5.3 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 9-8의 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)

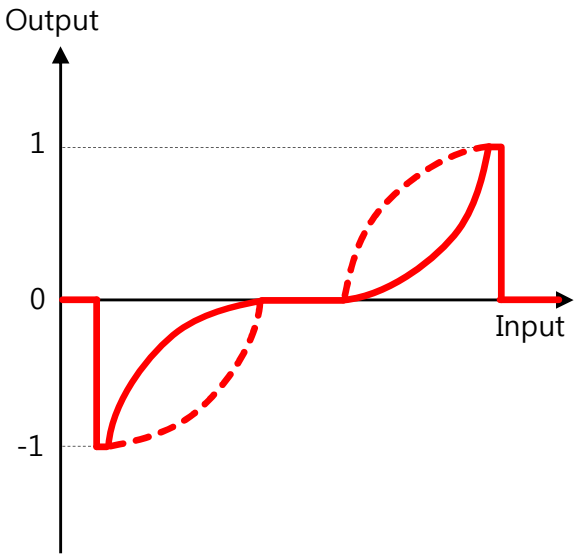

그림 9-8 Exponent 함수의 적용

9.5.4 센터 안전 검사

사용자가 **'Center Safety'** 파라미터의 설정을 통해, 센터 안전 검사(Center Safety Check) 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.

9.6 입력 값의 매핑, 중재, 믹싱

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

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

- **AI Function, PI Function, DI Function**
- **Control Mixing**

그리고 다음 아날로그/펄스 입력 값

- **AI Value**
- **PI Value**

은 매핑, 중재, 믹싱 과정을 거쳐 다음의 변환된 값을 만들어냅니다:

- **AI Converted Value**
- **PI Converted Value**

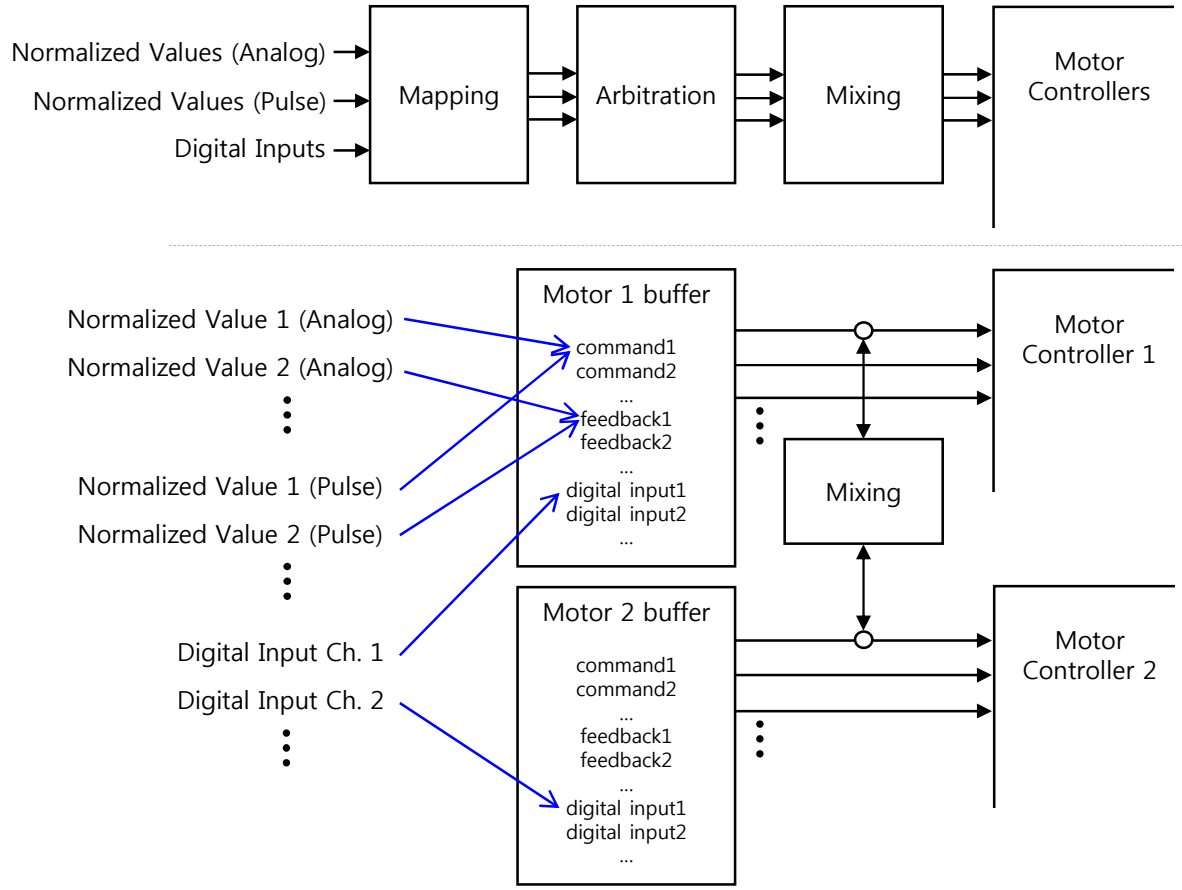

그림 9-9 아날로그/펄스/디지털 입력 신호의 흐름

매핑(Mapping)은 입력 채널을 모터의 특정 기능으로 연결합니다. 상기 그림에서 파란색 실성으로 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.

믹싱(Mixing) 기능은 듀얼 채널 제어기에서 유효합니다. 이 기능은 조이스틱 혹은 RC 조종기로 이동로봇을 조종할 때 유용하게 사용됩니다. 조이스틱으로부터 전진속도와 회전속도를 입력 받아 이를 좌우 모터의 속도로 믹싱하여 분배하는 기능입니다.

9.6.1 매핑

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

9.6.2 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 9-9와 같이 모터와 기능에 따라 중재를 위한 메모리 버퍼를 가지고 있습니다. 이 버퍼에 하나 이상의 값이 입력될 때 다음 규칙에 따라 중재됩니다:

- 명령(command)이나 피드백(feedback) 버퍼의 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 이동로봇에 연결하여 동시에 조종하는 경우 발생할 수 있습니다. 이때, 조이스틱은 조종에 사용되지 않아 입력이 0이고 RC 조종기로 조종하고 있는 상황이라 입력이 0이 아니라면 로봇은 RC 조종기에 의해 움직이게 됩니다.

9.6.3 신호의 믹싱

제어기의 믹싱 기능을 사용하려면, 듀얼 채널 제어기여야만 합니다. 싱글 채널 제어기에서는 믹싱 기능을 사용할 수 없습니다. 그리고 두 개의 아날로그 혹은 펄스 입력이 동시에 Enable 되어 있어야 하고 두 입력이 동일하게 전압 혹은 속도 명령으로 매핑되어 있어야 합니다.

다음 그림 9-10과 같이 일반적인 이동로봇은 두 개의 모터가 좌우 바퀴에 연결되어 구동됩니다. 이때 조이스틱이나 RC 조종기를 통해 내려지는 로봇의 구동 명령(전진 속도, 회전 속도)을 좌우 바퀴의 속도로 믹싱하여 분배해야 합니다.

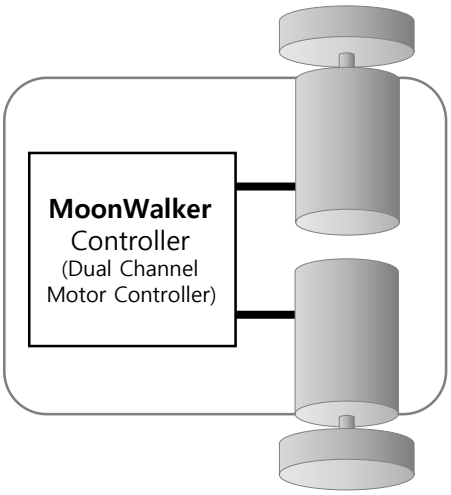

그림 9-10 신호의 믹싱에 의한 이동 로봇의 좌우 바퀴 제어

두 채널의 입력을 전진속도(v)와 회전속도(ω)로 보고 이동로봇의 좌우 바퀴의 속도(v_l, v_r)로 믹싱할 때, 사용자는 **'Control Mixing'** 파라미터 설정을 통해 다음과 같이 4가지 믹싱 모드를 선택하여 사용할 수 있습니다:

- Separate
- Mixing Mode 1
- Mixing Mode 2
- Mixing Mode 3
- Mixing Mode 4

Separate:
믹싱 기능을 사용하지 않습니다.

Mixing Mode 1:

최대값을 넘어가는 믹싱 결과에 대해 단지 최소 최대값 범위 내로 제한합니다.

$$v_l = v - \omega,$$
$$v_r = v + \omega.$$

$$v_l \leftarrow \text{Limit}(v_l),$$
$$v_r \leftarrow \text{Limit}(v_r).$$

여기서 Limit (x) 함수는 다음과 같이 정의됩니다:

$$x \leftarrow \begin{cases} x & (\text{-1} \le x \le 1) \\ \text{-1} & (x < -1) \\ 1 & (1<x) \end{cases}$$

Mixing Mode 2:

최대값을 넘어가는 믹싱 결과에 대해 최대값을 넘어간 비율대로 줄여줍니다.

$$v_l = v - \omega,$$
$$v_r = v + \omega.$$

$$v_l \leftarrow \frac{v_l}{r},$$
$$v_r \leftarrow \frac{v_r}{r}.$$

여기서 r은 다음과 같이 결정합니다:

$$r = \max(|v_l|, |v_r|)$$

Mixing Mode 3:

최대값을 넘어가는 믹싱 결과에 대해 전진속도를 우선 적용합니다.

$$r = |v| + |\omega| - 1$$

$$\omega \leftarrow \begin{cases} \omega - r & (0 < r \text{ and } 0 < \omega) \\ \omega + r & (0 < r \text{ and } 0 > \omega) \end{cases}$$

$$v_l = v - \omega,$$
$$v_r = v + \omega.$$

Mixing Mode 4:

최대값을 넘어가는 믹싱 결과에 대해 각속도를 우선 적용합니다.

$$r = |v| + |\omega| - 1$$

$$v \leftarrow \begin{cases} v - r & (0 < r \text{ and } 0 < v) \\ v + r & (0 < r \text{ and } 0 > v) \end{cases}$$

$$v_l = v - \omega,$$
$$v_r = v + \omega.$$

※ 아날로그/펄스 입력 신호의 믹싱은 속도 명령과 전압 명령에만 사용 가능합니다.

10 제어기 오브젝트

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

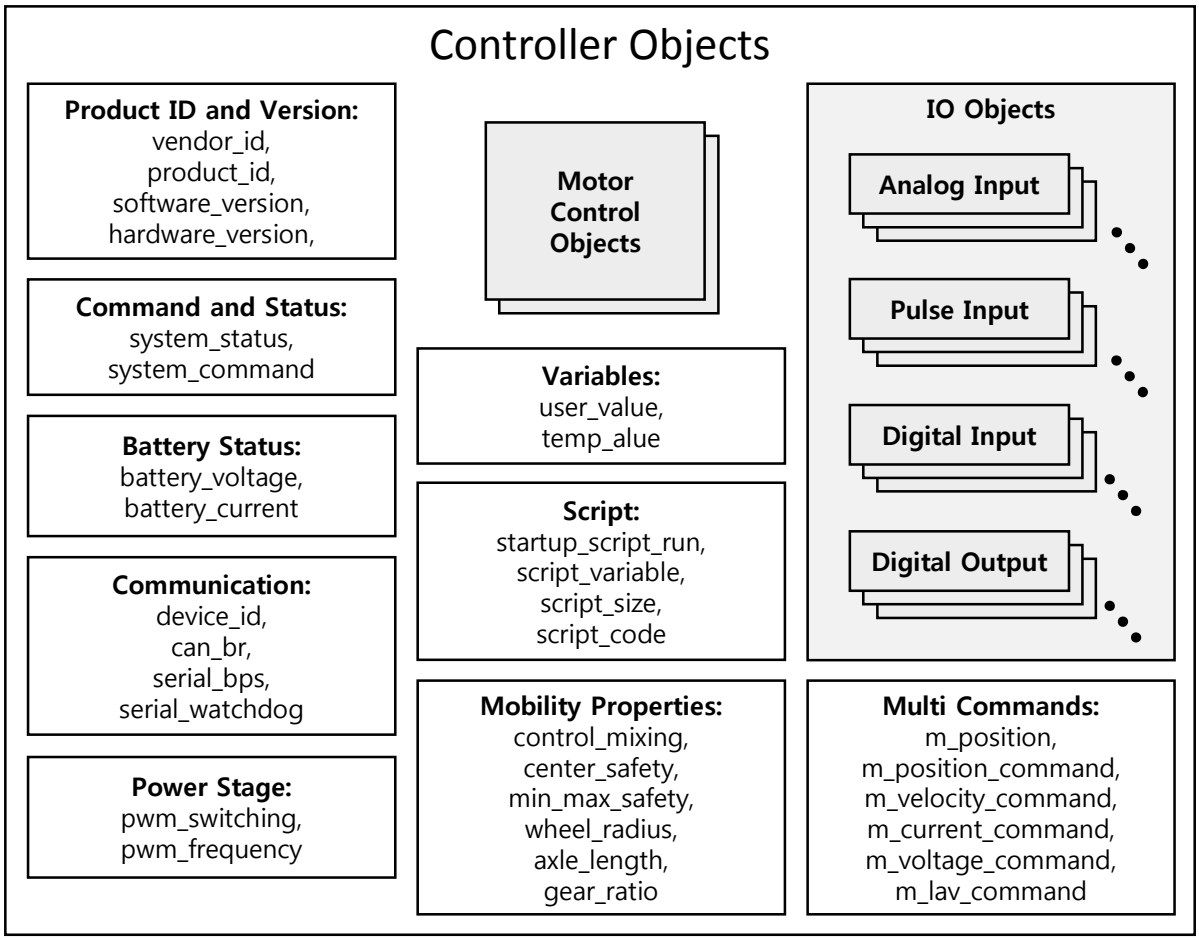

그림 10-1 Controller Objects

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 11장과 12장에서 설명합니다. 이번 장에서는 이 둘을 제외한 오브젝트들에 대해 설명합니다.

10.1 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 10-1 제품 ID 및 버전 정보 오브젝트

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

Type 열의 괄호 안 표기는 다음과 같습니다:

- (CN) - 상수(Constant) 오브젝트
- (CM) - 명령(Command) 오브젝트
- (ST) - 상태(Status) 오브젝트
- (CP) - 구성 파라미터(Configuration Parameter) 오브젝트
- (VA) - 변수(Variable) 오브젝트

10.1.1 vendor_id - Vendor ID

vendor_id는 제품 공급자 ID를 가집니다. 현재는 0으로 고정되어 있습니다.

10.1.2 product_id - Product ID

product_id는 제품의 ID를 가집니다. 현재 제어기에 적용된 제품 ID는 다음과 같습니다:

- 101 – 소형 싱글 채널 DC모터 드라이버 (30V, 3A)
- 102 – 소형 듀얼 채널 DC모터 드라이버 (30V, 3A)
- 201 – 중형 싱글 채널 DC모터 드라이버 (30V, 10A)

- 202 – 중형 듀얼 채널 DC모터 드라이버 (30V, 10A)
- 301 – 대형 싱글 채널 DC모터 드라이버 (50V, 80A)
- 302 – 대형 듀얼 채널 DC모터 드라이버 (50V, 40A)

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

10.1.3 software_version - Software Version

software_version은 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.

10.1.4 hardware_version - Hardware Version

hardware_version은 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.

10.2 제어기 명령 및 상태

제어기에 명령을 내리고 상태를 모니터링 합니다.

표 10-2 제어기 명령 및 상태 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
system_status, sst	5/0	I32 (CN)	제어기의 운용 상태
system_command, sco	7/0	I16 (CM)	제어기에 내려지는 명령

10.2.1 system_status - System Status

system_status는 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.

10.2.2 system_command - System Command

system_command에 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 동

안만 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.

10.3 배터리 상태

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

표 10-3 배터리 상태 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
battery_voltage, bv	8/0	F32 (ST)	전원 소스로부터 제어기에 공급되는 전압 (단위: V)
battery_current, bc	9/0	F32 (ST)	전원 소스로부터 제어기를 통과해 흐르는 전류 (단위: A)

10.3.1 battery_voltage - Battery Voltage

battery_voltage는 전원(파워서플라이, 배터리)으로부터 제어기에 공급되는 전압을 나타냅니다.

battery_voltage가 모터제어기의 **undervoltage_limit**과 **overvoltage_limit** 범위를 벗어나는 경우, 모터의 폴트(Fault)를 발생합니다. 또한 **high_voltage**보다 높은 경우, 디지털 출력을 트리거 하는 데도 사용됩니다.

10.3.2 battery_current - Battery Current

battery_current는 전원으로부터 제어기를 통과해서 흐르는 전류를 나타냅니다.

battery_current는 전원의 공급 선로에서 직접 측정되지 않습니다. 제어기의 회로가 소비하는 전류와 각 모터에서 소비하는 전류의 합으로 계산됩니다. 모터에서 소비하는 전류는 "6.2.3 전류 측정 방식"에서 설명한 것과 같이 모터 전류로부터 계산된 값입니다. 이 때문에 전원 공급장치에서 표시되는 전류와 **battery_current**는 차이가 날 수 있습니다.

10.4 통신 설정

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

표 10-4 통신 설정 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
device_id, id	11/0	I32 (CP)	장치 ID (범위: 1 ~ 255, 기본값: 1)
can_br, cb	12/0	I32 (CP)	CAN 통신 속도 (기본값: 1000)
serial_bps, sb	13/1	I32 (CP)	RS-232 통신 속도 (기본값: 115200)
	13/2	I32 (CP)	USB 통신 속도 (기본값: 115200)
serial_watchdog, sw	14/0	I32 (CP)	CAN, USB, RS-232 포트로 수신되는 명령어 패킷에 대한 와치독 타이머 타임아웃 값 (단위: ms, 범위: 0~10000, 기본값: 0)

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

10.4.1 deice_id - Device ID

CAN 또는 RS-422, RS-485 버스에는 여러 제어기가 동시에 연결되어 통신 선로를 서로 공유합니다. 이때, 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 마이크로컨트롤러의 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 USB 포트로 Motor Control UI 유틸리티를 연결합니다. 그리고 Configuration 탭에서 Serial/CAN Communication 그룹의 Device ID를 서로 중복되지 않도록 변경해야 합니다.

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

10.4.2 can_br - CAN Bitrate

만일 제어기를 CAN 버스에 연결한다면, CAN 버스에 연결되는 모든 제어기는 서로 통신 속도가 일치해야 합니다. **can_br** 은 CAN 통신 속도를 설정합니다.

can_br 파라미터 값으로 다음 중 하나를 설정하여 CAN 통신 속도를 변경합니다:

- 10 – 10Kbps
- 25 – 25Kbps
- 50 – 50Kbps
- 125 – 125Kbps
- 250 – 250Kbps
- 500 – 500Kbps
- 800 – 800Kbps
- 1000 – 1Mbps (기본값)

만일 상기 목록에 없는 값을 설정하게 되면, CAN 통신 속도는 1Mbps로 설정됩니다.

CAN 버스에 제어기를 연결하려면, 먼저 제어기의 USB 포트로 Motor Control UI 유틸리티를 연결합니다. 그리고 Configuration 탭에서 Serial/CAN Communication 그룹의 CAN Bitrate를 CAN 버스에 연결된 다른 장치들과 일치하도록 변경해야 합니다.

10.4.3 serial_bps - Serial Baudrate

PC나 마이크로컨트롤러의 USB 또는 RS-232 포트에 제어기를 연결하는 경우, **serial_bps**로 시리얼 통신 속도를 설정해야 합니다.

serial_bps 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 USB의 통신 속도를 설정하게 됩니다. 예로, 제어기에 연결된 텍스트 터미널로 "`serial_bps1 = 115200`"와 같이 입력하면 RS-232의 시리얼 통신 속도를 115200bps로 설정하게 됩니다. 만일 "`serial_bps2 = 921600`"와 같이 입력하면 USB의 시리얼 통신 속도를 921600bps로 설정하게 됩니다.

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

10.4.4 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 USB 또는 RS-232, CAN 포트를 통해 명령(**position_command, velocity_command, current_command, voltage_command** 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.

10.5 전원단 설정

전원단(Power stage)은 모터에 전력을 공급하기 위해 MOSFET 스위칭 소자로 구성된 H-bridge 회로를 말합니다.

표 10-5 전원단 설정 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
pwm_switching, ps	31/0	I8 (CP)	PWM 스위칭 방법
pwm_frequency, pf	32/0	I8 (CP)	PWM 주파수

전원단의 스위칭 방법과 PWM 주파수를 변경하는 것은 모터의 동작에 있어 가시적인 차이를 만들지는 않습니다. 되도록 설정 기본값을 사용하는 것이 좋습니다. 만일 구성을 변경하려면 "6.2 전력단"을 숙지하기를 권장합니다.

※ 상기 값을 변경한 경우에는 제어기를 새로 시작(제어기 전원을 끄고 켜거나 제어기를 리셋)하여야 변경된 값이 적용됩니다.

10.5.1 pwm_switching - PWM Switching

pwm_switching 파라미터는 전원단의 H-bridge 회로의 운용 방법을 결정합니다. H-bridge 회로의 각 스위칭 소자를 운용할 때 가장 많이 사용하는 방법은 Unipolar와 Bipolar 방식입니다.

이 파라미터의 설정 값으로 다음 중 하나를 선택합니다:

- 0 - Unipolar (기본값)
- 1 - Bipolar

일반적으로 'Bipolar' 구동은 저속에서의 속도 제어 특성이 좋지만, 모터의 인덕턴스가 낮은 경우 대기 소모 전력이 높으며 모터 발열을 일으킬 수 있습니다. 이런 경우에는 PWM 주파수를 높이거나 스위칭 방법을 'Unipolar' 방식으로 바꾸어야 합니다. 다른 방법으로 모터와 제어기 간에 인덕턴스가 높은 코일을 연결해서 해결할 수 있습니다.

10.5.2 pwm_frequency - PWM Frequency

pwm_frequency 파라미터는 전원단의 H-bridge 회로를 구성하는 MOSFET 스위칭 소자에 인가되는 PWM 신호의 주파수를 결정합니다.

이 파라미터의 설정 값으로 다음 중 하나를 선택합니다:

- 0 - 18kHz (기본값)
- 1 - 20kHz
- 2 - 24kHz
- 3 - 30kHz
- 4 - 36kHz,
- 5 - 40kHz

PWM 주파수가 가청주파수 대역에 있을 경우 모터에서 고주파 소음이 발생할 수 있습니다. 이때는 PWM 주파수를 높여 가청주파수 대역을 벗어날 수 있습니다. 하지만 H-bridge의 상하 MOSFET 쌍이 동시에 켜지지 못하도록 설정된 PWM 펄스의 데드밴드(Deadband)로 인해, 모터에 정격 전압을 공급하더라도 정격 회전속도에 도달하지 못하게 되는 폭이 조금씩 커집니다.

10.6 사용자/임시 변수

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

표 10-6 사용자 및 임시 변수 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
user_value, uv	56/1~32	I32 (CP)	사용자가 읽고 쓰는 변수들, 플래시 메모리에 저장 가능
temp_value, tv	57/1~32	I32 (VA)	사용자가 읽고 쓰는 변수들, RAM에만 저장 됨

10.6.1 user_value - User Value

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

system_command에서 "1 - Save properties in Flash Memory" 명령으로 제어기 구성 파라미터들을 저장한 경우, 이 값들은 제어기의 전원이 꺼지더라도 유지됩니다.

10.6.2 temp_value - Temp Value

일반적으로 **temp_value** 변수는 마스터 PC와 제어기에서 실행되는 스크립트간에 명령/상태를 주고받기 위해 사용합니다. 32개의 변수들을 저장할 수 있으며, 각각의 변수들은 Sub-index에 의해 구분됩니다.

이 값들은 제어기 전원이 꺼지면 소실됩니다.

10.7 스크립트 다운로드와 실행

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

표 10-7 스크립트 다운로드와 실행 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
startup_script_run, ssr	16/0	I8 (CP)	제어기 시작 시 스크립트의 실행 여부 (기본값: 0)
script_variable, sv	252/ 1~256	F32 (ST)	스크립트가 실행되는 동안 사용되는 변수
script_size, ss	250/0	I32 (VA)	바이트코드를 다운로드하기 전에 크기 전송 (단위: byte)
script_code, sc	251/0	I32 (VA)	바이트코드를 4-byte씩 묶어서 차례로 전송

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

10.7.1 startup_script_run - Startup Script Run

사용자가 **startup_script_run** 파라미터를 변경하여 제어기가 시작될 때 스크립트의 실행 여부를 설정합니다. 이 값으로 다음 중 하나를 선택합니다:

- 0 - 제어기 시작 시 스크립트 실행하지 않음 (기본값)
- 1 - 제어기 시작 시 스크립트 실행함

스크립트가 실행되면서 메모리의 오버플로우가 발생하는 경우에 시스템 충돌로 이어질 수 있습니다. 이때에는, 실행되는 스크립트를 중지하고 새로운 스크립트를 로드할 수 없게 되거나 PC와 통신할 수 없게 됩니다. 이러한 상황이 발생하는 것을 방지하기 위해, **startup_script_run**을 0으로 설정하고 사용자가 작성한 스크립트를 충분히 테스트 해야 합니다.

스크립트의 시작과 중지는 **system_command** 명령으로도 가능합니다. "10.2.2 system_command - System Command"을 참조하기 바랍니다.

10.7.2 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 1부터 순차적으로 부여됩니다. 변수의 값을 읽을 때는 변수 ID를 Sub-index로 지정합니다.

하이퍼터미널과 같은 유틸리티로 제어기의 USB 또는 RS-232 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, 스크립트가 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.

10.7.3 script_size - Script Size

스크립트의 소스코드는 제어기에 직접 다운로드 될 수 없고, 다운로드 되기 전에 반드시 바이트코드로 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.

10.7.4 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 엑세스 하는 것을 권장하지 않습니다. PC의 Motor Control UI 유틸리티를 사용하여 스크립트를 다운로드 하기를 권장합니다.

10.8 모바일 로봇 속성

듀얼 채널 제어기 모델(MW DCS02, MW DCM02, MW DCL02)은 이동로봇의 좌우 바퀴 모터의 구동에 바로 적용될 수 있습니다.

본 절의 오브젝트는 차동 구동형 이동로봇에 특화된 구성 파라미터들입니다. 이 구성 파라미터들은 듀얼 채널 제어기에서만 사용 가능합니다.

표 10-8 차동 구동형 이동로봇 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
control_mixing, cm	21/0	I8 (CP)	두 모터에 내려지는 속도/전압 명령의 믹싱 모드 (범위: 0 ~ 4, 기본값: 0)
center_safety, cs	22/0	I8 (CP)	모터에 내려지는 속도/전류/전압 명령의 Center Safety (범위: 0~1, 기본값: 0)
min_max_safety, ms	23/0	I8 (CP)	모터에 내려지는 속도/전류/전압 명령의 Min/Max Safety (범위: 0~1, 기본값: 0)
wheel_radius, wr	86/0	F32 (CP)	이동로봇의 바퀴 반지름 (단위: m)
axle_length, al	87/0	F32 (CP)	이동로봇 좌우 바퀴간 거리 (단위: m)
gear_ratio, gr	88/0	F32 (CP)	모터와 바퀴간 감속비율 (모터 회전수/바퀴 회전수)

10.8.1 control_mixing - Control Mixing

조이스틱 혹은 RC 조종기는 이동로봇을 조종하는 입력 장치로 주로 사용됩니다. 일반적으로 조이스틱의 입력을 전진속도(v)와 회전속도(ω)로 보고 차동 구동형 이동로봇의 좌우 바퀴의 속도(v_l, v_r)로 믹싱하여 분배합니다.

$v_l = v - \omega,$
$v_r = v + \omega.$

믹싱한 결과가 좌우 모터의 최대 속도를 넘어갈 때 처리하는 방식에 따라 4가지 믹싱 모드를 지원합니다. **control_mixing** 파라미터의 값으로 다음 중 하나를 선택합니다:

- 0 - Separate (기본값)
- 1 - Mixing Mode 1
- 2 - Mixing Mode 2
- 3 - Mixing Mode 3
- 4 - Mixing Mode 4

0 – Separate:
믹싱 기능을 사용하지 않습니다. 조이스틱의 입력은 믹싱되지 않고 이동로봇의 좌우 바퀴의 속도가 됩니다.

1 – Mixing Mode 1:
믹싱 결과가 최대 속도보다 크다면 최대 속도를 넘어가지 않도록 제한합니다.

2 – Mixing Mode 2:
믹싱 결과가 최대 속도보다 크다면 최대 속도를 넘어간 비율만큼 결과를 줄여줍니다.

3 – Mixing Mode 3:
믹싱 결과가 최대 속도보다 클 때 전진속도를 우선 적용하고 회전속도를 줄여 최대 속도 내로 제한합니다.

4 – Mixing Mode 4:
믹싱 결과가 최대 속도보다 클 때 회전속도를 우선 적용하고 전진속도를 줄여 최대 속도 내로 제한합니다.

믹싱 기능은 아날로그 입력 또는 펄스 입력에 적용됩니다. 아날로그/펄스 입력은 두 개의 모터에 각각 동일하게 속도 명령이나 전압 명령으로 매핑된 경우에만 활성화됩니다. "9.6.3 신호의 믹싱"을 참조하기 바랍니다.

10.8.2 center_safety - Center Safety

조이스틱이나 RC 조종기의 스틱이 중앙에 있지 않은 상태에서 제어기가 켜지면 모터가 갑자기 회전하여 위험한 상황을 초래할 수 있습니다. 이러한 상황을 방지하기 위해 센터 안전(Center Safety) 검사 기능을 사용합니다.

center_safety 파라미터의 값으로 다음 중 하나를 선택합니다:

- 0 - Disable (기본값)
- 1 - Enable

이 파라미터가 Enable로 설정되어 있다면, 모터에 전원이 공급되거나 조이스틱이나 RC 조종기가 연결될 때 속도 명령의 값이 100ms동안 0에 머무를 경우 입력 값을 받아들이기 시작합니다.

이 기능은 아날로그 입력 또는 펄스 입력에 대해 사용됩니다. 아날로그/펄스 입력이 모터의 속도 명령이나 전류 명령, 전압 명령으로 매핑된 경우에만 활성화됩니다. "9.5.4 센터 안전 검사"을 참조하기 바랍니다.

10.8.3 min_max_safety - Min/Max Safety

Min/Max 안전(Min/Max Safety) 검사 기능은 조이스틱의 연결이 단절되거나 망가져 잘못된 신호가 출력되는 것을 탐지하는 기능입니다. 이 기능은 조이스틱의 양단에 밴드 가드 저항을 닮으로 가능합니다.

아날로그/펄스 입력 값이 캘리브레이션의 입력 최소값과 최대값의 범위를 5% 이상 벗어나는 경우 입력 값이 유효하지 않은 것으로 판단하고 입력 값을 0으로 지정합니다.

min_max_safety 파라미터의 값으로 다음 중 하나를 선택합니다:

- 0 - Disable (기본값)
- 1 - Enable

이 기능은 아날로그 입력 또는 펄스 입력에 대해 사용됩니다. 아날로그/펄스 입력이 모터의 속도 명령이나 전류 명령, 전압 명령으로 매핑된 경우에만 활성화됩니다. "9.5.2 Min/Max 안전 검사"를 참조하기 바랍니다.

10.8.4 wheel_radius - Wheel Radius

wheel_radius 파라미터에는 이동로봇의 바퀴 반지름을 설정합니다.

이 파라미터는 **m_lav_command** 명령으로 이동로봇을 제어할 때, 전진속도와 각속도 명령을 좌/우 바퀴의 속도 명령으로 변환할 때 사용됩니다.

10.8.5 axle_length - Axle Length

axle_length 파라미터에는 이동로봇의 좌우 바퀴간 거리를 설정합니다.

이 파라미터는 **m_lav_command** 명령으로 이동로봇을 제어할 때, 전진속도와 각속도 명령을 좌/우 바퀴의 속도 명령으로 변환할 때 사용됩니다.

10.8.6 gear_ratio - Gear Ratio

gear_ratio 파라미터에는 이동로봇의 모터와 바퀴간 감속비율을 설정합니다. 감속비율은 바퀴가 1 회전 할 때 모터의 회전 비율을 의미합니다.

이 파라미터는 **m_lav_command** 명령으로 이동로봇을 제어할 때, 전진속도와 각속도 명령을 좌/우 바퀴의 속도 명령으로 변환할 때 사용됩니다.

※ wheel_radius, axle_length, gear_ratio 파라미터들은 m_lav_command 명령이 내려졌을 때 좌우 바퀴의 속도를 계산하는데 사용됩니다. 그리고 스크립트 프로그래밍에서 사용자의 이동 명령(전진 속도, 회전 속도)을 로봇의 좌우 바퀴 속도에 적용하거나 좌우 바퀴의 이동량으로부터 로봇의 위치를 추정하는데 사용될 수 있습니다.

10.9 듀얼 채널 명령

본 절의 듀얼 채널 명령들은 로봇의 좌우 바퀴에 동시에 구동 명령을 내리고 상태를 읽어 오는데 사용됩니다. 다음 명령은 시리얼(USB, RS-232) Text Packet 에서만 사용됩니다. 시리얼 Binary Packet 및 CAN 통신에서는 사용할 수 없습니다.

표 10-9 다중 명령 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
m_position, mp	91/0	(ST)	모터 채널 1,2의 엔코더 값을 읽음 (단위: pulse, pulse)
m_position_command, mpc	92/0	(CM) (ST)	모터 채널 1,2의 위치구동 명령을 내리고(단위: pulse, pulse) 엔코더 값을 읽음(단위: pulse, pulse)
m_velocity_command, mvc	93/0	(CM) (ST)	모터 채널 1,2의 속도 구동 명령을 내리고(단위: RPM, RPM) 엔코더 값을 읽음(단위: pulse, pulse)
m_current_command, mcc	94/0	(CM) (ST)	모터 채널 1,2의 전류 구동 명령을 내리고(단위: A, A) 전류 값을 읽음 (단위: A, A)
m_voltage_command, mvtc	95/0	(CM) (ST)	모터 채널 1,2의 전압 구동 명령을 내리고(단위: V, V) 엔코더 값을 읽음(단위: pulse, pulse)

m_lav_command, mla	96/0	(CM) (ST)	전진속도와 각속도로 이동로봇의 구동 명령을 내리고(단위: m/s, rad/s) 모터 1,2의 엔코더 값을 읽음(단위: pulse, pulse)

m_position_command, m_velocity_command, m_voltage_command, m_lav_command 명령의 리턴 값은 좌우 모터의 엔코더 값(단위: pulse, pulse)이며, **m_current_command** 명령의 리턴 값은 좌우 모터의 전류 값(단위: A, A)입니다.

10.9.1 m_position - Multi Position

m_position은 좌우 모터의 위치(주로 엔코더 카운트 값)를 2개의 32bit 정수로 읽어옵니다.

좌우 모터의 위치를 읽으려면 하이퍼터미널과 같은 유틸리티로 제어기의 USB 또는 RS-232 포트에 연결하여 다음과 같이 입력합니다.

```
mp↵
mp=41256,17448
```

여기서 ↵는 키보드의 Enter 키를 의미합니다. 터미널 설정에 따라 다르겠지만, 입력된 값은 일반적으로 터미널에 표시되지 않습니다. `mp`는 `m_psotion`의 Short Name 입니다.

10.9.2 m_position_command - Multi Position Command

m_position_command는 좌우 모터에 위치 명령을 동시에 내립니다. 그리고 좌우 모터의 위치를 읽어 옵니다.

좌우 모터를 10000, 10000 위치로 이동하기 위해서는 다음과 같이 입력합니다:

```
mpc=10000,10000↵
mpc=11256,27448
```

그러면 제어기는 좌우 모터의 현재 위치를 즉시 리턴하고, 명령을 실행하기 시작합니다. `mpc`는 `m_psotion_command`의 Short Name 입니다.

10.9.3 m_velocity_command - Multi Velocity Command

m_velocity_command는 좌우 모터에 속도 명령을 동시에 내립니다. 그리고 좌우 모터의 위치를 읽어옵니다.

좌우 모터를 500RPM, 500RPM 속도로 회전하기 위해서는 다음과 같이 입력합니다:

```
mvc=500,500↲
mvc=10000,10000
```

그러면 제어기는 좌우 모터의 현재 위치를 즉시 리턴하고, 명령을 실행하기 시작합니다. mvc는 m_velocity_command의 Short Name 입니다.

10.9.4 m_current_command - Multi Current Command

m_current_command는 좌우 모터에 전류 명령을 동시에 내립니다. 그리고 좌우 모터의 전류를 읽어옵니다.

좌우 모터에 1A, 1A 전류를 흘리기 위해서는 다음과 같이 입력합니다:

```
mcc=1,1↲
mcc=0.004,-0.002
```

그러면 제어기는 좌우 모터의 현재 전류 값을 즉시 리턴하고, 명령을 실행하기 시작합니다. mcc는 m_current_command의 Short Name 입니다.

10.9.5 m_voltage_command - Multi Voltage Command

m_voltage_command는 좌우 모터에 전압 명령을 동시에 내립니다. 그리고 좌우 모터의 위치를 읽어옵니다.

좌우 모터에 12V, 12V 전압을 가하기 위해서는 다음과 같이 입력합니다:

```
mvtc=12,12↲
mvtc=1561,1749
```

그러면 제어기는 현재 좌우 모터의 엔코더 위치를 즉시 리턴하고, 명령을 실행하기 시작합니다. mvtc는 m_voltage_command의 Short Name 입니다.

10.9.6 m_lav_command - Linear/Angular Velocity Command

전진속도와 각속도 명령을 이동로봇에 내립니다. 그리고 좌우 모터의 위치를 읽어옵니다.

m_lav_command 명령으로 전달받은 전진속도(v)와 각속도(ω)를 이동로봇의 좌우 바퀴에 대한 회전 속도(v_l, v_r)로 변환하는 식은 다음과 같습니다.

$$v_l = \frac{g}{r}\left(v - \omega\frac{b}{2}\right),$$
$$v_r = \frac{g}{r}\left(v + \omega\frac{b}{2}\right).$$

여기서 b 은 좌우 바퀴간 거리(**axle_length**), r 은 바퀴의 반지름(**wheel_radius**), g 는 바퀴의 감속 비율(**gear_ratio**)입니다.

이동로봇에 전진속도 1[m/s], 각속도 0.1[rad/s]로 구동 명령을 내리기 위해서는 다음과 같이 입력합니다:

```
mla=1,0.1↵
mla=1961,2749
```

그러면 제어기는 좌우 모터의 현재 위치를 즉시 리턴하고, 명령을 실행하기 시작합니다. `mla`는 `m_lav_command`의 Short Name 입니다.

11 모터제어기 오브젝트

제어기는 모델에 따라 한 개 또는 두 개의 모터를 연결하여 제어할 수 있습니다. 이 장에서는 모터 제어부의 구성 파라미터 설정 및 명령과 상태에 관련된 오브젝트들에 대해 설명합니다.

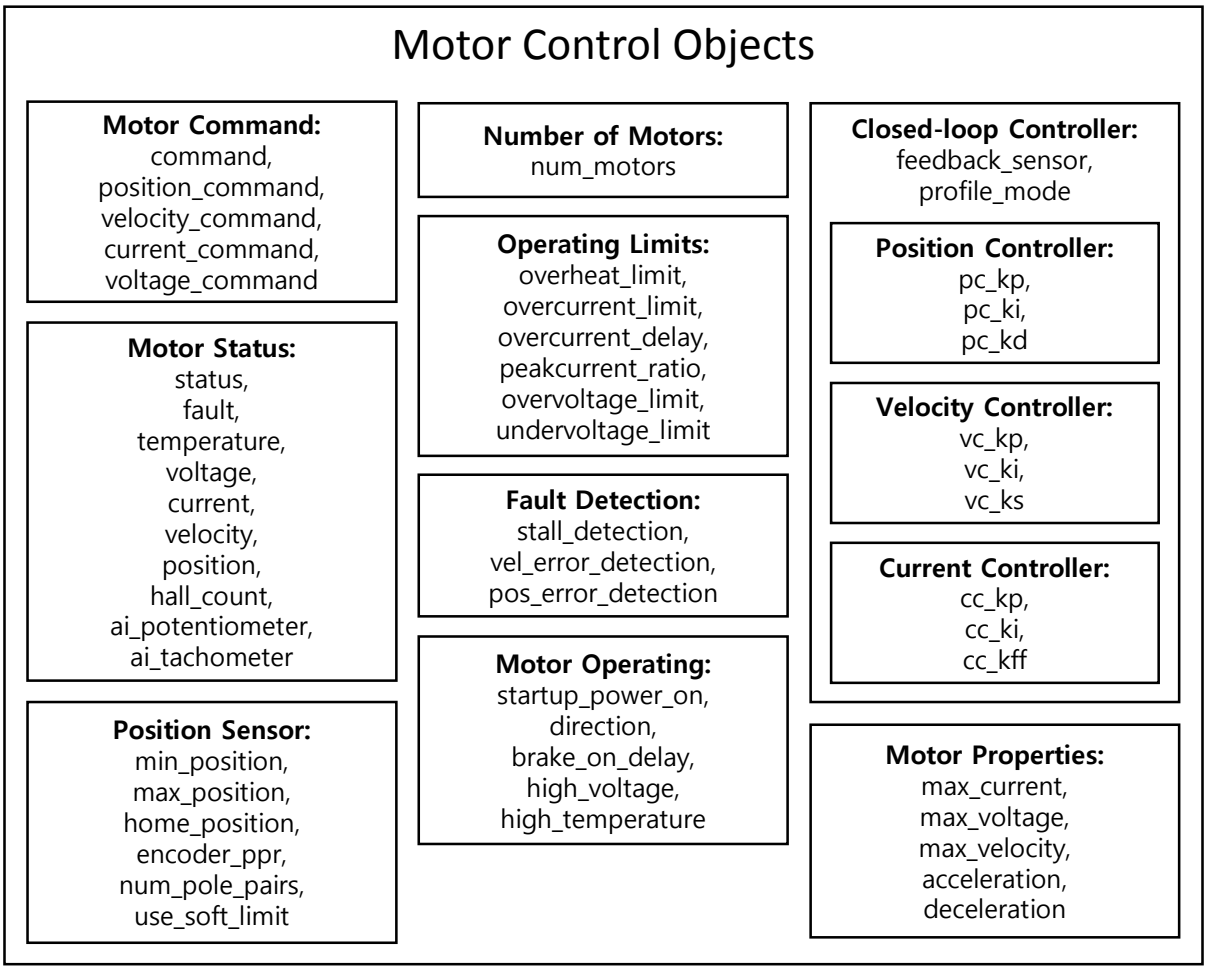

그림 11-1 Motor Controller Objects

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.

11.1 모터 개수

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

표 11-1 모터 개수 확인 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
num_motors, nm	80/0	I8 (CN)	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

Type 열의 괄호 안 표기는 다음과 같습니다:

- (CN) - Constant 오브젝트
- (CM) - Command 오브젝트
- (ST) - Status 오브젝트
- (CP) - Configuration Parameter 오브젝트
- (VA) - Variable 오브젝트

11.1.1 num_motors - Number of Motors

제어기에 연결 가능한 모터의 수는 최대 2개 입니다. **num_motors** 상수의 값으로 다음 중 하나를 가집니다:

- 1 - 싱글 채널 제어기
- 2 - 듀얼 채널 제어기

싱글 채널 제어기에는 최대 1개의 모터를 연결 가능합니다. 듀얼 채널 제어기에는 최대 2개의 모터를 연결 가능합니다.

11.2 모터 명령

이 절에서는 모터에 내려지는 위치, 속도, 전류, 전압 명령에 대해 설명합니다.

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

표 11-2 모터 명령 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
command, co	101/ 0~2	I16 (CM)	모터제어기에 내려지는 명령 코드
position_command, pc	111/ 0~2	I32 (CM)	모터의 폐루프 위치 제어 명령 (단위: pulse)
velocity_command, vc	112/ 0~2	F32 (CM)	모터의 폐루프 속도 제어 명령 (단위: RPM)
current_command, cc	113/ 0~2	F32 (CM)	모터의 폐루프 전류 제어 명령 (단위: A)
voltage_command, vtc	114/ 0~2	F32 (CM)	모터에 인가되는 전압 출력 (단위: V)

11.2.1 command – Command

command에 명령 코드를 쓰는 것으로 모터제어기의 해당 기능을 실행할 수 있습니다. 다음은 명령 코드 목록입니다:

- 0 - Motor Power OFF
- 1 - Motor Power ON
- 2 - Clear Fault Flags
- 6 - Deceleration Stop
- 7 - Quick Stop

0 - Motor Power OFF:
모터의 전원을 차단하고 모터를 제어하지 않습니다. 모터의 전원을 차단한 것은 모터가 전원으로부터 분리된 것과 같은 효과를 냅니다. 만일 모터가 회전하고 있는 상태에서 모터에 전원을 차단한다면, 모터의 속도가 줄 지 않고 계속 회전하는 상태로 남아있게 됩니다(마찰이 없을 때).

1 - Motor Power ON:
모터에 0V 전원을 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. 모터의 제어 모드가 위치 제어 모드라면 속도 제어 모드로 변경됩니다. 전류 제어 모드라면 전압 출력 모드로 변경됩니다.

7 – Quick Stop
모터를 최대한 빨리 정지합니다. 모터의 제어 모드가 위치, 속도, 전류 제어 모드라면 전압 출력 모드로 변경됩니다. 디지털 입력에서의 'Emergency Stop' 기능과는 구별됩니다.

모터가 Power ON 될 때는 다음 사항을 체크합니다. 이 중 하나라도 통과되지 못하면 모터는 Power OFF 상태에 머무르게 됩니다:

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

모터 Power ON을 성공하면 다음 과정이 진행됩니다:

1. 아날로그 입력과 펄스 입력의 센터 안전 검사 플래그를 리셋
2. 모터제어기는 전압 출력 모드로 설정하고 0V 전압 출력
3. 위치, 속도, 전류, 전압 명령을 모두 0으로 리셋

11.2.2 position_command - Position Command

모터의 페루프 위치 제어기가 추종해야 할 위치 명령을 내립니다. 명령을 내리기 전에 모터제어기가 속도 제어나 전류 제어, 전압 출력 모드에서 구동 중인 상황에 있었더라도 위치 제어 모드로 즉시 변경되고 **position_command**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. Profile Mode를 켠 상태라면 가속과 감속을 위해 **acceleration**과 **deceleration** 값이 사용됩니다.

position_command를 사용하려면, 모터제어기는 다음 사항을 만족하여야 합니다:

- 모터는 'Motor Power ON' 상태에 있어야 함
- 모터제어기의 디지털 입력 중 'Emergency Stop', '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

위치 제어 모드에서는 위치 제어기와 속도 제어기, 전류 제어기가 동시에 실행됩니다. "7.3 모터제어기 구조"를 참고하십시오.

11.2.3 velocity_command - Velocity Command

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 **velocity_command**로 주어진 명령 인수를 추종합니다.

속도 명령의 절대값은 **max_velocity**보다 클 수 없으며, –**max_velocity**와 **max_velocity**사이의 값을 가질 수 있습니다. Profile Mode를 켠 상태라면 가속과 감속을 위해 **acceleration**과 **deceleration** 값이 사용됩니다.

velocity_command를 사용하려면, 모터제어기는 다음 사항을 만족하여야 합니다:

- 모터는 "Power ON" 상태에 있어야 함
- 모터제어기의 디지털 입력 중 "Emergency Stop", "Quick Stop", "Declaration Stop" 가 꺼져 있거나 비활성화 상태(제어기의 디지털 입력 포트로 매핑되어있지 않은 상태)여야 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

속도 제어 모드에서는 전류 제어기가 동시에 실행됩니다. "7.3 모터제어기 구조"를 참고하십시오.

11.2.4 current_command - Current Command

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

전류 명령의 절대값은 **max_current**보다 클 수 없으며, –**max_current**와 **max_current** 사이의 값을 가질 수 있습니다.

current_command를 사용하려면, 모터제어기는 다음 사항을 만족하여야 합니다:

- 모터는 'Motor Power ON' 상태에 있어야 함

- 모터제어기의 디지털 입력 중 'Emergency Stop', '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 전류 명령이 0보다 작아야 함
- 모터제어기의 디지털 입력 중 'Reverse Limit Switch'가 켜져 있는 상태라면 전류 명령이 0 보다 커야 함

11.2.5 voltage_command - Voltage Command

voltage_command로 지정된 전압을 모터에 직접 내보냅니다. 명령을 내리기 전에 모터제어기가 위치 제어나 속도 제어, 전류 제어 모드에서 구동 중인 상황에 있었더라도 전압 출력 모드로 즉시 변경되고 **voltage_command**로 주어진 명령 인수를 추종합니다.

전압 명령의 절대값은 **max_voltage**보다 클 수 없으며, –**max_voltage**와 **max_voltage**사이의 값을 가질 수 있습니다.

current_command를 사용하려면, 모터제어기는 다음 사항을 만족하여야 합니다:

- 모터는 'Motor Power ON' 상태에 있어야 함
- 모터제어기의 디지털 입력 중 'Emergency Stop', '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 전압 명령이 0보다 작아야 함
- 모터제어기의 디지털 입력 중 'Reverse Limit Switch'가 켜져 있는 상태라면 전압 명령이 0 보다 커야 함

11.3 모터 상태

모터의 전압, 전류, 회전속도, 위치 카운트 값 등의 모터의 운용 상태를 읽어오는 오브젝트입니다.

표 11-3 모터 상태 확인 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
status, s	102/0~2	I32 (ST)	모터제어기의 현재 상태
fault, f	103/0~2	I32 (ST)	모터제어기 및 관련 I/O에서 발생한 폴트
temperature, tp	121/ 0~2	F32 (ST)	FET와 방열판의 온도 (단위: ℃)

voltage, vt	122/ 0~2	F32 (ST)	모터에 가해지는 전압 (단위: V)
current, c	123/ 0~2	F32 (ST)	모터에 흐르는 전류 (단위: A)
velocity, v	124/ 0~2	F32 (ST)	모터의 회전속도 (단위: RPM)
position, p	125/ 0~2	I32 (VA)	모터의 회전위치 (단위: pulse)
hall_count, h	126/ 0~2	I32 (ST)	BLDC 모터의 홀 센서 카운트 값 (단위: pulse)
ai_potentiometer, pt	131/ 0~2	I32 (ST)	아날로그 입력 포트에 매핑된 위치 센서의 피드백 값 (범위: -1 ~ 1)
ai_tachometer, ta	132/ 0~2	I32 (ST)	아날로그 입력 포트에 매핑된 속도 센서의 피드백 값 (범위: -1 ~ 1)

11.3.1 status - Status

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

- 0x0001 - Motor Power ON
- 0x0002 - Motor Moving
- 0x0004 - Fault Detected
- 0x0008 - Emergency Stop
- 0x0010 - Position Controller
- 0x0020 - Velocity Controller
- 0x0040 - Current Controller
- 0x0100 - AI Center Safety Check Failed
- 0x0200 - PI Center Safety Check Failed
- 0x0400 - AI Min/Max Safety Check Failed
- 0x0800 - PI Min/Max Safety Check Failed
- 0x1000 - DI Stop Requested
- 0x2000 - Exceed the Limit Range
- 0x4000 - Serial Command
- 0x8000 - AI/PI Command
- 0x10000 - Script Running

0x0001 - Motor Power ON:
모터에 전원이 공급되고 위치, 속도, 전류, 전압 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. 모터의 속도는 단위 시간당 엔코더 카운트 값의 변화량으로 계산합니다. 모터에 구동 명령이 내려지지 않은 상황에서 모터를 손으로 돌려 엔코더 값이 변하게 되어도 이 플래그가 켜집니다.

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.

0x0008 - Emergency Stop:
모터제어기의 디지털 입력에 매핑 된 'Emergency Stop' 스위치가 켜진 상태입니다. 이 경우 모터는 Power OFF 상태가 됩니다. 'Emergency stop' 스위치가 꺼질 때까지 'Motor Power ON' 상태가 되지 못합니다.

0x0010 - Position Controller:
모터의 폐루프 위치제어기가 동작 중입니다. 위치 제어기는 속도제어기와 전류제어기를 내장하고 있기 때문에 함께 동작합니다.

0x0020 - Velocity Controller:
모터의 폐루프 속도제어기가 동작 중입니다. 속도 제어기는 전류제어기를 내장하고 있기 때문에 함께 동작합니다.

0x0040 - Current Controller:
모터의 폐루프 전류 제어기가 동작 중입니다.

0x0100 - AI Center Safety Check Failed:

모터제어기의 속도나 전압 명령에 매핑된 아날로그 입력 값이 모터 전원 공급 후 0에 있지 않습니다. 입력 값이 0이 될 때까지 입력 값은 강제로 0이 됩니다.

0x0200 - PI Center Safety Check Failed:
모터제어기의 속도나 전압 명령에 매핑된 펄스 입력 값이 모터 전원 공급 후 0에 있지 않습니다. 입력 값이 0이 될 때까지 입력 값은 강제로 0이 됩니다.

0x0400 - AI Min/Max Safety Check Failed:
모터제어기의 속도나 전압 명령에 매핑된 아날로그 입력 값이 캘리브레이션의 Min/Max 범위를 벗어났습니다. 이 경우 입력 값은 강제로 0이 됩니다.

0x0800 - PI Min/Max Safety Check Failed:
모터제어기의 속도나 전압 명령에 매핑된 펄스 입력 값이 캘리브레이션의 Min/Max 범위를 벗어

났습니다. 이 경우 입력 값은 강제로 0이 됩니다.

0x1000 - DI Stop Requested:
모터제어기의 디지털 입력 포드 중 'Declaration Stop'이나 'Quick Stop'이 켜진 상태입니다. 이 상태가 꺼질 때까지 모터제어기에 내려지는 위치와 속도, 전류, 전압 명령은 차단됩니다.

0x2000 - Exceed the Limit Range:
모터제어기의 디지털 입력 포트 중 'Forward Limit Switch'나 'Reverse Limit Switch'가 켜진 상태입니다. 이 상태가 꺼질 때까지 모터제어기에 내려지는 위치와 속도, 전류, 전압 명령은 제한적으로 실행됩니다. 즉, 리미트를 벗어나는 방향으로 움직이는 명령만 수행됩니다.

0x4000 - Serial Command:
시리얼(RS-232, USB) 포트와 CAN 포트로부터 모터제어기에 구동(위치와 속도, 전류, 전압) 명령이 내려지고 있습니다. 이 플래그는 구동 명령 외(구성 파라미터 읽고 쓰기와 같은 것들)에는 반응하지 않습니다.

0x8000 - AI/PI Command:
모터제어기에 매핑 된 아날로그 입력 혹은 펄스 입력 포트로부터 구동 명령이 입력되고 있습니다.

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

'0x0010 - Position Controller', '0x0020 - Velocity Controller', '0x0040 - Current Controller' 플래그가 모두 꺼지고 '0x0001 - Motor Power ON' 상태가 켜진 후에는 전압 출력 모드가 됩니다.

'0x0100 - AI Center Safety Check Failed'와 '0x0200 - PI Center Safety Check Failed' 플래그가 작동하려면 아날로그 입력 포트 혹은 펄스 입력 포트가 제어기의 모터 구동 명령에 매핑되어 있어야 합니다. 이 상태를 체크하는 기능은 **center_safety** 파라미터에 의해 켜고 끌 수 있습니다. 조이스틱이나 RC 조종기의 입력 값이 0이 아닌 상황에서 모터에 전력이 공급되어 모터가 갑자기 움직이면 위험한 상황이 발생할 수 있기 때문에 이를 방지하기 위한 기능입니다.

'0x0400 - AI Min/Max Safety Check Failed'와 '0x0800 - PI Min/Max Safety Check Failed' 플래그가 작동하려면 아날로그 입력 포트 혹은 펄스 입력 포트가 제어기의 모터 구동 명령에 매핑되어 있어야 합니다. 이 상태를 표시하는 기능은 **min_max_safety** 파라미터에 의해 켜고 끌 수 있습니다. 조이스틱이나 RC 조종기의 연결이 끊어지거나 하여 잘못된 신호가 입력되고, 이로 인해 모터가 오작동하면 위험한 상황이 발생할 수 있기 때문에 이를 방지하기 위한 기능입니다.

'0x0100 - AI Center Safety Check Failed', '0x0200 - PI Center Safety Check Failed', '0x0400 - AI Min/Max Safety Check Failed', '0x0800 - PI Min/Max Safety Check Failed' 플래그가 켜지면 모터제어기의 입력 값은 0이 됩니다. 이 플래그가 켜지더라도 USB나 RS-232, CAN 포트로 내려지는 명

령은 처리 됩니다.

'0x4000 - Serial Command'나 '0x8000 - AI/PI Command' 플래그는 통신 포트나 I/O 포트로부터 모터 구동 명령이 내려지는 상태로 250ms 동안 ON 상태가 유지됩니다.

※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.

11.3.2 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.

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

- 0x0001 - Overcurrent
- 0x0002 - Overvoltage
- 0x0004 - Undervoltage
- 0x0008 - Overheat
- 0x0010 - Short Circuit
- 0x0020 - Stall Detection
- 0x0040 - Velocity Error Detection
- 0x0080 - Position Error Detection

0x0001 - Overcurrent:
모터에 흐르는 전류가 **overcurrent_limit**에서 설정한 값 이상으로 흘렀습니다. 과전류(Overcurrent)는 다음 중 하나가 원인이 될 수 있습니다:

- 전압 명령으로 모터를 직접 구동할 때(위치/속도/전류 제어기가 동작하지 않는 상황)
- 폐루프 전류 제어기의 이득을 잘못 조정한 상태에서 위치, 속도, 전류 명령을 실행할 때
- 회전 관성이 큰 부하가 연결된 모터를 빠르게 감속하는 경우

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

- 제어기 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:

- 제어기 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. **profile_mode**를 켜고 **acceleration**을 낮게 설정하여 모터가 급격히 가속하지 않도록 해야 합니다.

0x0008 - Overheat:
MOSFET와 방열판의 온도가 **overheat_limit**에서 설정한 값 이상으로 올라갔습니다.

0x0010 - Short Circuit:
모터의 +, - 선이 서로 단락 되었거나 전원과 단락 되었습니다. 또는 MOSFET 스위칭 소자의 파괴로 인해 H-bridge 회로가 잘못된 조합으로 켜진 상태가 되었습니다.

0x0020 - Stall Detection:
모터에 전력이 공급되는 상황에서 모터의 회전이 검출되지 않는 경우 스톨 상황으로 판단합니다. **stall_detection** 파라미터를 참조하십시오.

0x0040 - Velocity Error Detection:
폐루프 속도제어 모드에서 속도 오차가 커지면 속도제어 에러 상황으로 판단합니다. **vel_error_detection** 파라미터를 참조하십시오.

0x0080 - Position Error Detection:
위치제어 모드에서 위치 오차가 지정된 값 이상이면 위치제어 에러 상황으로 판단합니다. **pos_error_detection** 파라미터를 참조하십시오.

11.3.3 temperature –Temperature

temperature 상태는 MOSFET와 방열판의 온도를 나타냅니다.

이 값이 **high_temperature** 파라미터로 설정한 값 이상으로 올라가면 냉각 팬을 켜 온도 상승을 억제할 수 있습니다. 이 기능을 사용하려면, 제어기의 디지털 출력 포트 중 적어도 하나에 'High Temperature (Cooling Fan ON)' 기능이 선택되어 있어야 합니다.

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

※ MOSFET의 내부 온도는 온도 센서에의 해 측정되는 온도보다 훨씬 높을 수 있습니다.

11.3.4 voltage - Voltage

voltage 상태는 모터에 공급되는 전압을 나타냅니다.

이 값이 **high_voltage** 파라미터로 설정한 값 이상으로 올라가면 제동저항(Brake Resistor)을 켜 전압 상승을 억제할 수 있습니다. 이 기능을 사용하려면, 제어기의 디지털 출력 포트 중 적어도 하나에 ''High Voltage (Brake Resistor ON)' 기능이 선택되어 있어야 합니다.

또한 이 값이 **undervoltage_limit**에서 설정한 값 이하로 내려가면 '0x0004 - Undervoltage' 폴트를 발생하고 **overvoltage_limit**에서 설정한 값 이상으로 올라가면 '0x0002 - Overvoltage' 폴트를 발생합니다. 그리고 모터는 Power OFF 상태가 됩니다.

11.3.5 current - Current

current 상태는 모터에 흐르는 전류를 나타냅니다. **battery_current** 오브젝트와 구분하기 바랍니다.

이 값이 **overcurrent_delay** 시간 동안 **overcurrent_limit**에서 설정한 값 이상으로 올라가면 '0x0001 - Overcurrent' 폴트를 발생하고 모터는 Power OFF 상태가 됩니다.

또한 이 값이 **peakcurrent_ratio** x **overcurrent_limit** 이상으로 한 번이라도 올라가면 '0x0001 - Overcurrent' 폴트를 발생하고 모터는 Power OFF 상태가 됩니다.

11.3.6 velocity - Velocity

velocity 상태는 단위 시간당 모터의 위치 변화량(회전속도)을 나타냅니다. 회전속도를 계산하려면 제어기에 엔코더나 홀센서가 연결되어 있어야 합니다. 또한 회전속도는 **ai_potentiometer**나 **ai_tachometer** 로부터 측정되거나 계산될 수 있습니다.

11.3.7 position - Position

position 변수는 모터의 위치를 나타냅니다. 위치를 카운트하려면 제어기에 엔코더나 홀센서가 연

결되어 있어야 합니다. 또한 위치는 **ai_potentiometer**나 **ai_tachometer** 로부터 측정되거나 계산될 수 있습니다.

feedback_sensor가 '1 - Encoder'로 설정된 경우 **position** 변수는 엔코더 카운터를 읽은 값이 됩니다. '2 - Hall Sensors'로 설정된 경우 홀센서 카운터를 읽은 값이 됩니다.

이 절의 다른 상태 오브젝트와 달리 **position**은 쓰기 가능합니다. 특정한 값을 쓰면, 이 값은 바로 모터제어기에 적용됩니다. 이 때, 만일 **feedback_sensor**가 '2 - Hall Sensors'로 설정되어 홀센서 카운터 값이 위치에 연결되어 있더라도 **hall_count** 값은 영향을 받지 않습니다.

※주의※ 모터가 구동중인 상황에서 위치를 바꾸는 것은 위험한 상황을 초래할 수 있습니다. 값을 바꾸기 전에 반드시 모터는 Power OFF 상태에 있어야 합니다.

11.3.8 hall_count - Hall Count

hall_count 상태는 BLDC 모터에서만 사용됩니다. **feedback_sensor**가 '2 - Hall Sensors'로 설정된 경우, 이 상태는 **position** 변수에 업데이트 됩니다. 그리고 이 값은 엔코더를 대신하여 폐루프 위치 제어기와 속도 제어기의 피드백으로 사용됩니다.

11.3.9 ai_potentiometer - AI Potentiometer

ai_potentiometer 상태는 모터제어기의 위치 피드백 센서 값을 가집니다. 제어기의 아날로그 입력 포트나 펄스 입력 포트 중 하나가 여기로 매핑될 수 있습니다('Motor feedback: Position' 기능 매핑).

feedback_sensor가 '3 - Potentiometer'로 설정된 경우, 이 상태는 **position** 변수에 업데이트 됩니다. 그리고 이 값은 엔코더를 대신하여 폐루프 위치 제어기의 피드백으로 사용됩니다.

11.3.10 ai_tachometer - AI Tachometer

ai_tachometer 상태는 모터제어기의 속도 피드백 센서 값을 가집니다. 제어기의 아날로그 입력 포트나 펄스 입력 포트 중 하나가 여기로 매핑될 수 있습니다('Motor feedback: Velocity' 기능 매핑).

feedback_sensor가 '4 - Tachometer로 설정된 경우, 이 상태는 **velocity** 변수에 업데이트 됩니다. 그리고 이 값은 엔코더를 대신하여 폐루프 속도 제어기의 피드백으로 사용됩니다.

11.4 위치 센서 설정

모터의 회전 위치와 속도를 검출하는데 사용하는 위치센서(Encoder, Hall Sensors)에 대한 설정입니다.

표 11-4 위치 센서 설정 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
min_position, np	141/ 0~2	I32 (CP)	모터가 이동 가능한 최소 위치 (단위: pulse, 범위: 0 ~ 음수)
max_position, xp	142/ 0~2	I32 (CP)	모터가 이동 가능한 최대 위치 (단위: pulse, 범위: 0 ~ 양수)
home_position, hp	143/ 0~2	I32 (CP)	홈센서 감지 시 position에 로드 되는 위치 값 설정 (단위: pulse)
encoder_ppr, ep	144/ 0~2	I16 (CP)	모터 1회전당 엔코더 펄스 수 (단위: pulse/rev)
num_pole_pairs, npp	145/ 0~2	I16 (CP)	BLDC 모터에서 홀센서의 폴페어(Pole Pairs) 수 (단위: pulse/rev)
use_soft_limit, usl	146/ 0~2	I8 (CP)	소프트 리미트 스위치 사용 여부 (기본값: 0)

11.4.1 min_position, max_position - Min, Max Position

min_position과 **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_limit이 켜진 경우, **position_command**와 **home_position** 오브젝트는 최소 위치와 최대 위치 범위 내로 조정됩니다.

또한 **min_position**과 **max_position**은 **encoder_ppr**과 함께 사용되어 정규화된 아날로그 입력이나 펄스 입력을 모터제어기의 위치 명령이나 위치 피드백으로 스케일을 변환합니다.

11.4.2 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. 전압 명령으로 모터를 천천히 움직입니다.
3. 홈 센서가 트리거 되면 **position**에 **home_postion** 값이 로드 됩니다.
4. 0V 전압 명령으로 모터를 정지합니다.

11.4.3 encoder_ppr - Encoder PPR

encoder_ppr 파라미터는 모터가 1회전할 때 카운트 되는 엔코더 펄스 수입니다. 모터제어기는 엔코더의 A/B상 펄스 수를 4체배 하여 카운트 합니다.

모터의 엔코더 펄스 수를 모르는 경우는, 모터를 Power OFF 하고 손으로 모터 축을 한 바퀴 돌렸을 때 카운트 되는 펄스 수를 사용하면 됩니다. 감속기가 적용된 모터에서는 감속비를 고려해야 합니다.

※ 이 파라미터는 모터의 회전속도 계산에 사용됩니다. 그리고 min_position, max_position과 함께 정규화된 아날로그 입력이나 펄스 입력을 모터제어기의 위치 명령이나 위치 피드백으로 스케일을 변환하는데도 사용되기 때문에, 모터제어기에 엔코더가 연결되지 않은 경우에도 필히 설정되어야 하는 값입니다.

11.4.4 num_pole_pairs - Number of Pole Pairs

num_pole_pairs 파라미터는 BLDC모터에서 홀센서(Hall Sensors)의 폴페어(Pole Pairs) 수입니다. 모터가 1회전 하였을 때 발생하는 펄스 수는 6 x **num_pole_pairs**가 됩니다. 이 파라미터는 BLDC 모터의 속도 계산에 사용될 수 있습니다. BLDC 모터 제어기에서 엔코더를 사용하지 않고 홀센서 만으로 위치 제어와 속도 제어를 수행하는 경우 이 값은 올바르게 설정되어야 합니다. 그렇지 않은 경우 모터의 속도를 정확히 계산할 수 없습니다. 만일 엔코더를 사용하는 BLDC 모터라면 이 값은 제어기의 동작에 영향을 미치지 않습니다.

11.4.5 use_soft_limit - Use Soft Limit

use_soft_limit는 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 벗어나

면 리미트 스위치가 동작한 것과 같은 효과를 내도록 합니다.

11.5 모터 속성 설정

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

표 11-5 모터 속성 설정 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
max_current, xc	151/ 0~2	F32 (CP)	모터에 흐르는 최대 연속 전류 (단위: A)
max_voltage, xvt	152/ 0~2	F32 (CP)	모터에 가해지는 최대 전압 (단위: V)
max_velocity, xv	153/ 0~2	F32 (CP)	모터의 최대 회전 속도 (단위: RPM)
acceleration, ac	154/ 0~2	F32 (CP)	모터의 회전 가속도 (단위: RPM/s)
deceleration, dc	155/0~2	F32 (CP)	모터의 회전 감속도 (단위: RPM/s)

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

11.5.1 max_current - Max Current

max_current 파라미터에는 모터에 연속해서 흘릴 수 있는 최대 전류를 설정합니다. 이 값은 **overcurrent_limit** 보다 낮게 설정되어야 합니다.

위치제어나 속도제어 모드에서 폐루프 전류제어기는 모터에 흐르는 전류가 **max_current**를 넘지 않는 범위 내에서 제어합니다.

11.5.2 max_voltage - Max Voltage

max_voltage 파라미터에는 모터에 가해질 수 있는 최대 전압을 설정합니다. 이 값은 **overvoltage_limit** 보다 낮게 설정되어야 합니다.

위치제어나 속도제어, 전류제어, 전압 출력 모드에서 모터에 가해지는 전압은 **max_voltage**를 넘지 않도록 제한됩니다.

11.5.3 max_velocity - Max Velocity

max_velocity 파라미터는 모터의 최대 회전속도를 설정합니다.

위치제어 모드에서 폐루프 속도제어기는 모터의 속도가 **max_velocity**를 넘지 않는 범위 내에서 제어합니다.

11.5.4 acceleration, deceleration - Acceleration, Deceleration

acceleration과 **deceleration** 파라미터에는 각각 모터의 회전 가속도와 감속도를 설정합니다.

profile_mode가 1로 설정되어 있는 경우, 가속도와 감속도는 위치제어, 속도제어, 전압출력 모드에서 속도 프로파일을 만드는데 사용됩니다.

11.6 모터 구동 한계 설정

이 절에서는 모터의 구동 한계를 설정하는 오브젝트들에 대해 다룹니다.

표 11-6 모터 구동 한계 설정 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
overheat_limit, ohl	161/0~2	F32 (CP)	FET와 방열판의 과열 한계 (단위: °C)
overcurrent_limit, ocl	162/0~2	F32 (CP)	모터의 과전류 한계 (단위: A)
overcurrent_delay, ocd	163/0~2	I16 (CP)	과전류 허용 지연 시간 (단위: ms)
peakcurrent_ratio, pcr	164/0~2	F32 (CP)	모터의 순간 피크 전류 (단위: %)
overvoltage_limit, ovl	165/0~2	F32 (CP)	전원단의 과전압 한계 (단위: V)
undervoltage_limit, uvl	166/0~2	F32 (CP)	전원단의 저전압 한계 (단위: V)

모터의 상태(전압, 전류, 열 등)가 한계를 벗어나는 경우에 폴트(Fault)를 발생하고 모터는 Power

OFF 상태가 됩니다.

11.6.1 overheat_limit - Overheat Limit

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

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

이 파라미터는 MOSFET와 방열판의 열 전달을 고려하여 값을 설정해야 합니다. MOSFET의 온도는 온도 센서에의 해 측정되는 온도보다 훨씬 높을 수 있습니다.

11.6.2 overcurrent_delay, overcurrent_limit - Overcurrent Limit, Overcurrent Delay

overcurrent_delay와 **overcurrent_limit** 파라미터는 모터의 과전류(Overcurrent) 상황을 감지하기 위한 최대 한계전류를 설정합니다.

모터에 흐르는 전류(**current**)가 **overcurrent_delay** 동안 **overcurrent_limit** 값을 넘어갈 경우, '0x0001 - Overcurrent' 폴트를 발생하고 모터는 Power OFF 상태가 됩니다.

overcurrent_limit 값은 **max_current**보다 120% 이상 높게 설정되어야 합니다. 모터의 전류 측정 값에는 정류자에서 발생하는 아크 및 MOSFET 스위칭시 발생하는 전압 스파이크에 의해 노이즈가 많이 포함되어 있습니다. 만일 **overcurrent_limit**과 **max_current** 간에 값의 차이가 미소하다면 노이즈로 인해 폴트가 발생할 가능성이 높습니다.

11.6.3 peakcurrent_ratio - Peak Current Ratio

peakcurrent_ratio 파라미터는 모터의 과전류(Overcurrent) 상황을 감지하기 위한 피크 전류를 설정합니다.

모터에 흐르는 전류(**current**)가 한 순간이라도 **peakcurrent_ratio** x **overcurrent_limit** 값을 넘어갈 경우, '0x0001 - Overcurrent' 폴트를 발생하고 모터는 Power OFF 상태가 됩니다.

11.6.4 overvoltage_limit - Overvoltage Limit

overvoltage_limit 파라미터는 모터의 과전압(Overvoltage) 상황을 감지하기 위한 최대 한계 전압

을 설정합니다.

전원에서 공급되는 전압이 **overvoltage_limit** 값을 넘어갈 경우, '0x0002 - Overvoltage' 폴트를 발생하고 모터는 Power OFF 상태가 됩니다.

11.6.5 undervoltage_limit - Undervoltage Limit

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

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

11.7 모터 구동오류 감지조건 설정

이 절에서는 모터에 구동 명령이 인가되었는데 모터가 정지해 있거나 명령을 추종하지 못하는 상황을 감지하기 위한 오브젝트들에 대해 설명합니다.

표 11-7 모터 구동오류 감지조건 설정 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
stall_detection, sd	167/0~2	I8 (CP)	모터의 구동 명령에 대해 움직이지 않는 상황 감지 (기본값: 0)
vel_error_detection, ved	168/0~2	I8 (CP)	폐루프 속도제어기에서 명령과 피드백 사이의 오차에 의한 이상 감지 (기본값: 0)
pos_error_detection, ped	169/0~2	I8 (CP)	폐루프 위치제어기에서 명령과 피드백 사이의 오차에 의한 이상 감지 (기본값: 0)

11.7.1 stall_detection - Stall Detection

stall_detection 파라미터는 모터의 구동 명령에 대해 움직이지 않는 상황을 감지하기 위한 조건을 설정합니다.

지정된 시간 동안 모터에 지정된 전압(PWM 듀티비로 전압 생성) 이상이 가해지는데도 모터가 회전하지 않으면 스톨(stall) 상황으로 감지합니다. 그리고 '0x0020 - Stall Detection' 폴트를 발생하며 모터를 Power OFF 합니다.

이 파라미터의 값으로 다음 중 하나를 선택합니다:

- 0 - Not used (기본 값)
- 1 - 100ms at 10% PWM duty ratio
- 2 - 200ms at 20% PWM duty ratio
- 3 - 400ms at 30% PWM duty ratio
- 4 - 700ms at 40% PWM duty ratio
- 5 - 1s at 50% PWM duty ratio

스톨 감지 기능은 모든 구동모드(폐루프 위치제어, 속도제어, 전류제어 모드 및 개루프 전압 출력 모드)에서 활성화 됩니다. 또한 엔코더나 홀센서와 같은 위치를 읽기 위한 센서가 제어기에 연결되어야 합니다.

11.7.2 vel_error_detection - Velocity Error Detection

vel_error_detection 파라미터는 폐루프 속도제어기에서 명령과 피드백 사이의 오차에 의한 이상을 감지하기 위한 조건을 설정합니다.

폐루프 속도제어기에서 지정된 시간 동안 명령 속도와 피드백 속도의 오차가 지정된 속도보다 커지면 속도에러 상황으로 감지합니다. 그리고 '0x0040 - Velocity Error Detection' 폴트를 발생하고 모터를 Power OFF 합니다.

이 파라미터의 값으로 다음 중 하나를 선택합니다:

- 0 - Not used (기본값)
- 1 - 100ms and error > 100 RPM
- 2 - 200ms and error > 200 RPM
- 3 - 400ms and error > 500 RPM
- 4 - 700ms and error > 1500 RPM
- 5 - 1s and error > 3000 RPM

속도에러 감지 기능은 폐루프 위치제어와 속도제어 모드에서 활성화 됩니다. 또한 엔코더나 홀센서와 같은 위치를 읽기 위한 센서가 제어기에 연결되어야 합니다.

※ 만일 속도 명령이 드문드문 불연속적으로 내려지는 경우에는 Velocity Error Detection 기능을 사용하지 않는 것이 좋습니다. 명령이 내려질 때 큰 속도오차가 발생하여 폴트가 발생하게 됩니다.

11.7.3 pos_error_detection - Position Error Detection

pos_error_detection 파라미터는 폐루프 위치제어기에서 명령과 피드백 사이의 오차에 의한 이상을 감지하기 위한 조건을 설정합니다.

폐루프 위치제어기에서 지정된 시간 동안 명령 위치와 피드백 위치의 오차가 지정된 거리보다 커지면 위치에러 상황으로 감지합니다. 그리고 '0x0080 - Position Error Detection' 폴트를 발생하고 모터를 Power OFF 합니다.

이 파라미터의 값으로 다음 중 하나를 선택합니다:

- 0 - Not used (기본값)
- 1 - 100ms and error > 100 pulse
- 2 - 200ms and error > 500 pulse
- 3 - 400ms and error > 2000 pulse
- 4 - 700ms and error > 5000 pulse
- 5 - 1s and error > 20000 pulse

위치에러 감지 기능은 폐루프 위치제어 모드에서 활성화 됩니다. 또한 엔코더나 홀센서와 같은 위치를 읽기 위한 센서가 제어기에 연결되어야 합니다.

※ 만일 위치 명령이 드문드문 불연속적으로 내려지는 경우에는 Position Error Detection 기능을 사용하지 않는 것이 좋습니다. 명령이 내려질 때 큰 위치오차가 발생하여 폴트가 발생하게 됩니다.

11.8 모터 및 I/O 구동관련 설정

이 절에서는 모터 및 I/O의 구동에 관련된 오브젝트들에 대해 설명합니다.

표 11-8 모터 및 I/O 구동 관련 설정

Long name, Short name	Index/ Sub-index	Type	Description
startup_power_on, spo	173/ 0~2	I8 (CP)	제어기가 시작될 때 모터 Power ON/OFF 설정 (기본값: 1)
direction, dir	174/ 0~2	I8 (CP)	모터의 정방향/역방향 회전 설정 (기본값: 0)
brake_on_delay, bod	175/ 0~2	I16 (CP)	브레이크 작동 시 지연시간 설정 (단위: ms, 기본값: 0, 범위: 0 ~ 30000)
high_voltage, hv	176/ 0~2	I16 (CP)	제동저항(Brake Resistor)을 켜는 전압 설정 (단위: V)
high_temperature, ht	177/ 0~2	I16 (CP)	냉각 팬을 켜는 온도 설정 (단위: ℃)

11.8.1 startup_power_on - Startup Power ON

startup_power_on 파라미터는 제어기가 시작될 때 모터의 Power ON/OFF를 결정합니다. 이 파라미터의 값으로 다음 중 하나를 선택합니다:

- 0 - Disable
- 1 - at Start-up Controller (기본값)
- 2 - when Motor Power OFFed

'0 – Disable'로 설정된 경우, 제어기가 시작될 때 모터는 Power OFF 상태가 됩니다.

'1 – at Start-up Controller'로 설정된 경우, 제어기가 시작될 때에만 모터 Power ON 상태로 전환을 시도합니다. Power ON에 관련된 모든 조건이 만족되면 모터는 Power ON 상태가 되고 그렇지 않으면 Power OFF 상태에 머무르게 됩니다.

'2 – when Motor Power OFFed'로 설정된 경우, 모터가 Power OFF 상태가 될 때마다 Power ON 상태로 전환을 시도합니다. 이러한 시도는 위험한 상황을 초래할 수 있기 때문에 특별한 경우를 제외하고는 되도록 사용하지 않는 것이 좋습니다.

11.8.2 direction - Direction

direction은 모터의 회전 방향과 엔코더의 카운트 방향을 결정합니다. 이 파라미터의 값으로 다음 중 하나를 선택합니다:

- 0 - Not Change (기본값)
- 1 - Reverse Direction

듀얼 채널 제어기를 이동로봇에 사용하는 경우, 이 설정은 효과적입니다. 이동로봇의 좌우 바퀴에 연결되는 두 모터는 서로 등지도록 장착되기 때문에, 같은 속도 명령에 대해 두 모터의 회전 방향은 반대가 됩니다. 이럴 경우 한 쪽 모터의 **direction**을 1로 설정하여 회전 방향을 반대로 설정합니다.

※주의※ direction의 값이 1로 설정된 경우 모터에 가해지는 전압의 부호를 바꿉니다. 그리고 엔코더도 반대로 카운트(증가 카운트인 경우 감소 카운트로, 감소 카운트인 경우 증가 카운트로 바꿈) 합니다. 모터에 양의 전압을 가했을 때 엔코더가 감소 카운트 되는 경우에 혹은 이와 반대의 경우에, 이를 바로잡기 위해 direction 설정을 사용하면 안됩니다.

11.8.3 brake_on_delay - Brake ON Delay

brake_on_delay 파라미터의 변경은 모터에 전원이 차단된 시점부터 브레이크를 활성화하기까지의 지연 시간을 설정합니다.

이 기능을 사용하려면, 제어기의 디지털 출력 포트 중 적어도 하나에 'Motor Power ON (Brake Release)' 기능이 선택되어 있어야 합니다.

11.8.4 high_voltage - High Voltage

high_voltage 파라미터의 변경은 모터의 역기전력으로 인한 파워 소스 및 제어기의 파손을 방지하기 위한 허용 전압을 설정합니다. 제어기의 전원단 전압이 상승하여 이 파라미터로 설정한 값 이상이 되면 제동저항(Brake Resistor)을 켜 전압의 상승을 억제합니다.

이 기능을 사용하려면, 제어기의 디지털 출력 포트 중 적어도 하나에 'High Voltage (Brake Resistor ON)' 기능이 선택되어 있어야 합니다. 그리고 역기전력을 열로 소모하기에 충분한 용량의 션트 저항과 관련 회로가 필요합니다.

11.8.5 high_temperature - High Temperature

high_temperature 파라미터의 변경은 제어기 온도가 너무 높아 고장이 나거나 오작동 하는 것을 방지하기 위한 허용 온도를 설정합니다. 제어기의 방열판 온도가 상승하여 이 파라미터로 설정한 값 이상이 되면 냉각 팬을 켜 온도 상승을 억제합니다.

이 기능을 사용하려면, 제어기의 디지털 출력 포트 중 적어도 하나에 'High Temperature (Cooling Fan ON)' 기능이 선택되어 있어야 합니다. 그리고 제어기의 열을 식히기에 충분한 냉각 팬과 관련 회로가 필요합니다.

11.9 페루프 제어 설정

이 절에서는 페루프 제어와 관련된 구성 파라미터들에 대해 설명합니다.

표 11-9 페루프 제어 설정 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
feedback_sensor, fs	171/ 0~2	I8 (CP)	제어기 피드벡 센서 선택 (기본값: 1)
profile_mode, pm	172/ 0~2	I8 (CP)	속도 명령에 사다리꼴 프로파일 적용여부 (기본값: 1)

11.9.1 feedback_sensor - Feedback Sensor

feedback_sensor 파라미터의 변경은 폐루프 위치제어기와 속도제어기의 피드백 값을 읽을 센서를 결정합니다. 이 파라미터의 값으로 다음 중 하나를 선택합니다:

- 0 - None
- 1 - Encoder (기본값),
- 2 - Hall Sensors
- 3 - Potentiometer
- 4 - Tachometer

'3 – Potentiometer'로 설정된 경우, **ai_potentiometer** 값은 위치제어기의 피드백으로 사용됩니다. 이때 입력 범위 -1과 1 사이 값은 **min_position**과 **max_position** 사이 값으로 스케일 조정됩니다.

'4 – Tachometer'로 설정된 경우, **ai_tachometer** 값은 속도제어기의 피드백으로 사용됩니다. 이때 입력 범위 -1과 1 사이 값은 -**max_velocity**와 **max_velocity** 사이 값으로 스케일 조정됩니다.

11.9.2 profile_mode - Profile Mode

프로파일 모드의 사용 목적은 기본적으로 모터나 모터가 연결된 시스템에 급격한 속도 변화로 인한 충격을 방지하는데 있습니다. 프로파일 생성기는 속도 명령이 급격하게 변하더라도 일정한 가속도와 감속도가 적용된 속도 프로파일을 생성하게 됩니다.

profile_mode는 위치제어와 속도제어 모드에서 사다리꼴 속도 프로파일의 사용 여부를 결정 합니다. 이 파라미터의 값으로 다음 중 하나를 선택합니다:

- 0 - None
- 1 - Trapezoidal Velocity Profile Mode (기본값)

profile_mode가 1로 설정된 경우, 사다리꼴 속도 프로파일의 가속과 감속 구간의 기울기는 **acceleration**과 **deceleration**에 의해 결정되고 최고 속도는 **max_velocity**에 의해 결정됩니다.

11.10 위치 제어기 이득 설정

폐루프 위치제어기는 기본적으로 PID 제어기로 구성됩니다. 이 절은 PID 제어기의 이득에 대해 설명합니다.

표 11-10 위치 제어기 이득 설정 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
pc_kp, pp	191/ 0~2	F32 (CP)	PID 위치제어기의 비례제어 이득

pc_ki, pi	192/ 0~2	F32 (CP)	PID 위치제어기의 적분제어 이득
pc_kd, pd	193/ 0~2	F32 (CP)	PID 위치제어기의 미분제어 이득

11.10.1 pc_kp, pc_ki, pc_kd - Kp, Ki, Kd

폐루프 위치제어기에 사용되는 PID 제어기의 비례제어 이득 **pc_kp**와 적분제어 이득 **pc_ki**, 미분제어 이득 **pc_kd**을 설정합니다.

폐루프 위치제어기의 이득 설정에 대한 자세한 내용은 "7.4.3 위치 제어기 PID 이득 조정"을 참고하기 바랍니다.

※ 각각의 이득은 제어기 동작 중에도 변경이 가능합니다. 하지만 제어기 동작 중 이득을 변경하는 것은 제어기의 출력이 갑자기 변하게 되어 위험한 상황을 초래할 수 있습니다.

11.11 속도 제어기 이득 설정

폐루프 속도제어기는 기본적으로 PI 제어기로 구성됩니다. 이 절은 PI 제어기의 이득에 대해 설명합니다.

표 11-11 속도 제어기 이득 설정 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
vc_kp, vp	186/ 0~2	F32 (CP)	PI 속도제어기의 비례제어 이득
vc_ki, vi	187/ 0~2	F32 (CP)	PI 속도제어기의 적분제어 이득
vc_ks, vff	190/ 0~2	F32 (CP)	속도 레퍼런스에 대한 스케일 펙터

11.11.1 vc_kp, vc_ki - Kp, Ki

폐루프 속도 제어기에 사용되는 PI 제어기의 비례제어 이득 **vc_kp**과 적분제어 이득 **vc_ki**을 설정합니다.

폐루프 속도제어기의 이득 설정에 대한 자세한 내용은 "7.4.2 속도 제어기 PI 이득 조정"을 참고하기 바랍니다.

※ 각각의 이득은 제어기 동작 중에도 변경이 가능합니다. 하지만 제어기 동작 중 이득을 변경하는 것은 제어기의 출력이 갑자기 변하게 되어 위험한 상황을 초래할 수 있습니다.

11.11.2 vc_ks - Ks

vc_ks 파라미터의 변경은 폐루프 속도 제어기에서 속도 레퍼런스에 대한 스케일팩터를 설정합니다.

만약 속도 제어기의 비례제어 이득과 적분제어 이득이 0이고 스케일팩터가 어떠한 값을 가진다면, 속도 제어기 내부의 PI 제어기는 동작하지 않고 속도 레퍼런스에 이득 **vc_ks**만 곱해져 출력으로 내보내는 개루프 속도제어기로 동작합니다.

개루프 속도제어기는 엔코더의 해상도가 너무 낮거나 포텐셔미터와 같이 속도를 측정하기 어려운 센서가 위치제어 피드백으로 사용될 때 속도제어기를 바이패스 하기 위해 사용합니다.

11.12 전류 제어기 이득 설정

폐루프 전류제어기는 기본적으로 PI 제어기로 구성됩니다. 이 절은 PI 제어기의 이득에 대해 설명합니다.

표 11-12 전류 제어기 이득 설정

Long name, Short name	Index/ Sub-index	Type	Description
cc_kp, cp	181/ 0~2	F32 (CP)	PI 전류제어기의 비례제어 이득
cc_ki, ci	182/ 0~2	F32 (CP)	PI 전류제어기의 적분제어 이득
cc_kff, cff	185/ 0~2	F32 (CP)	PI 전류제어기의 모터 회전속도 전향 보상 이득

11.12.1 cc_kp, cc_ki - Kp, Ki

폐루프 전류제어기에 사용되는 PI 제어기의 비례제어 이득 **cc_kp**와 적분제어 이득 **cc_ki**를 설정합니다.

폐루프 전류제어기의 이득 설정에 대한 자세한 내용은 "7.4.1 전류 제어기 PI 이득 조정"을 참고하기 바랍니다.

※ 각각의 이득은 제어기 동작 중에도 변경이 가능합니다. 하지만 제어기 동작 중 이득을 변경하는 것은 제어기의 출력이 갑자기 변하게 되어 위험한 상황을 초래할 수 있습니다.

11.12.2 cc_kff - Kff

cc_kff 파라미터의 변경은 모터의 회전속도로부터 역기전력을 전향 보상하기 위한 이득을 설정합니다.

모터의 역기전력은 모터의 회전속도에 역기전력 상수를 곱해 계산 가능합니다. 그러므로 **cc_kff**는 역기전력 상수와 유사한 값을 가질 것입니다.

※ 역기전력 상수로부터 cc_kff를 계산할 때 단위 변환에 주의해야 합니다.

12 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.

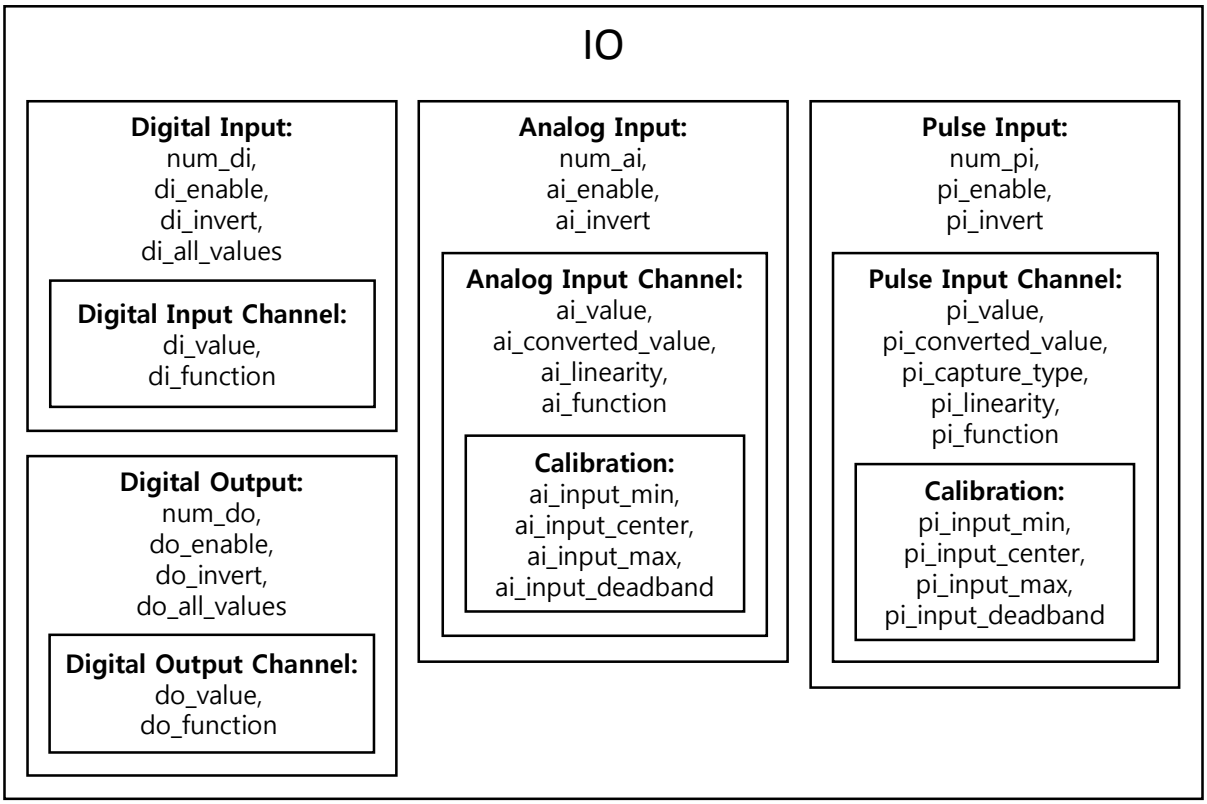

그림 12-1 I/O Objects

제어기 모델에 따라 I/O 채널의 수는 달라집니다. 모델별 I/O채널의 수는 데이터시트를 확인하도록 합니다.

그리고 I/O의 각 채널을 구분하기 위해 Sub-index를 사용합니다.

12.1 디지털 입력

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

표 12-1 디지털 입력 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
num_di, ndi	60/0	I8 (CN)	디지털 입력 채널의 수
di_enable, die	61/0	I32 (CP)	각 비트별 해당 디지털 입력 채널의 사용 여부
di_invert, dii	62/0	I32 (CP)	각 비트별 해당 디지털 입력 채널의 반전 여부
di_all_values, di	63/0	I32 (ST)	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

Type 열의 괄호 안 표기는 다음과 같습니다:

- (CN) - Constant 오브젝트
- (CM) - Command 오브젝트
- (ST) - Status 오브젝트
- (CP) - Configuration Parameter 오브젝트
- (VA) - Variable 오브젝트

12.1.1 num_di - Number of DI

num_di 상수는 제어기가 보유한 디지털 입력 채널의 수를 나타냅니다. 디지털 입력 채널의 수는 제어기 모델에 따라 다릅니다. 해당 제품의 데이터시트에서도 참조 가능합니다.

12.1.2 di_enable - DI Enable

di_enable 파라미터의 각 비트는 해당 디지털 입력 채널의 사용 여부를 결정합니다:

- Bit0 - 디지털 입력 채널 1의 사용 여부
- Bit1 - 디지털 입력 채널 2의 사용 여부
- …

해당 비트가 0인 경우 디지털 입력 채널은 사용 불가능합니다. 1인 경우 디지털 입력 채널은 사용 가능합니다.

12.1.3 di_invert - DI Invert

di_invert 파라미터의 각 비트는 해당 디지털 입력 채널의 반전 여부를 결정합니다:

- Bit0 - 디지털 입력 채널 1의 반전 여부
- Bit1 - 디지털 입력 채널 2의 반전 여부
- …

해당 비트가 0인 경우 디지털 입력 채널의 신호를 그대로 읽습니다. 1인 경우는 디지털 입력 채널의 입력 신호를 반전하여 읽습니다. 즉, 0인 경우 1로, 1인 경우 0으로 읽습니다.

12.1.4 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.

12.2 디지털 입력 채널

다음은 디지털 입력의 각 채널 별로 설정되는 오브젝트입니다.

표 12-2 디지털 입력 채널 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
di_value, div	201/ 1~n	I8 (ST)	디지털 입력 채널의 값 (범위: 0 or 1)

di_function, dif	202/ 1~n	I16 (CP)	디지털 입력 채널을 특정 모터의 동작으로 매핑

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

12.2.1 di_value - DI Value

di_value 상태는 디지털 입력 채널의 값을 나타냅니다. 값으로 1bit의 0이나 1을 가지며, **di_invert** 에서 설정된 반전 여부가 적용된 값입니다.

12.2.2 di_function - DI Function

di_function 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
- 0x0100 - 채널 2번에 연결된 모터 선택

하위 8bit로는 동작을 선택합니다:

- 0 - None
- 1 - Emergency Stop
- 2 - Quick Stop
- 3 - Stop
- 4 - Run Script
- 5 - Forward Limit Switch
- 6 - Reverse Limit Switch
- 7 - Invert Direction
- 8 - Load Home Counter

12.3 디지털 출력

이 절에서는 디지털 출력 채널들에 공통으로 사용되는 오브젝트들에 대해 다룹니다.

표 12-3 디지털 출력 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
num_do, ndo	65/0	I8 (CN)	디지털 출력 채널의 수

do_enable, doe	66/0	I32 (CP)	각 비트별 해당 디지털 출력 채널의 활성화 여부
do_invert, doi	67/0	I32 (CP)	각 비트별 해당 디지털 출력 채널의 반전 여부
do_all_values, do	68/0	I32 (ST)	디지털 출력 채널들을 하나의 32bit 값으로 모음

12.3.1 num_do - Number of DO

num_do 상수는 제어기가 가진 디지털 출력 채널의 수를 나타냅니다. 디지털 출력 채널의 수는 제어기 모델에 따라 다릅니다. 해당 제품의 데이터시트에서도 참조 가능합니다.

12.3.2 do_enable - DO Enable

do_enable 파라미터의 각 비트는 해당 디지털 출력 채널의 사용 여부를 결정합니다:

- Bit0 - 디지털 출력 채널 1의 사용 여부
- Bit1 - 디지털 출력 채널 2의 사용 여부
- …

해당 비트가 0인 경우 디지털 출력 채널은 사용 불가능합니다. 1인 경우 디지털 출력 채널은 사용 가능합니다.

12.3.3 do_invert - DO Invert

do_invert 파라미터의 각 비트는 해당 디지털 출력 채널의 반전 여부를 결정합니다:

- Bit0 - 디지털 출력 채널 1의 반전 여부
- Bit1 - 디지털 출력 채널 2의 반전 여부
- …

해당 비트가 0인 경우 디지털 출력 채널의 신호는 변환 없이 출력됩니다. 1인 경우 디지털 출력 채널의 신호는 반전되어 출력됩니다.

12.3.4 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.

12.4 디지털 출력 채널

다음은 디지털 출력의 각 채널 별로 설정되는 오브젝트입니다.

표 12-4 디지털 출력 채널 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
do_value, dov	211/ 1~m	I32 (VA)	디지털 출력 채널의 값 (범위: 0 or 1)
do_function, dof	212/ 1~m	I32 (CP)	디지털 출력 채널을 특정 모터의 상태로 매핑

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

12.4.1 do_value - DO Value

do_value 변수는 디지털 출력 채널에 값을 읽고 씁니다. 값으로 1bit의 0이나 1을 가지며, **do_invert**에서 설정된 반전 여부가 적용되기 전의 값입니다. 반전 여부는 디지털 출력 포트로 신호가 나갈 때 적용됩니다.

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

12.4.2 do_function - DO Function

do_function 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
- 0x0100 - 채널 2번에 연결된 모터 선택

하위 8bit로는 상태를 선택합니다:

- 0 - None

- 1 - Motor Power ON (Break Release)
- 2 - Motor is Reversed (Warning Buzzer ON)
- 3 - High Voltage (Brake Resistor ON)
- 4 - High Temperature (Cooling Fan ON)

12.5 아날로그 입력

이 절에서는 아날로그 입력 채널에 공통으로 사용되는 오브젝트들에 대해 다룹니다.

표 12-5 아날로그 입력 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
num_ai, nai	70/0	I8 (CN)	아날로그 입력 채널의 수
ai_enable, aie	71/0	I32 (CP)	각 비트별 해당 아날로그 입력 채널의 활성화 여부
ai_invert, aii	72/0	I32 (CP)	각 비트별 해당 아날로그 입력 채널의 극성 반전 여부

12.5.1 num_ai - Number of AI

num_ai 상수는 제거기가 보유한 아날로그 입력 채널의 수를 나타냅니다. 아날로그 입력 채널의 수는 제어기 모델에 따라 다릅니다. 해당 제품의 데이터시트에서도 참조 가능합니다.

12.5.2 ai_enable - AI Enable

ai_enable 파라미터의 각 비트는 해당 아날로그 입력 채널의 사용 여부를 결정합니다:

- Bit0 - 아날로그 입력 채널 1의 사용 여부
- Bit1 - 아날로그 입력 채널 2의 사용 여부
- …

해당 비트가 0인 경우 아날로그 입력 채널은 사용 불가능합니다. 1인 경우 아날로그 입력 채널은 사용 가능합니다.

12.5.3 ai_invert - AI Invert

ai_invert 파라미터의 각 비트는 해당 아날로그 입력 채널의 반전 여부를 결정합니다:

- Bit0 - 아날로그 입력 채널 1의 반전 여부
- Bit1 - 아날로그 입력 채널 2의 반전 여부
- ...

해당 비트가 1인 경우, 아날로그 입력 값이 변환 과정을 거쳐 정규화 된 **ai_converted_value** 값의 극성이 반전됩니다. 즉, -1은 1로 반전되고 1은 -1로 반전됩니다. 0의 값은 그대로 유지됩니다.

12.6 아날로그 입력 채널

다음은 아날로그 입력의 각 채널 별로 설정되는 오브젝트입니다.

표 12-6 아날로그 입력 채널 오브젝트

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

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

12.6.1 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.

12.6.2 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.

12.6.3 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

12.6.4 ai_function - AI Function

ai_function 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
- 0x0100 - 채널 2번에 연결된 모터 선택

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

- 0 - None
- 1 - Command: Voltage
- 2 - Command: Current

- 3 - Command: Velocity
- 4 - Command: Position
- 5 - Feedback: Position
- 6 - Feedback: Velocity
- 7 - DI: Emergency Stop
- 8 - DI: Quick Stop
- 9 - DI: Slowdown Stop
- 10 - DI: Run Script
- 11 - DI: Forward Limit Switch
- 12 - DI: Reverse Limit Switch
- 13 - DI: Invert Direction
- 14 - DI: Load Home Counter

여기서 'Command:'로 표기된 기능은 명령 버퍼로 매핑되는 기능입니다. 'Feedback:'으로 표기된 기능은 피드백 버퍼로 매핑되는 기능입니다. 'DI'로 표기된 기능은 디지털 입력 버퍼로 매핑되는 기능입니다. 디지털 입력 버퍼로 매핑된 경우, 아날로그 입력 값이 0보다 크면 버퍼 값은 1, 0을 포함하고 0보다 작으면 버퍼 값은 0이 됩니다.

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

ai_input_min, ai_input_center, ai_input_max는 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.

12.6.6 ai_input_deadband - AI Input Deadband

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

12.7 펄스 입력

이 절에서는 펄스 입력 채널에 공통으로 사용되는 오브젝트들에 대해 다룹니다.

표 12-7 펄스 입력 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
num_pi, npi	75/0	I8 (CN)	펄스 입력 채널의 수
pi_enable, pie	76/0	I32 (CP)	각 비트별 해당 펄스 입력 채널의 활성화 여부
pi_invert, pii	77/0	I32 (CP)	각 비트별 해당 펄스 입력 채널의 극성 반전 여부

12.7.1 num_pi - Number of PI

num_pi 상수는 펄스 입력 채널의 수를 나타냅니다. 펄스 입력 채널의 수는 제어기 모델에 따라 다릅니다. 해당 제품의 데이터시트에서도 참조 가능합니다.

12.7.2 pi_enable - PI Enable

pi_enable 파라미터의 각 비트는 해당 펄스 입력 채널의 사용 여부를 결정합니다:

- Bit0 - 펄스 입력 채널 1의 사용 여부
- Bit1 - 펄스 입력 채널 2의 사용 여부
- …

해당 비트가 0인 경우 펄스 입력 채널은 사용 불가능합니다. 1인 경우는 펄스 입력 채널은 사용 가능합니다.

12.7.3 pi_invert - PI Invert

pi_invert 파라미터의 각 비트는 해당 펄스 입력 채널의 반전 여부를 결정합니다:

- Bit0 - 펄스 입력 채널 1의 반전 여부
- Bit1 - 펄스 입력 채널 2의 반전 여부
- …

해당 비트가 1인 경우, 펄스 입력 값이 변환 과정을 거쳐 정규화 된 **pi_converted_value** 값의 극

성이 반전됩니다. 즉, -1은 1로 반전되고 1은 -1로 반전됩니다. 0의 값은 그대로 유지됩니다.

12.8 펄스 입력 채널

다음은 펄스 입력의 각 채널 별로 설정되는 오브젝트들입니다.

표 12-8 펄스 입력 채널 오브젝트

Long name, Short name	Index/ Sub-index	Type	Description
pi_value, piv	231/ 1~p	I32 (ST)	펄스 입력 채널의 원시 값(raw value)
pi_converted_value, picv	232/ 1~p	F32 (ST)	변환 과정을 거쳐 정규화 된 값(normalized value) (범위: -1 ~ 1)
pi_capture_type, pit	233/ 1~p	I8 (CP)	펄스 캡쳐의 종류
pi_linearity, pil	234/ 1~p	I8 (CP)	지수/로그 변환 형식 지정
pi_function, pif	235/ 1~p	I16 (CP)	펄스 입력을 특정 모터의 명령이나 피드백과 매핑
pi_input_min, pin	236/ 1~p	I32 (CP)	캘리브레이션: 입력 최소값
pi_input_center, pic	237/ 1~p	I32 (CP)	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.

12.8.1 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.

12.8.2 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.

12.8.3 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.

12.8.4 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

12.8.5 pi_function - PI Function

pi_function 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
- 0x0100 - 채널 2번에 연결된 모터 선택

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

- 0 - None
- 1 - Command: Voltage
- 2 - Command: Current
- 3 - Command: Velocity
- 4 - Command: Position
- 5 - Feedback: Position
- 6 - Feedback: Velocity
- 7 - DI: Emergency Stop
- 8 - DI: Quick Stop
- 9 - DI: Slowdown Stop
- 10 - DI: Run Script
- 11 - DI: Forward Limit Switch
- 12 - DI: Reverse Limit Switch
- 13 - DI: Invert Direction
- 14 - DI: Load Home Counter

여기서 'Command:'로 표기된 기능은 명령 버퍼로 매핑되는 기능입니다. 'Feedback:'으로 표기된 기능은 피드백 버퍼로 매핑되는 기능입니다. 'DI'로 표기된 기능은 디지털 입력 버퍼로 매핑되는 기능입니다. 디지털 입력 버퍼로 매핑된 경우, 아날로그 입력 값이 0보다 크면 버퍼 값은 1, 0을 포함하고 0보다 작으면 버퍼 값은 0이 됩니다.

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

pi_input_min, pi_input_center, pi_input_max는 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.

12.8.7 pi_input_deadband - PI Input Deadband

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

13 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.

통신 프로토콜은 크게 CAN 포트에서 사용되는 프로토콜과 시리얼(USB, RS-232) 포트에서 사용되는 프로토콜로 구분됩니다. 시리얼 포트 프로토콜은 다시 패킷의 구조에 따라 **바이너리(binary)** 형태와 **텍스트(text)** 형태로 구분됩니다.

13.1 용어의 정의

제어기의 객체를 통신 프로토콜로 읽고 쓰는데 사용되는 용어를 표 13-1에서 정의합니다.

표 13-1 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 제어기 내부의 오브젝트를 참조하는 색인으로, CAN 및 시리얼 바이너리 패킷에서 사용됩니다.

Sub-index는 오브젝트를 참조하는 부-색인으로, 통상 모터의 채널 번호를 의미합니다. 채널 1에 연결된 모터에 대한 명령이나 설정이라면 Sub-index로 1을, 채널 2에 연결된 모터의 경우는 2를 사용하면 됩니다. 연결된 모터가 아니라 제어기 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 제어기 내부에서 Index 값으로 변환되어 Index와 동일하게 사용됩니다.

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

표 13-1에서는 오브젝트의 4가지 속성을 보여줍니다. 각각의 오브젝트 타입에 따라 패킷의 길이가 달라질 수 있습니다.

표 13-2 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 13-2에서는 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 13-3 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)

13.2 CAN 메시지

제어기의 구성 파라미터(Configuration Parameter)를 설정하고 명령(Command)을 내리거나 상태(Status)를 읽기 위해서 제어기에 존재하는 여러 오브젝트들을 CAN 통신으로 엑세스 할 수 있습니다. 이 절은 제어기에 존재하는 오브젝트들을 엑세스하기 위한 CAN 통신의 메시지 구조에 대해 기술합니다.

※ 마스터 PC와 제어기가 CAN 버스에 연결되어 통신하기 위해서는, 네트워크에 연결된 모든 노드의 통신 속도가 일치해야 하고 각각의 장치 ID는 모두 달라야 합니다.

13.2.1 CAN 패킷의 기본 구조

마스터 PC와 제어기 간에 CAN 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 CAN 메시지를 보내거나 장치가 마스터로 회신 할 때, Node ID에는 장치 ID가 동일하게 지정되어야 합니다. Length는 메시지(Message)의 길이를 나타내는데, 메시지의 최대 크기가 8byte 이므로 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 13-4과 같은 액세스 형식(Access Code)과 오브젝트 형식(Object Type)의 조합으로 1byte를 구성하게 됩니다.

표 13-4 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.

13.2.2 오브젝트 읽기 요청

마스터 PC가 제어기의 오브젝트를 읽기 위해, 마스터 PC가 제어기에 보내는 쿼리 패킷을 구성합니다.

오브젝트의 값을 읽을 때는 표 13-4의 Access Code 0x30과 읽고자 하는 Object Type을 조합해서 Command를 먼저 구성해야 합니다. 읽고자 하는 오브젝트의 형이 INT16 이라면 Command에는

0x34를 사용합니다.

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

13.2.3 오브젝트 쓰기 요청

마스터 PC가 제어기의 오브젝트에 값을 쓰기 위해, 마스터 PC가 제어기에 보내는 쿼리를 구성합니다.

오브젝트에 값을 쓸 때는, 4th byte 지점 까지는 오브젝트 읽기 요청과 동일하게 구성합니다. 단, Command는 표 13-4의 Access Code에 의해 0x10과 Object Type에 맞는 조합으로 작성되어야 하며, 그 형식에 맞춰 오브젝트의 값(Value)을 작성합니다.

1st byte	2nd byte	3rd byte	4th byte	5th byte	6th byte	7th byte	8th byte
Command	Index		Sub-index	Value(INT8)			
				Value(INT16)			
				Value(INT32)			
				Value(FLOAT)			

13.2.4 오브젝트 읽기/쓰기 요청에 대한 성공 응답

제어기가 마스터 PC의 오브젝트 읽기/쓰기 요청에 응답하기 위해, 제어기가 마스터 PC에 보내는 응답 패킷을 구성합니다.

오브젝트의 값을 읽거나 쓰기가 성공한 경우에는 오브젝트의 값(Value)이 응답으로 돌아옵니다. 패킷의 구성은 오브젝트 쓰기 요청에서와 같습니다. 단, Command는 표 13-4의 Access Code에 제시된 것처럼 0x20이나 0x40중 하나와 Object Type에 맞는 조합으로 작성됩니다.

1st byte	2nd byte	3rd byte	4th byte	5th byte	6th byte	7th byte	8th byte
Command	Index		Sub-index	Value(INT8)			
				Value(INT16)			
				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.

13.2.5 오브젝트 읽기/쓰기 요청에 대한 실패 응답

제어기가 마스터 PC의 오브젝트 읽기/쓰기 요청에 실패를 알리기 위해, 제어기가 마스터 PC에 보내는 실패 응답 패킷을 구성합니다.

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

1st byte	2nd byte	3rd byte	4th byte	5th byte	6th byte	7th byte	8th byte
Command	Error Code						

오류 메시지는 표 13-5에 나타나 있습니다. CAN 패킷에서 오류 메시지 내용은 전달되지 않으며 Error Code만 반환됩니다. 그리고 Command는 0x80이 반환됩니다.

표 13-5 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

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

13.3 시리얼 바이너리 패킷

제어기의 오브젝트들은 시리얼(USB나 RS-232) 통신으로도 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.

13.3.1 바이너리 패킷의 기본 구조

바이너리 패킷의 메시지 구조는 CAN 패킷에서 사용되는 메시지 구조와 동일합니다. 단 CAN에서는 Object Type에 따라 패킷의 길이가 바뀌지만, 바이너리 패킷 구조에서는 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 CAN 통신에서의 메시지 구조와 동일합니다. 단 CAN은 8byte의 메시지를 모두 채우지 않아도 되지만, 시리얼 바이너리 패킷에서는 8byte의 메시지를 모두 채워야 합니다. 만약 메시지의 크기가 8byte가 되지 않는다면, 남는 공간에 0을 채우면 됩니다.

시리얼 바이너리 패킷에서도 CAN 패킷에서와 같이 리틀 인디안(Little-Endian)방식으로 패킷을 구성합니다.

13.3.2 오브젝트 읽기 요청

오브젝트의 값을 읽을 때는, 패킷의 4th ~ 11th byte 지점에 위치하는 Message를 아래와 같이 구성하면 됩니다. Command는 표 13-4의 Access Code 0x30과 읽고자 하는 Object Type을 조합해서 Command를 먼저 구성해야 합니다.

4th byte	5th byte	6th byte	7th byte	8th byte	9th byte	10th byte	11th byte
Command	Index		Sub-index	0	0	0	0

13.3.3 오브젝트 쓰기 요청

오브젝트에 값을 쓸 때는, 7th byte 지점 까지는 오브젝트 읽기 요청과 동일하게 구성합니다. 단 Command는 표 13-4의 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>

13.3.4 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 13-4의 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="4">Command</td><td colspan="2" rowspan="4">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>

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

13.3.5 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 13-5에 나타나 있습니다. 시리얼 바이너리 패킷에서 오류 메시지 내용은 전달되지 않으며 Error Code만 반환됩니다. 그리고 Command는 0x80이 반환됩니다.

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

13.4 시리얼 텍스트 패킷

시리얼(USB나 RS-232) 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 13-1의 Short Name이나 Long Name과 Sub-index를 사용합니다.

13.4.1 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)

만약 채널 1에 연결된 모터에 내려진 전압 명령의 값을 알고자 할 때는 아래와 같이 입력합니다.

```
voltage_command1↲
vtc1↲
```

모터의 전압 명령은 **'Voltage Command'**로, 이에 대한 Long Name은 **voltage_command**이고 Short Name은 **vtc**입니다. 그러므로 상기 두 명령은 동일한 결과를 가져옵니다.

13.4.2 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.

만약 채널 1에 연결된 모터를 10V 전압으로 구동시키고자 할 때 아래와 같이 입력합니다.

```
vtc1=10↲
voltage_command1=10↲
```

모터의 전압 명령은 **'Voltage Command'**로, 이에 대한 Long Name은 **voltage_command**이고 Short Name은 **vtc**입니다. 그러므로 상기 두 명령은 동일한 결과를 가져옵니다.

그리고 본 제어기의 가장 큰 장점 중의 하나는 듀얼 채널 제어기에 특화된 명령을 가지고 있습니다. 듀얼 채널 제어기에서는 다음과 같은 명령을 사용할 수 있습니다:

- **m_position_command**
- **m_velocity_command**
- **m_current_command**

- **m_velocity_command**
- **m_lav_command**

상기 명령을 인가하는 방법은 다음과 같은 형식을 가집니다. 여기서 Value1은 1번 채널을, Value2는 2번 채널을 의미합니다.

Short/Long Name	=	Value1	,	Value2	CR/LF

여기서 Sub-Index는 의미가 없기 때문에 사용하지 않습니다. 대신 각각의 채널에 대한 값을 ','로 구분하여 패킷을 구성합니다.

만약 두 모터에 위치 명령을 동시에 인가하는 **m_position_command/mpc** 오브젝트를 통해 명령을 내리고자 할 때 다음과 같이 입력하면 됩니다.

```
mpc=100,100↲
m_position_command=1000,2000↲
```

13.4.3 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.

13.4.4 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 13-5에 나타나 있습니다.

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

13.4.5 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 채널 1에 연결된 모터를 10V 전압으로 구동시키고자 할 때 아래와 같이 입력합니다.

```
1;vtc1=10↲
1;voltage_command1=10↲
```

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 제어기 간에 Device ID가 서로 충돌하지 않도록 주의하십시오.

14 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 "14.2 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.

14.1 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.

14.2 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 상수가 정의되어 있습니다:

- `_E, _PI, _EULER, _status, _s, _fault, _f`

14.3 문장

스크립트로 작성된 프로그램은 문장(Statement)들로 구성됩니다. 문장은 프로그램을 실행하기 위한 기본적인 실행 단위라고 볼 수 있습니다. 다음 그림 14-1은 스크립트 프로그램의 구조를 보여주고 있습니다.

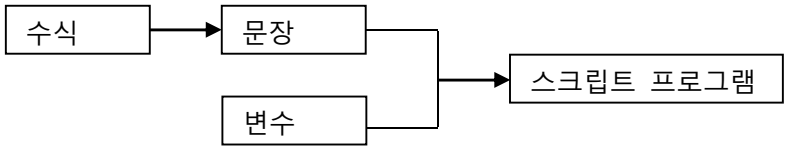

그림 14-1 스크립트 프로그램의 구조

14.3.1 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;
```

14.3.2 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;
}
```

14.3.3 제어문

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

제어문에 대해서는 "14.8 제어문"에서 상세하게 설명합니다.

14.4 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.

14.4.1 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
    ...
 */
```

14.4.2 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
```

14.5 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 부울형(`TRUE, FALSE`)이나 문자 상수(`'A', 'a'`), 문자열 상수(`"abc", "DEF"`)를 지원하지 않습니다.

14.5.1 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.

14.5.2 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.

14.5.3 미리 정의된 상수

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

표 14-1 정의된 수학 상수

상수	값	설명
_E	2.7182818284590452354	자연상수
_PI	3.14159265358979323846	원주율
_EULER	0.57721566490153286061	오일러-마스케로니 상수

수학 상수는 다음과 같이 사용할 수 있습니다.

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

또한 사용자가 제어기의 오브젝트를 참조하는 오브젝트 Index를 프로그램 작성시 쉽게 사용할 수 있도록 오브젝트 이름에 대한 상수를 정의하고 있습니다. 이 상수는 오브젝트의 Long Name과 Short Name 앞에 밑줄 문자 '_'를 추가하여 정의됩니다.

표 14-2 스크립트에서 사용 가능한 상수

Long Name, Short Name	Index	Description
_vendor_id, _vid	1	제품 공급자 ID
_product_id, _pid	2	제품 ID
_software_version, _swv	3	제어기 펌웨어 버전
_hardware_version, _hwv	4	제어기 하드웨어 버전
_system_status, _sst	5	제어기의 운용 상태
_system_command, _sco	7	제어기에 내려지는 명령
_battery_voltage, _bv	8	전원 소스로부터 제어기에 공급되는 전압 (단위: V)
_battery_current, _bc	9	전원 소스로부터 제어기를 통해 흐르는 전류 (단위: A)
_device_id, _id	11	장치 ID
_can_br, _cb	12	CAN 통신 속도
_serial_bps, _sb	13	USB/RS-232의 통신 속도
_serial_watchdog, _sw	14	CAN, USB, RS-232 포트로 수신되는 명령어 패킷에 대한 와치독 타이머 타임아웃 값 (단위: ms)
_startup_script_run, _ssr	16	제어기 시작 시 스크립트의 실행 여부
_control_mixing, _cm	21	두 모터에 내려지는 속도/전압 명령의 믹싱 모드
_center_safety, _cs	22	모터에 내려지는 속도/전류/전압 명령의 Center Safety
_min_max_safety, _ms	23	모터에 내려지는 속도/전류/전압 명령의 MinMax

		Safety
_pwm_switching, _ps	31	PWM 스위칭 방법
_pwm_frequency, _pf	32	PWM 주파수
_user_value, _uv	56	사용자가 읽고 쓰는 변수들, 플래시 메모리에 저장됨
_temp_value, _tv	57	사용자가 읽고 쓰는 변수들, RAM에만 저장됨
_num_di, _ndi	60	디지털 입력 채널의 수
_di_enable, _die	61	각 비트별 해당 디지털 입력 채널의 사용 여부
_di_invert, _dii	62	각 비트별 해당 디지털 입력 채널의 반전 여부
_di_all_values, _di	63	디지털 입력을 하나의 32bit 값으로 모음
_num_do, _ndo	65	디지털 출력 채널의 수
_do_enable, _doe	66	각 비트별 해당 디지털 출력 채널의 활성화 여부
_do_invert, _doi	67	각 비트별 해당 디지털 출력 채널의 반전 여부
_do_all_values, _do	68	디지털 출력을 하나의 32bit 값으로 모음
_num_ai, _nai	70	아날로그 입력 채널의 수
_ai_enable, _aie	71	각 비트별 해당 아날로그 입력 채널의 활성화 여부
_ai_invert, _aii	72	각 비트별 해당 아날로그 입력 채널의 극성 반전 여부
_num_pi, _npi	75	펄스 입력 채널의 수
_pi_enable, _pie	76	각 비트별 해당 펄스 입력 채널의 활성화 여부
_pi_invert, _pii	77	각 비트별 해당 펄스 입력 채널의 극성 반전 여부
_num_motors, _nm	80	제어기에 연결 가능한 모터의 수
_wheel_radius, _wr	86	이동로봇의 바퀴 반지름 (단위: m)
_axle_length, _al	87	이동로봇 좌우 바퀴간 거리 (단위: m)
_gear_ratio, _gr	88	모터와 바퀴간 감속비율 (모터 회전수/바퀴 회전수)
_m_position, _mp	91	모터 채널 1,2의 엔코더 값을 읽음 (단위: pulse, pulse)
_m_position_command, _mpc	92	모터 채널 1,2의 위치구동 명령을 내리고(단위: pulse, pulse) 엔코더 값을 읽음(단위: pulse, pulse)
_m_velocity_command, _mvc	93	모터 채널 1,2의 속도 구동 명령을 내리고(단위: RPM, RPM) 엔코더 값을 읽음(단위: pulse, pulse)
_m_current_command, _mcc	94	모터 채널 1,2의 전류 구동 명령을 내리고(단위: A, A) 전류 값을 읽음 (단위: A, A)
_m_voltage_command, _mvtc	95	모터 채널 1,2의 전압 구동 명령을 내리고(단위: V, V) 엔코더 값을 읽음(단위: pulse, pulse)
_m_lav_command, _mla	96	전진속도와 각속도로 이동로봇의 구동 명령을 내리고 (단위: m/s, rad/s) 모터 1,2의 엔코더 값을 읽음(단위: pulse, pulse)
_status, _s	102	모터제어기의 현재 상태
_fault, _f	103	모터제어기 및 관련 I/O에서 발생한 폴트
_command, _co	101	모터제어기에 내려지는 명령 코드

_position_command, _pc	111	모터의 폐루프 위치 제어 명령 (단위: pulse)
_velocity_command, _vc	112	모터의 폐루프 속도 제어 명령 (단위: RPM)
_current_command, _cc	113	모터의 폐루프 전류 제어 명령 (단위: A)
_voltage_command, _vtc	114	모터에 인가되는 전압 출력 (단위: V)
_temperature, _tp	121	FET와 방열판의 온도 (단위: ℃)
_voltage, _vt	122	모터에 가해지는 전압 (단위: V)
_current, _c	123	모터에 흐르는 전류 (단위: A)
_velocity, _v	124	모터의 회전속도 (단위: RPM)
_position, _p	125	모터의 회전위치 (단위: pulse)
_hall_count, _h	126	BLDC 모터의 홀 센서 카운트 값 (단위: pulse)
_ai_potentiometer, _pt	131	아날로그 입력 포트에 매핑된 위치 센서의 피드백 값
_ai_tachometer, _ta	132	아날로그 입력 포트에 매핑된 속도 센서의 피드백 값
_min_position, _np	141	모터가 이동 가능한 최소 위치 (단위: pulse)
_max_position, _xp	142	모터가 이동 가능한 최대 위치 (단위: pulse)
_home_position, _hp	143	홈센서 감지 시 position에 로드 되는 위치 값 설정 (단위: pulse)
_encoder_ppr, _ep	144	모터 1회전당 엔코더 펄스 수 (단위: pulse/rev)
_num_pole_pairs, _npp	145	BLDC 모터에서 홀센서의 폴페어 수 (단위: pulse/rev)
_use_soft_limit, _usl	146	소프트 리미트 스위치 사용 여부
_max_current, _xc	151	모터에 흐르는 최대 연속 전류 (단위: A)
_max_voltage, _xvt	152	모터에 가해지는 최대 전압 (단위: V)
_max_velocity, _xv	153	모터의 최대 회전 속도 (단위: RPM)
_acceleration, _ac	154	모터의 회전 가속도 (단위: RPM/s)
_deceleration, _dc	155	모터의 회전 감속도 (단위: RPM/s)
_overheat_limit, _ohl	161	FET와 방열판의 과열 한계 (단위: ℃)
_overcurrent_limit, _ocl	162	모터의 과전류 한계 (단위: A)
_overcurrent_delay, _ocd	163	과전류 허용 지연 시간 (단위: ms)
_peakcurrent_ratio, _pcr	164	모터의 순간 피크 전류 (단위: %)
_overvoltage_limit, _ovl	165	전원단의 과전압 한계 (단위: V)
_undervoltage_limit, _uvl	166	전원단의 저전압 한계 (단위: V)
_stall_detection, _sd	167	모터의 구동 명령에 대해 움직이지 않는 상황 감지
_vel_error_detection, _ved	168	폐루프 속도제어기에서 명령과 피드백 사이의 오차에 의한 이상 감지
_pos_error_detection, _ped	169	폐루프 위치제어기에서 명령과 피드백 사이의 오차에 의한 이상 감지
_feedback_sensor, _fs	171	제어기 피드벡 센서 선택
_profile_mode, _pm	172	속도 명령에 사다리꼴 프로파일 적용여부
_startup_power_on, _spo	173	제어기가 시작될 때 모터 Power ON/OFF 설정

_direction, _dir	174	모터의 정방향/역방향 회전 설정
_brake_on_delay, _bod	175	브레이크 작동 시 지연시간 설정 (단위: ms)
_high_voltage, _hv	176	제동저항(Brake Resistor)을 켜는 전압 설정 (단위: V)
_high_temperature, _ht	177	냉각 팬을 켜는 온도 설정
_cc_kp, _cp	181	PI 전류제어기의 비례제어 이득
_cc_ki, _ci	182	PI 전류제어기의 적분제어 이득
_cc_kff, _cff	185	PI 전류제어기의 모터 회전속도 전향 보상 이득
_vc_kp, _vp	186	PI 속도제어기의 비례제어 이득
_vc_ki, _vi	187	PI 속도제어기의 적분제어 이득
_vc_ks, _vs	190	속도 레퍼런스에 대한 스케일 펙터
_pc_kp, _pp	191	PID 위치제어기의 비례제어 이득
_pc_ki, _pi	192	PID 위치제어기의 적분제어 이득
_pc_kd, _pd	193	PID 위치제어기의 미분제어 이득
_di_value, _div	201	디지털 입력 채널의 값
_di_function, _dif	202	디지털 입력 채널을 특정 모터의 동작으로 매핑
_do_value, _dov	211	디지털 출력 채널의 값 (0 or 1)
_do_function, _dof	212	디지털 출력 채널을 특정 모터의 상태로 매핑
_ai_value, _aiv	221	아날로그 입력 채널의 원시 값
_ai_converted_value, _aicv	222	변환 과정을 거쳐 정규화된 값
_ai_linearity, _ail	223	지수/로그 변환 형식 지정
_ai_function, _aif	224	아날로그 입력을 특정 모터의 명령이나 피드백과 매핑
_ai_input_min, _ain	225	캘리브레이션: 입력 최소값
_ai_input_center, _aic	226	캘리브레이션: 입력 중앙값
_ai_input_max, _aix	227	캘리브레이션: 입력 최대값
_ai_input_deadband, _aidb	228	캘리브레이션: 입력의 데드밴드 값
_pi_value, _piv	231	펄스 입력 채널의 원시 값
_pi_converted_value, _picv	232	변환 과정을 거쳐 정규화 된 값
_pi_capture_type, _pit	233	펄스 캡쳐의 종류
_pi_linearity, _pil	234	지수/로그 변환 형식 지정
_pi_function, _pif	235	펄스 입력을 특정 모터의 명령이나 피드백과 매핑
_pi_input_min, _pin	236	캘리브레이션: 입력 최소값
_pi_input_center, _pic	237	캘리브레이션: 입력 중앙값
_pi_input_max, _pix	238	캘리브레이션: 입력 최대값
_pi_input_deadband, _pidb	239	캘리브레이션: 입력의 데드밴드 값
_script_size, _ss	250	바이트코드를 다운로드하기 전에 크기 전송 (단위: byte)
_script_code, _sc	251	바이트코드를 4-byte씩 묶어서 차례로 전송
_script_variable, _sv	252	스크립트가 실행되는 동안 사용되는 변수

오브젝트이름 상수는 다음과 같이 사용할 수 있습니다.

```
v = getv (_temp_value, 1)/1000;
w = getv (_temp_value, 2)/1000;

setv (_velocity_command, 1, v+w);
setv (_velocity_command, 2, v-w);
```

14.6 변수

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

14.6.1 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(_PI, _E, _status, _s, …)는 변수명으로 사용할 수 없음

변수 명은 다음과 다음과 같이 만들 수 있습니다.

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

14.6.2 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.

14.6.3 자료형 변환

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

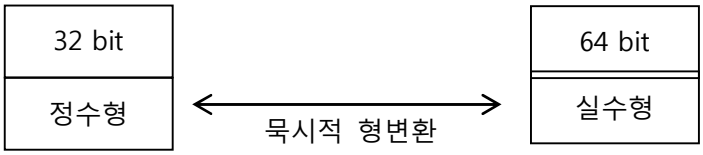

그림 14-2 스크립트에서의 자료 형변환

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
```

14.7 연산자

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

14.7.1 산술, 부호 연산자

스크립트 언어에서는 산술 연산을 표현할 수 있도록 산술 연산자를 제공합니다. 더하기는 '+', 빼기는 '-', 곱하기는 '*', 나누기는 '/', 나머지 연산은 '%' 기호를 사용하여 각각의 연산을 표현합니

다. '%' 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

14.7.2 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.

14.7.3 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

14.7.4 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.

연산자	연산식	예제		설명
		연산식	결과	
!	!a	!1	0	a가 거짓(false; 0)이면 참(true; 1)
&&	a && b	0&&1	0	a와 b가 모두 참이면 참
||	a || b	1||0	1	a가나 b 둘 중 하나라도 참이면 참

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

14.7.5 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 쉬프트함

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.

14.7.6 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
+=	a += b	n = n + 1; → n += 1;	a = a + b
-=	a -= b	n = n - 1; → n -= 1;	a = a – b
*=	a *= b	n = n * 1; → n *= 1;	a = a * b
/=	a /= b	n = n / 1; → n /= 1;	a = a / b
%=	a %= b	n = n % 1; → n %= 1;	a = a % b
&=	a &= b	n = n & 1; → n &= 1;	a = a & b
|=	a |= b	n = n | 1; → n |= 1;	a = a | b
^=	a ^= b	n = n ^ 1; → n ^= 1;	a = a ^ b
<<=	a <<= b	n = n << 1; → n <<= 1;	a = a << b
>>=	a >>= b	n = n >> 1; → n >>= 1;	a = a >> b

14.7.7 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 우선순위를 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.

우선순위	연산자	결합법칙
기본연산자	[ ] . ( ) ++ --	←
전치	+ - ++ -- ~ !	→
승제	* / %	→

가감	+ -	→
비트 이동	<< >>	→
관계	< <= > >=	→
등식	== !=	→
비트 AND	&	→
비트 XOR	^	→
비트 OR	|	→
논리곱	&&	→
논리합	||	→
대입	= *= /= %= += -= <<= >>= >>>= &= ^= |=	←

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.

14.8 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.

14.8.1 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) {
   …
}
```

14.8.2 if-else 문

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

14.8.3 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 C 기능 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
}
```

14.8.4 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++;
}
```

14.8.5 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;
}
```

14.8.6 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);
```

14.8.7 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++;
}
```

14.8.8 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);
```

14.8.9 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;
```

14.9 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
- getrv(device_id, index,sub_index) - device_id로 지정된 제어기에서 index와 sub_index로 지정된 오브젝트의 값을 읽어옴
- setrv(device_id, index,sub_index,x) - device_id로 지정된 제어기에서 index와 sub_index로 지정된 오브젝트의 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

14.9.1 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;
```

14.9.2 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;
```

14.9.3 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 1번 모터에 1초 마다 전압 +5V와 -5V을 교대로 가합니다.
while (1) {
      setv (_voltage_command, 1, 5);
      sleep (1000);
      setv (_voltage_command, 1, -5);
      sleep (1000);
}
```

14.9.4 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. index는 USB/RS-232 통신에서 사용하는 Short Name과 Long Name으로 대체할 수 있으며, 스크립트에서 사용하기 위해서는 Short Name과 Long Name 앞에 언더스코어 문자('_')를 붙여야 합니다.

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.

사용 가능한 index와 Short Name, Long Name에 대해서는 "**14.5.3 미리 정의된 상수**"를 참조하기 바랍니다.

Examples:

```
// 모터 채널 1의 전압 명령 값을 읽어옴, 다음 3개의 함수는 동일한 기능을 수행함
a = getv (114, 1);
b = getv (_vtc, 1);
c = getv (_voltage_command, 1);
```

14.9.5 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:

```
// 모터 채널 1에 전압 명령 5를 내림, 다음 3개의 함수는 동일한 기능을 수행함
setv (114, 1, 5);
setv (_vtc, 1, 5);
setv (_voltage_command, 1, 5);
```

14.9.6 getrv

Declaration: getrv (device_id, index, sub_index)

인수 device_id로 지정된 원격 제어기에서 index와 sub_index로 지정된 오브젝트 값을 읽어옵니다. `getrv()` 함수로 원격 제어기의 모든 오브젝트 값을 읽어올 수 있습니다.

device_id는 엑세스 하고자 하는 원격 제어기의 장치 ID 입니다.

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

Remarks: 그림 14-3의 스크립트에서 사용된 `getrv()`와 `setrv()` 함수는 원격 제어기의 오브젝트를 엑세스 하는데 사용됩니다.

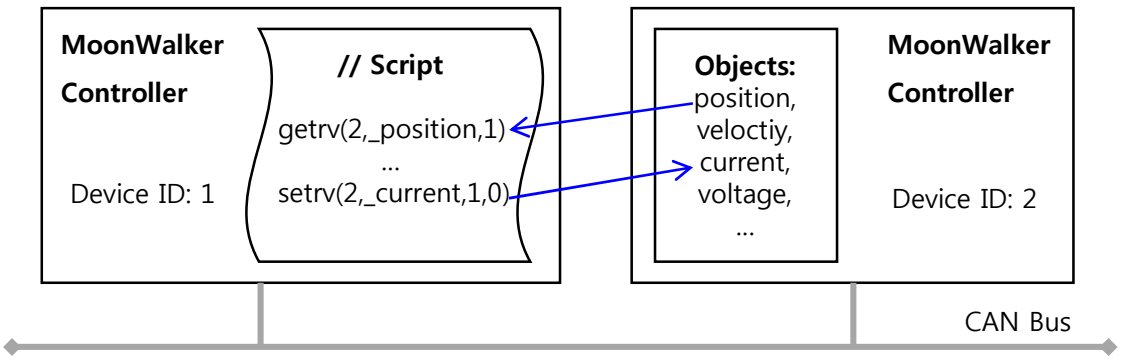

그림 14-3 스크립트에서 원격 제어기의 오브젝트 엑세스

스크립트를 실행하는 제어기는 마스터(상기 그림에서 Device ID: 1)가 되고 오브젝트의 값을 제공하는 제어기는 슬레이브(상기 그림에서 Device ID: 2)가 됩니다. 마스터와 슬레이브 제어기는 동일한 CAN bus에 연결되어 있어야 하며, 동일한 CAN 통신 속도를 사용하지만 장치 ID는 서로 달라야 합니다. 마스터와 슬레이브 제어기를 USB나 RS-232로 연결하였을 경우, `getrv()/setrv()` 함수는 제대로 동작하지 않습니다.

Examples:

```
// 장치ID 2번 제어기의 모터 채널 1의 전압 명령 값을 읽어옴
a = getrv (2, 114, 1);
b = getrv (2, _vtc, 1);
c = getrv (2, _voltage_command, 1);
```

14.9.7 setrv

Declaration: setrv (device_id, index, sub_index, x)

인수 device_id로 지정된 원격 제어기에서 index와 sub_index로 지정된 오브젝트에 x를 변경합니다. `setrv()` 함수로 원격 제어기의 모든 오브젝트에 값을 변경할 수 있습니다. 변경된 값은 원격

제어기에 즉시 적용됩니다. 하지만 변경된 값이 원격 제어기의 Flash Memory에 저장되지는 않습니다.

device_id에 대한 설명은 `getrv()` 함수의 설명을 참조하십시오.

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

Remarks: `setrv()` 함수는 원격 제어기의 오브젝트에 기록되는 값 x가 유효한지 검사하지 않습니다. 사용자는 오브젝트의 값을 변경할 때 주의하여야 합니다.

Examples:

```
// 장치ID 2번 제어기의 모터 채널 1에 전압 명령 5를 내림
setrv (2, 114, 1, 5);
setrv (2, _vtc, 1, 5);
setrv (2, _voltage_command, 1, 5);
```

14.9.8 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
```

14.9.9 sin

Declaration: sin(x)

라디안 각 x의 sine 값을 계산합니다.

인수 x에는 모든 값을 사용할 수 있습니다.

반환 값은 -1과 1 사이의 값입니다.

Examples:

```
// 정현파 전압을 가해 모터를 구동
```

```
for (i=0; i<100000000; i++) {
      v = 5*sin(i*0.01);
      setv (_voltage_command, 1, v);
      sleep (10);
}
```

14.9.10 cos

Declaration: cos(x)

라디안 각 x의 cosine 값을 계산합니다.

인수 x에는 모든 값을 사용할 수 있습니다.

반환 값은 -1과 1 사이의 값입니다.

14.9.11 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.

14.9.12 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.

14.9.13 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.

14.9.14 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.

14.9.15 sinh

Declaration: sinh(x)

x의 쌍곡선 sine을 계산합니다.

인수 x에는 모든 값을 사용할 수 있습니다.

반환 값은 모든 값이 될 수 있습니다.

14.9.16 cosh

Declaration: cosh(x)

x의 쌍곡선 cosine을 계산합니다.

인수 x에는 모든 값을 사용할 수 있습니다.

반환 값은 모든 값이 될 수 있습니다.

14.9.17 tanh

Declaration: tanh(x)

x 의 쌍곡선 tangent를 계산합니다.

인수 x에는 모든 값을 사용할 수 있습니다.

반환 값은 모든 값이 될 수 있습니다.

14.9.18 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
```

14.9.19 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
```

14.9.20 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
```

14.9.21 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
```

14.9.22 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(_E, 2);     // b는 7.389...
```

14.9.23 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(_E);        // a는 1
b = log(_E*_E);     // b는 2
```

14.9.24 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
```

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

14.9.26 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...
```

14.9.27 min

Declaration: min (x, y)

주어진 두 인수 x, y의 최소값을 선택합니다.

인수 x, y에는 모든 값을 사용할 수 있습니다.

14.9.28 max

Declaration: max (x, y)

주어진 두 인수 x, y의 최대값을 선택합니다.

인수 x, y에는 모든 값을 사용할 수 있습니다.

15 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.

15.1 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, **"14.2 C언어와의 차이"**만 살펴보더라도 바로 프로그램을 작성할 수 있습니다.

15.1.1 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.

15.1.2 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.

```
// 채널 1의 모터에 대하여 최대 전류, 전압, 속도, 가속도 설정
setv (_max_current, 1, 5);
setv (_max_voltage, 1, 24);
setv (_max_velocity, 1, 3000);
setv (_acceleration, 1, 1500);
setv (_deceleration, 1, 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.

```
// 채널 1의 모터에 대하여 최대 속도, 가속도 설정
setv (_max_velocity, 1, 1000);
setv (_acceleration, 1, 500);
setv (_deceleration, 1, 500);

// 아날로그 입력 채널 1의 값을 읽어 모터 채널 1의 속도를 조절
while (1) {
      speed = getv (_ai_converted_value, 1);
      speed = speed * 1000;
      setv (_velocity_command, 1, 0);
      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.

15.1.3 스크립트 예제

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*_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*_PI/enc;
	de2 *= 2*_PI/enc;

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

※ 엔티렉스 로봇연구소 홈페이지(www.ntrexgo.com)에 접속하면 제어기의 다양한 스크립트 예제를 다운로드 받을 수 있습니다.

15.2 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.

15.2.1 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 "15.2.2 컴파일 오류 메시지"를 참고하시기 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.

15.2.2 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'

15.3 다운로드 및 실행

15.3.1 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 60Kbyte의 바이트코드를 저장할 수 있는 공간이 예약되어 있습니다. 이를 소스코드로 환산하면 약 3천 라인 이상입니다. 이는 사용자가 스크립트로 필요한 기능을 구현하는데 충분한 양이 될 것입니다.

만일 제어기의 가상머신에서 프로그램이 실행 중일 때 새로운 파일을 다운로드 하면, 현재 실행중인 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.

15.3.2 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, USB, RS-232, CAN 포트를 통해 명령을 전송하여 실행합니다. Motor control UI 유틸리티에서 Script 탭의 [Run/Stop] 버튼으로 스크립트 프로그램을 실행하거나 중단할 수 있습니다.

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

```
sco=8  - 스크립트 시작 명령
sst    - 스크립트의 실행 상태 읽기
sco=9  - 스크립트 종료 명령
sst
```

자세한 내용은 "10.2.2 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 "제품 초기 설정 값" 상태로 되돌려야 합니다. "1.3.1 리셋 스위치"를 참조하기 바랍니다.

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

16 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.

16.1 소프트웨어 다운로드 및 실행

16.1.1 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

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

16.1.2 다운로드

UI 유틸리티 프로그램은 엔티렉스 로봇연구소 홈페이지에서 다운로드 받을 수 있습니다:

- 다운로드 경로: http://www.ntrexgo.com/archives/19482

다운로드 파일은 zip으로 압축되어 있습니다. 압축 파일을 풀면 "Motor Control UI v1.xx" 폴더 안에 MotorControlUI.exe 파일이 있습니다.

MoonWalker Motor Control UI Program 다운로드 받기

Motor Control UI v1.02

16.1.3 실행

Motor Control UI 유틸리티는 설치 과정이 필요 없습니다. 압축을 푼 폴더 안에 있는 MotorControlUI.exe 파일을 실행하면 됩니다. 이 유틸리티의 첫 실행 화면을 그림 16-1에서 보여주고 있습니다.

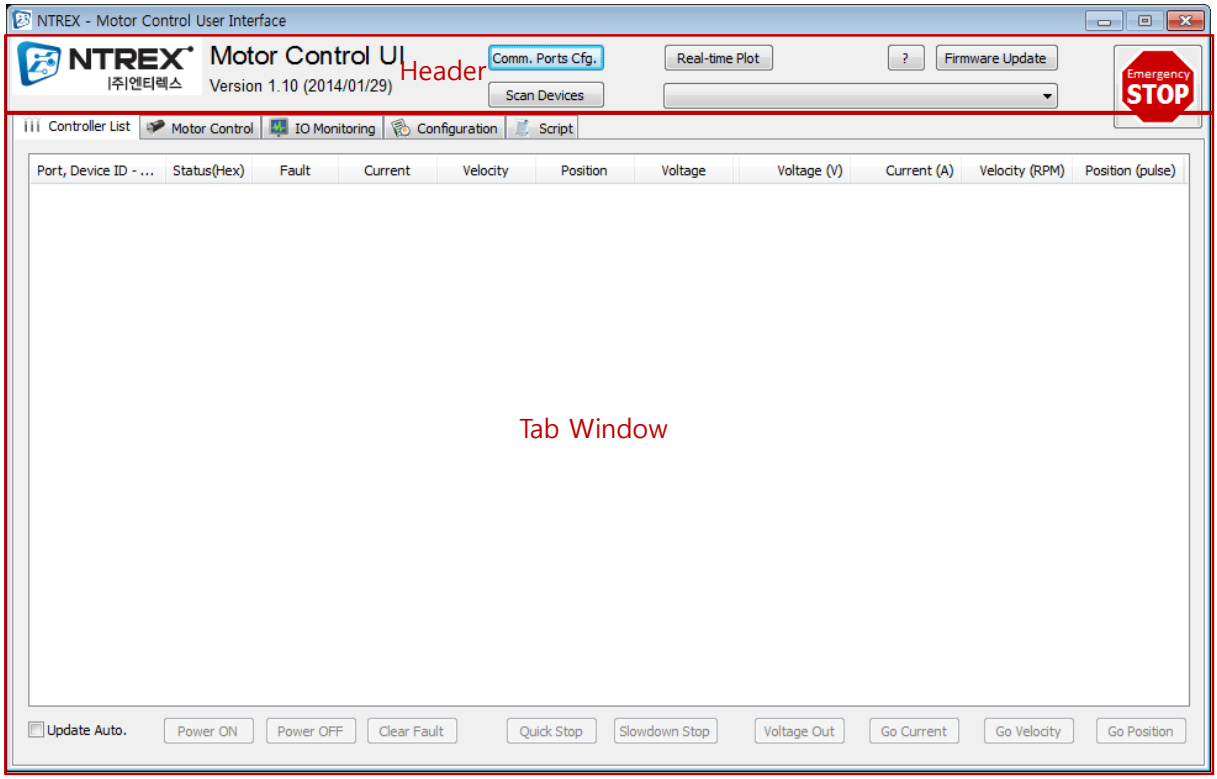

그림 16-1 Motor Control UI 유틸리티의 실행화면

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.

16.2 메인 화면 구성

MCUI 프로그램의 메인 화면은 헤더와 탭 윈도우 부분으로 구성됩니다.

16.2.1 헤더

헤더에는 UI 유틸리티를 제어기에 연결하기 위한 통신 설정, 장치의 검색, 검색된 장치 목록 표시 등의 기능이 있습니다.

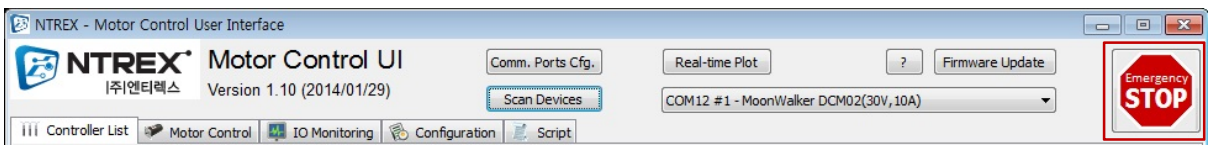

그림 16-2 헤더 파트

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

[Port Config.] 버튼을 누르면 PC에 연결된 제어기를 검색하기 위한 설정을 하는 대화상자를 표시합니다. [Scan Devices] 버튼은 CAN이나 USB, RS-232 포트를 통해 PC에 연결된 제어기를 검색합니다. 검색된 제어기는 버튼의 오른쪽 드롭 리스트 박스에 삽입됩니다. 만일 하나 이상의 제어기가 검색되었다면 첫 번째 검색된 제어기가 드롭 리스트 박스에 표시되고 자동으로 연결됩니다.

[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.

[?] 버튼을 누르면 UI 유틸리티 사용 팁 및 제어기 정보를 확인할 수 있습니다.

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

[Emergency STOP] 버튼은 비상 정지 버튼으로, UI 유틸리티에 연결된 제어기의 모터를 Power OFF 합니다. **※ 이 버튼은 검색된 제어기(연결되지 않은 제어기)에는 영향을 주지 않습니다.**

만일 하나 이상의 제어기가 검색되었다면, 이후 탭 윈도우에서 제어기를 설정하고 연결된 모터를 구동할 수 있습니다.

16.2.2 탭 윈도우

탭 윈도우는 기본적으로 Controller List, Motor Control, I/O Monitoring, Configuration, Script의 5가지 탭으로 구성되어 있습니다.

- Controller List Tab: 검색된 모든 제어기를 모터 별로 모니터링하고 명령을 인가
- Motor Control Tab: 제어기의 채널 별로 모터 제어 명령 인가 및 상태 모니터링
- I/O Monitoring Tab: 아날로그, 디지털, 펄스 입출력 채널 제어 및 모니터링
- Configuration Tab: 제어기, 모터, I/O 구성 설정
- Script Tab: 스크립트 작성과 다운로드 및 실행

사용자는 필요에 따라 해당 탭 선택하여 사용하면 됩니다.

16.3 Port Config

PC의 CAN, USB, RS-232 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.

Port Config 대화상자는 다음 그림과 같습니다.

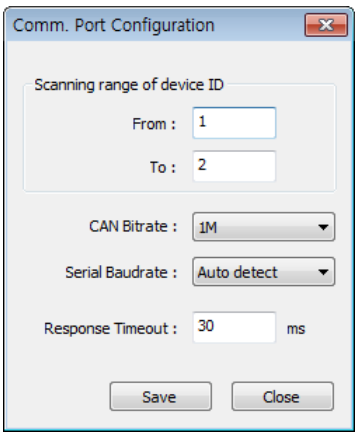

그림 16-3 Port Configuration 창 화면

먼저 'From'과 'To' 편집 박스에 장치(제어기) ID의 검색 범위를 설정합니다. 상기 그림과 같이 "From: 1, To: 10"으로 설정하면 제어기 ID를 1부터 10까지 검색하겠다는 의미입니다. 이때 UI 프로그램은 많은 제어기 ID를 검색해야 하기 때문에 검색 시간이 오래 걸립니다. 만약 제어기 ID 값이 1인 경우만 검색하고 싶다면 "From: 1, To: 1"로 설정하면 됩니다.

'CAN Bitrate'에서는 CAN 포트의 통신 속도(10K, 25K, 50K, 125K, 250K, 500K, 800K, 1Mbps)를 선택합니다. 제품 초기 설정 값으로 CAN 통신 속도는 1Mbps로 설정되어 있습니다.

또한 '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.

'Response Time'에는 제어기에 패킷을 보내고 되돌아올 때까지 기다리는 응답 시간을 밀리세컨드 단위로 설정합니다.

모든 설정이 끝나면 [Save] 버튼을 눌러 변경된 설정 사항을 저장합니다. 만일 [Close] 버튼을 누르면 변경된 설정을 저장하지 않고 대화상자를 닫습니다.

16.4 Real-time Plot

Real-time Plot 창은 배터리 전압/전류, 사용자 변수, 모터에 내려지는 위치, 속도, 전류, 전압 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 16-4과 같습니다.

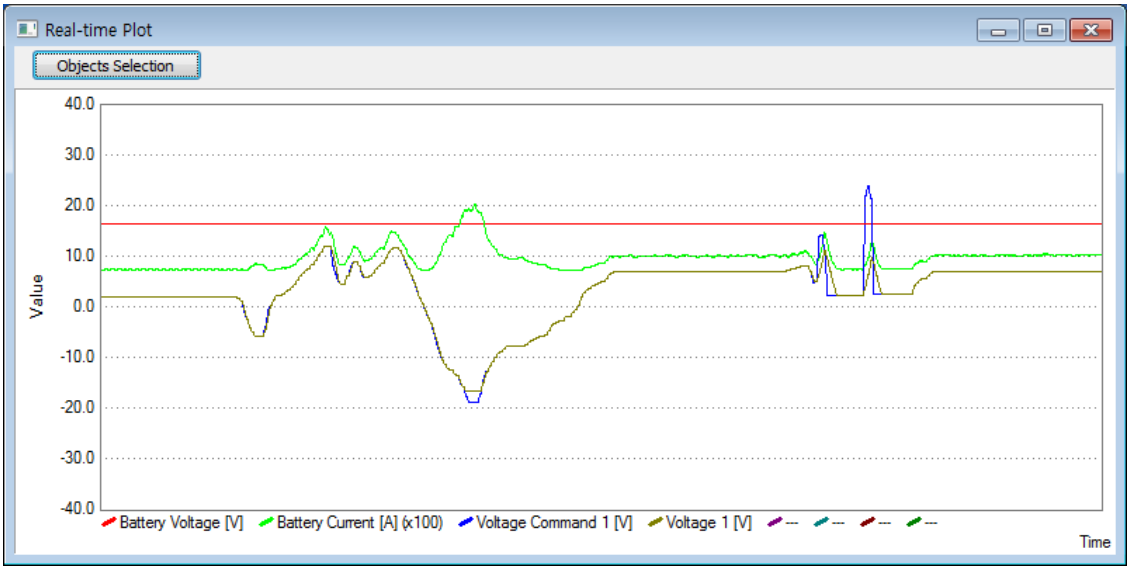

그림 16-4 Real-time Plot 창 화면

그래프의 왼쪽 에 표시되는 스케일(Scale)은 **마우스 휠**을 위아래로 굴려 확대하거나 축소할 수 있습니다. 또한, 그래프의 아래에는 그래프로 표시되는 항목들에 대한 세부 정보가 표시됩니다.

[Objects Selection] 버튼을 클릭하면 그림 16-5과 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.

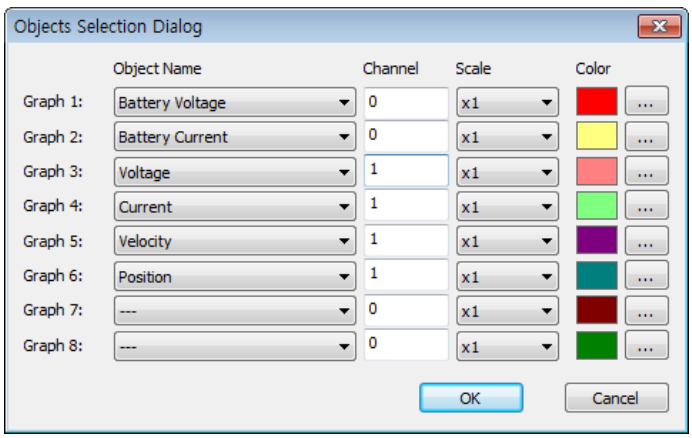

그림 16-5 Objects Selection Dialog 창 화면

오브젝트(Object)는 아래와 같은 항목을 선택할 수 있습니다.

- Battery Voltage, Battery Current
- User Value, Temp Value
- Position Command, Velocity Command, Current Command, Voltage Command
- Temperature, Voltage, Current, Velocity
- Position, Hall Count
- AI Potentiometer, AI Tachometer

- DI Value, Do Value
- AI Raw Value, AI Converted Value
- PI Raw Value, PI Converted Value

16.5 Controller List 탭

Controller List 탭에서는 그림 16-6과 같이 검색된 모든 제어기의 모터 상태를 모니터링하고 선택된 모터에 전압과 전류, 속도, 위치 명령을 내릴 수 있습니다.

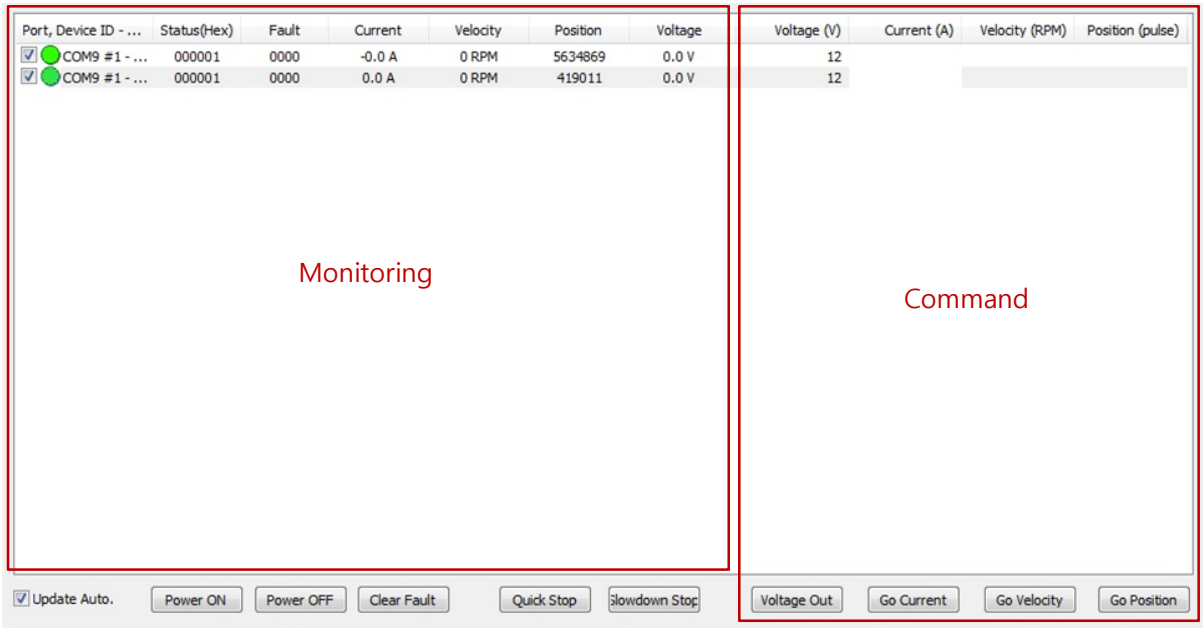

그림 16-6 Controller List 탭 화면

검색된 제어기의 모터 상태를 모니터링하기 위해서는 먼저 'Update Auto.' 체크 박스가 체크되어 있어야 합니다. 그리고 각 항목의 체크 박스가 체크되어 있어야 합니다. 그러면 상기 윈도우에서와 같이 모터의 Status, Fault, Current, Velocity, Position, Voltage 값을 표시합니다.

제어기의 모터에 구동 명령을 내리기 위해서는 Voltage, Current, Velocity, Position 칼럼에 명령 인수를 적습니다. 그리고 [Voltage Out], [Go Current], [Go Velocity], [Go Position] 버튼을 눌러 선택된 제어기의 모터에 전압 출력, 전류 제어, 속도 제어, 위치 제어 명령을 내립니다.

[Quick Stop]과 [Slowdown Stop] 버튼은 선택된 제어기의 모터를 긴급히 멈추거나 감속하여 멈추게 됩니다. 정지와 관련된 버튼은 제어기의 모든 명령으로 구동 중인 모터를 정지할 수 있습니다.

[Power ON]과 [Power OFF] 버튼은 선택된 제어기의 모터를 Power ON 하거나 Power OFF 합니다. 모터에 명령을 내리고 구동하기 전에 모터는 먼저 Power ON 상태가 되어야 합니다. Power OFF 상태에서는 모든 명령이 차단됩니다.

[Clear Fault] 버튼은 선택된 제어기의 모터에 발생한 폴트 플래그를 모두 리셋 합니다. 폴트 플래그를 리셋 한다고 해서 제어기의 문제가 바로잡히는 것은 아닙니다. 사용자는 반드시 폴트를 유발하는 원인을 찾아 문제를 없앤 후 제어기를 운영하는 것이 좋습니다.

16.6 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.

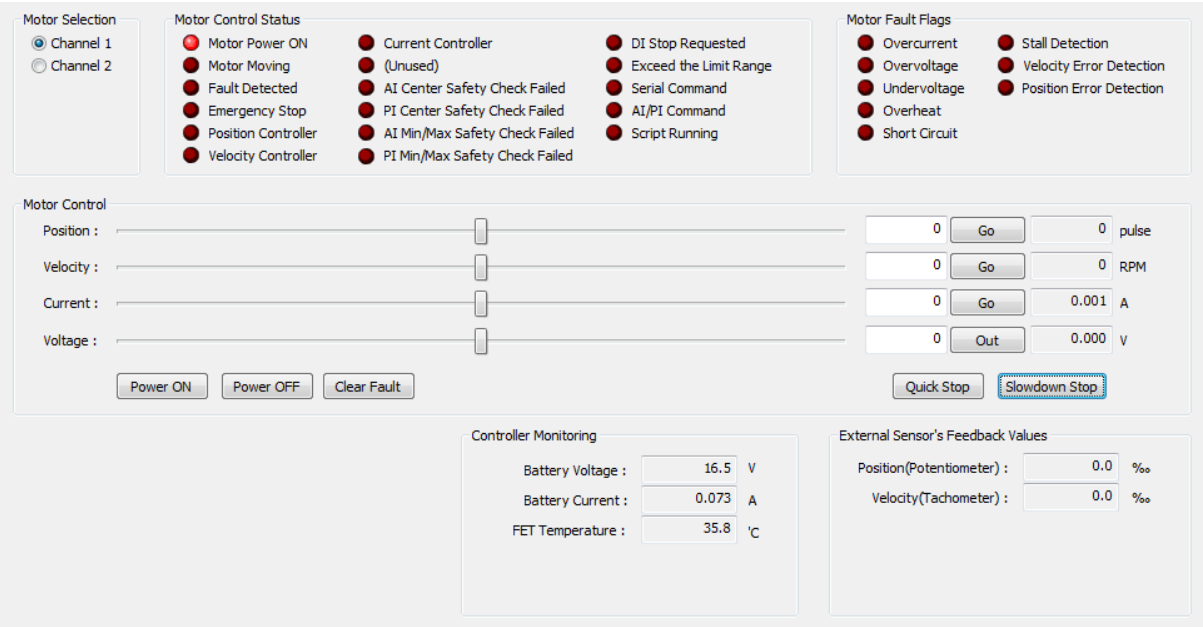

그림 16-7 Motor Control 탭 화면

16.6.1 Motor Selection

듀얼 채널 제어기에는 두 개의 모터가 연결 가능합니다. 이때 어느 채널의 모터를 모니터링하고 제어할지를 먼저 결정해야 합니다. 이를 위해 Motor Selection 그룹에서 Channel 1과 Channel 2 중 하나를 선택합니다.

싱글 채널 제어기에서는 한 개의 모터만 연결할 수 있기 때문에, 이 기능은 비활성화되고 Channel 1이 기본적으로 선택됩니다.

16.6.2 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 Power ON - 모터에 전원이 공급되고 있는 상태
- Motor Moving - 모터가 회전하고 있는 상태
- Fault Detected - 모터에 폴트가 발생한 상황
- Emergency Stop - 긴급 정지 버튼이 작동한 상태
- Position Controller - 위치 제어기가 동작중인 상태
- Velocity Controller - 속도 제어기가 동작중인 상태
- Current Controller - 전류 제어기가 동작중인 상태
- AI Center Safety Check Failed - 아날로그 입력 값이 센터에서 시작되지 않음
- PI Center Safety Check Failed - 펄스 입력 값이 센터에서 시작되지 않음
- AI Min/Max Safety Check Failed - 아날로그 입력에서 적절한 값이 들어오지 않음
- PI Min/Max Safety Check Failed - 펄스 입력에서 적절한 값이 들어오지 않음
- DI Stop Requested - I/O에 연결된 정지 스위치가 켜진 상태
- Exceed the Limit Range - 모터의 위치가 리미트 범위를 벗어난 상태
- Serial Command - RS-232, USB, CAN에서 명령어가 입력되는 상태
- AI/PI Command - 아날로그/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

'Fault Detection'이 활성화되면 Motor Fault Flags 창에 어느 폴트 플래그가 활성화 되었는지 확인할 수 있습니다. 이는 다음 절에 자세히 설명됩니다.

'AI Outside Valid Range'나 'PI Outside Valid Range'가 활성화되면 아날로그나 펄스 입력이 센터에서 시작되지 않은 상황입니다. 자세한 상황은 "10.8.2center_safety - Center Safety"를 참조하기 바랍니다.

'Loss Detection of AI'나 'Loss Detection of PI'가 활성화되면 아날로그나 펄스 입력이 Min/Max 범위를 벗어났거나 제대로 들어오지 않은 상황입니다. 자세한 상황은 "10.8.3min_max_safety - Min/Max Safety"를 참조하기 바랍니다.

'Stopped by External Input'이 활성화되면 디지털 입력에서 Quick Stop, Declaration Stop가 작동한 상태입니다. 이 상태를 벗어나기 위해서는 해당 디지털 입력이 OFF 상태가 되어야 합니다.

'Passed the Limit Position'이 활성화되면 디지털 입력에서 'Forward Limit Switch'나 'Reverse Limit

Switch'가 작동한 상태입니다. 그리고 소프트 리미트 스위치가 작동한 상태입니다. 자세한 상황은 "11.4 위치 센서 설정"을 참조하기 바랍니다.

16.6.3 Motor Fault Flags

Motor Control Status는 모터에서 폴트가 발생한 세부 내역을 확인할 수 있습니다. 모터의 폴트 플래그 목록은 다음과 같습니다:

- Overcurrent - 모터에 과전류가 흐름
- Overvoltage - 제어기에 과전압이 걸림
- Undervoltage - 제어기에 저전압이 걸림
- Overheat - 내부 MOSFET와 방열판이 과열 됨
- Short Circuit - 모터의 +, -단자가 단락 되거나 FET가 잘못된 조합으로 운용
- Stall Detection - 모터에 전력이 공급되나 회전하지 않는 상태 감지
- Velocity Error Detection - 속도 제어기에서 속도 오차가 지정된 값 이상이 됨
- Position Error Detection - 위치 제어기에서 위치 오차가 지정된 값 이상이 됨

제어기에서 폴트가 발생하면 모터에 공급되는 전원은 차단(Power OFF)되며, 폴트 조건이 해제된 이후에 모터는 Power ON 가능합니다.

폴트가 발생하는 조건과 대처에 대해서는 "11.3.2 fault - Fault"를 참조하기 바랍니다.

16.6.4 Motor Control

Motor Control 그룹에서는 모터에 위치, 속도, 전류, 전압 명령을 내리고 모터의 현재 위치, 속도, 전류, 전압을 읽을 수 있습니다.

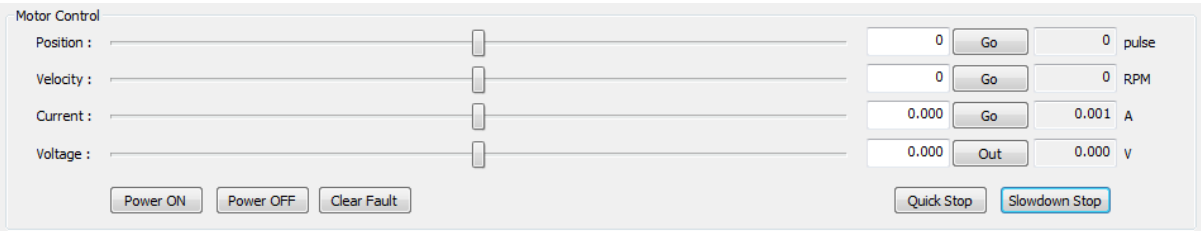

그림 16-8 Motor Control 창

위치, 속도, 전류, 전압에 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(위치, 속도, 전류, 전압)가 표시됩니다. 이 값은 주기적으로 업데이트 됩니다.

[Power ON]과 [Power OFF] 버튼은 모터에 Power ON이나 Power OFF 명령을 내립니다. [Clear Fault]

버튼은 제어기의 모터에 발생한 모든 폴트 플래그를 리셋 합니다.

[Quick Stop]과 [Slowdown Stop] 버튼은 선택된 제어기의 모터를 긴급히 멈추거나 감속하여 멈추게 됩니다.

16.6.5 Controller Monitoring

Controller Monitoring 그룹에서는 배터리 전압과 전류, FET 온도를 모니터링 합니다.

16.6.6 External Sensor's Feedback Values

External Sensor's Feedback Values 그룹에서는 속도제어나 위치제어의 피드백에 사용되는 타코미터나 포텐셔미터 입력 값을 모니터링 합니다.

16.7 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.

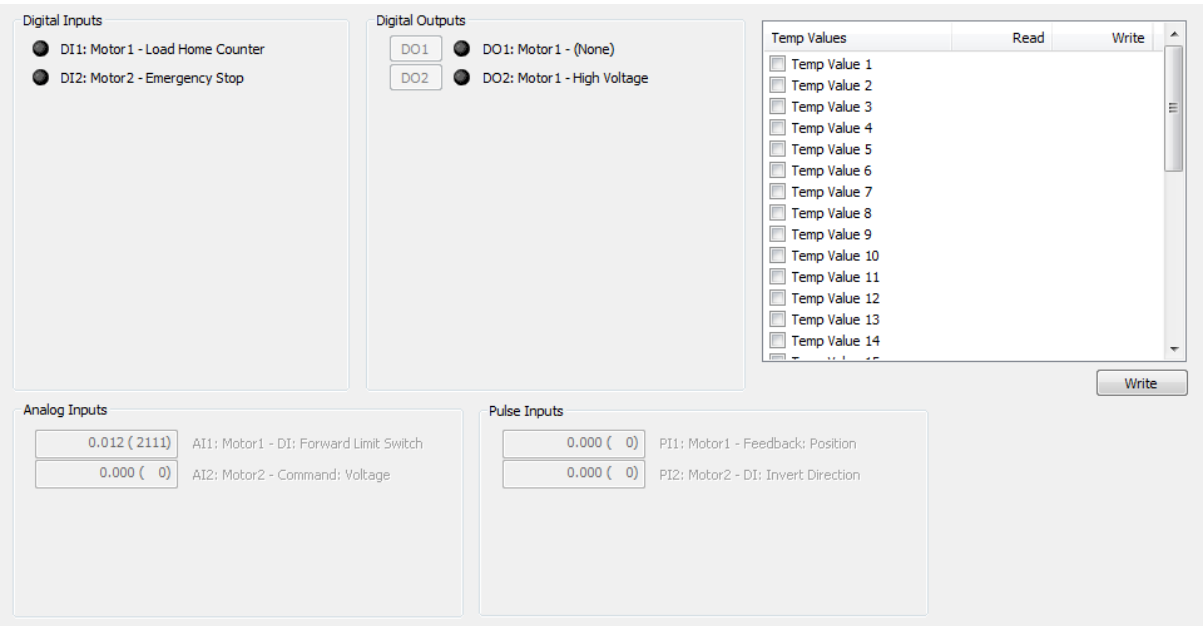

그림 16-9 I/O Monitoring 탭 화면

16.7.1 Digital Inputs

디지털 입력 그룹은 Configuration 탭에서 Enable 된 디지털 입력 채널의 값을 모니터링하고 매

핑된 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

16.7.2 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.

16.7.3 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.

16.7.4 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.

16.7.5 Temp Values

Temp Values 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.

16.8 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.

이 탭은 두 개의 창으로 나누어져 있습니다. 왼쪽은 모터와 제어기에 관련된 구성 파라미터의 설정 창이고 오른쪽은 입출력 채널에 관련된 구성 파라미터의 설정 창입니다.

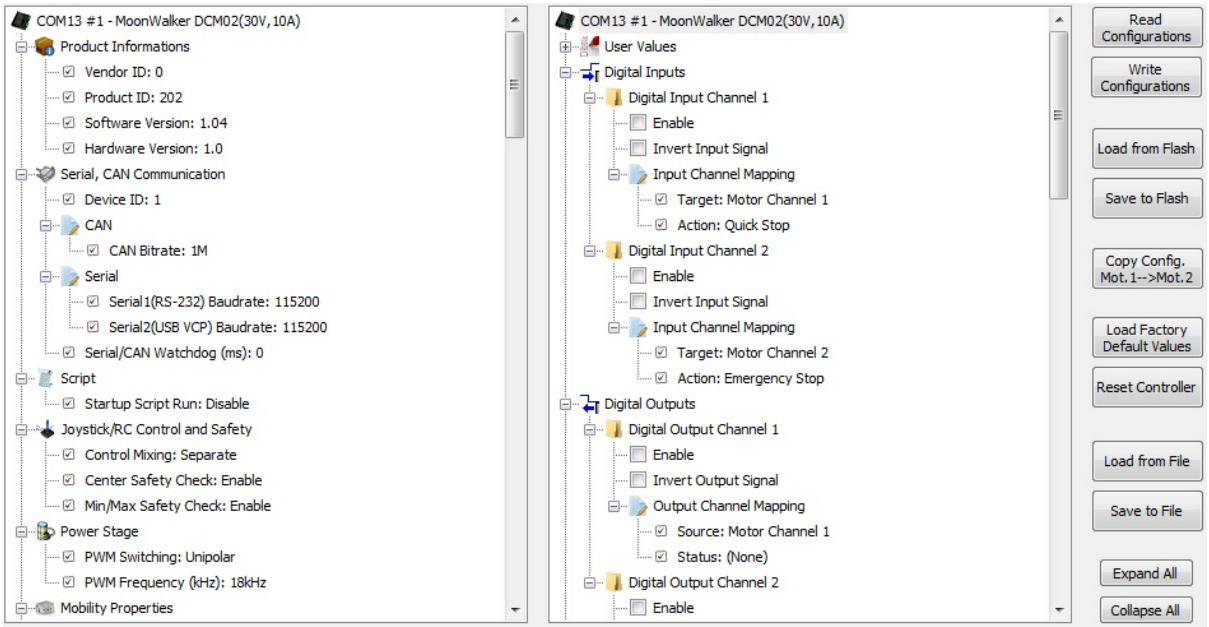

그림 16-10 Configuration 탭 화면

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 from Flash - 플래시 메모리 저장된 구성 파라미터 값을 불러오기
- Save to Flash - 제어기의 구성 파라미터 값을 플래시 메모리에 저장
- Copy Config. Mot.1->Mot.2 - Motor 1 설정 값을 Motor 2에 복사하기

- Load Factory Default Values - 제품 초기 설정 값으로 되돌리기
- Reset Controller - 제어기의 소프트웨어 리셋
- Load from File - PC에서 구성 파라미터 파일 읽어 제어기로 쓰기
- Save to File - 제어기의 구성 파라미터를 읽어 PC에 파일로 저장
- Expand All - 트리 메뉴를 모두 펼침
- Collapse All - 트리 메뉴를 모두 접기

아래 그림 16-11은 제어기에서 구성 파라미터가 복사되는 경로와 탭의 오른쪽 버튼 기능과의 관계를 보여줍니다.

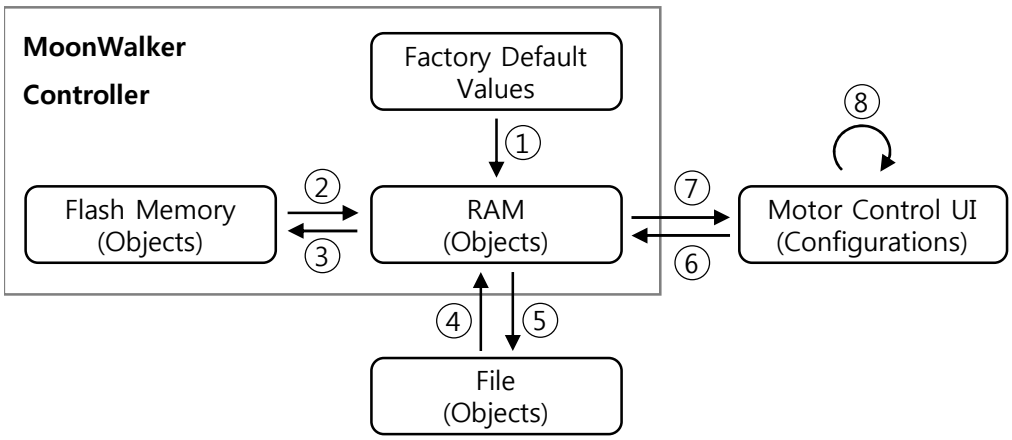

그림 16-11 Configuration 탭 버튼 구성도

제어기에는 RAM, Flash Memory, Factory Default Values가 있습니다. RAM은 전원을 끄면 모든 데이터가 초기화가 되는 임시저장장치이고 Flash Memory는 전원이 끊겨도 저장된 정보가 지워지지 않는 기억장치입니다. 그리고 Factory Default Value은 공장 출하시 초기 설정 값(제품 초기 설정 값)입니다.

[Read Configuration] 버튼을 클릭하면 RAM에 저장된 설정 값을 UI 유틸리티로 읽어오고(그림 16-11에서 ⑦), [Write Configuration] 버튼을 클릭하면 UI 유틸리티의 설정 값을 RAM에 저장합니다(그림 16-11에서 ⑥). 하지만 전원을 ON/OFF하면 모든 설정 값이 초기화됩니다.

[Load Factory Default Values] 버튼을 클릭하면 제어기의 초기 설정 값을 RAM으로 읽어 온 후 UI 유틸리티로 읽어옵니다 (그림 16-11에서 ①, ⑦).

[Copy Config. Mot1->Mot2] 버튼을 클릭하면 UI 유틸리티 내에서 Motor 1에서 설정한 값을 Motor 2에 복사합니다(그림 16-11에서 ⑧).

[Load from File] 버튼을 클릭하면 PC에 저장된 cfg 파일을 RAM으로 읽어온 후 다시 UI 유틸리티로 읽어옵니다(그림 16-11에서 ④, ⑦). [Save to File] 버튼을 클릭하면 RAM에 저장된 설정 값을 cfg 파일로 저장합니다(그림 16-11에서 ⑤).

[Load from Flash] 버튼을 클릭하면 Flash Memory에 있는 설정 값을 RAM으로 읽어 온 후 다시 UI 유틸리티로 읽어옵니다(그림 16-11에서 ②, ⑦). 그리고 [Save to Flash] 버튼을 클릭하면 RAM에 저장된 설정 값을 Flash Memory에 저장하게 됩니다(그림 16-11에서 ③). 이때 저장된 설정 값은 전원을 ON/OFF해도 지워지지 않습니다.

※ Configuration탭에 있는 [Save to Flash] 버튼과 [Write Configurations] 버튼 기능을 혼동하지 마시기 바랍니다. [Save to Flash] 버튼은 제어기의 RAM에서 플래시 메모리로 설정 값을 저장(쓰기)하는 기능이고, [Write Configurations] 버튼은 UI 유틸리티의 설정 값을 제어기의 RAM에 저장(쓰기)하는 기능입니다.

16.8.1 Product Information

Product Information은 현재 연결된 제품 공급자 ID, 제품 ID, 제어기와 펌웨어 버전을 확인할 수 있습니다. 이 설정은 사용자가 임의로 변경할 수 없습니다.

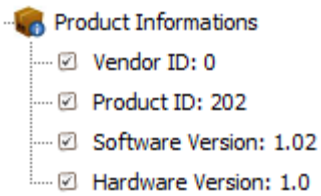

그림 16-12 Product Information

이 그룹의 항목과 관련되는 오브젝트는 다음과 같습니다:

- Vendor ID - vendor_id
- Product ID - product_id
- Software Version - software_version
- Hardware Version - hardware_version

16.8.2 Serial/CAN Communication

Serial/CAN Communication은 제어기 ID, 현재 연결된 CAN 통신 속도, Serial 통신 속도, 와치독 등을 설정하고 확인할 수 있습니다. 여기서 Serial 1은 RS-232이며, Serial 2는 USB 가상 시리얼 포트입니다.

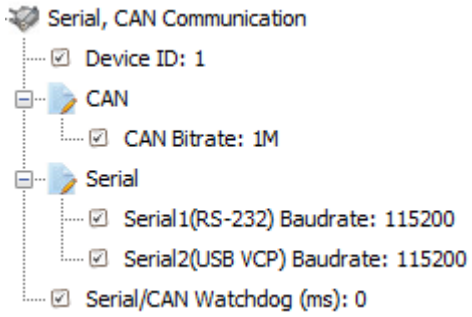

그림 16-13 Serial/CAN Communication

이 그룹의 항목과 관련되는 오브젝트는 다음과 같습니다:

- Device ID - device_id
- CAN Bitrate - can_br
- Serial 1(RS-232) Baudrate - serial_bps1
- Serial 2(USB VCP) Baudrate - serial_bps2
- Serial/CAN Watchdog - serial_watchdog

이 그룹에서 Device ID, CAN Bitrate, Serial 1(RS-232) Baudrate, Serial 2(USB VCP) Baudrate 값을 변경한 경우에는 제어기를 재시작해야 변경된 값이 반영됩니다.

이 오브젝트들에 대한 설명은 "10.4 통신 설정"을 참조하기 바랍니다.

16.8.3 Script

Script는 제어기에 전원이 인가되었을 때 제어기에 저장한 스크립트가 자동으로 실행되게 설정할 수 있습니다.

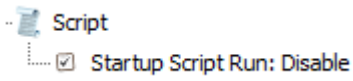

그림 16-14 Script

이 그룹의 항목과 관련되는 오브젝트는 다음과 같습니다:

- Startup Script Run - startup_script_run

16.8.4 Joystick/RC Control and Safety

Joystick/RC Control and Safety는 Control Mixing mode와 Center Safety 그리고 Min/Max Safety 기능과 사용 여부를 설정할 수 있습니다.

그림 16-15 Joystick/RC Control and Safety

이 그룹의 항목과 관련되는 오브젝트는 다음과 같습니다:

- Control Mixing - control_mixing
- Center Safety Check - center_safety
- Min/Max Safety Check - min_max_safety

이 오브젝트들에 대한 설명은 "10.8 모바일 로봇 속성"을 참조하기 바랍니다.

16.8.5 Power Stage

Power Stage는 unipolar와 bipolar 그리고 PWM 신호 주파수를 설정할 수 있습니다.

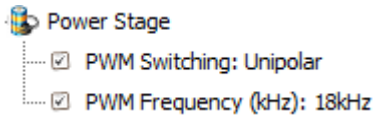

그림 16-16 Power Stage

이 그룹의 항목과 관련되는 오브젝트는 다음과 같습니다:

- PWM Switching - pwm_switching
- PWM Frequency - pwm_frequency

이 오브젝트들에 대한 설명은 "10.5 전원단 설정"을 참조하기 바랍니다.

16.8.6 Mobility Properties

Mobility Properties는 이동 로봇의 바퀴의 지름, 좌우 바퀴간의 거리, 모터와 바퀴간 감속비율(모터 회전수/바퀴 회전수) 등 로봇의 기본 설정 값을 설정할 수 있습니다. 이 설정은 2채널 제어기에만 적용된 설정입니다.

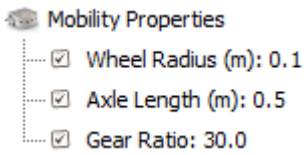

그림 16-17 Mobility Properties

이 그룹의 항목과 관련되는 오브젝트는 다음과 같습니다:

- Wheel Radius - wheel_radius
- Axle Length - axle_length
- Gear Ratio - gear_ratio

이 오브젝트들에 대한 설명은 "10.8 모바일 로봇 속성"을 참조하기 바랍니다.

16.8.7 Motors

제어기 제품에 따라 1축 제어기는 Motor 1로, 2축 제어기는 Motor 1과 Motor 2로 구성됩니다.

Motor 1과 Motor 2는 Position Sensor, Motor Characteristics, Fault Condition, Operations, Closed Loop Controller 등 5개의 설정으로 구분되어 있습니다. 여기서 Motor 1과 Motor 2의 메뉴는 동일하므로 Motor 1에 대해서만 설명합니다.

16.8.8 Motors – Position Sensors

Position Sensor는 위치 센서에 대한 설정입니다. 위치 제어를 하기 위해 최소/최대 위치 값, 홈센서 감지 시 Position에 로드 되는 위치 값, 모터 1회전당 엔코더 펄스 수, Soft Limit 사용 여부를 설정할 수 있습니다.

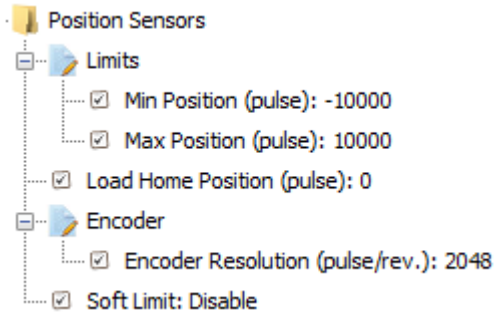

그림 16-18 Position Sensor

Soft Limit은 Encoder, Hall Sensors, Potentiometer 값이 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 Passed the Limit Position이 활성화 됩니다.

이 그룹의 항목과 관련되는 오브젝트는 다음과 같습니다:

- Min Position - min_position
- Max Position - max_position
- Load Home Position - home_position
- Encoder Resolution - encoder_ppr

- Soft Limit - use_soft_limit

이 오브젝트들에 대한 설명은 "11.4 위치 센서 설정"을 참조하기 바랍니다.

16.8.9 Motors – Motor Characteristics

Motor Characteristics는 사용자가 사용하고자 하는 모터 자체의 특성치에 대한 설정입니다. 모터에 흐르는 최대 연속 전류, 모터를 구동하는 최대 전압, 모터의 최대 회전 속도, 모터의 회전 가속도, 모터의 회전 감속도를 설정할 수 있습니다.

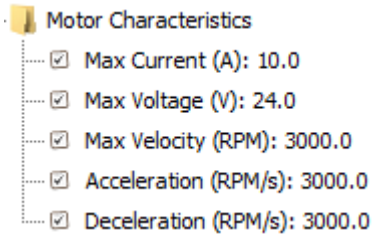

그림 16-19 Motor Characteristics

사용자가 12V 모터와 24V 모터를 동시에 제어기 연결해서 사용하고 싶다면 Motor Characteristics 설정을 이용해서 제어기와 모터를 안전하게 사용할 수 있습니다. 먼저 12V 모터와 24V 모터를 동시에 사용하기 위해서는 24V 배터리가 필요합니다. 여기서 문제는 24V 배터리를 24V 모터를 구동하는데 문제가 없지만 12V 모터를 구동할 경우 모터와 제어기가 파손될 위험이 있습니다. 따라서 안전을 위해 12V 모터는 최대 사용 전압을 12V로 설정해 줄 필요가 있습니다.

만약 12V 모터가 제어기의 Motor 1에 24V 모터가 제어기의 Motor 2에 연결되었다고 가정하면 Motor 1과 Motor 2의 Max Voltage를 각각 12V와 24V로 설정해 주면 됩니다. 그러면 Motor 1에서는 최대 사용 전압이 12V로 Motor 2에서는 최대 사용 전압 24V로 설정됩니다. 그러면 제어기와 모터를 안전하게 사용할 수 있습니다.

이 그룹의 항목과 관련되는 오브젝트는 다음과 같습니다:

- Max Current - max_current
- Max Voltage - max_voltage
- Max Velocity - max_velocity
- Acceleration - acceleration
- Deceleration - deceleration

이 오브젝트들에 대한 설명은 "11.5 모터 속성 설정"을 참조하기 바랍니다.

16.8.10 Motors – Fault Conditions

Fault Condition은 사용자 그리고 제품의 안전을 위해 Fault 감지 조건을 설정할 수 있습니다. Fault Condition Limits는 과열온도, 과전류, 과전류 흐르는 시간, 제어기의 FET가 흘릴 수 있는 순간 최대 전류범위, 과전압, 저전압 한계값을 설정할 수 있고, Fault Condition Detection은 모터의 Fault 감지 조건을 설정할 수 있습니다.

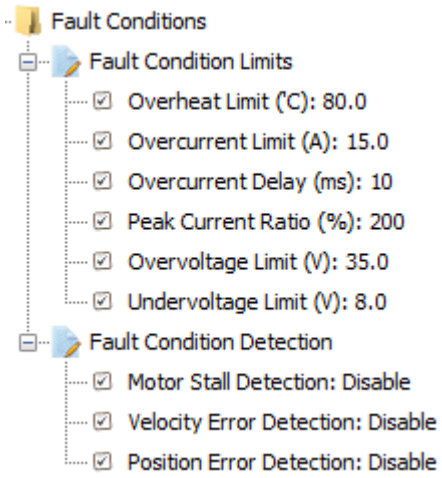

그림 16-20 Fault Condition

만약 위와 같이 설정되었을 때 방열판 온도가 80도 이상, 과전압 35V 이상, 저전압 8V 이하, 그리고 과전류가 10ms(Overcurrent Dealy: 10)동안 15A 이상 또는, 30A(Peak Current Ratio: 200) 이상 한 번이라도 감지되면 폴트(Fault)가 발생하게 됩니다. 여기서 주의할 점은 과전류와 과전압을 Motor Characteristics에 Max Current와 Max Voltage 값과 같은 값으로 설정하면 안됩니다. 만약 같은 값으로 설정할 경우 계속 과전압과 과전류가 감지돼서 폴트가 발생하게 됩니다.

이 그룹의 항목과 관련되는 오브젝트는 다음과 같습니다:

- Overheat Limit - overheat_limit
- Overcurrent Limit - overcurrent_limit
- Overcurrent Delay - overcurrent_delay
- Peak Current Ratio - peak_current_ratio
- Overvoltage Limit - overvoltage_limit
- Undervoltage Limit - undervoltage_limit
- Motor Stall Detection - stall_detection
- Velocity Error Detection - vel_error_detection
- Position Error Detection - pos_error_detection

이 오브젝트들에 대한 설명은 "11.6 모터 구동 한계 설정"과 "11.7 모터 구동오류 감지조건 설정"을 참조하기 바랍니다.

※주의※ 사용자는 사용하고자 하는 제어기의 매뉴얼와 스펙을 잘 숙지한 후 과열 온도, 과전류, 과전류 흐르는 시간, 제어기의 FET가 흘릴 수 있는 순간 최대 전류범위, 과전압, 저전압 한계 값을 적용해 주시기 바랍니다.

※주의※ 제어기의 스펙 이상의 값이 설정되면 제어기가 파손될 위험이 있으니 주의하기 바랍니다.

16.8.11 Motors – Operations

Operations는 제어기 시작 시 모터를 어떻게 구동할지를 설정할 수 있습니다. 제어기 시작 시 자동으로 POWER ON, 모터 POWER OFF 시 브레이크 활성화하기 위한 지연시간, 모터 회전방향 등 모터 구동과 관련된 모든 설정을 할 수 있습니다.

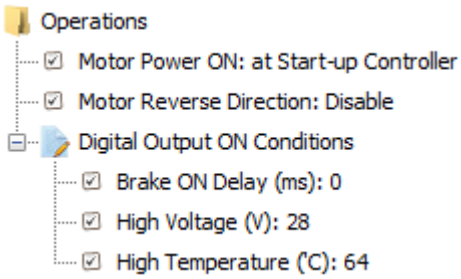

그림 16-21 Operations

이 그룹의 항목과 관련되는 오브젝트는 다음과 같습니다:

- Motor Power ON - startup_power_on
- Motor Reverse Direction - direction
- Brake ON Delay - brake_on_delay
- High Voltage - high_voltage
- High Temperature - high_temperature

이 오브젝트들에 대한 설명은 "11.8 모터 및 I/O 구동관련 설정"을 참조하기 바랍니다.

16.8.12 Motors – Closed Loop Controller

Closed Loop Controller는 제어기 피드백 센서 선택, 위치, 속도 제어 기능에서 사다리꼴 프로파일 적용여부 그리고 전류, 속도, 위치 제어기의 PID 이득, 안티와인드업 이득을 설정할 수 있습니다.

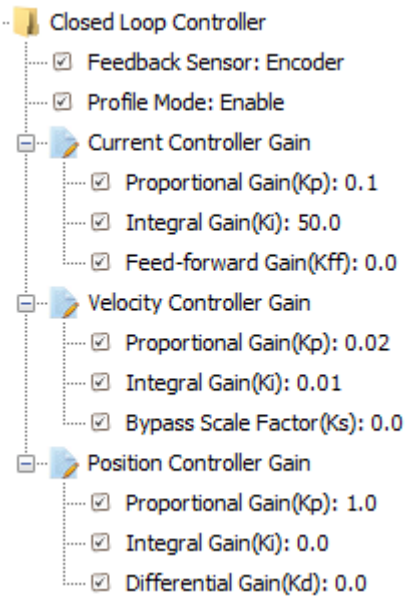

그림 16-22 Closed Loop Controller

이 그룹의 항목과 관련되는 오브젝트는 다음과 같습니다:

- Feedback Sensor - feedback_sensor
- Profile Mode - profile_mode
- Current Controller Gain - Proportional Gain(Kp) - cc_kp
- Current Controller Gain - Integral Gain(Ki) - cc_ki
- Current Controller Gain – Feed-forward Gain(Kff) - cc_kff
- Velocity Controller Gain - Proportional Gain(Kp) - vc_kp
- Velocity Controller Gain - Integral Gain(Ki) - vc_ki
- Velocity Controller Gain – Bypass Scale Factor(Ks) - vc_ks
- Position Controller Gain - Proportional Gain(Kp) - pc_kp
- Position Controller Gain - Integral Gain(Ki) - pc_ki
- Position Controller Gain - Differential Gain(Kd) - pc_kd

이 오브젝트들에 대한 설명은 "11.9 폐루프 제어 설정", "11.10 위치 제어기 이득 설정", "11.11 속도 제어기 이득 설정", "11.12 전류 제어기 이득 설정"을 참조하기 바랍니다.

16.8.13 User Values

User Value는 주로 제어기와 관련된 값들을 제어기에 저장해 두기 위해 사용하며, 총 32개의 변수 값을 설정할 수 있습니다.

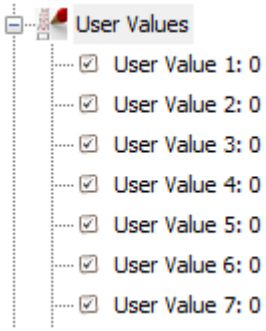

그림 16-23 User Values

User Value는 Temp value와 달리 플래시 메모리에 저장하면 제어기의 전원이 꺼지더라도 값이 유지됩니다.

이 그룹의 항목과 관련되는 오브젝트는 다음과 같습니다:

- User Value - user_value

16.8.14 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.

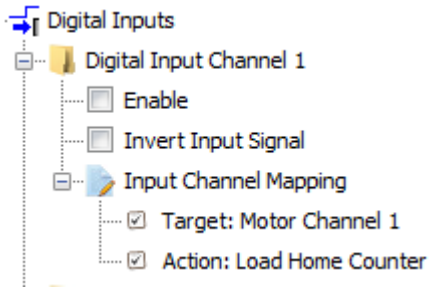

그림 16-24 Digital Inputs

이 그룹의 항목과 관련되는 오브젝트는 다음과 같습니다:

- Enable - di_enable
- Invert Input Signal - di_invert
- Target, Action - di_function

이 오브젝트들에 대한 설명은 "12.1 디지털 입력"과 "12.2 디지털 입력 채널"을 참조하기 바랍니다.

16.8.15 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.

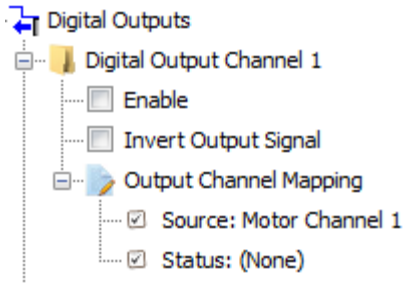

그림 16-25 Digital Outputs

이 그룹의 항목과 관련되는 오브젝트는 다음과 같습니다:

- Enable - do_enable
- Invert Output Signal - do_invert
- Source, Status - do_function

이 오브젝트들에 대한 설명은 "12.3 디지털 출력"과 "12.4 디지털 출력 채널"을 참조하기 바랍니다.

16.8.16 Analog Inputs

Analog Input은 모터의 구동에 필요한 정보 및 명령을 아날로그 신호로 받아들이기 위해 사용합니다. 아날로그 입력 활성화, 아날로그 입력 극성 반전 활성화, Linearity 설정, Mapping 기능, Calibration을 설정할 수 있습니다.

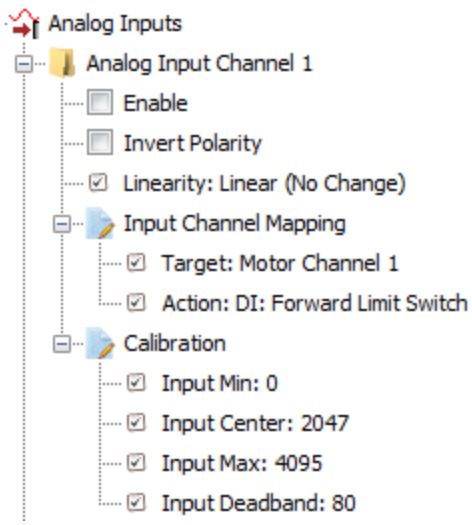

그림 16-26 Analog Input

이 그룹의 항목과 관련되는 오브젝트는 다음과 같습니다:

- Enable - ai_enable
- Invert Polarity - ai_invert
- Linearity - ai_linearity
- Target, Action - ai_function
- Input Min - ai_input_min
- Input Center - ai_input_center
- Input Max - ai_input_max
- Input Deadband - ai_input_deadband

이 오브젝트들에 대한 설명은 "12.5 아날로그 입력"과 "12.6 아날로그 입력 채널"을 참조하기 바랍니다.

16.8.17 Pulse Inputs

Pulse Input은 RC 조종기 및 디지털 신호를 받아들이기 위해 사용합니다. 펄스 입력 활성화, 펄스 입력 극성 반전 활성화, Capture Type 설정, Linearity 설정, Mapping 기능, Calibration을 설정할 수 있습니다.

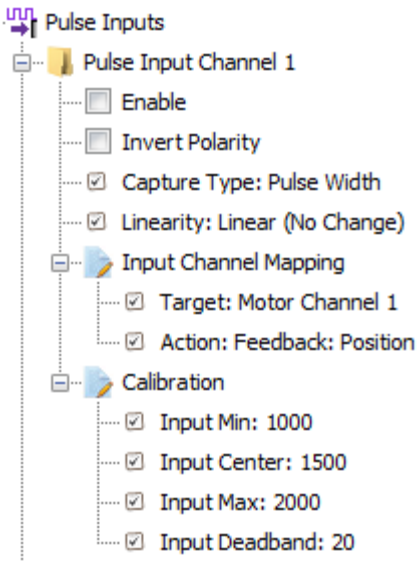

그림 16-27 Pulse Input

이 그룹의 항목과 관련되는 오브젝트는 다음과 같습니다:

- Enable - pi_enable
- Invert Polarity - pi_invert
- Capture Type - pi_capture_type
- Linearity - pi_linearity

- Target, Action - pi_function
- Input Min - pi_input_min
- Input Center - pi_input_center
- Input Max - pi_input_max
- Input Deadband - pi_input_deadband

이 오브젝트들에 대한 설명은 "12.7 펄스 입력"과 "12.8 펄스 입력 채널"을 참조하기 바랍니다.

16.8.18 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.

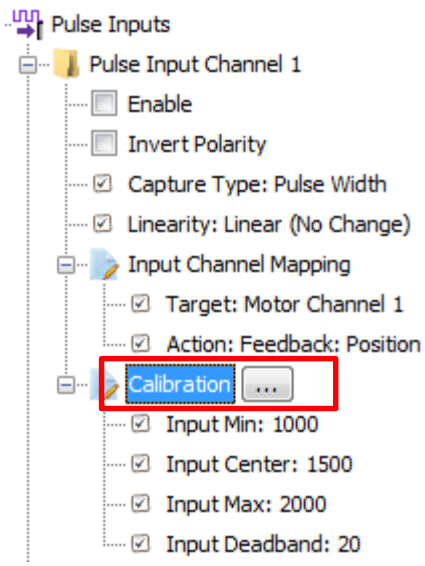

그림 16-28 Pulse Input - Calibration

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.

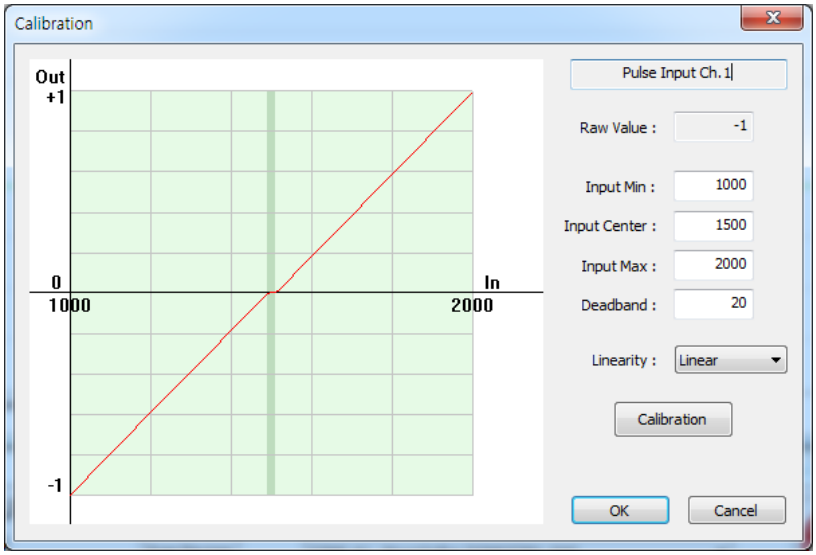

그림 16-29 Calibration - Dead-band

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.

※ 사용자는 캘리브레이션을 하기 전에 먼저 사용하고자 하는 아날로그 입력 포트와 펄스 입력 포트를 활성화(Enable)한 후 설정을 플래시 메모리에 저장([Save to Flash] 버튼 클릭)해야 합니다. 그리고 난 후 캘리브레이션을 진행해야 합니다.

※주의※ 캘리브레이션을 진행하기에 앞서 해당 채널에 연결된 모터는 Power OFF 상태에 있어야 합니다. 만일 Power ON 상태에 있다면 캘리브레이션 도중 모터가 갑자기 회전하여 위험한 상황을 초래할 수 있습니다.

16.9 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.

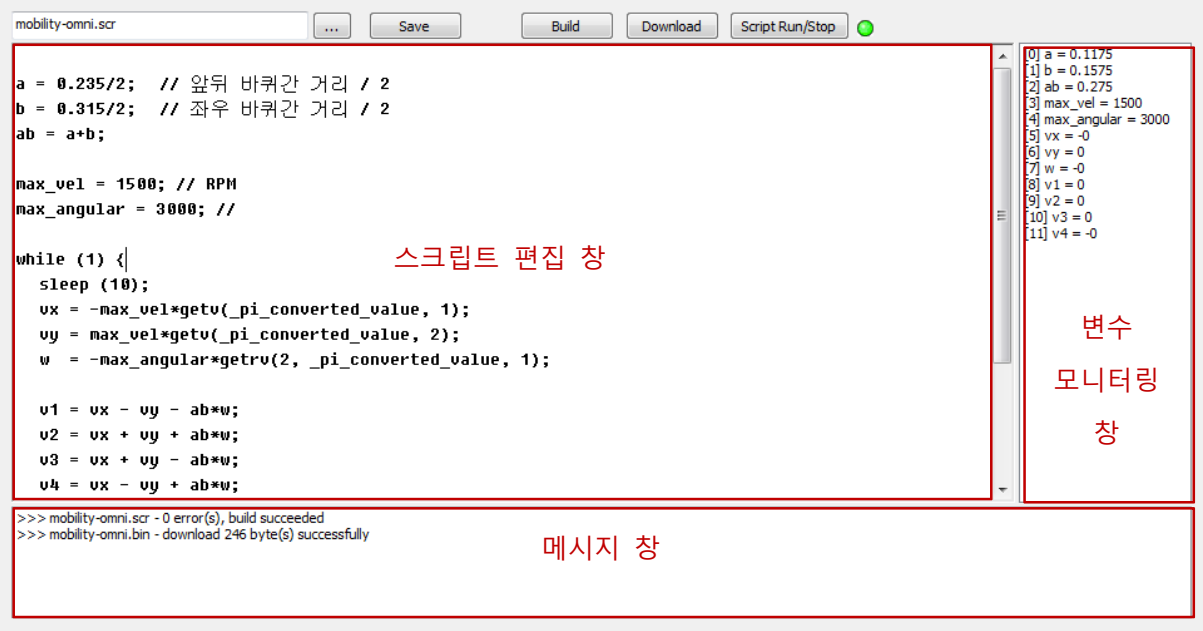

그림 16-30 Script 탭 화면

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

16.9.1 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.

16.9.2 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 다

운로드 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.

16.9.3 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.

16.10 제어기 펌웨어 업데이트

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

16.10.1 펌웨어 다운로드

펌웨어를 업데이트하기 위해서는 먼저 엔티렉스 연구소 홈페이지에서 최신 펌웨어를 다운로드 받아야 합니다.

- 다운로드 경로: http://www.ntrexgo.com/archives/23054

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

16.10.2 펌웨어 업데이트

Firmware Update 대화상자는 Motor Control UI 유틸리티 헤더의 [Firmware Update] 버튼을 눌러 실행합니다. 만일 현재 제어기와 연결되어 있다면, 그림 16-31과 같이 연결된 제어기의 Device ID, Vender ID, Product ID, H/W Version, S/W Version 정보를 표시합니다. 제어기 정보를 확인하고 [Next] 버튼을 눌러 다음 스탭으로 이동하거나 [Cancel] 버튼을 눌러 펌웨어 업데이트를 취소할

수 있습니다.

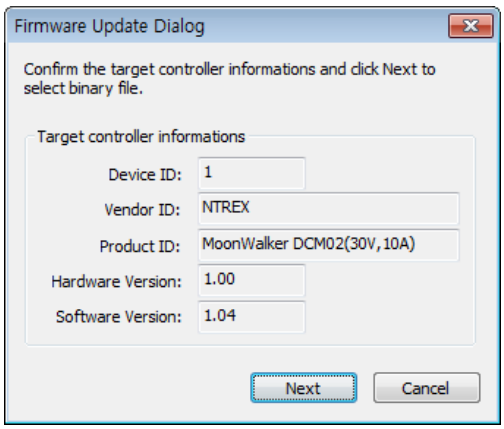

그림 16-31 Firmware Update - Target controller information 창 화면

※ 펌웨어 업데이트는 USB와 RS-232 연결로만 가능합니다. CAN 연결로 펌웨어 업데이트를 할 수 없습니다.

현재 제어기 버전을 확인한 후 [Next] 버튼을 누르면 그림 16-32와 같이 업데이트할 펌웨어를 선택할 수 있는 대화상자가 표시됩니다. 만일 현재 Motor Control UI 유틸리티가 제어기와 연결되어 있지 않다면, 그림 16-31의 대화상자는 표시되지 않고 바로 아래 그림의 대화상자가 제일 먼저 표시됩니다.

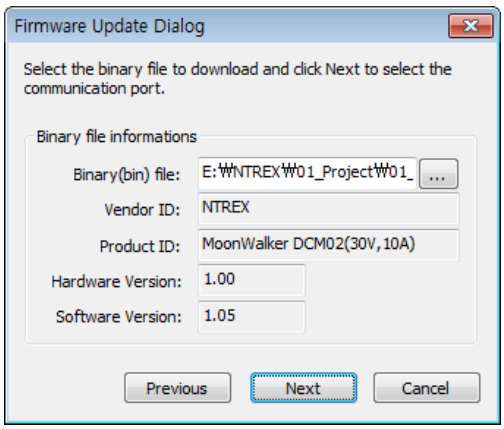

그림 16-32 Firmware Update - Binary file information 창 화면

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 16-33과 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.

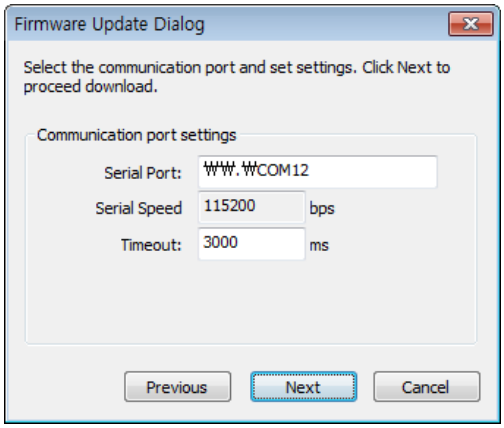

그림 16-33 Firmware Update – Communication port settings 창 화면

여기서는 제어기에 펌웨어를 다운로드 할 시리얼 포트와 와 통신속도, 패킷 전송 Timeout을 설정합니다. 그리고 [Next] 버튼을 누르면 그림 16-34와 같이 펌웨어를 다운로드 하기위한 대화상자가 표시됩니다.

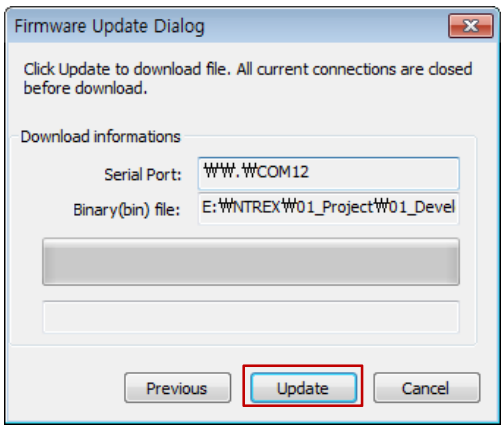

그림 16-34 Firmware Update – Download information 창 화면

여기서 다시한번 펌웨어를 다운로드 하기위한 시리얼 포트와 펌웨어 파일을 확인합니다. 마지막으로 [Update] 버튼을 누르면 기존의 모든 제어기 연결이 종료되고 그림 16-35과 같이 최신 펌웨어 다운로드가 시작됩니다.

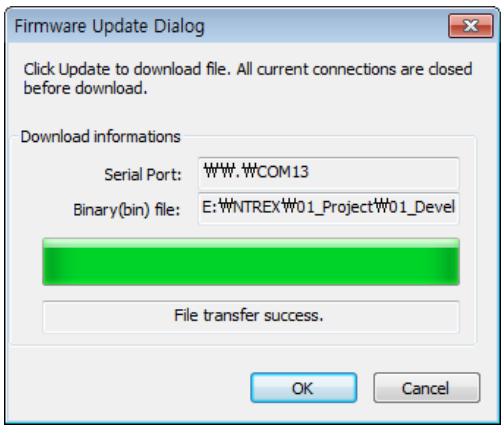

그림 16-35 펌웨어 다운로드 화면

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.

16.10.3 펌에어 업데이트 확인

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

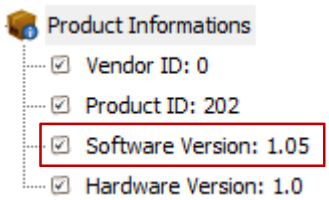

그림 16-36 업데이트 된 버전 화인

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

17 관련 자료

아래 홈페이지에서 MoonWalker 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://devicemart.co.kr/goods/brand.php?seq=1475

- MoonWalekr UI Utility:
  http://www.ntrexgo.com/archives/19482

- MoonWalker 자료:
  http://www.ntrexgo.com/archives/category/ntrex-lab/moonwalker

- MoonWalker 예제:
  http://www.ntrexgo.com/archives/category/ntrex-lab/moonwalker_applications

- MoonWalker 액세서리:
  http://www.ntrexgo.com/archives/category/moonwalker_accessory

18 문서 변경 이력

Data	Version	Charges
2013. 10. 08	1.00	- 첫 출시
2013. 11. 04	1.02	- 상세 내용 수정 및 추가
2013. 11. 15	2.00	- 상세 내용 수정 및 추가
2014. 01. 13	2.20	- 상세 내용 수정 및 추가 - Motor Control UI Program이 Motor Control UI Utility로 변경 - Motor Control UI Utility 업그레이드 내용 추가

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