

MW-SPM36D200S



User's Manual

©Copyright NTREXLAB



※ 사용하시기 전에

- MoonWalker SPM36D200S 드라이버를 구입해 주셔서 감사합니다.
- 사용자 설명서에는 SPM36D200S 드라이버의 주의사항, 제품사양, 취급방법 등이 기재되어 있습니다.
- 사용자 설명서를 잘 읽어보신 후 SPM36D200S 드라이버를 안전하게 사용하여 주십시오.
- 사용자 설명서는 사용하는 사람이 언제든지 볼 수 있는 위치에 잘 보관해 주기 바랍니다.

※ 일반 주의사항

- 사용자 설명서에 포함된 정보는 정확하고 신뢰성이 있는 내용입니다. 그러나 출판 당시 발견되지 않은 오류가 있을 수 있으니 사용자는 자신의 제품 검증을 수행하시기 바라며, 전적으로 사용자 설명서에 포함된 정보에 의존하지 마시기 바랍니다.
- 사용자가 임의로 제품을 개조하는 것은 당사의 보증 범위 밖이므로 책임지지 않습니다.

※ 안전 주의사항

- 드라이버의 설치, 점검, 보수 전에는 반드시 사용자 설명서를 숙지하신 후 실시하여 주시기 바랍니다.
- 드라이버의 설치, 점검, 보수시에는 적합한 자격을 가진 사람이 실시하여 주십시오.
- 드라이버가 손상되어 있거나 또는, 부품이 빠져 있는지 확인하십시오. 비정상적인 제품을 설치하거나 운전 시 기계 파손 또는, 사용자 부상의 위험이 있습니다.
- 운반 시에는 충분히 주의하시고 제품을 던지지 마십시오. 이런 행동은 제품의 오작동을 유발합니다.
- 드라이버를 취급할 장소에서는 불연물을 사용해 주십시오.
- 드라이버의 주변에 폭발성 물질이나, 인화성 가스가 있는 장소, 부식성 분위기, 물이 닿을 가능성이 있는 장소, 가연성 물질이 있는 부근에서는 사용을 피하여 주십시오.
- 드라이버의 정격 (전압 또는 전력) 이내의 범위에서 사용하시기 바랍니다.
- 드라이버의 전원 입력 전압이 OFF 되어 있는 것을 확인한 후 작업하여 주십시오.
- 드라이버의 배선 접속은 배선도에 따라 정확하게 실시하여 주십시오.
- 드라이버에 전원을 투입할 때에는 드라이버의 제어 입력을 모두 OFF로 한 후에 투입하여 주십시오.
- 드라이버를 비 정상적으로 전원을 차단 하였을 경우, 다시 전원 인가 시 전체적인 설정 상태를 재 확인 하시기 바랍니다. 갑작스런 모터 기동으로 인한 상해의 위험이 있습니다.
- 드라이버의 보호 기능이 작동하면 원인을 제거한 후 보호 기능을 해제하여 주십시오.
- 모든 배선장치의 연결 또는 해제 시 전원이 꺼져 있는지 확인 하시기 바랍니다.
- 통전 중에 배선 변경을 하지 마십시오.
- 보조 제동장치가 설치되지 않은 조건에서 수직 상하 운동(Z축)에 사용하지 마십시오.
- 여러대의 드라이버를 작고 좁은 밀폐된 공간에 설치할 경우, 냉각팬 등을 설치하여 주위 온도를 50℃ 이하가 되도록 하십시오.

1	전원 연결시 주의사항	1
1.1	전원 스위치 및 비상 정비 버튼의 사용	1
1.2	파워서플라이 사용시 주의사항	1
1.3	배터리 충전시 주의사항	2
1.4	전기 노이즈 감소 방법	2
2	모터 사양 및 크기	3
2.1	BLDC 시리즈	3
2.1.1	BLDC 모터 시리즈 명명법	3
2.1.2	BLDC 모터 사양	3
2.1.3	BLDC 모터 크기	4
2.1.4	BLDC 모터 & 엔코더 배선	4
2.1.5	BLDC 모터 토크 특성	5
2.2	AC 시리즈	6
2.2.1	AC 모터 시리즈 명명법	6
2.2.2	AC 모터 사양	6
2.2.3	AC 모터 크기	6
2.2.4	AC 모터 & 엔코더 배선	7
2.2.5	AC 모터 토크 특성	7
2.3	STEP 시리즈	8
2.3.1	STEP 모터 시리즈 명명법	8
2.3.2	STEP 모터 사양	8
2.3.3	STEP 모터 크기	8
2.3.4	STEP 모터 & 엔코더 배선	9
2.3.5	STEP 모터 토크 특성	9
3	드라이버 주요 특징	10
4	드라이버 사양 및 크기	11
4.1	드라이버 사양	11

4.2	드라이버 크기	12
5	드라이버 연결 및 배선	13
5.1	드라이버 연결시 주의사항	13
5.2	드라이버 외부 구성	13
5.3	커넥터 핀맵	15
5.3.1	Power	15
5.3.2	Motor	15
5.3.3	Hall sensor/Encoder	15
5.3.4	I/O	16
5.3.5	RS-232	16
5.4	드라이버 외부 배선도	17
5.5	드라이버 외부 LED	18
5.5.1	상태 표시 LED 기능	18
5.5.2	Running LED 기능	18
5.5.3	Fault LED 기능	18
5.6	통신 연결 구성	19
6	시스템 구성	19
6.1	드라이버 & 모터 & 모션제어기 구성	19
6.2	드라이버 & 모터 & 액추에이터 구성	20
7	MOTOR CONTROL UI	21
7.1	소프트웨어 다운로드 및 실행	21
7.1.1	시스템 요구사항	21
7.1.2	다운로드	21
7.1.3	실행	21
7.2	메인 화면 구성	22
7.2.1	헤더	23

7.2.2	탭 윈도우	23
7.2.3	메시지 표시	24
7.3	Chart Recorder 창	24
7.4	Monitor 창	26
7.4.1	Status/Fault 탭	26
7.4.2	I/O 탭	28
7.4.3	Variables 탭	33
7.4.4	Etc. 탭	34
7.5	Probe 창	35
7.6	연결	36
7.6.1	RS-232/485 연결	36
7.6.2	Ethernet 연결	37
7.7	Motor Control 탭	39
7.7.1	Status 그룹	40
7.7.2	Motor Control Inputs 그룹	40
7.7.3	Serial Control 그룹	41
7.7.4	Jog 그룹	43
7.7.5	Motion Test 그룹	43
7.8	Configuration 탭	43
7.8.1	모터 드라이버 오토 튜닝	45
7.8.2	세부 튜닝	58
7.8.3	PID 제어기 이해	60
7.8.4	수동 Gain 조정	62
7.8.5	Product Information	63
7.8.6	Start-up/Stop	63
7.8.1	Connection – Serial port	64
7.8.2	Connection – Ethernet	65
7.8.3	Pulse Input	65
7.8.4	Position Sensors	66
7.8.5	Motors & Mechanics	67
7.8.6	Motors & Mechanics – Motion Profile	68

7.8.7	Motors & Mechanics – Electrical Parameters	68
7.8.8	Motors & Mechanics – Mechanical Parameters	69
7.8.9	Controller – Fault Protection – Fault Limits	70
7.8.10	Controller – Fault Protection - Fault Detection	70
7.8.11	Controller – Feedforward Controller	71
7.8.12	Controller – Feedback Controller	72
7.8.13	Controller – I/O Parameters	73
7.9	Script Edit/Run 탭	74
7.9.1	Script 작성	75
7.9.2	빌드 및 다운로드	75
7.9.3	실행 및 디버깅	75
7.10	모터 드라이버 펌웨어 업데이트	76
7.10.1	펌웨어 다운로드	76
7.10.2	펌웨어 업데이트	76
8	통신 프로토콜	80
8.1	Communication Protocol	80
8.1.1	용어의 정리	80
8.2	CAN 메시지	81
8.2.1	CAN 패킷의 기본 구조	82
8.2.2	오브젝트 읽기 요청	83
8.2.3	오브젝트 쓰기 요청	83
8.2.4	오브젝트 읽기/쓰기 요청에 대한 성공 응답	84
8.2.5	오브젝트 읽기/쓰기 요청에 대한 실패 응답	84
8.3	시리얼 바이너리 패킷	85
8.3.1	바이너리 패킷의 기본 구조	85
8.3.2	오브젝트 읽기 요청	86
8.3.3	오브젝트 쓰기 요청	86
8.3.4	오브젝트 읽기/쓰기 요청에 대한 성공 응답	86
8.3.5	오브젝트 읽기/쓰기 요청에 대한 실패 응답	87
8.4	시리얼 텍스트 패킷	87

8.4.1	오브젝트 읽기 요청	88
8.4.2	오브젝트 쓰기 요청	88
8.4.3	오브젝트 읽기/쓰기 요청에 대한 성공 응답	89
8.4.4	오브젝트 읽기/쓰기 요청에 대한 실패 응답	89
8.4.5	Device ID의 부여	89
8.4.6	여러 오브젝트의 읽기/쓰기	90
9	오브젝트	91
9.1	Object 설명	91
9.2	오브젝트의 기본 단위	91
9.3	오브젝트 테이블	93
9.4	Product Information	97
9.4.1	vendor_id	97
9.4.2	product_id	97
9.4.3	software_version	98
9.4.4	hardware_version	98
9.4.5	units	98
9.5	Startup/Stop	99
9.5.1	startup_drive_input	99
9.5.2	startup_control_mode	99
9.5.3	startup_enable_delay	100
9.5.4	script_stop_action	100
9.5.5	disable_oper_action	100
9.6	System Parameters	101
9.6.1	system_status	101
9.6.2	system_control	102
9.6.3	system_argument	105
9.6.4	power_source_voltage	106
9.6.5	power_source_current	106
9.7	Connection	106
9.7.1	device_name	106

9.7.2	device_id	107
9.7.3	serial_watchdog	107
9.7.4	serial_baudrate	107
9.8	Pulse/Direction Input	108
9.8.1	pulse_input_mode	109
9.8.2	pulse_count_edge	109
9.8.3	dir_signal_polarity	110
9.8.4	gear_numerator	110
9.8.5	gear_denominator	110
9.9	Variables	110
9.9.1	user_variable	110
9.9.2	temp_variable	111
9.10	Error Code	111
9.10.1	error_code	111
9.11	Motor Status	114
9.11.1	control	114
9.11.2	status	115
9.11.3	fault	116
9.11.4	temperature	118
9.11.5	load_torque	118
9.12	Motor Drive	118
9.12.1	target_position	118
9.12.2	target_velocity	119
9.12.3	target_current_d/q	119
9.12.4	target_voltage_d/q	119
9.12.5	torque_limit	119
9.12.6	position_demand	120
9.12.7	velocity_demand	120
9.12.8	current_demand_d/q	120
9.12.9	voltage_demand_d/q	121
9.12.10	acceleration_demand	121
9.12.11	position_actual	121

9.12.12	velocity_actual	121
9.12.13	current_actual_d/q	121
9.12.14	acceleration_actual	121
9.12.15	position_error	122
9.12.16	velocity_error	122
9.12.17	current_error_d/q	122
9.13	Position Limit	122
9.13.1	soft_limit_check	123
9.13.2	min_position	123
9.13.3	max_position	123
9.13.4	limit_action	123
9.14	Homing	124
9.14.1	homing_method	124
9.14.2	homing_velocity	128
9.14.3	home_offset	128
9.14.4	home_position	128
9.15	Rated Values	128
9.15.1	rated_voltage	128
9.15.2	rated_current	128
9.15.3	rated_velocity	129
9.15.4	max_current	129
9.15.5	max_velocity	129
9.16	Velocity Profile	129
9.16.1	profile_type	129
9.16.2	profile_velocity	130
9.16.3	profile_acceleration	131
9.16.4	profile_deceleration	131
9.16.5	profile_jerk	131
9.17	Fault Limits	131
9.17.1	overheat_limit	131
9.17.2	overcurrent_limit	132
9.17.3	overvoltage_limit	132

9.17.4	undervoltage_limit	133
9.18	Fault Detection	133
9.18.1	vibration_detect	133
9.18.2	stall_current_detect	134
9.18.3	position_track_error	134
9.18.4	velocity_track_error	135
9.19	Motor Properties	135
9.19.1	motor_type	135
9.19.2	motor_direction	136
9.20	Feed-forward Controller	137
9.20.1	feedforward_option	137
9.21	Sensor Properties	139
9.21.1	position_sensor	139
9.21.2	encoder_direction	139
9.21.3	encoder_resolution	140
9.21.4	no_pole_pairs	140
9.21.5	hallsensor_phase	141
9.21.6	linear_sensor_pitch	142
9.22	Electrical Parameters	143
9.22.1	bemf_constant	143
9.22.2	resistance	143
9.22.3	inductance_d/q	143
9.22.4	electric_angle_bias	143
9.23	Mechanical Parameters	144
9.23.1	torque_constant	144
9.23.2	moment_of_inertia	144
9.23.3	viscous_friction	144
9.23.4	coulomb_friction	144
9.23.5	load_torque_bias	145
9.24	Controller's Gain	145
9.24.1	position_p_gain	145

9.24.2	velocity_p_gain	145
9.24.3	velocity_i_gain	145
9.24.4	velocity_d_gain	145
9.24.5	current_p_gain_d/q	146
9.24.6	current_i_gain_d/q	146
9.25	Moving Average Filter	146
9.25.1	pmaf_window_size	146
9.25.2	vmaf_window_size	147
9.26	I/O Parameters	148
9.26.1	in_position_threshold	148
9.26.2	in_velocity_threshold	148
9.26.3	brake_on_delay	148
9.26.4	jog_velocity	149
9.26.5	regen_voltage_cutoff	149
9.27	I/O Common	150
9.27.1	no_digital_input	150
9.27.2	no_digital_output	150
9.27.3	no_analog_input	150
9.27.4	no_pulse_input	150
9.27.5	digital_inputs	150
9.27.6	digital_outputs	151
9.27.7	digital_outputs_mask	151
9.28	Digital Input	151
9.28.1	di_option	151
9.28.2	di_value	151
9.28.3	di_function	152
9.29	Digital Output	152
9.29.1	do_option	152
9.29.2	do_value	152
9.29.3	do_function	153
9.30	Analog Input	153

9.30.1	ai_option	153
9.30.2	ai_value	153
9.30.3	ai_function	154
9.30.4	ai_raw_value	154
9.30.5	ai_cutoff_freq	154
9.30.6	ai_cal_min, ai_cal_center, ai_cal_max	154
9.30.7	ai_cal_deadband	155
9.31	Pulse Input	155
9.31.1	pi_option	155
9.31.2	pi_value	155
9.31.3	pi_function	155
9.31.4	pi_raw_value	155
9.31.5	pi_capture_type	156
9.31.6	pi_cutoff_freq	156
9.31.7	pi_cal_min, pi_cal_center, pi_cal_max	156
9.31.8	pi_cal_deadband	156
10	MINI-C 스크립트 언어	157
10.1	스크립트 관련 구성	157
10.2	C언어와의 차이	157
10.3	문장	159
10.3.1	수식과 문장	159
10.3.2	복합문	159
10.3.3	제어문	159
10.4	주석	159
10.4.1	블록 주석문	160
10.4.2	라인 주석문	160
10.5	상수	160
10.5.1	정수형 상수	160
10.5.2	실수형 상수	161
10.5.3	미리 정의된 상수	161

10.6	변수	168
10.6.1	변수명	168
10.6.2	변수의 선언과 초기화	169
10.6.3	지역변수와 전역변수	169
10.6.4	시스템 변수	170
10.6.5	자료형 변환	175
10.7	연산자	176
10.7.1	산술, 부호 연산자	176
10.7.2	증감 연산자	176
10.7.3	관계 연산자	177
10.7.4	논리 연산자	177
10.7.5	비트 연산자	178
10.7.6	대입 연산자	179
10.7.7	연산자 우선순위	180
10.8	제어문	180
10.8.1	if 문	181
10.8.2	if-else 문	181
10.8.3	if-else-if-else 문	182
10.8.4	while 문	182
10.8.5	for 문	183
10.8.6	do-while 문	184
10.8.7	break 문	184
10.8.8	continue 문	185
10.8.9	goto 문	185
10.9	함수	186
10.9.1	함수 선언	186
10.9.2	인트럽트 함수	186
10.9.3	함수 호출	187
10.10	내장 함수	187
10.10.1	rand	188
10.10.2	int	188
10.10.3	sin	188

10.10.4	cos	189
10.10.5	tan	189
10.10.6	asin	189
10.10.7	acos	189
10.10.8	atan	190
10.10.9	sinh	190
10.10.10	cosh	190
10.10.11	tanh	190
10.10.12	fabs	191
10.10.13	floor	191
10.10.14	ceil	191
10.10.15	sqrt	192
10.10.16	exp	192
10.10.17	log	192
10.10.18	log10	193
10.10.19	atan2	193
10.10.20	pow	193
10.10.21	min	194
10.10.22	max	194
10.10.23	clock	194
10.10.24	wait	195
10.10.25	ei	195
10.10.26	di	195
10.10.27	timer0	196
10.10.28	timer1	196
10.10.29	timer2	196
10.10.30	sleep	197
10.10.31	getv	197
10.10.32	setv	197
11	프로그램의 작성과 실행	199
11.1	프로그램의 작성	199
11.1.1	소스코드 작성	199

11.1.2	소스코드 구조	199
11.2	빌드	201
11.2.1	컴파일	201
11.2.2	컴파일 오류 메시지	202
11.3	다운로드 및 실행	203
11.3.1	다운로드	203
11.3.2	실행	203
12	관련 자료	204

1 전원 연결시 주의사항

드라이버에 전원(Power Source, 배터리 또는 파워서플라이)을 연결하는 방법에 대해 설명합니다.

1.1 전원 스위치 및 비상 정비 버튼의 사용

파워서플라이와 같이 전원 스위치가 없는 배터리를 전원으로 사용할 때는, 드라이버의 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 1-1 비상 정지 버튼

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

모터가 고속으로 회전하는 동안에는 전원 스위치를 ON/OFF하지 마십시오. 드라이버가 손상될 수도 있으니 주의하시기 바랍니다.

1.2 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. 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

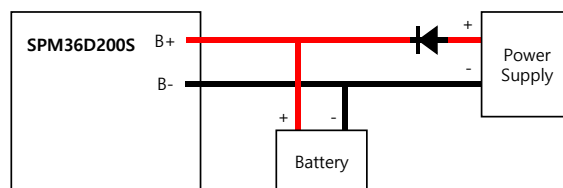


그림 1-2 파워서플라이와 배터리 병렬 연결

1.3 배터리 충전시 주의사항

충전식 배터리를 사용할 경우 드라이버 및 전원 보호 회로를 외부적으로 따로 설계한 후, 전원 스위치를 끈 상태에서 배터리를 충전해야 합니다. 만약 드라이버와 연결된 상태에서 충전할 경우, 드라이버 및 주변 회로에 배터리 전압 이상의 전압이 인가될 수 있습니다.

1.4 전기 노이즈 감소 방법

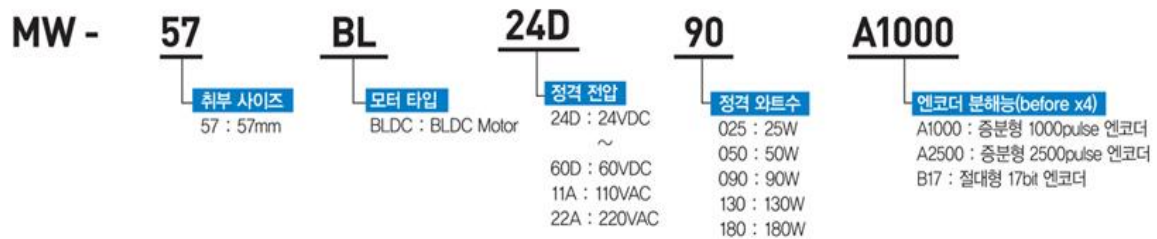
전기적 노이즈를 감소시키기 위해 드라이버 내에 여러 회로들이 설계되어 있지만, 드라이버를 모터와 전원에 연결하여 사용할 때는 노이즈를 저감하기 위한 추가적인 작업이 필요합니다. 다음은 전기적 노이즈를 감소시키는 방법들입니다:

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

2 모터 사양 및 크기

2.1 BLDC 시리즈

2.1.1 BLDC 모터 시리즈 명명법



2.1.2 BLDC 모터 사양

- 전기적 사양

Parameters	57BL24D025	57BL24D050	57BL36D090	57BL36D130	57BL36D180	57BL36D180-C
Rated Voltage (VDC)	24	24	36	36	36	36
Rated Power (W)	25	50	90	130	180	180
Rated Tprque (N.M)	0.08	0.16	0.29	0.41	0.57	0.43
Peak Torque (N.M)	0.24	0.48	0.87	1.23	1.71	1.27
Rated Speed (RPM)	3000	3000	3000	3000	3000	4000
Rated Current (A)	1.6	3	3.45	5.3	6.7	7
Peak Current (A)	4.8	9	10.35	15.9	20	20.5
Torque Const. (N.M/A)	0.05	0.053	0.084	0.078	0.085	0.063
Back EMF Const (V/RPM)	5.2	5.55	8.8	8.2	8.9	6.6
Resistance (ohms)	1.73	0.88	1.35	0.63	0.9	0.35
Inductance (mH)	3.36	2.2	4.1	2.17	2	1
Inertia (Kgm ² x 10 ⁻⁴)	30	75	119	173	230	230
Motor Length L (mm)	70	80	100	120	140	140
Mass (Kg)	0.25	0.5	0.75	1	1.25	1.25

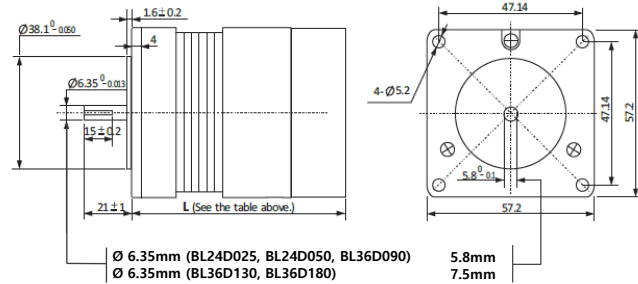
- 일반적 사양

Parameters	
Winding connection	△ Delta connection
Hall sensor	120°
Pole Pairs	2
Phase	3
Shaft Radial Play	0.025 mm
Shaft Axial Play	0.025mm@460g
Allowable radial load	75N@20mm from th flange
Allowable axial load	15N
Isolation Level	Class B
Isolation Strength	500 VDC for one minute
Isolation Resistance	100 M ohms

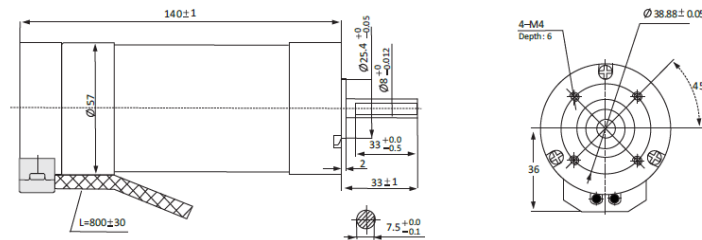
2.1.3 BLDC 모터 크기

Parameters	
Frame Size	57mm (NEMA23)

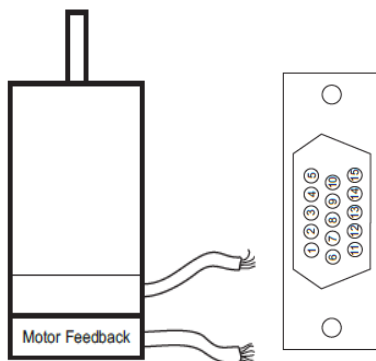
- 57BL24D025, 57BL24D050, 57BL36D090, 57BL36D130, 57BL36D180



- 57BL36D180-C



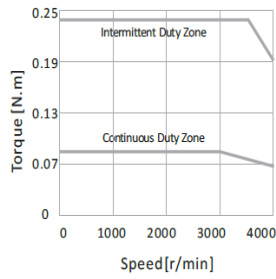
2.1.4 BLDC 모터 & 엔코더 배선



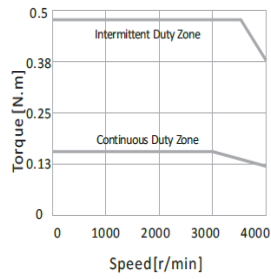
pin	Description	pin	Description	pin	Description
1	EA+	6	NC	11	EA-
2	EB+	7	NC	12	EB-
3	GND	8	NC	13	VCC
4	Hall C	9	Hall B	14	NC
5	Hall A	10	NC	15	NC

- Motor Cable :** U (BRN), V (BLU), W (BLK)
- Hall Sensor Cable :** Hall A (BRN), Hall B (GRY), Hall C (ORG), VCC+5V (RED), GND (BLK)
- Encoder Cable :** EA+ (GRN), EA- (GRN/BLK), EB+ (WHT), EB- (WHT/BLK), VCC+5V (ORN/BLK), GND (BRN/BLK)

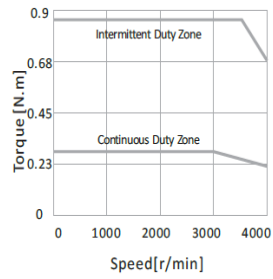
2.1.5 BLDC 모터 토크 특성



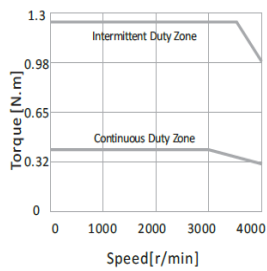
MW-57BL24D025



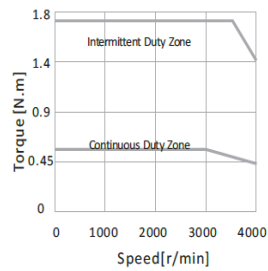
MW-57BL24D050



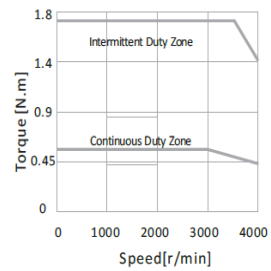
MW-57BL36D090



MW-57BL36D130



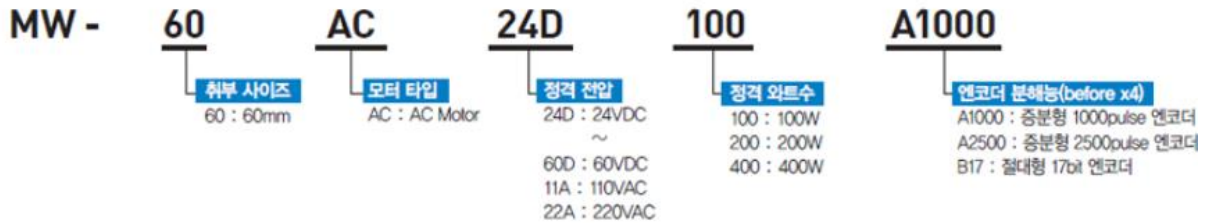
MW-57BL36D180



MW-57BL36D180-C

2.2 AC 시리즈

2.2.1 AC 모터 시리즈 명명법



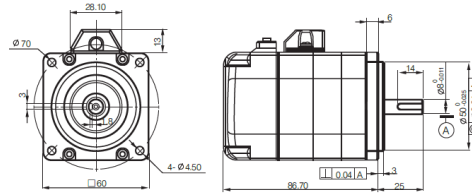
2.2.2 AC 모터 사양

Parameters	60AC36D100	60AC36D200
Rated Voltage (VDC)	36	36
Rated Power (W)	100	200
Rated Tprque (N.M)	0.318	0.64
Peak Torque (N.M)	0.95	1.91
Rated Speed (RPM)	3000	3000
Peak Speed (RPM)	4000	4000
Rated Current (A)	4	7.6
Peak Current (A)	11	22
Torque Const. (N.M/A)	0.0866	0.0918
Back EMF Const (V/RPM)	3.03×10^{-3}	3.213×10^{-3}
Resistance (ohms)	0.38	0.16
Inductance (mH)	0.91	0.41
Inertia (Kg $m^2 \times 10^{-4}$)	0.1032	0.176
Allowable radial load (N)	89.6	245
Allowable axial load (N)	38.2	68
Flange Size (mm)	60	60
Mounting Diameter (mm)	70	70
Shaft Diamter (mm)	8	11
Motor Length (mm)	86.7	100.7
Pole Pairs (-)	4	4
Encoder Res. (counts/rev.)	2500 **	2500 **
Mass (Kg)	0.701	0.966
Ambient Temperature (°C)	0 to 40	0 to 40

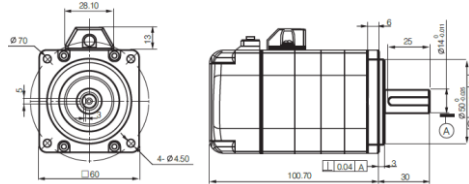
2.2.3 AC 모터 크기

Parameters	
Frame Size	60mm (NEMA24)

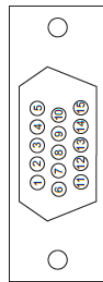
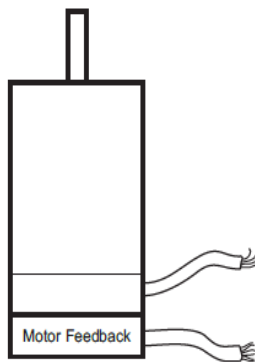
- 60AC36D100



- 60AC36D200



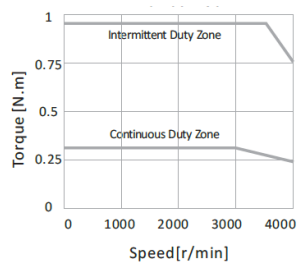
2.2.4 AC 모터 & 엔코더 배선



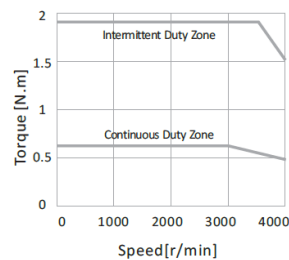
Pin	Description	Pin	Description	pin	Description
1	EA+	6	Shield	11	EA-
2	EB+	7	NZ+	12	EB-
3	GND	8	NZ-	13	VCC
4	Hall W+	9	Hall V+	14	Hall W-
5	Hall U+	10	Hall V-	15	Hall U-

- **Motor Cable** : U (RED), V (BLU), W (BLK), Shield (YEL)
- **Hall Sensor Cable** : Hall U+ (BRN), Hall U- (BRN/BLK), Hall V+ (GRY), Hall V- (GRY/BLK), Hall W+ (ORG), Hall W- (ORG/BLK), VCC+5V (RED), GND (BLK)
- **Encoder Cable** : EA+ (GRN), EA- (GRN/BLK), EB+ (WHT), EB- (WHT/BLK), EZ+ (YEL), EZ- (YEL/BLK), VCC+5V (RED), GND (BLK)

2.2.5 AC 모터 토크 특성



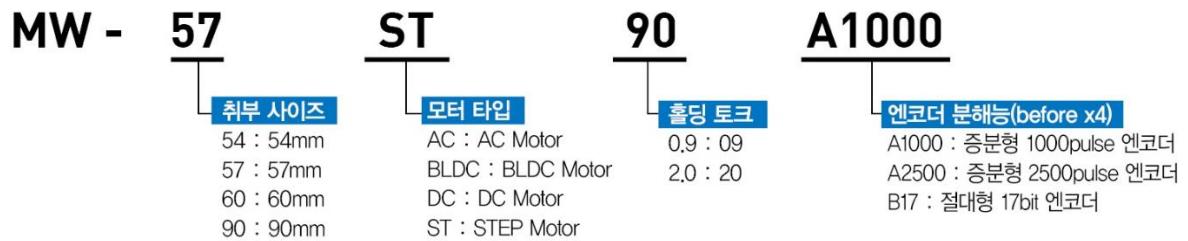
MW-60AC36D100



MW-60AC36D200

2.3 STEP 시리즈

2.3.1 STEP 모터 시리즈 명명법



2.3.2 STEP 모터 사양

• 모터 사양

Parameters	57ST09-A1000	57ST20-A1000
Phase	3	3
Step Angle (°)	1.2	1.2
Leads	3	3
Holding Torque (N.m)	0.9	2.0
Phase Current (A)	5.8	5.8
Phase Resistance (Ohm)	0.37	0.62
Phase Inductance (mH)	0.92	1.85
Rotor Inertia (kg.cm ²)	0.3	0.5
Weight (Kg)	0.9	1.35
Shaft Diameter (mm)	8	8

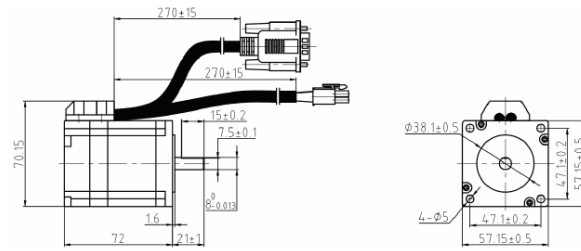
• 엔코더 사양

Parameters	
Supply Voltage (VDC)	4.5 to 5.5 (Typ. 5.0)
Output Current per Channel (mA)	-1 to 5
Low Level Output Voltage (VDC)	Max 0.4
High Level Output Voltage (VDC)	Min 2.4
Count Frequency (kHz)	Max 100

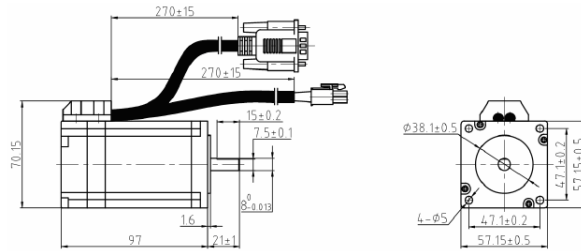
2.3.3 STEP 모터 크기

Parameters	
Frame Size	57mm (NEMA23)

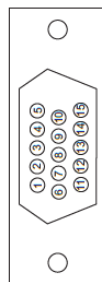
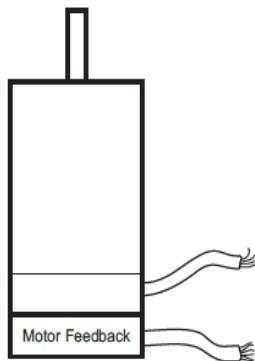
- 57ST09-A1000



- 57ST20-A1000



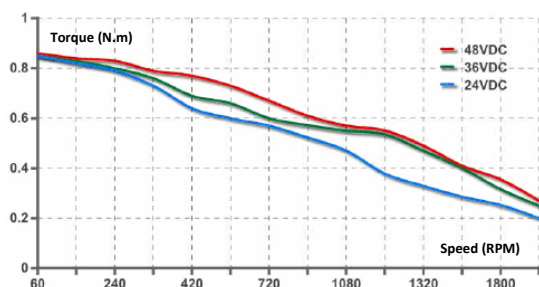
2.3.4 STEP 모터 & 엔코더 배선



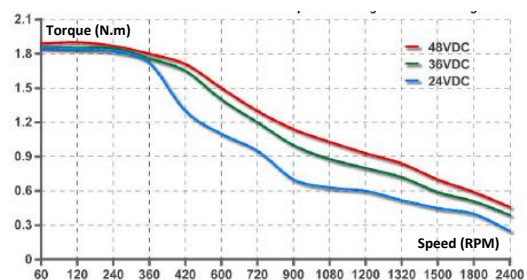
Pin	Description	Pin	Description	Pin	Description
1	EA+	6	NC	11	EB+
2	VCC	7	NC	12	EB-
3	GND	8	NC	13	EA-
4	NC	9	NC	14	NC
5	NC	10	NC	15	NC

- Motor Cable** : U (BRN), V (BLU), W (BLK)
- Encoder Cable** : EA+ (BLK), EA- (BLU), EB+ (YEL), EB- (GRE), VCC+5V (RED), GND (WHT)

2.3.5 STEP 모터 토크 특성



57ST09-A1000



57ST20-A1000

3 드라이버 주요 특징

- BLDC, STEP(3상)모터 통합 제어용 드라이버
- PC 기반의 User Interface를 사용하여 제어 파라미터 설정 및 구동
- 시리얼 명령어를 통한 위치, 속도, 전류, 전압 지령
- 사다리꼴 과 Sine 프로파일 을 사용한 위치&속도 구동
- PID 피드백 위치 제어기 사용. Anti-Wind-UP 기능 포함
- 모터의 전기적 파라미터 (L, R, Ke) 탐지 및 전류제어기 이득 자동 설정 기능 탑재
- 모터의 전기적 & 기계적 모델로부터 Feed Forward 제어 기능 사용
- 모터의 기계적 파라미터 (J, B, TL) 탐지 및 속도&위치 제어기 이득 자동 설정 기능 탑재
- FOC (전류벡터 제어) 방식 제어를 통한 저소음, 저 리플 구동
- Notch Filter 내장으로 부하 (기구부) 진동 저감
- 펄스 입력 방식과 RS-232 통신 방식에 의한 모터 제어
- 1펄스 (Pulse, Direction) 방식과, 2펄스 (CW, CCW) 방식 중 선택 구동
- 모터의 단락, 과전압, 저전압, 과온도 등 상태 감지에 의한 알람 발생 기능



4 드라이버 사양 및 크기

4.1 드라이버 사양

전기적 데이터	
정격 전압	+24 ~ +36 VDC
최소 전압	+12 VDC
최대 전압	+40 VDC
연속 전류	10 A
최대 전류	20 A
대기 전류	+24 V 기준 80 mA
정격 출력	200W
모터 타입	BLDC, STEP(3상)
PWM 구동 주파수	20 KHz
제어 방법	전류벡터 제어 (FOC)
전류제어 샘플링 주파수	10 KHz
속도제어 샘플링 주파수	1 KHz
위치제어 샘플링 주파수	1 KHz

전압/전류 출력	
Encoder	+5V, 120 mA
Hall Sensor	+5V, 150 mA

Position Sensor Feedback	
Encoder Type	Incremental, Line Driver or Single Ended (A, B, I)
Encoder 최대 입력 주파수	10 Mpps
Hall Sensor Type	Line Driver or Open-Collector (U, V, W)

Digital Input / Output	
Input 1	모터 온/오프 (SERVO-ON), Photo Coupler
Input 2	알람 해제 (ALARM RESET), Photo Coupler
Output 1	위치 결정 완료 (IN-POSITION), Photo Coupler
Output 2	알람 (ALARM), Photo Coupler
Output 3	BRAKE 온/오프 (N-ch MOSFET, Open Drain)

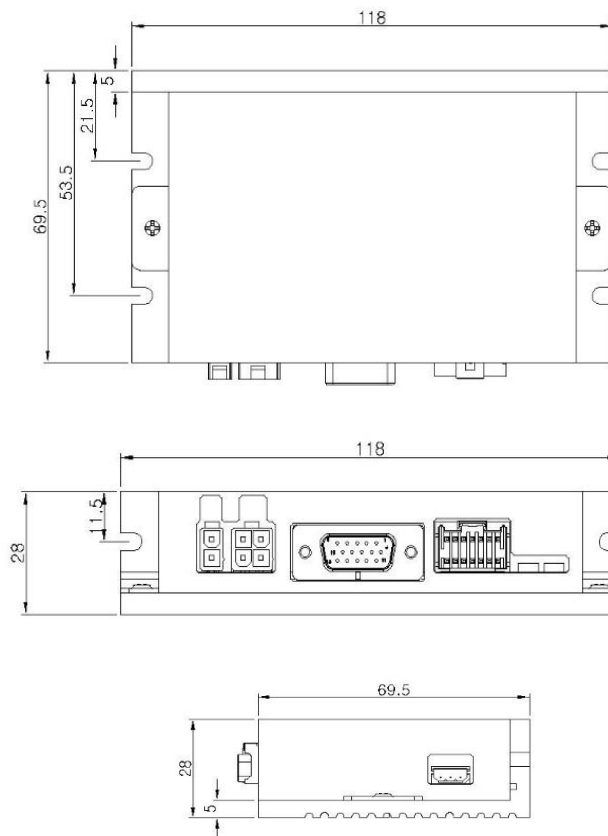
Communication	
RS-232C	PC와의 접속 – GUI 운용 타 호스트 장치와의 1 : 1 연결 (Default - 115200 bps)

상태 표시	
BLUE LED	Running
RED LED	Error / Fault

환경 조건	
동작 조건 온도	0 ~ 80 °C
보관 온도	0 ~ 50 °C
습도	40 ~ 90 % RH
방열	Heat Sink Plane

4.2 드라이버 크기

기계적 데이터		
길이 (mm)		118
너비 (mm)		69.5
높이 (mm)		28
중량 (g)		220



5 드라이버 연결 및 배선

5.1 드라이버 연결시 주의사항

- 실내에서 사용해 주시기 바랍니다.
- 실내 주의 온도는 0~50 °C에서 사용해 주시기 바랍니다. (여러대의 드라이버를 작고 좁은 밀폐된 공간에 설치할 경우, 냉각팬 등을 설치하여 주십시오.)
- 드라이버의 온도가 50 °C 이상이 되면 방열을 해주시기 바랍니다.
- 주변에 폭발성 물질이나, 인화성 가스가 있는 장소, 부식성 분위기, 물이 닿을 가능성이 있는 장소, 가연성 물질이 있는 부근에서는 사용을 피하여 주시기 바랍니다.
- 통전 중에 절대로 배선 변경을 하지 마십시오.
- 배선의 연결 또는 해제 시 전원이 꺼져 있는지 확인해주시기 바랍니다.
- 드라이버를 여러대를 나란히 설치 시에는 수직 방향으로 20mm, 수평 방향으로 50mm 이상 거리를 두고 설치해 주시기 바랍니다.

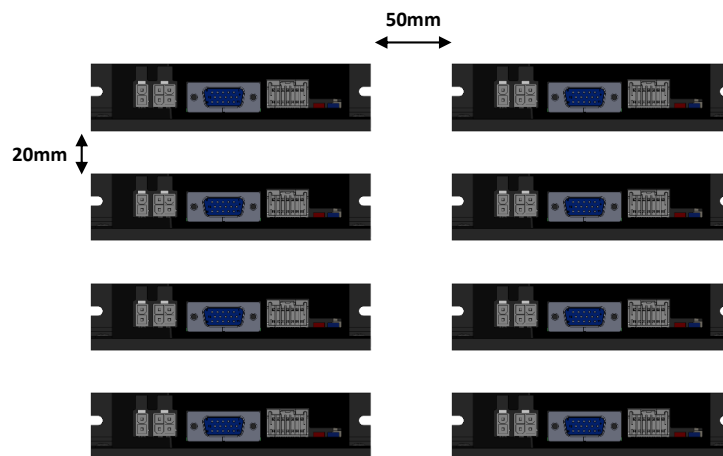


그림 5-1 드라이버 여러대 설치 시 간격

5.2 드라이버 외부 구성

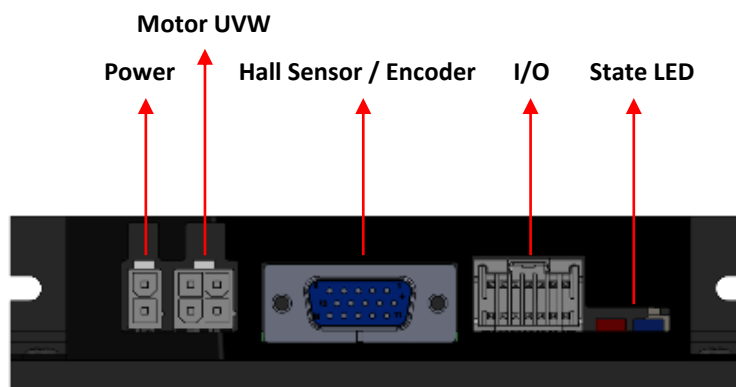




그림 5-2 드라이버 외부 구성

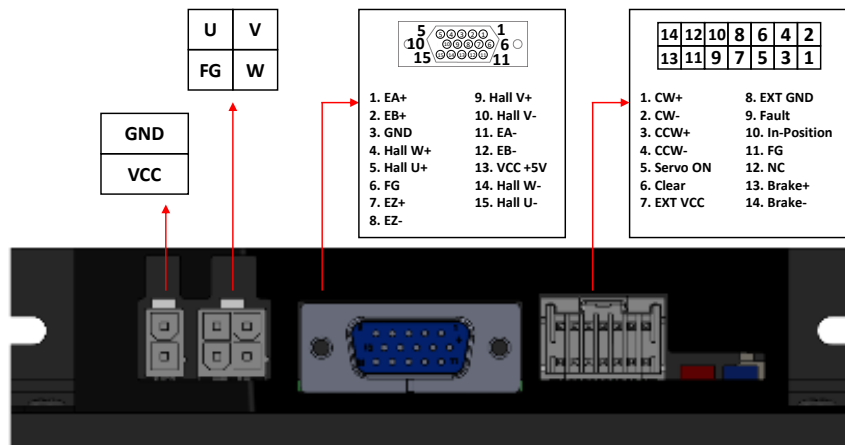




그림 5-3 드라이버 핀맵 구성

5.3 커넥터 핀맵

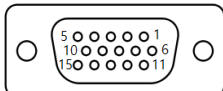
5.3.1 Power

Pin No.	Function	I/O	Pin Layout
1	Power Input : 12 ~ 48VDC	Input	
2	Power Input: GND	Input	

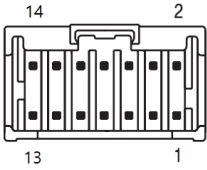
5.3.2 Motor

Pin No.	Function			I/O	Pin Layout
	DC	BLDC/PMSM/STEP(3p)	STEP(2p)		
1		Motor W	Motor B-	Output	
2			Motor A-	Output	
3	Motor -	Motor V	Motor B+	Output	
4	Motor +	Motor U	Motor A+	Output	

5.3.3 Hall sensor/Encoder

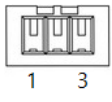
Pin No.	Function	I/O	Pin Layout
1	Encoder A+	Input	
2	Encoder B+	Input	
3	GND		
4	Hallsensor W+	Input	
5	Hallsensor U+	Input	
6	F.G.		
7	Encoder Z+	Input	
8	Encoder Z-	Input	
9	Hallsensor V+	Input	
10	Hallsensor V-	Input	
11	Encoder A-	Input	
12	Encoder B-	Input	
13	Power +5V	Output	
14	Hallsensor W-	Input	
15	Hallsensor U-	Input	

5.3.4 I/O

Pin No.	Function		I/O	Pin Layout
	Default	Extension		
1	PUL+/CW+	Digital Input 1+	Input	
2	PUL-/CW-	Digital Input 1-	Input	
3	DIR+/CCW+	Digital Input 2+	Input	
4	DIR-/CCW-	Digital Input 2-	Input	
5	Servo On	Digital Input 3	Input	
6	Clear	Digital Input 4/Pulse Input 1	Input	
7	External VCC	24V+	Input	
8	External GND	GND		
9	Fault	Digital Output 1	Output	
10	In-Position	Digital Output 2	Output	
11	F.G.			
12				
13	Brake +	Digital Output 3+	Output	
14	Brake -	Digital Output 3-	Output	

***주의:** Pulse/Direction 구동 모드에서 DI1, DI2 가 Enable로 설정되어 있다면, Enable 상태가 해제됩니다.

5.3.5 RS-232

Pin No.	Function	I/O	Pin Layout
1	TxD	Output	
2	RxD	Input	
3	GND		

5.4 드라이버 외부 배선도

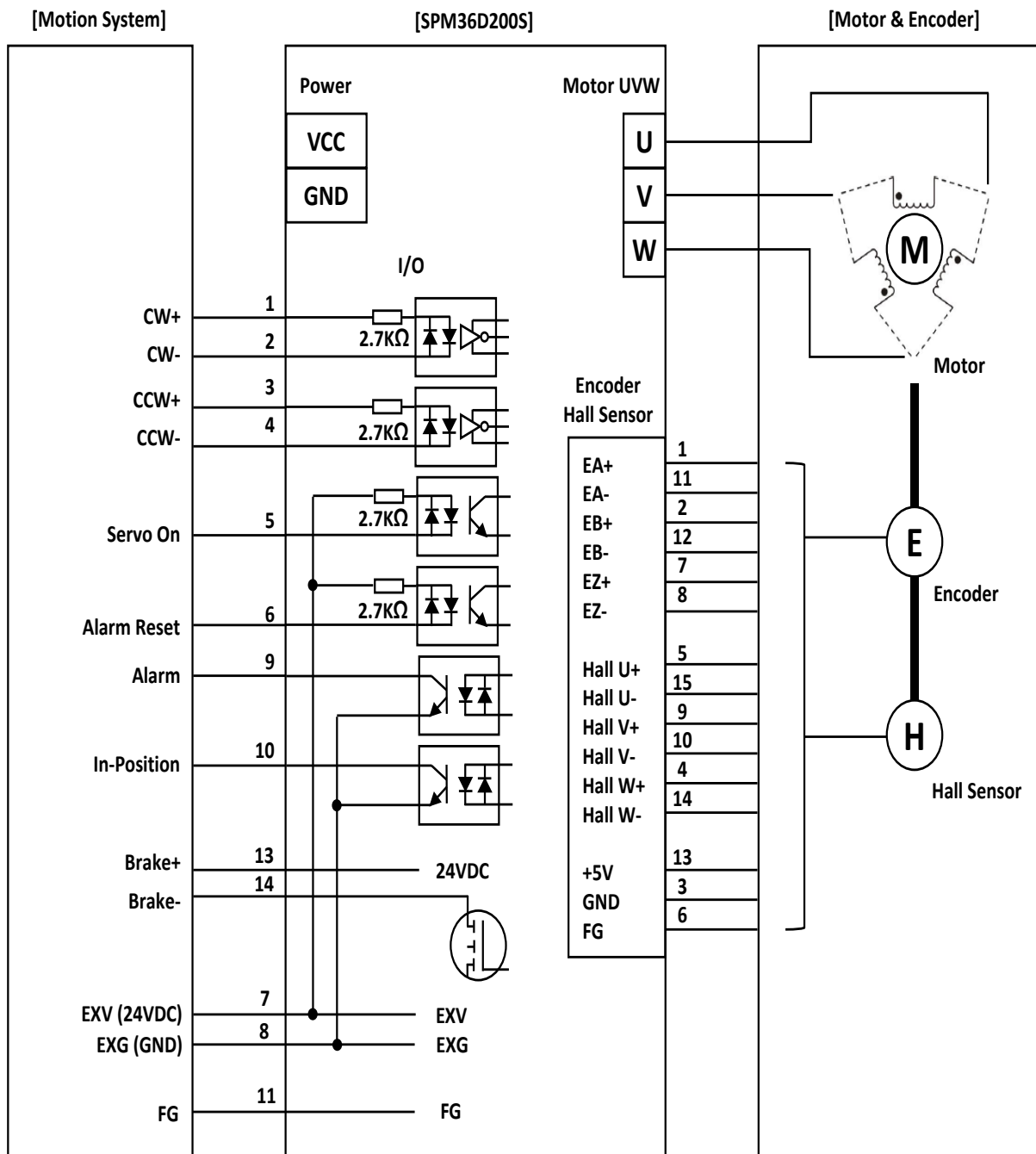


그림 5-4 SPM36D200S 외부 배선도

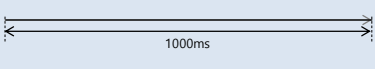
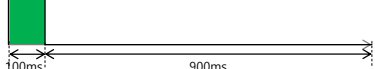
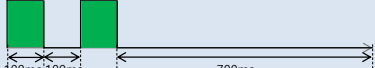



***주의:** BLDC, AC, STEP 모터의 엔코더 및 홀 센서 핀맵이 각각 다르니 사용자 매뉴얼 “2장 모터 사양 및 크기” 내용을 숙지한 후 SPM36D200S 드라이버에 연결해 주시기 바랍니다.

5.5 드라이버 외부 LED

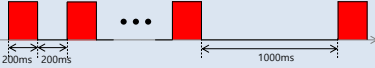
5.5.1 상태 표시 LED 기능

상태 표시	
BLUE LED	Running
RED LED	Error / Fault

5.5.2 Running LED 기능

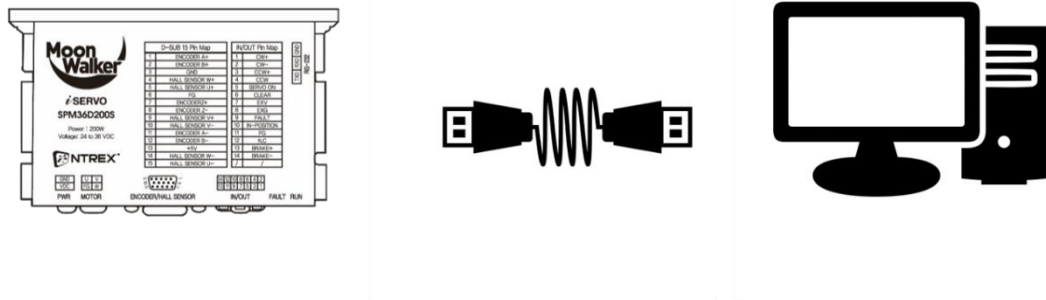
LED State	State	Description
	Motor Disable and Fault Occur	모터 Disable 상태로 모터에 전원이 투입되지 않으며 Fault가 발생한 상황입니다. Fault의 종류는 Fault LED에서 확인 가능합니다
	Motor Disable and No Fault	모터 Disable 상태지만 Fault가 발생하지 않은 상황입니다.
	Motor Enable and Limit/Stop by IO	모터 Enable 상태지만 Digital Input의 Limit Switch나 Stop Switch가 작동하여 모터를 움직일 수 없는 상황입니다.
	Motor Enable and Moving/Homing	모터 Enable 상태에서 모터가 구동중인 상황입니다.
	Motor Enable only	모터 Enable 상태에서 모터가 움직이지 않는 상황입니다.
	Motor Enable and In-position	위치 구동 명령에 대하여 모터가 목적지에 도달하여 In-position 상황임을 나타냅니다.

5.5.3 Fault LED 기능

LED State	Blinking	Description
	0	(정상)
	1	Overcurrent Fault 발생
	2	Overvoltage / Undervoltage Fault 발생
	3	Overheat Fault 발생
	4	Stall Current Fault 발생 / Vibration Fault 발생
	5	Position Tracking / Velocity Tracking / Overspeed Fault 발생
	6	Hallsensor 관련 Fault 발생
	7	Encoder Connection Fault 발생
	8	Motor Connection / Power Down Fault / Short circuit Fault 발생
	9	Positive Limit Fault / Negative Limit Fault 발생, Emergency Stop Swich 작동
	10	Electric Parameter (R, L, Ke) / Mechanic Parameter (J, B, C, TI)가 잘못 설정됨

5.6 통신 연결 구성

사용자가 PC를 사용하여 제어기의 구성(Configuration)을 설정하고 운용 하는 가장 간단한 방법은 제어기와 PC 간에 RS-232(or RS-485) 연결을 구성하고 PC에서 Motor Control UI 유틸리티를 사용하는 것입니다. PC에서 Motor Control UI 유틸리티를 실행하여 제어기를 설정하고 운용 할 수 있습니다.



6 시스템 구성

6.1 드라이버 & 모터 & 모션제어기 구성

SPM36D200S 드라이버와 각 BLDC, AC, STEP 모터, 그리고 모션제어기를 아래 그림과 같이 연결 후 바로 사용이 가능합니다.

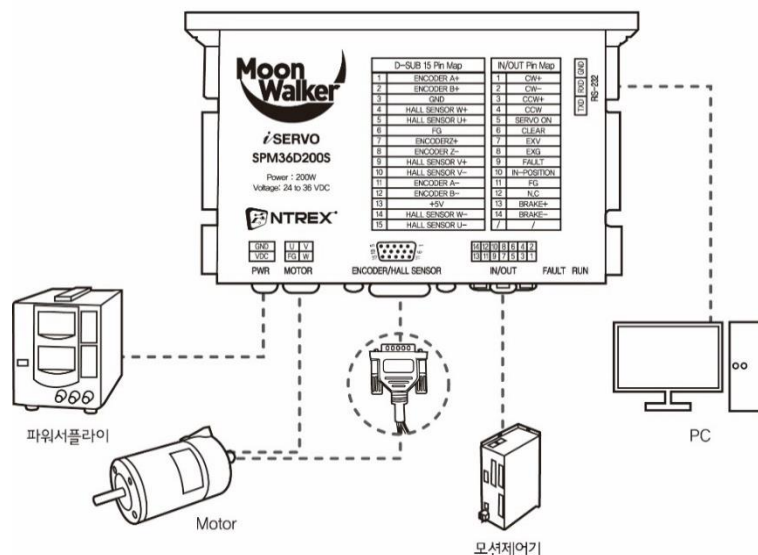


그림 6-1 드라이버 & 모터 & 모션제어기 구성의 예

6.2 드라이버 & 모터 & 액추에이터 구성

SPM36D200S 드라이버와 각 BLDC, AC, STEP 모터, 그리고 액추에이터를 가지고 아래 그림과 같이 연결 후 2 & 3축 직교로봇을 구성할 수 있습니다.



그림 6-2 드라이버 & 모터 & 액추에이터 구성의 예

7 Motor Control UI

7.1 소프트웨어 다운로드 및 실행

7.1.1 시스템 요구사항

이 유틸리티를 실행하기 위해서는 다음과 같은 PC 환경이 필요합니다.

- Window XP/7/10 32bit/64bit OS
- 10Mbyte의 HDD 여유공간
- 1GByte 이상의 RAM
- RS-232커넥터

***주의: Motor Control UI 유틸리티는 제품에 동봉되어 있지 않습니다. 이 유틸리티를 다운로드 하기 위해서 PC는 인터넷과 연결되어 있어야 합니다.**

7.1.2 다운로드

UI 유틸리티 프로그램은 디바이스마트내 제품 관련자료에서 다운받을 수 있습니다.

<http://www.devicezine.com/>

7.1.3 실행

Motor Control UI 유틸리티는 설치 과정이 필요없습니다. 압축을 푼 폴더 안에 있는 MotorControlUI.exe 파일을 실행하면 됩니다. 이 유틸리티의 첫 실행 화면은 그림 7-1에서 보여주고 있습니다.

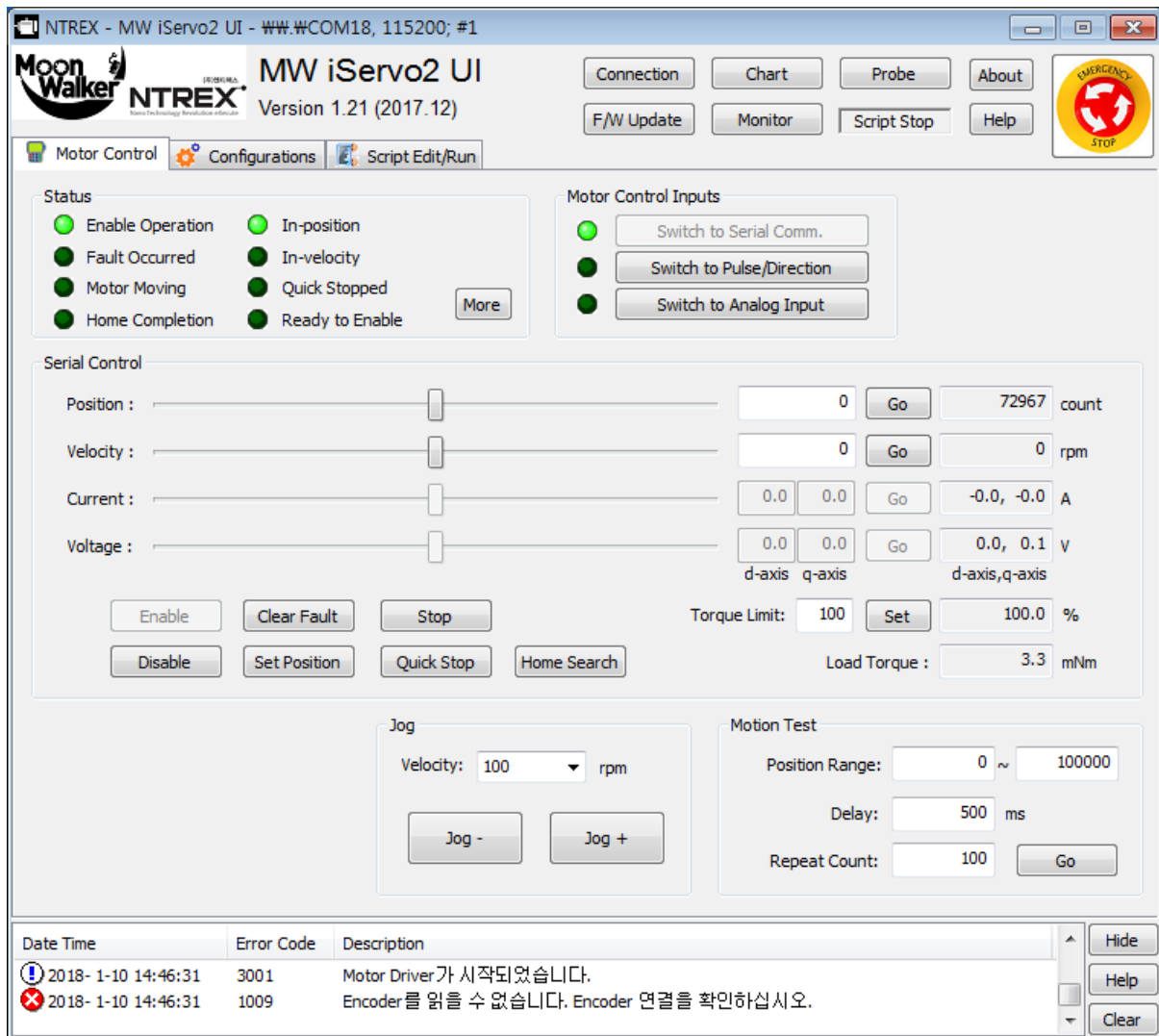


그림 7-1 SPM36D200S의 Motor Control UI 메인화면

이 유틸리티를 올바르게 사용하려면 모터 제어가 PC에 연결되어 있어야 합니다. 그렇지 않으면 유틸리티의 모든 기능이 활성화 되지 않습니다.

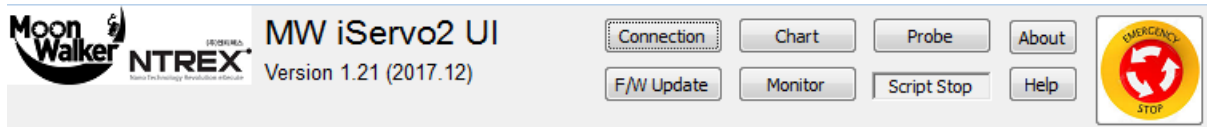
※ **Motor Control UI** 유틸리티는 수시로 업데이트 될 수 있습니다. 따라서 사용자는 최신 버전을 확인하고 사용하기 바랍니다.

7.2 메인 화면 구성

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

7.2.1 헤더

헤더에는 UI 유틸리티를 모터 드라이버에 연결하거나 펌웨어를 다운로드 할 수 있는 버튼들로 구성됩니다.



헤더의 제일 왼쪽에는 (주)엔티렉스 상호와 MoonWalker 상표명, 프로그램 이름(MW iServo2 UI)과 버전이 표시됩니다. 그리고 우측에 표시되는 버튼들은 다음과 같습니다.

- [Connection] 버튼 - 모터 드라이버와 연결을 위한 대화상자가 표시
- [Firmware Update] 버튼 - 모터 드라이버에 펌웨어 다운로드를 진행하는 대화상자 표시
- [Chart] 버튼 - Chart Recorder 창을 화면에 보이거나 숨기도록 토글 함, 현재 창이 표시되지 않는 상태에서 버튼을 누르면 창이 표시되고, 반대로 창이 표시되는 상태에서 버튼을 누르면 창을 숨김
- [Monitor] 버튼 - 모터 드라이버의 상태나 폴트 정보를 자세하게 표시하는 창을 보이거나 숨기도록 토글 함
- [Probe] 버튼 - 모터의 위치나 속도, 전류 값 등을 제어 주기에 따라 샘플링 한 값을 표시하는 그래프 창을 보이거나 숨기도록 토글 함
- [About] 버튼 - 프로그램에 대한 정보를 보여주는 대화상자 표시
- [Help] 버튼 - 도움말 표시
- [Emergency STOP] 버튼 - 비상 정지 버튼으로, 모터 드라이버를 Disable 상태로 만들고 모터는 Free run 상태가 됨

7.2.2 탭 윈도우

탭 윈도우는 Motor Control, Configurations, Script Edit/Run의 3개 탭으로 구성되어 있습니다.



다음은 각각의 탭에 대한 간략한 설명입니다.

- Motor Control Tab - 모터 드라이버에 모터 제어 명령을 내리고 상태를 모니터링 함
- Configuration Tab - 모터 드라이버의 각종 파라미터를 설정 함
- Script Edit/Run Tab - 스크립트 작성과 다운로드 및 실행

사용자는 필요에 따라 해당 탭을 선택하여 UI의 기능을 사용할수 있습니다.

7.2.3 메시지 표시

메시지 표시부에는 모터 드라이버에서 발생한 information, Warning, Error 메시지가 순서대로 표시됩니다.

Date Time	Error Code	Description	
2018- 1-10 14:46:31	3001	Motor Driver가 시작되었습니다.	Hide
2018- 1-10 14:46:31	1009	Encoder를 읽을 수 없습니다. Encoder 연결을 확인하십시오.	Help
			Clear

우측의 버튼들은 다음의 기능을 수행합니다.

- [Hide] 버튼 - 메시지 표시부를 접어 보이지 않게 함, 새로운 메시지가 발생하면 메시지 표시부가 열리게 됨
- [Help] 버튼 - 에러 메시지에 대한 설명과 조치 방법에 대한 문서를 오픈
- [Clear] 버튼 - 모터 드라이버에서 발생한 모든 메시지를 삭제

7.3 Chart Recorder 창

Chart Recorder 창은 모터에 내려지는 위치, 속도, 전류, 전압 명령과 센서 측정 그리고 입출력 채널 중 선택된 항목의 값을 실시간 그래프로 모니터링 할 수 있습니다.

이 창은 메인 화면의 헤더에서 [Chart] 버튼을 눌러 표시하거나 닫을 수 있습니다..



현재 출력되는 그래프를 파일로 저장하고자 한다면, [Save] 버튼을 눌러 파일 이름을 지정하고

*.csv 포맷으로 저장 가능합니다.

그래프의 왼쪽에 표시되는 스케일(Scale)은 **마우스 휠**을 위아래로 굴려 확대하거나 축소할 수 있습니다. 또한, 그래프의 아래에는 그래프로 표시되는 항목들에 대한 세부 정보가 표시됩니다.

[Channel] 버튼을 클릭하면 다음 그림 7.3과 같이 Channel Selection 대화상자가 표시됩니다. 사용자는 최대 8개의 오브젝트를 선택할 수 있으며, 선택한 오브젝트의 Sub-index 값과 Scale, Color를 지정 가능합니다.

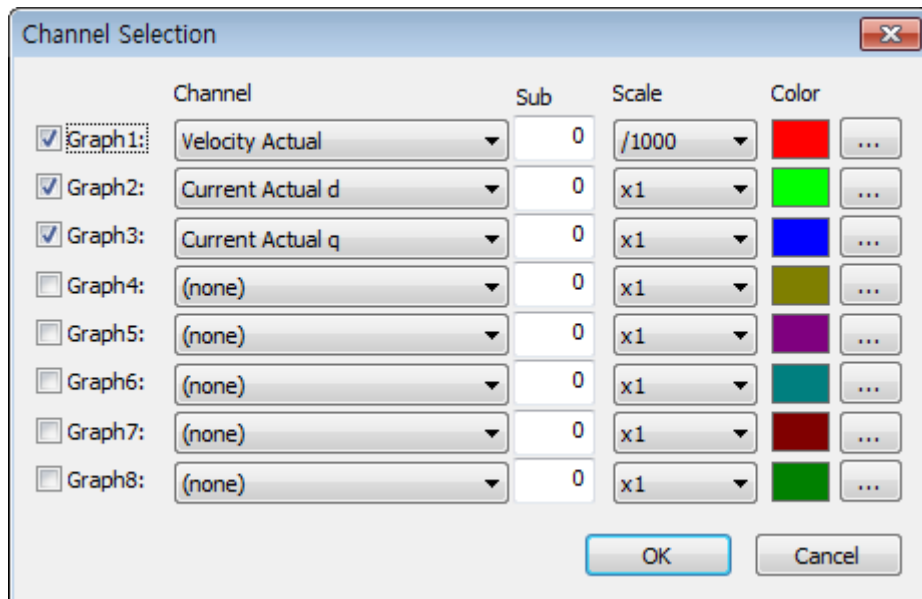
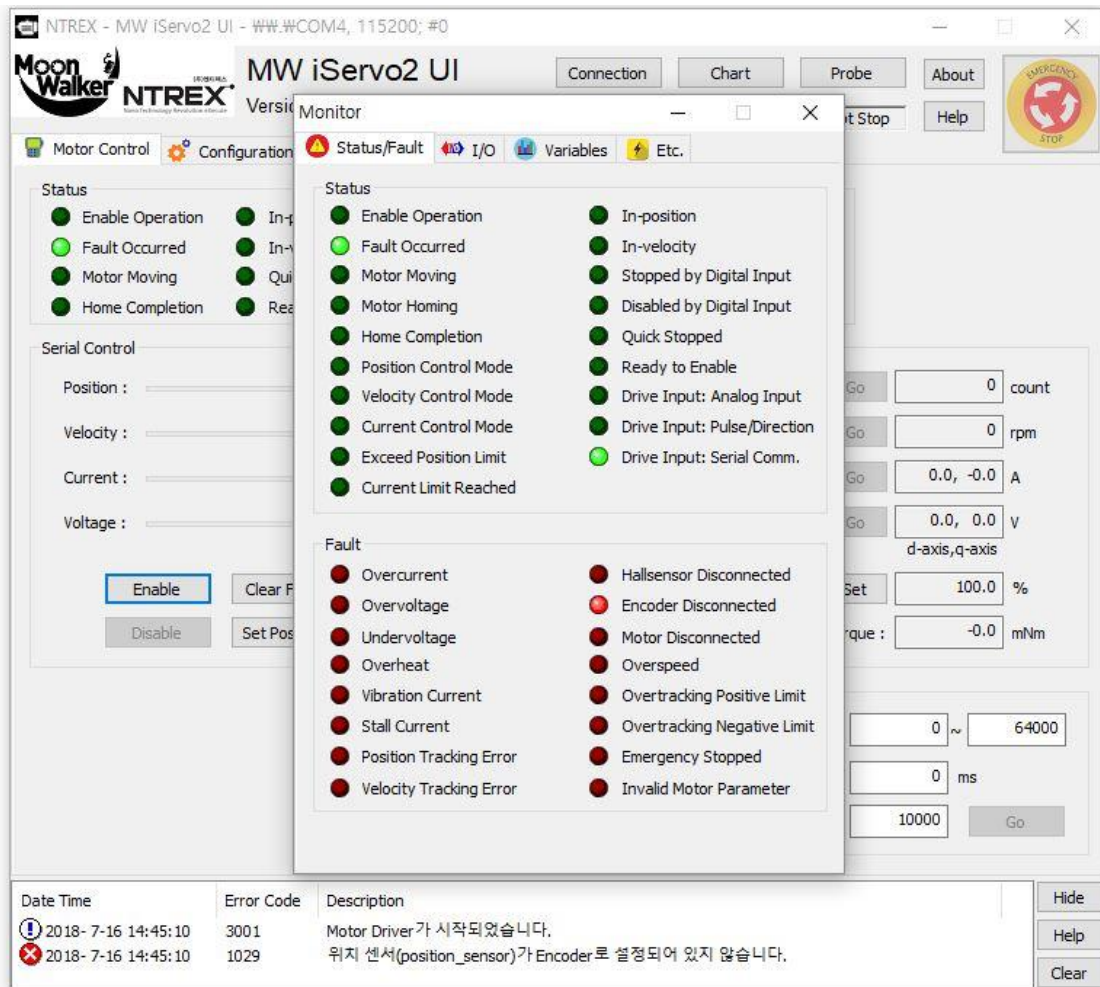


그림 7-3 Channel Selection 대화 상자

7.4 Monitor 창

Monitor 창에서는 모터 드라이버의 상태와 폴트, I/O의 입출력 상태를 모니터링 합니다. Monitor 창은 Status/Fault, I/O, Variables, Etc.의 4개 탭으로 구성됩니다.



7.4.1 Status/Fault 탭

Status/Fault 탭에서는 모터 드라이버의 상태 및 모터 드라이버에서 발생한 폴트 플래그를 확인할 수 있습니다.



7.4.1.1 Status 그룹

Status 그룹에서 표시되는 상태는 다음과 같습니다.

- - 상태 플래그가 활성 상태를 나타냄
- - 상태 플래그가 비활성 상태를 나타냄
- - 모터 드라이버와 연결이 끊겼거나 모터 드라이버의 전원이 꺼진 상태

Status 그룹에서 표시되는 상태를 정리하면 다음과 같습니다.

- Enable Operation - 모터에 전원이 공급되고 구동 준비가 됨
- Fault Occurred - 모터에 폴트가 발생, 모터의 전원이 차단되고 구동 불가능한 상태
- Motor Moving - 모터가 회전하고 있는 상태
- Motor Homing - 모터가 홈위치 이동 중인 상태
- Home Completion - 모터가 홈 위치이동 완료한 상태
- Position Control Mode - 위치 명령을 수행 중이거나 끝난 상태
- Velocity Control Mode - 속도 명령을 수행 중이거나 끝난 상태
- Current Control Mode - 전류 명령을 수행 중이거나 끝난 상태
- Exceed Position Limit - 최소, 최대 위치 리미트를 벗어난 상태
- Current Limit Reached - 모터에 흐르는 전류가 max_current에 가깝게 흐르는 상태
- In-position - 모터의 위치 구동 명령에 대하여 목표 위치에 도달

- In-velocity - 모터의 속도 구동 명령에 대하여 목표 속도에 도달
- Stopped by Digital Input - 디지털 입력의 Stop/Quick Stop 기능에 의해 모터가 정지
- Disabled by Digital Input - 디지털 입력의 Disable 기능에 의해 모터가 정지
- Quick Stopped - 모터에 Quick stop 명령이 내려진 상태, [Enable] 버튼을 눌러 해제
- Ready to Enable - 현재 모터는 구동 불가능한 상태이며, Enable 가능함을 표시
- Drive Input: Analog Input - 모터의 제어권이 아날로그 입력으로부터 내려짐
- Drive Input: Pulse/Direction- 모터의 구동 명령이 Pulse/Direction 입력으로부터 내려짐
- Drive Input: Serial Comm. - 모터의 구동 명령이 시리얼(RS-232/485, CAN, Ethernet) 포트로부터 내려짐

7.4.1.2 Fault 그룹

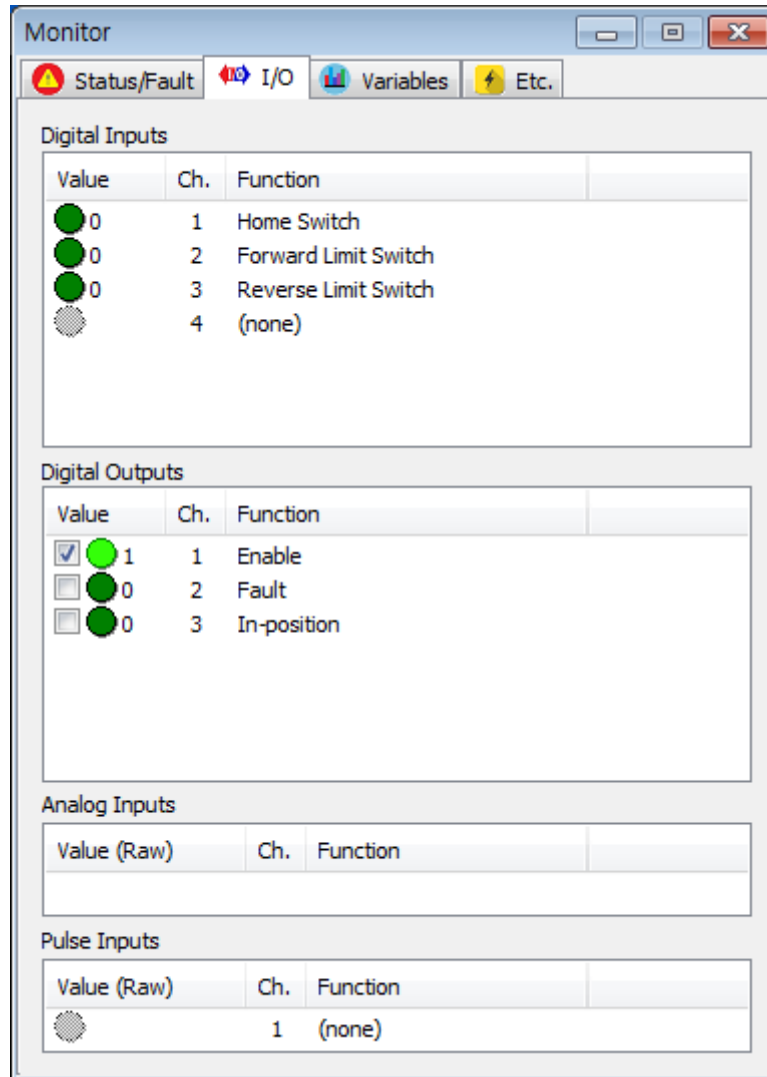
Fault 그룹은 모터에서 폴트가 발생한 세부 내역을 확인할 수 있습니다. 모터의 폴트 플래그 목록은 다음과 같습니다.

- Overcurrent - 모터에 과전류가 흐름
- Overvoltage - 모터 드라이버에 과전압이 걸림
- Undervoltage - 모터 드라이버에 저전압이 걸림
- Overheat - 내부 MOSFET와 방열판이 과열 됨
- Vibration Circuit - 모터의 전류가 설정 주기 이상으로 진동함
- Stall Current - 모터에 전력이 공급되나 회전하지 않는 상태 감지
- Position Tracking Error - 위치 제어 모드에서 위치 오차가 지정된 값 이상이 됨
- Velocity Error Detection - 속도 제어 모드에서 속도 오차가 지정된 값 이상이 됨
- Hallsensor Disconnected - 홀센서 연결이 감지되지 않음
- Encoder Disconnected - 엔코더 연결이 감지되지 않음
- Motor Disconnected - 모터 연결이 감지되지 않음
- Overspeed - 모터의 속도가 max_velocity의 150%를 초과함
- Overtracking Positive Limit- 모터의 위치가 정방향 위치 리미트를 벗어남
- Overtracking Negative Limit- 모터의 위치가 역방향 위치 리미트를 벗어남
- Emergency Stopped - 디지털 입력이나 통신 명령으로 긴급정지가 실행됨
- Invalid Motor Parameter - 모터의 파라미터가 구동 조건에 맞지 않음

모터 드라이버에서 폴트가 발생하면 모터에 공급되는 전원은 차단(Disable)되며, 폴트 조건이 해제된 이후에 모터는 Enable 가능합니다.

7.4.2 I/O 탭




I/O Monitoring 탭에서는 디지털 입력, 디지털 출력, 아날로그 입력, 펄스 입력 채널의 값을 모니터링하고 디지털 출력 채널을 ON/OFF 제어할 수 있습니다.



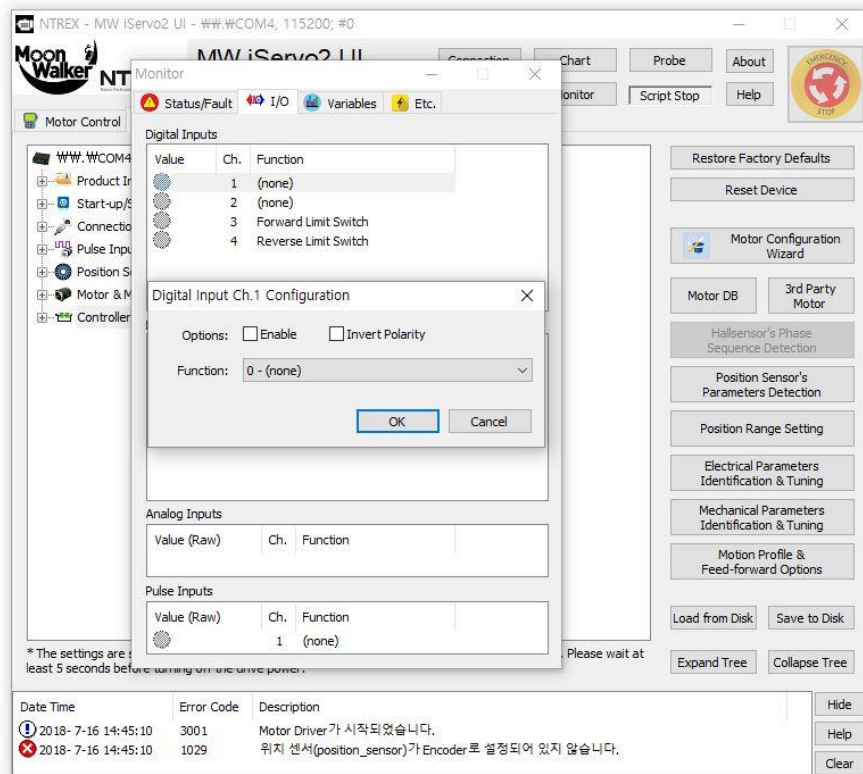
7.4.2.1 Digital Inputs

디지털 입력 그룹은 Enable 된 디지털 입력 채널의 값을 입력 및 모니터링 합니다.

디지털 입력 값의 상태를 표시하는 LED는 다음과 같습니다.

-  - 입력이 ON(1)인 상태 표시
-  - 입력이 OFF(0)인 상태 표시
-  - 모터 드라이버와 연결이 끊겼거나 입력이 사용 불가(disable) 상태

첫 번째 칼럼에는 디지털 입력 값의 상태를 LED와 숫자로 표시하고, 두 번째 칼럼에는 디지털 입력 채널의 숫자를 표시합니다. 마지막 칼럼에는 디지털 입력 채널에 매핑된 기능을 표시합니다.



Digital Input 채널 Configuration 창에서 디지털 입력 활성화, 디지털 입력 극성 반전, 디지털 입력의 기능을 설정 할 수 있습니다..




디지털 입력 채널은 총 4개가 있습니다. I/O Monitoring 탭에서 사용하고자 하는 채널을 마우스로 더블 클릭 해주면 Digital Input Channel Configuration 창이 뜹니다.

Digital Input Channel Configuration 창에서 디지털 입력 활성화, 디지털 입력 극성 반전, 디지털 입력의 기능을 설정 할 수 있습니다..

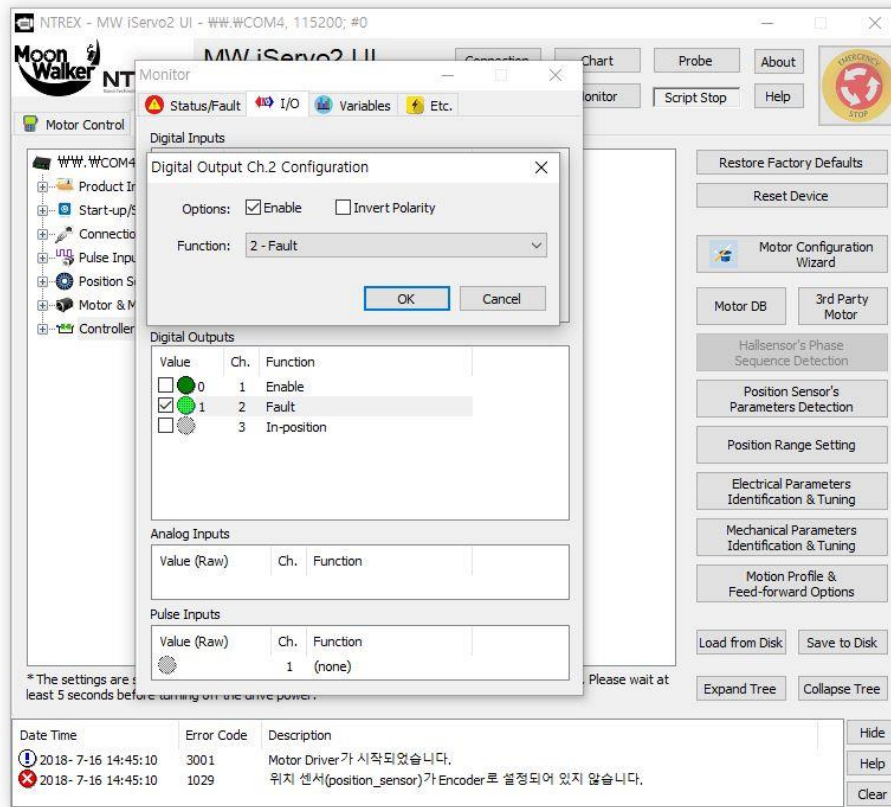
7.4.2.2 Digital Outputs

디지털 출력 그룹은 디지털 출력 채널의 값을 입력 및 모니터링하고 매핑된 기능을 읽을 수 있습니다.

디지털 출력 값의 상태를 표시하는 LED는 디지털 입력 값의 상태 LED와 같습니다.

-  - 출력이 ON(1)인 상태 표시
-  - 출력이 OFF(0)인 상태 표시
-  - 모터 드라이버와 연결이 끊겼거나 출력이 사용 불가(disable) 상태

첫 번째 칼럼에는 디지털 출력 값의 상태를 LED와 숫자로 표시하고, 두 번째 칼럼에는 디지털 출력 채널의 숫자를 표시합니다. 마지막 칼럼에는 디지털 출력 채널에 매핑된 기능을 표시합니다.



디지털 출력 채널은 총 3개가 있습니다. I/O Monitoring 탭에서 사용하고자 하는 채널을 마우스로 더블 클릭 해주면 Digital Output Channel Configuration 창이 뜹니다.

Digital Input Channel Configuration 창에서 디지털 출력 활성화, 디지털 출력 극성 반전, 디지털 출력의 기능을 설정 할 수 있습니다..

사용자가 디지털 출력 값을 직접 설정하고자 할 때는 디지털 출력 채널이 Enable 상태에서 매핑된 기능은 (none)이 선택되어야 합니다.

7.4.2.3 Analog Inputs

Analog Inputs은 SST-36D200S-C모터 드라이버만 사용 가능 합니다. 아날로그 입력은 모터의 구동에 필요한 정보 및 명령을 아날로그 신호로 받아들이기 위해 사용하며, 아날로그 입력 활성화, 아날로그 입력 극성 반전, 아날로그 입력 기능을 설정할 수 있습니다.

아날로그 입력 그룹은 Enable 된 아날로그 입력 채널의 원래(raw) 값과 변환된 값을 모니터링하고 매핑된 기능을 읽을 수 있습니다. 괄호 안의 숫자는 원래 값(0 ~ 4094)이고 괄호 바깥의 숫자는 변환된 값(-1 ~ 1) 입니다.

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.

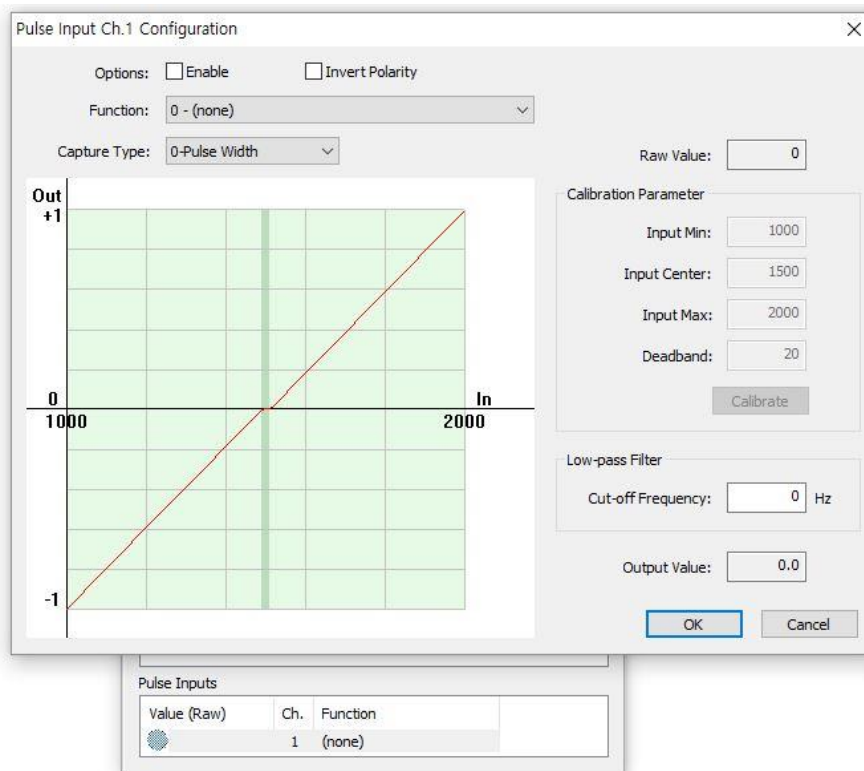
7.4.2.4 Pulse Inputs

Pulse Inputs은 RC 조종기와 같이 디지털 신호를 발생하는 장치로부터 모터의 구동 명령을 받아들이기 위해 사용됩니다.

펄스 입력 그룹은 펄스 입력 채널의 원래(raw) 값과 변환된 값을 모니터링하고 매핑된 기능을 읽을 수 있습니다. 괄호 안의 숫자는 원래 값이고 괄호 바깥의 숫자는 변환된 값(-1 ~ 1)입니다.

펄스 입력 채널의 원래 값은 펄스 입력 포트에 들어오는 신호의 Pulse Width, Frequency, Duty Cycle을 읽은 값입니다. 이 값의 범위는 캡처 타입에 따라 달라집니다. 이 값은 변환 과정을 거쳐 -1 ~ 1 사이의 값으로 변환됩니다.

I/O Monitoring 탭에서 사용하고자 하는 채널을 마우스로 더블 클릭 해주면 Pulse Input Channel Configuration 창이 뜹니다



Pulse Input 채널 Configuration 창에서 펄스 입력 활성화, 펄스 입력 극성 반전, 디지털 입력의 기능, 캘리브레이션 등을 설정 할 수 있습니다..

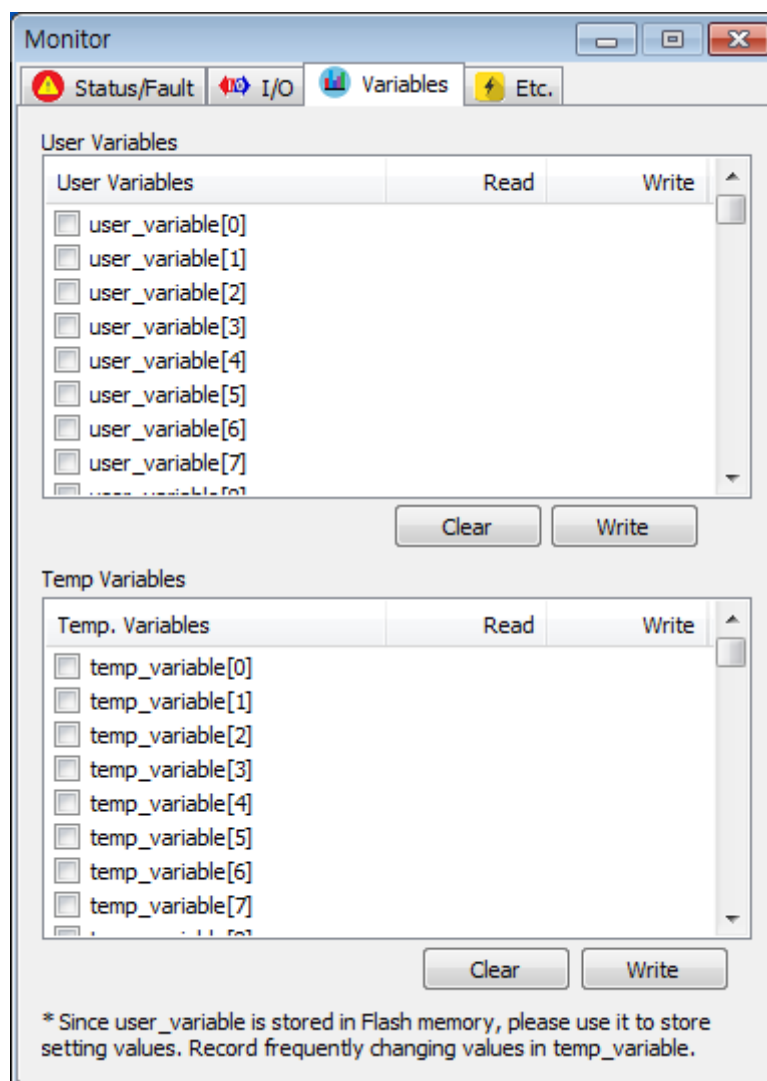
아날로그 입력이나 펄스 입력 채널로부터의 원시 값은 캘리브레이션 과정을 거쳐 -1과 1 사이의 정규화 된 값으로 변환됩니다. 캘리브레이션 과정에는 min, max, center, deadband 파라미터의 설정이 필요합니다. 자동으로 캘리브레이션 파라미터의 설정을 위해서는 아날로그 입력이나 펄스 입력에서 Calibration을 선택해서 [...] 버튼을 클릭합니다.

캘리브레이션을 진행하기 위해 앞서 펄스 입력 채널은 Enable 상태로 설정되어 있어야 합니다. 여기서는 설명을 쉽게 하기 위해 펄스 입력 채널 1에 조이스틱이 연결되었다고 가정하겠습니다.

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.

7.4.3 Variables 탭

Variable 탭에서는 User Variables과 Temp. Variables 값을 읽고 쓸 수 있습니다.



7.4.3.1 User Variables

User Variables은 모터 드라이버의 운용과 관련된 사용자 설정 값들을 저장하는데 사용하며, 변경된 값은 Flash Memory에 저장되기 때문에 모터 드라이버의 전원이 꺼지고 켜지더라도 변경된 값

이 유지됩니다.

첫 칼럼의 체크박스를 체크하면 현재 User Variables에 저장된 값을 두 번째 칼럼에 보여줍니다. 만일 값을 변경하고자 한다면, 세 번째 칼럼에 값을 쓰고 [Write] 버튼을 누르면 됩니다.

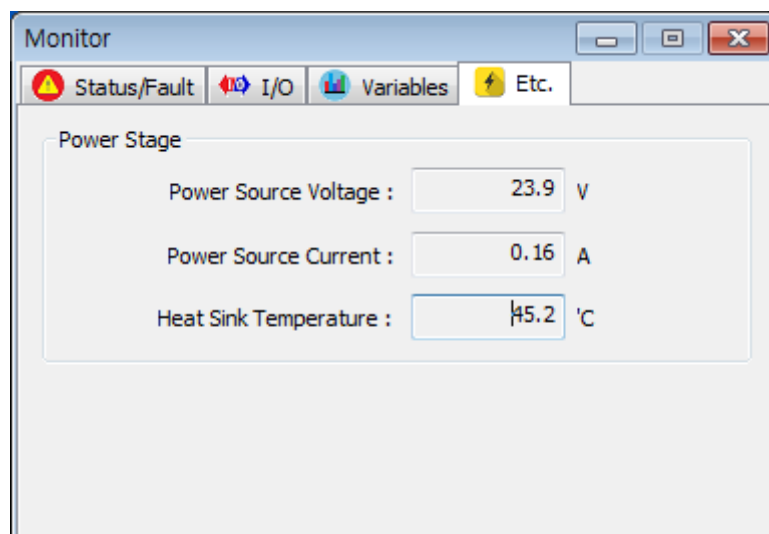
7.4.3.2 Temp Variables

Temp. Variables은 주로 Script 와 주고받는 값을 읽고 쓰기위해 사용하며, 값을 변경하더라도 RAM에만 남아있기 때문에 모터 드라이버의 전원이 꺼지고 켜지면 0으로 초기화 됩니다. 각각의 변수들은 Sub-index에 의해 구분됩니다.

첫 칼럼의 체크박스를 체크하면 현재 User Variables에 저장된 값을 두 번째 칼럼에 보여줍니다. 만일 값을 변경하고자 한다면, 세 번째 칼럼에 값을 쓰고 [Write] 버튼을 누릅니다.

7.4.4 Etc. 탭

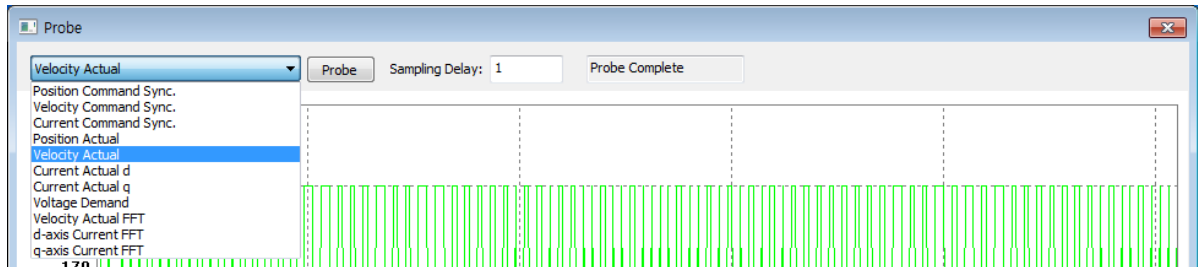
Etc. 탭에서는 모터 드라이버의 Power Stage와 관련된 변수들을 모니터링 합니다.



- Power Source Voltage - 모터 드라이버의 전원 입력단의 전압
- Power Source Current - 모터 드라이버의 전원에 흐르는 전류 (예측값)
- Heat Sink Temperature - FET와 방열판의 온도

7.5 Probe 창

Probe 창은 모터 드라이버의 내부 변수를 일정 간격의 제어 주기와 동기하여 모니터링 하는 창입니다. 최대 3개의 변수를 동시에 모니터링 가능합니다.



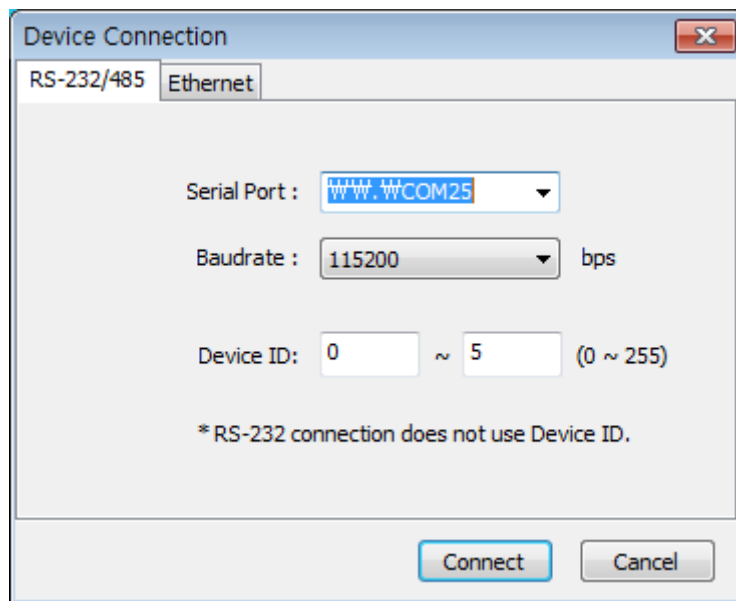
- Position Command Sync. - 위치 명령이 내려진 후의 위치 관련 변수 표시
- Velocity Command Sync. - 속도 명령이 내려진 후의 속도 관련 변수 표시
- Current Command Sync. - 전류 명령이 내려진 후의 전류 관련 변수 표시
- Position Actual - [Probe] 버튼이 눌린 이후의 실제 위치 관련 변수 표시
- Velocity Actual - [Probe] 버튼이 눌린 이후의 실제 속도 관련 변수 표시
- Current Actual d - [Probe] 버튼이 눌린 이후의 실제 d축 전류 관련 변수 표시
- Current Actual q - [Probe] 버튼이 눌린 이후의 실제 q축 전류 관련 변수 표시
- Voltage Demand - [Probe] 버튼이 눌린 이후의 전압 관련 변수 표시
- Velocity Actual FFT - [Probe] 버튼이 눌린 이후의 실제 속도의 FFT 결과 표시
- d-axis Current FFT - [Probe] 버튼이 눌린 이후의 실제 d축 전류의 FFT 결과 표시
- q-axis Current FFT - [Probe] 버튼이 눌린 이후의 실제 q축 전류의 FFT 결과 표시

7.6 연결

메인 화면의 헤더에서 [Connection] 버튼을 클릭하면, UI 유틸리티를 모터 드라이버에 연결하기 위한 대화상자가 화면에 표시됩니다. 모터 드라이버는 종류에 따라 RS-232, RS-485, Ethernet 연결을 제공합니다.

7.6.1 RS-232/485 연결

UI가 실행되는 PC와 모터 드라이버가 RS-232 또는 RS-485로 연결된 경우 RS-232/485 탭에서 연결을 시도합니다.



Serial Port에는 연결 가능한 RS-232 또는 RS-485 통신 포트가 검색되어 표시됩니다. 표시되는 포트 이름에서 모터 드라이버가 연결된 통신 포트를 선택합니다.

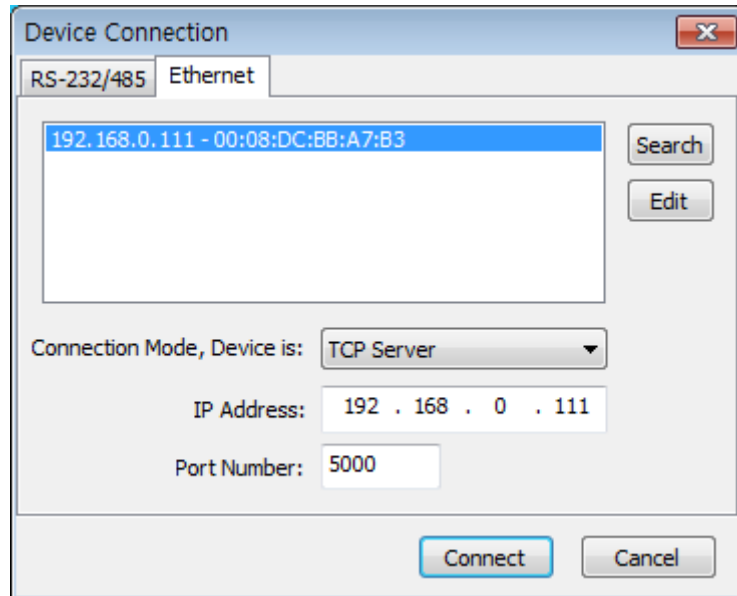
Baudrate에서는 COM 포트의 통신 속도(9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600 bits/s)를 선택합니다. 만일 'Auto detect'를 선택하면, 모든 통신 속도를 검색하여 모터 드라이버가 응답하는지 체크합니다.

Device ID에서는 모터 드라이버 ID의 검색 범위를 설정합니다. 위 그림에서와 같이 0에서 5 값으로 설정하면 모터 드라이버 ID를 0부터 5까지 검색하여 제일 먼저 응답하는 모터 드라이버와 연결하게 됩니다. 만일 검색 범위를 넓게 설정하게 되면 UI 프로그램은 많은 모터 드라이버 ID를 검색해야 하기 때문에 검색 시간이 오래 걸립니다.

연결과 관련된 정보 설정이 끝나면 [Connect] 버튼을 눌러 모터 드라이버와 연결을 시도합니다. 만일 [Cancel] 버튼을 누르면 연결을 시도하지 않고 않고 대화상자를 닫습니다.

7.6.2 Ethernet 연결

모터 드라이버가 이더넷으로 연결된 경우 Ethernet 탭을 클릭하여 연결을 시도합니다.



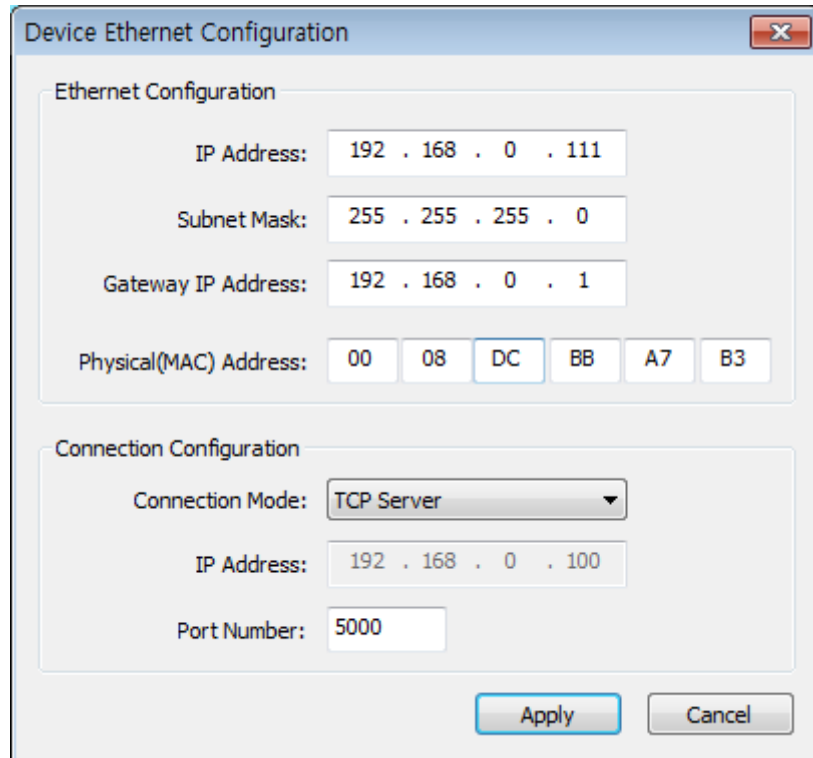
우측 상단의 [Search] 버튼을 누르면 호스트 PC와 동일한 서브넷에 연결된 모터 드라이버를 검색하여, 연결 가능한 모터 드라이버를 리스트 박스에 표시합니다.

Connection Mode, Device is에서는 호스트 PC와 모터 드라이버간 이더넷 연결 방법을 선택합니다. IP Address에는 UDP, TCP Client 모드에서 모터 드라이버가 접속해야 할 호스트 PC의 IP 주소를 설정합니다. Port Number에는 UDP, TCP Client 모드에서 모터 드라이버가 접속해야 할 호스트 PC의 포트 번호를 설정합니다. TCP Server 모드에서 모터 드라이버가 접속을 기다리는 TCP 서버 포트 번호가 됩니다.

호스트 PC와 모터 드라이버가 다른 서브넷에 연결되어 있는 경우, 접속 대상이 되는 모터 드라이버의 IP Address와 Port Number를 직접 입력하여 연결해야 합니다.

7.6.2.1 Ethernet Edit

검색된 모터 드라이버를 선택하고 [Edit] 버튼을 누르면 다음 그림과 같은 대화상자가 화면에 표시되면서 모터 드라이버의 Ethernet 포트에 관련된 설정을 편집할 수 있습니다.



The image shows a 'Device Ethernet Configuration' dialog box. It has two main sections: 'Ethernet Configuration' and 'Connection Configuration'. In the 'Ethernet Configuration' section, there are input fields for 'IP Address' (192 . 168 . 0 . 111), 'Subnet Mask' (255 . 255 . 255 . 0), 'Gateway IP Address' (192 . 168 . 0 . 1), and 'Physical(MAC) Address' (00 08 DC BB A7 B3). The 'Connection Configuration' section has a 'Connection Mode' dropdown menu set to 'TCP Server', an 'IP Address' field (192 . 168 . 0 . 100), and a 'Port Number' field (5000). At the bottom right, there are 'Apply' and 'Cancel' buttons.

Ethernet Configuration 창에서는 모터 드라이버 자체의 IP 주소와 MAC 주소를 설정하는 부분입니다.

- IP address – 모터 드라이버의 IP 주소 설정
- Subnet Mask – 모터 드라이버의 서브넷 마스크 설정
- Gateway IP Address – 게이트웨이의 IP 주소 설정
- Physical(MAC) Address – 모터 드라이버의 MAC 주소 설정

Connection Mode, Device is에서는 호스트 PC와 모터 드라이버간 이더넷 연결 방법을 선택하는데, 다음의 3가지 중 하나를 선택할 수 있습니다.

- TCP Server – 모터 드라이버가 TCP 서버가 되어 호스트 PC에서 연결하기를 기다림; 모터 드라이버에는 하나의 호스트 PC만 연결 가능
- TCP Client – 모터 드라이버가 TCP 클라이언트가 되어 호스트 PC의 서버에 연결 시도; 모터 드라이버가 DHCP로 설정되었을 때 주로 사용됨
- UDP – UDP/IP 프로토콜로 연결; 호스트 PC와 모터 드라이버가 동일한 서브넷 안에 있을 때 사용 가능

IP Address에는 UDP, TCP Client 모드에서 모터 드라이버가 접속해야 할 호스트 PC의 IP 주소를 설정합니다.

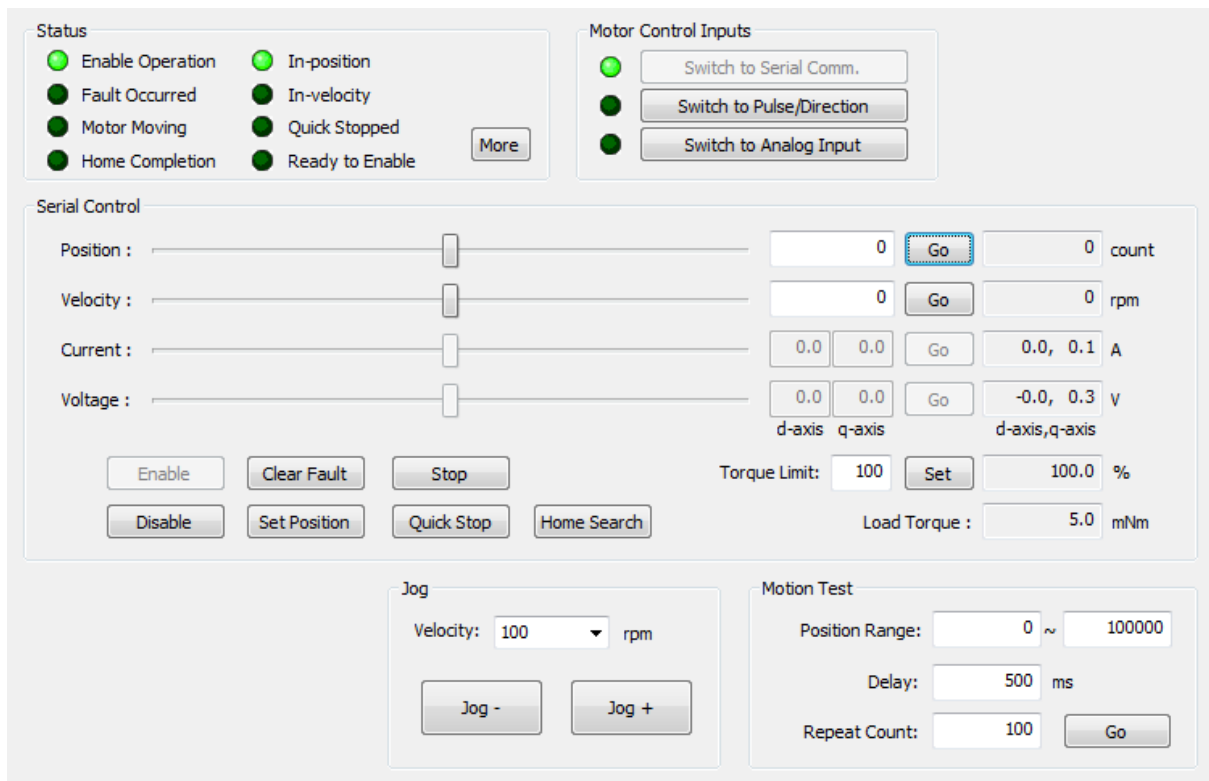
Port Number에는 UDP, TCP Client 모드에서 모터 드라이버가 접속해야 할 호스트 PC의 포트 번호를 설정합니다. TCP Server 모드에서 모터 드라이버가 접속을 기다리는 TCP 서버 포트 번호가 됩니다.

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

모터 드라이버는 TCP 서버 모드에서 시리얼 포트 당 하나의 TCP 세션을 지원합니다. 그래서 이미 연결이 성립된 경우, 추가 TCP 연결 요청은 거부됩니다.

7.7 Motor Control 탭

Motor Control 탭은 모터 드라이버의 상태를 표시하고, 모터의 위치, 속도, 전류, 전압 명령을 내리거나 현재 값들을 읽어올 수 있습니다.



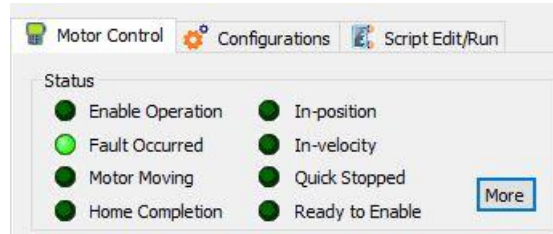
The screenshot displays the Motor Control interface with the following sections:

- Status:** Includes indicators for Enable Operation (green), Fault Occurred (red), Motor Moving (red), Home Completion (red), In-position (green), In-velocity (green), Quick Stopped (red), and Ready to Enable (green). A 'More' button is also present.
- Motor Control Inputs:** Features three buttons: 'Switch to Serial Comm.', 'Switch to Pulse/Direction', and 'Switch to Analog Input'.
- Serial Control:** Contains sliders for Position, Velocity, Current, and Voltage. To the right are input fields for numerical values and 'Go' buttons. Below these are buttons for 'Enable', 'Clear Fault', 'Stop', 'Disable', 'Set Position', 'Quick Stop', and 'Home Search'. Further right are 'Torque Limit' and 'Load Torque' settings.
- Jog:** Includes a 'Velocity' dropdown set to 100 rpm and 'Jog -' and 'Jog +' buttons.
- Motion Test:** Features 'Position Range' (0 to 100000), 'Delay' (500 ms), and 'Repeat Count' (100) fields, along with a 'Go' button.

Motor Control 탭에는 크게 Status, Motor Control Inputs, Serial Control, Jog, Motion Test 그룹으로 나누어집니다.

7.7.1 Status 그룹

Status 그룹에서는 모터 드라이버의 구동 상태 중에서 주요한 몇 가지 상태만 표시합니다.



표시되는 상태는 다음과 같습니다.

- - 상태 플래그가 활성 상태를 나타냄
- - 상태 플래그가 비활성 상태를 나타냄
- - 모터 드라이버와 연결이 끊겼거나 모터 드라이버의 전원이 꺼진 상태

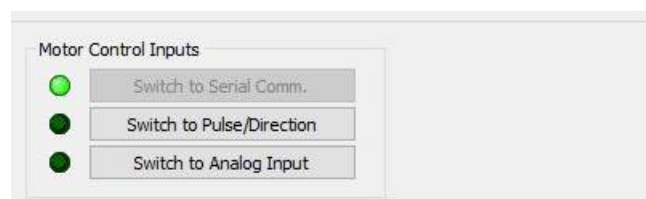
표시되는 상태를 정리하면 다음과 같습니다.

- Enable Operation - 모터에 전원이 공급되고 구동 준비가 됨
- Fault Occurred - 모터에 폴트가 발생, 모터의 전원이 차단되고 구동 불가능한 상태
- Motor Moving - 모터가 회전하고 있는 상태
- Home Completion - 모터가 홈 위치이동 완료한 상태
- In-position - 모터의 위치 구동 명령에 대하여 목표 위치에 도달한 상태를 표시
- In-velocity - 모터의 속도 구동 명령에 대하여 목표 속도에 도달한 상태를 표시
- Quick Stopped - 모터에 Quick stop 명령이 내려진 상태, [Enable] 버튼을 눌러 해제
- Ready to Enable - 현재 모터는 구동 불가능한 상태이며, Enable 가능함을 표시

상기 목록보다 더 자세한 Status는 [More] 버튼을 눌러 Monitor 창에서 확인할 수 있습니다.

7.7.2 Motor Control Inputs 그룹

Motor Control Inputs 그룹에서는 모터에 내려지는 위치, 속도, 전류, 전압 명령의 소스를 선택할 수 있습니다.



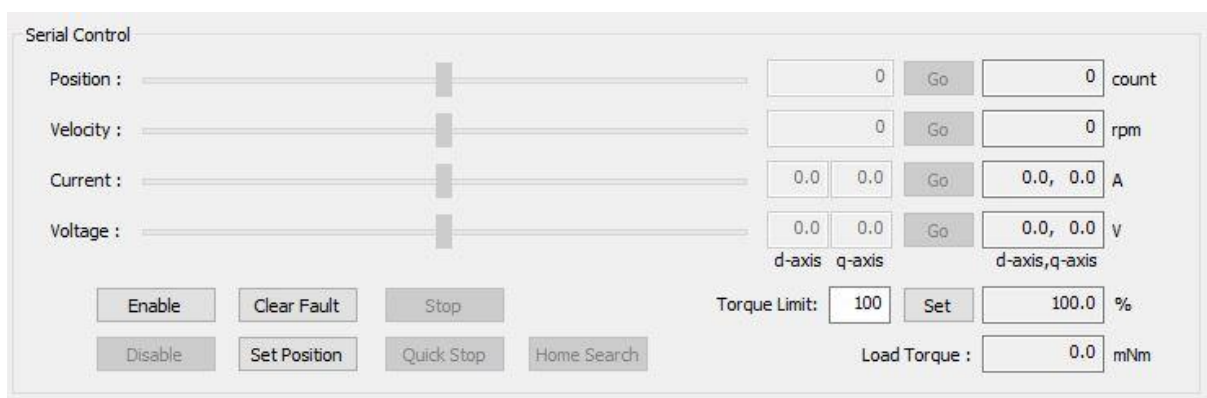
- Switch to Serial Comm. - RS-232, RS-485 또는 Ethernet과 같은 시리얼 통신을 통해 모터에 구동 명령을 내림
- Switch to Pulse/Direction - Pulse/Direction 입력을 통해 모터에 위치 구동 명령을 내림
- Switch to Analog Input - Analog Input이나 Pulse Input 채널을 모터의 구동 명령에 매핑하여 모터를 구동함

모든 종류의 모터 드라이버는 시리얼 통신으로 모터 드라이버의 설정 값을 읽고 쓸 수 있으며 모터 구동도 가능합니다. 반면 Pulse/Direction 구동이나 Analog Input 구동은 모터 드라이버가 인터페이스를 제공할 경우에만 가능합니다. 다음 표에서 모터 드라이버의 종류 별로 지원하는 인터페이스 종류를 확인할 수 있습니다.

Model Name	Serial Comm.	Pulse/Direction	Analog Input
SST-36D200S-P	RS-232	O	Pulse Input 1ch.
SST-36D200S-Pv2	RS-232, CAN	O	Pulse Input 1ch.
SST-36D200S-C	RS-485, Ethernet	X	Pulse Input 1ch. Analog Input 1ch.

7.7.3 Serial Control 그룹

Serial Control 그룹에서는 Position, Velocity, Current, Voltage에 대한 슬라이드 바를 움직이거나 값을 직접 입력하고 [Go] 버튼을 누름으로 모터에 위치, 속도, 전류, 전압 명령을 내린다. 그리고 모터의 현재 위치, 속도, 전압과 전류 값을 표시합니다.



The screenshot shows the 'Serial Control' interface. It features four horizontal sliders for Position, Velocity, Current, and Voltage. To the right of each slider are input fields and a 'Go' button. For Position, the unit is 'count'. For Velocity, the unit is 'rpm'. For Current, there are two input fields for 'd-axis' and 'q-axis', with the unit 'A'. For Voltage, there are two input fields for 'd-axis, q-axis', with the unit 'V'. Below the sliders are buttons for 'Enable', 'Clear Fault', 'Stop', 'Disable', 'Set Position', 'Quick Stop', and 'Home Search'. At the bottom right, there are fields for 'Torque Limit' (set to 100) and 'Load Torque' (set to 0.0 mNm).

- Position - 목표 위치를 지정하고 [Go] 버튼을 눌러 목표 위치로 이동 명령을 내림, 슬라이드 바의 이동 범위는 min_position과 max_position에 의해 결정됨
- Velocity - 목표 속도를 지정하고 [Go] 버튼을 눌러 목표 속도로 이동 명령을 내림, 슬라이드 바의 이동 범위는 -rated_velocity와 +rated_velocity에 의해 결정됨
- Current - d축과 q축에 대한 목표 전류를 지정하고 [Go] 버튼을 눌러 목표 전류로 구동

명령을 내림, 슬라이드 바로는 q축 목표 전류를 조절하며 슬라이드 바의 이동 범위는 -rated_current와 +rated_current에 의해 결정됨

- Voltage - d축과 q축에 대한 출력 전압을 지정하고 [Go] 버튼을 눌러 전압 구동 명령을 내림, 슬라이드 바로는 q축 출력 전류를 조절하며 슬라이드 바의 이동 범위는 -rated_voltage와 +rated_voltage에 의해 결정됨

그리고 최대 토크 한계 값을 설정하고 현재 모터에 작용하는 부하 토크 값을 표시합니다.



The interface shows a 'Torque Limit' section with a text input '100', a 'Set' button, and a display '100.0 %'. Below it is a 'Load Torque' section with a text input '0.0' and a unit 'mNm'.

- Torque Limit - 모터에 흐르는 최대 전류를 max_current에 대한 퍼센트 값(0 ~ 100)으로 [Set] 버튼을 눌러 설정함, 이 값은 모터 드라이버가 Disable 되었다가 Enable 될 때 100으로 초기화 됨
- Load Torque - 현재 모터에 작용하는 추정 부하 토크 값, 모터의 기구 파라미터에 의해 계산된 값으로 실제 값과 차이가 있을 수 있음

모터를 활성화 하거나 비활성화하고 구동을 정지하는 기능들은 다음과 같이 버튼들로 구성됩니다.



The interface contains seven buttons arranged in two rows: 'Enable', 'Clear Fault', 'Stop' in the top row, and 'Disable', 'Set Position', 'Quick Stop', 'Home Search' in the bottom row.

- [Enable] 버튼 - 모터가 Disable 상태일 때 Enable 상태로 만듦, Enable 상태에서는 모터에 전원이 공급되고 위치, 속도, 전류, 전압 명령에 의해 모터를 구동 가능함, [Enable] 버튼은 모터가 Quick Stop 중인 상태에서 빠져 나오는데도 사용됨
- [Disable] 버튼 - 모터가 Enable 상태일 때 Disable 상태로 만듦, Disable 상태에서는 모터에 전원이 차단되고 모터를 외력에 의해 구동 가능한 Free run 상태가 됨
- [Clear Fault] 버튼 - 모터 드라이버에 발생한 Fault 플래그를 모두 해제함, 만일 모든 Fault 플래그가 정상적으로 해제되었다면 Ready to Enable 상태가 커짐
- [Set Position] 버튼 - 모터의 현재 위치 값을 새로 설정함
- [Stop] 버튼 - 모터가 위치, 속도 명령에 의해 구동 중일 때는 모터의 속도가 0이 되도록 함; 모터가 전류, 전압 명령에 의해 구동 중일 때는 모터에 출력 되는 전압이 0이 되도록 함
- [Quick Stop] 버튼 - 모터를 정지하고 모터는 Enable 상태이나 새로운 명령을 처리할 수 없도록 Quick Stop 상태를 유지함, Quick Stop 상태는 [Enable] 버튼을 눌러 해제 가능함
- [Home Search] 버튼 - homing_method에 설정된 방법으로 홈 위치이동을 시작함, 자세한 내용은 homing_method 오브젝트의 설명 참조

7.7.4 Jog 그룹

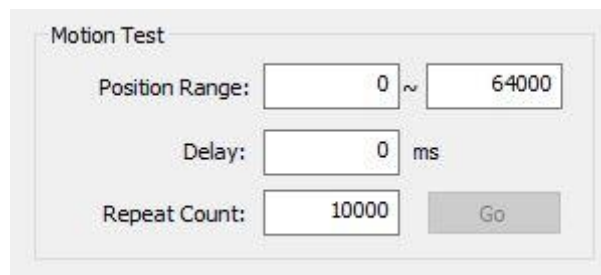
Jog 그룹은 조그 명령으로 모터의 정방향 /역방향 구동합니다.



- Velocity – 조그 이동 속도를 설정함
- [Jog -] 버튼 – 조그 이동 속도에 설정된 속도로 모터를 역방향 구동함
- [Jog +] 버튼 – 조그 이동 속도에 설정된 속도로 모터를 정방향 구동함

7.7.5 Motion Test 그룹

Motion Test 그룹에서는 모터가 지정된 두 위치 사이를 반복적으로 이동하도록 명령을 생성합니다. 이 명령은 UI에서 생성되기 때문에 정밀한 명령 생성 시간이 보장되지 않습니다.

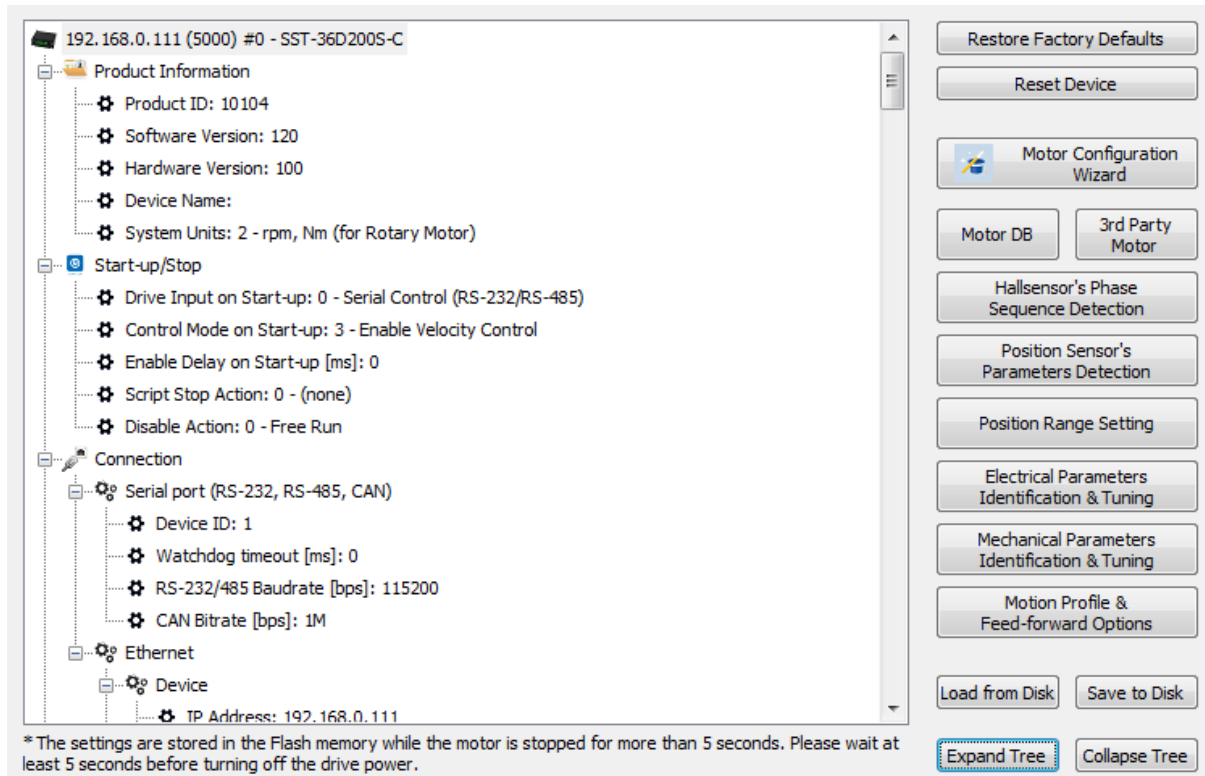


- Position Range – 모터가 왕복 이동하는 두 개의 목표 위치를 설정
- Delay – 모터가 목표 위치에 도달 후 멈춰 대기하는 시간 설정
- Repeat Count – 반복 횟수 설정

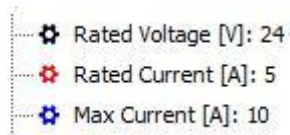
상기 3가지 항목을 설정하고 [Go] 버튼을 눌러 모터의 구동을 시작합니다. 모터가 설정된 반복 횟수만큼 이동을 완료하면 정지합니다.

7.8 Configuration 탭

Configuration 탭은 모터 드라이버의 각종 파라미터를 설정할 수 있다. 설정 값들을 모터 드라이버의 플래시 메모리에 저장되고 Factory default 값으로 초기화 할 수 있으며, PC의 파일로 저장하고 불러올 수 있습니다.



UI가 모터 드라이버에 연결되면, 모터 드라이버의 파라미터 값을 읽고 왼편 창에 표시합니다. 사용자가 파라미터의 값을 변경하면, 변경된 값은 즉시 모터 드라이버에 다운로드 되고 적용됩니다. 사용자가 오른쪽 버튼들을 이용해서 값을 바꿀시 톱니바퀴 색은 파란색으로 변하고, 사용자가 직접 값을 입력해서 바꾼 경우에는 톱니바퀴 색은 붉은색으로 변합니다.



※ 값을 변경하실 경우 톱니바퀴 색이 바뀌었는지 체크하시기 바랍니다.

***주의:** 모터 드라이버에 다운로드 된 파라미터 값은 모터 드라이버에 즉시 적용되지만 플래시 메모리에 바로 저장되지는 않습니다. 플래시 메모리에 저장되기 위해서는 모터 드라이버가 구동을 멈춘 상태에서 5초 정도의 시간이 필요합니다. 즉, 5초 이상 모터를 구동하지 않는 조건에서 변경된 파라미터를 플래시 메모리에 기록하기 때문에, 모터 드라이버의 파라미터 값을 바꾼 경우에는 즉시 전원을 끄지 않도록 주의하셔야 합니다.

탭의 오른쪽 버튼들에 대한 기능을 요약하면 다음과 같습니다.

- [Reset Factory Defaults] 버튼 - 제품을 출시할 때의 설정 값으로 모든 파라미터 설정을 초기화 함
- [Reset Device] 버튼 - 모터 드라이버를 소프트웨어 리셋 함, 모터 드라이버 펌웨어가 재 시작 됨
- [Motor Configuration Wizard] 버튼 - 모터 설정 마법사를 실행함
- [Motor DB] 버튼 - Motor Database 대화상자를 열어 모터의 종류를 선택함으로 모터 드라이버의 모터관련 파라미터를 설정함
- [3rd Party Motor] - 모터 데이터베이스에서 지원하지 않는 모터를 사용하는 경우, 모터의 종류 및 모터에 대한 정격 전압과 전류 값을 설정하는 대화상자를 실행함
- [Hallsensor's Phase Sequence Detection] 버튼 - 홀센서의 배열 순서와 위상을 탐지하는 대화상자를 실행함
- [Position Sensor's Parameters Detection] 버튼 - 모터에 연결된 위치 센서(Hallsensor, Encoder) 정보를 탐지하는 대화상자를 실행함
- [Position Range Setting] 버튼 - 모터의 구동 범위를 설정하는 대화상자를 실행함
- [Electrical Parameters Identification & Tuning] 버튼 - 모터의 전기 파라미터를 탐지하고 d 축과 q축 전류 제어기 이득을 설정하는 대화상자를 실행함
- [Mechanical Parameters Identification & Tuning] 버튼 - 모터의 기계 파라미터를 탐지하고 위치 제어기와 속도 제어기 이득을 설정하는 대화상자를 실행함
- [Motion Profile & Feed-forward Options] 버튼 - 모션 프로파일과 피드 포워드 옵션을 선택하는 대화상자를 실행함
- [Load from Disk] 버튼 - PC에서 구성 파라미터 파일 읽어 모터 드라이버로 다운로드 함
- [Save to Disk] 버튼 - 모터 드라이버에서 구성 파라미터를 읽어 PC의 파일로 기록 함
- [Expand Tree] 버튼 - 트리 메뉴를 모두 펼침
- [Collapse Tree] 버튼 - 트리 메뉴를 모두 접음

7.8.1 모터 드라이버 오토 튜닝

문워커 서보 모터드라이버를 사용하시려면 전원 - 모터드라이버 - 모터 연결을 끝낸 후

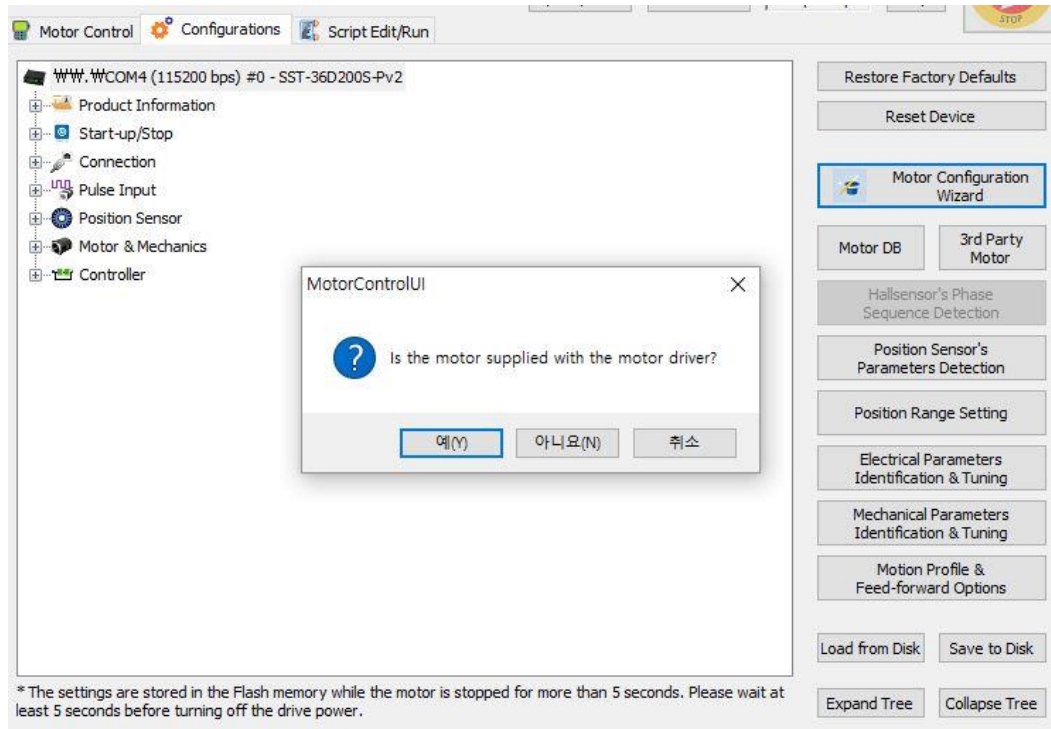
모터 드라이버 와 모터를 오토 튜닝 기능으로 매칭 시켜주어야 정상적으로 동작 합니다.

***주의: 오토튜닝 기능을 사용 하실 때는 모터가 무부하 상태로 있어야 합니다.**

Mechanical Parameters Identification & Tuning 버튼은 액추에이터등 기구부와 연결 한 후에 한 번 더 설정하여 값을 맞춰야 합니다.

7.8.1.1 모터 드라이버 설정 마법사

[Motor Configuration Wizard] 버튼을 눌러 모터 드라이브를 설정할 때는 설정 마법사를 사용할 수 있습니다.

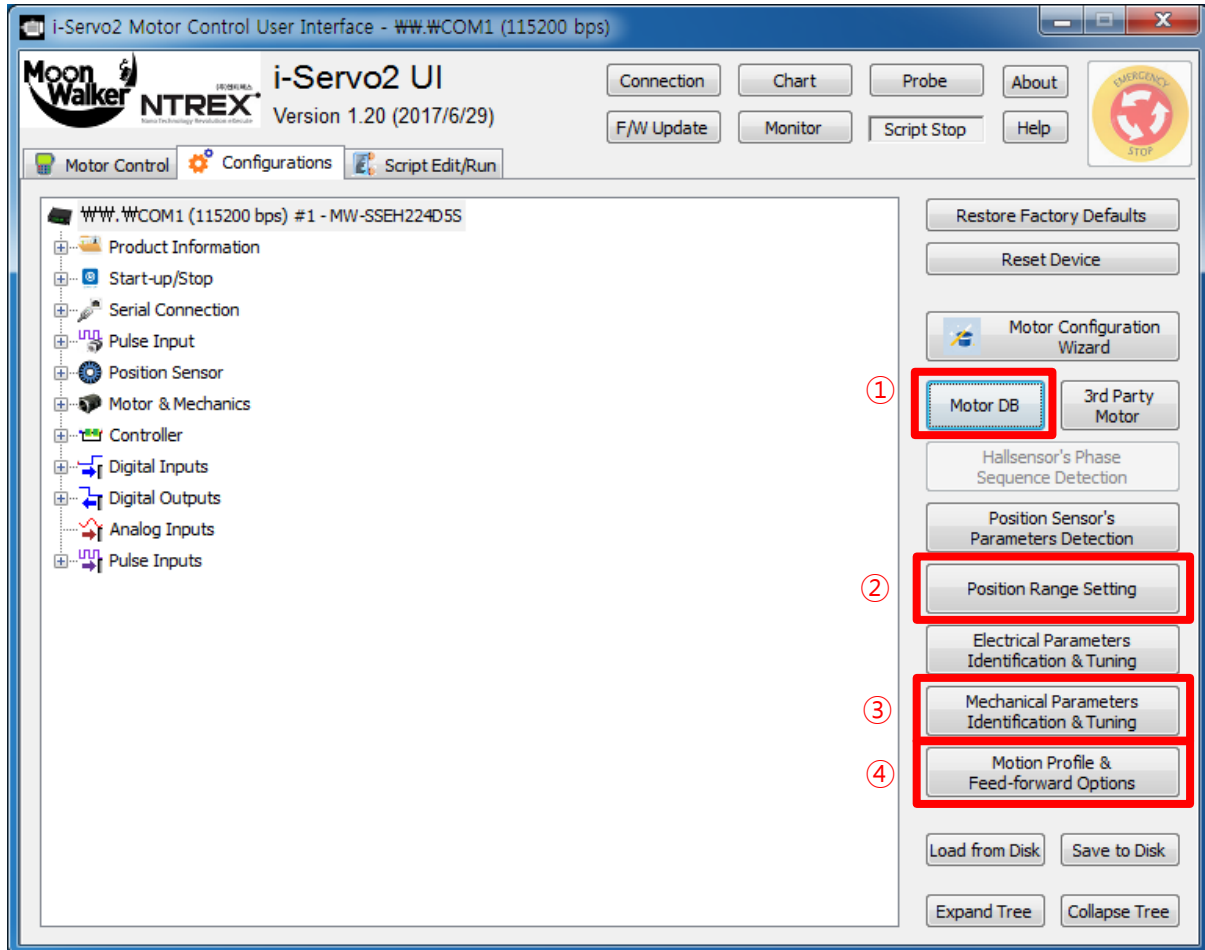


모터 드라이브가 지원하는 모터(이미 모터 상수들을 알고 있는 모터)를 사용한다면, 메시지 박스에서 [예] 버튼을 눌러 Motor DB에서 모터를 선택할 수 있습니다. 이 때는 모터 드라이브 설정 과정이 많이 생략됩니다.

모터 드라이브에서 모터 정보를 가지고 있지 않은 모터를 사용한다면, [아니오] 버튼을 눌러 모터를 구동하기 위한 주요 파라미터를 설정하는 세부 과정을 거쳐야 합니다.

7.8.1.2 Motor DB에 등록된 모터

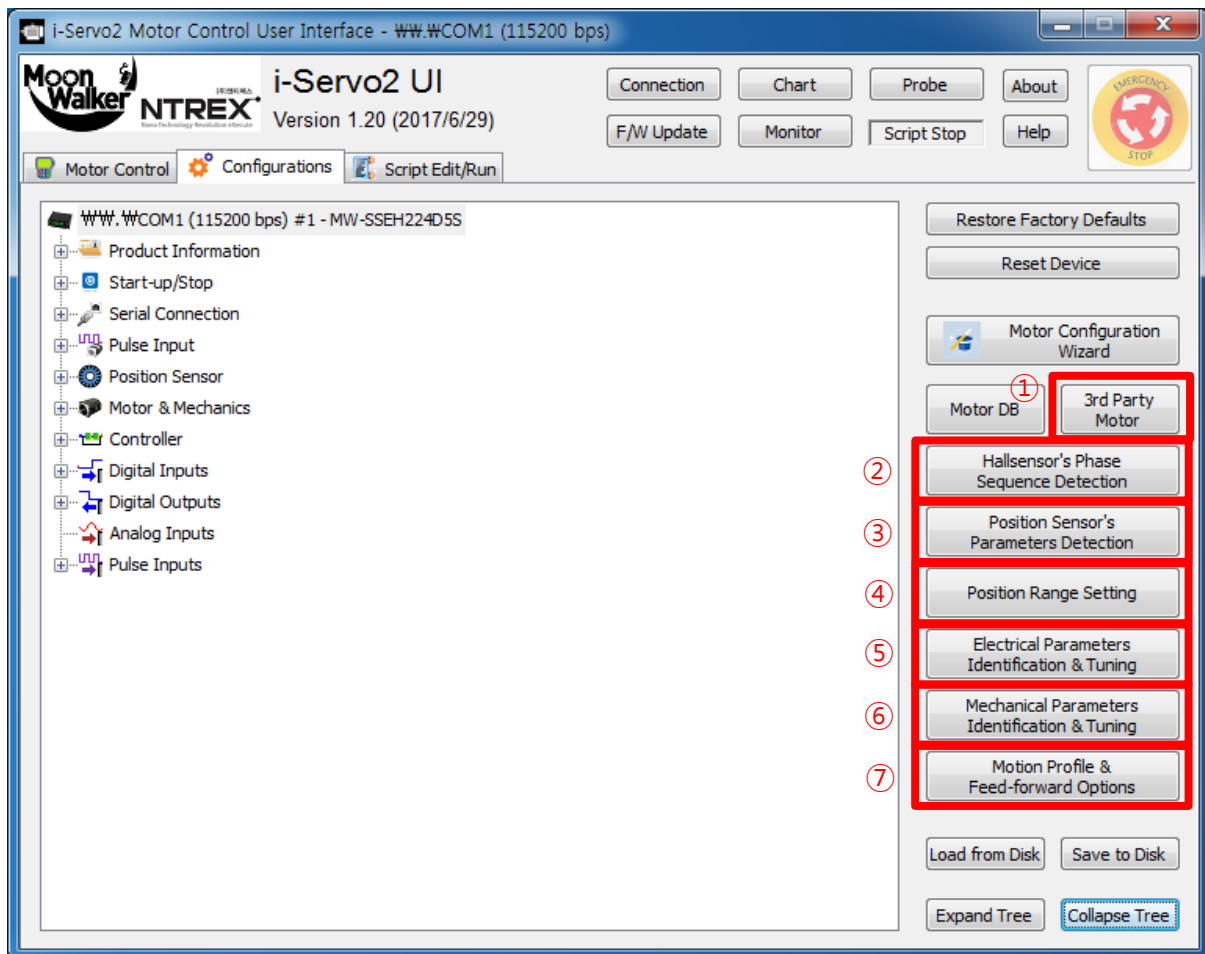
Motor DB에 등록된 모터를 사용하는 경우는 다음 과정을 따릅니다.



이 과정은 3rd Party Motor를 설정하는 과정의 일부이기 때문에 여기서는 설명을 생략하겠습니다.

7.8.1.3 Motor DB에 등록되지 않은 모터

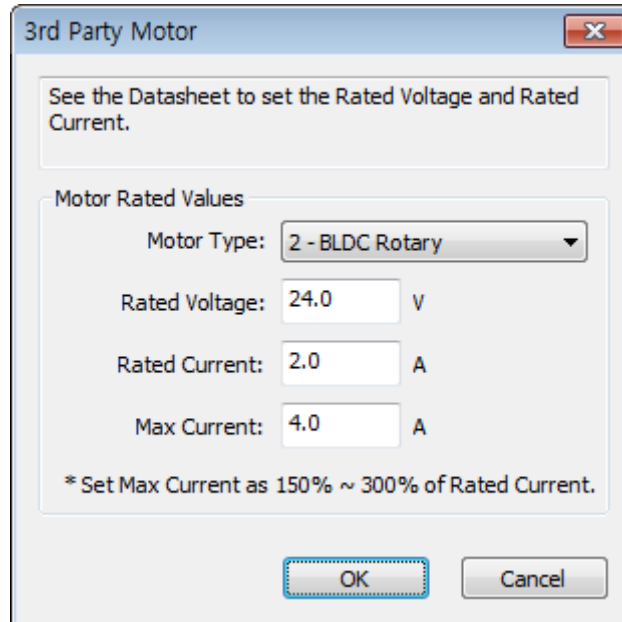
Motor DB에 등록되지 않은 모터를 사용하는 경우는 다음 과정을 따릅니다.



먼저 ①번의 [3rd Party Motor] 버튼을 눌러 모터 타입 및 정격 전압, 전류를 설정한 후 ②, ③, ④, ⑤, ⑥, ⑦ 과정을 순차적으로 수행합니다.

7.8.1.4 3rd Party Motor

[3rd Party Motor] 버튼을 눌러 모터의 종류와 정격 전압 및 전류를 설정합니다.



The dialog box titled "3rd Party Motor" contains the following elements:

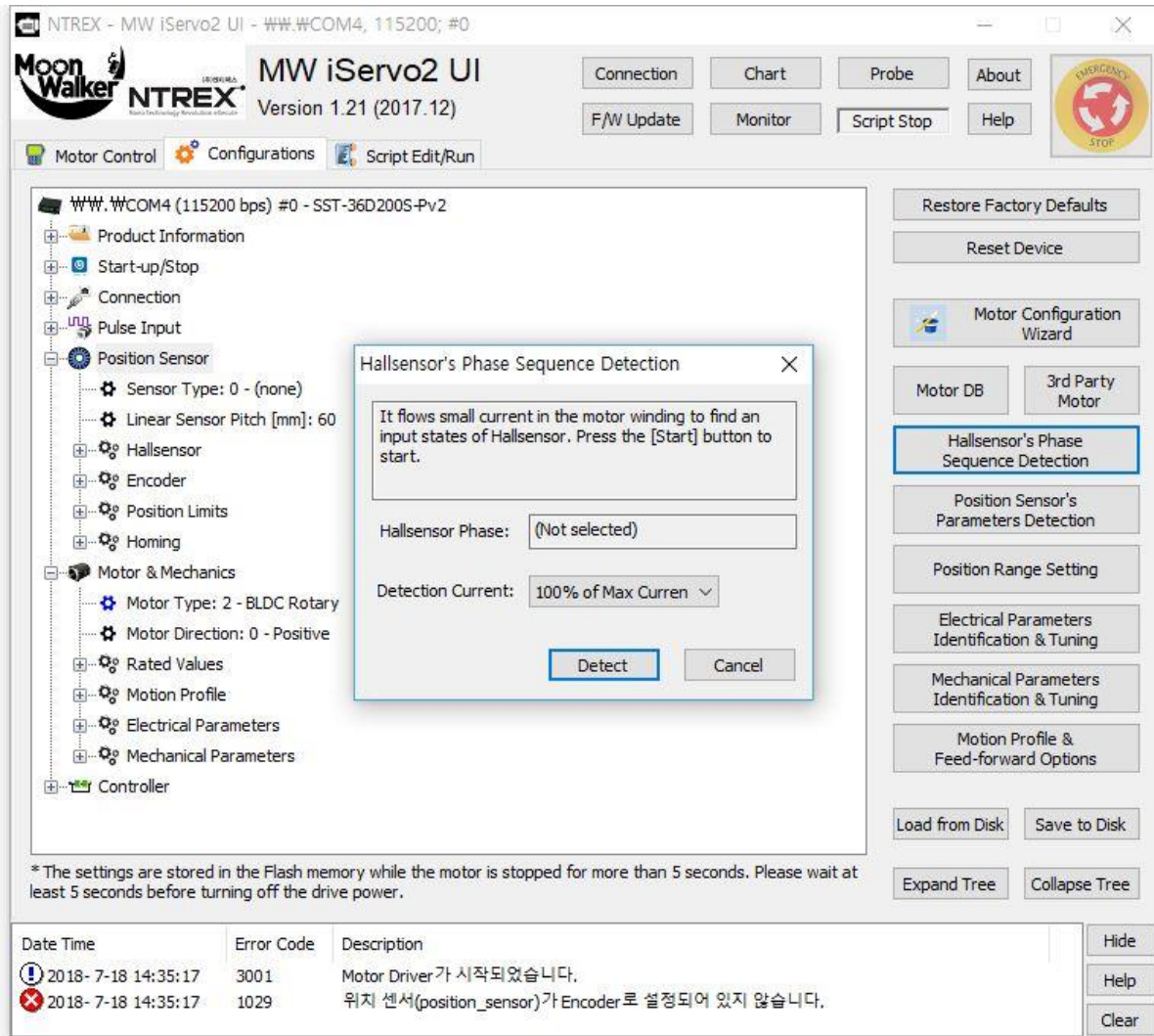
- A message box: "See the Datasheet to set the Rated Voltage and Rated Current."
- A section titled "Motor Rated Values" containing:
 - A "Motor Type" dropdown menu set to "2 - BLDC Rotary".
 - A "Rated Voltage" input field with the value "24.0" and a unit "V".
 - A "Rated Current" input field with the value "2.0" and a unit "A".
 - A "Max Current" input field with the value "4.0" and a unit "A".
- A note at the bottom: "* Set Max Current as 150% ~ 300% of Rated Current."
- "OK" and "Cancel" buttons at the bottom right.

모터 드라이버마다 지원하는 모터가 다릅니다. 모터 드라이버에 맞는 모터를 선택하여 Motor Type을 정해주시기 바랍니다. DC, BLDC, PMSM, STEP(2-phase, 3-phase) 가 있습니다. 또한 각 모터에 대한 Rotary/Linear 타입이 있습니다. Motor Type에서 사용하는 모터의 타입을 정확하게 선택해야 합니다. 만일 모터 타입이 정확하지 않다면, 이후 모터 설정 과정 및 모터 구동은 정상적으로 수행될 수 없습니다.

Rated Voltage에서는 모터에 가하는 정격 전압을, Rated Current에서는 모터에 연속으로 흘릴 수 있는 정격 전류(또는 데이터 시트의 continuous current)를 설정하고, Max Current(또는 데이터 시트의 peak current)는 보통 정격 전류의 300% 이내로 설정합니다.

7.8.1.5 HallSensor's Phase Sequence Detection

[HallSensor's Phase Sequence Detection] 버튼을 눌러 홀센서의 배열 순서를 찾습니다.

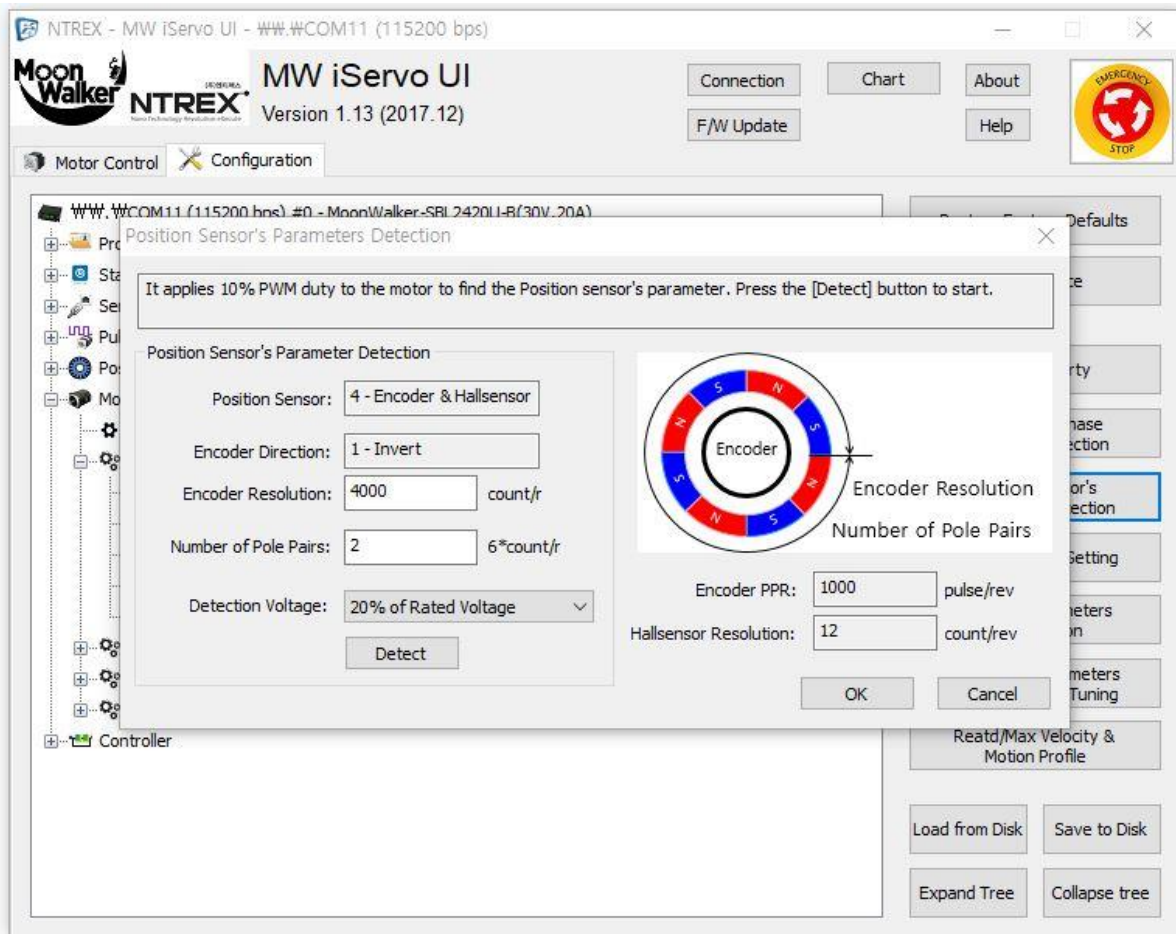


[Detect] 버튼을 누르면, 홀센서의 배열 순서를 찾기 위해 모터는 수 바퀴 회전합니다. 만일 모터가 일정한 속도로 회전하지 않거나 탐지에 실패하는 경우는 Detection Current를 높여서 다시 탐지 과정을 수행하면 됩니다.

***주의: 이 과정은 DC 모터나 STEP 모터에서는 사용할 수 없습니다. BLDC모터와 PMSM 모터에서만 사용 가능하도록 버튼이 활성화 됩니다.**

7.8.1.6 Position Sensor's Parameter Detection

[Position Sensor's Parameter Detection] 버튼을 눌러 Encoder와 Hallsensor 정보를 찾습니다



[Detect] 버튼을 누르면, Encoder와 Hallsensor 정보를 찾기 위해 모터는 수에서 수십 바퀴 회전하게 됩니다. 만일 모터가 일정한 속도로 회전하지 않거나 탐지에 실패하는 경우는 Detection Voltage를 높여 탐지 과정을 다시 수행하면 됩니다.

Encoder에서 Index 신호를 제공하는 경우는 Encoder Direction, Encoder Resolution, Number of Pole Pairs 값을 정확하게 찾습니다.

하지만 Index 신호를 사용할 수 없을 때는 Hallsensor 신호간의 균일하지 않은 시간차 데이터를 통계적으로 비교하여 Encoder Resolution, Number of Pole Pairs 값을 찾습니다. 이 경우에는 찾은 값의 비율은 맞을 수 있으나 실제 값과 다르게 찾을 수도 있습니다. 또한, DC 모터에서는 Hallsensor 신호가 없어 Encoder Resolution을 근본적으로 찾을 방법이 없기 때문에 회전속도로부터 대략의 값을 유추하게 됩니다.

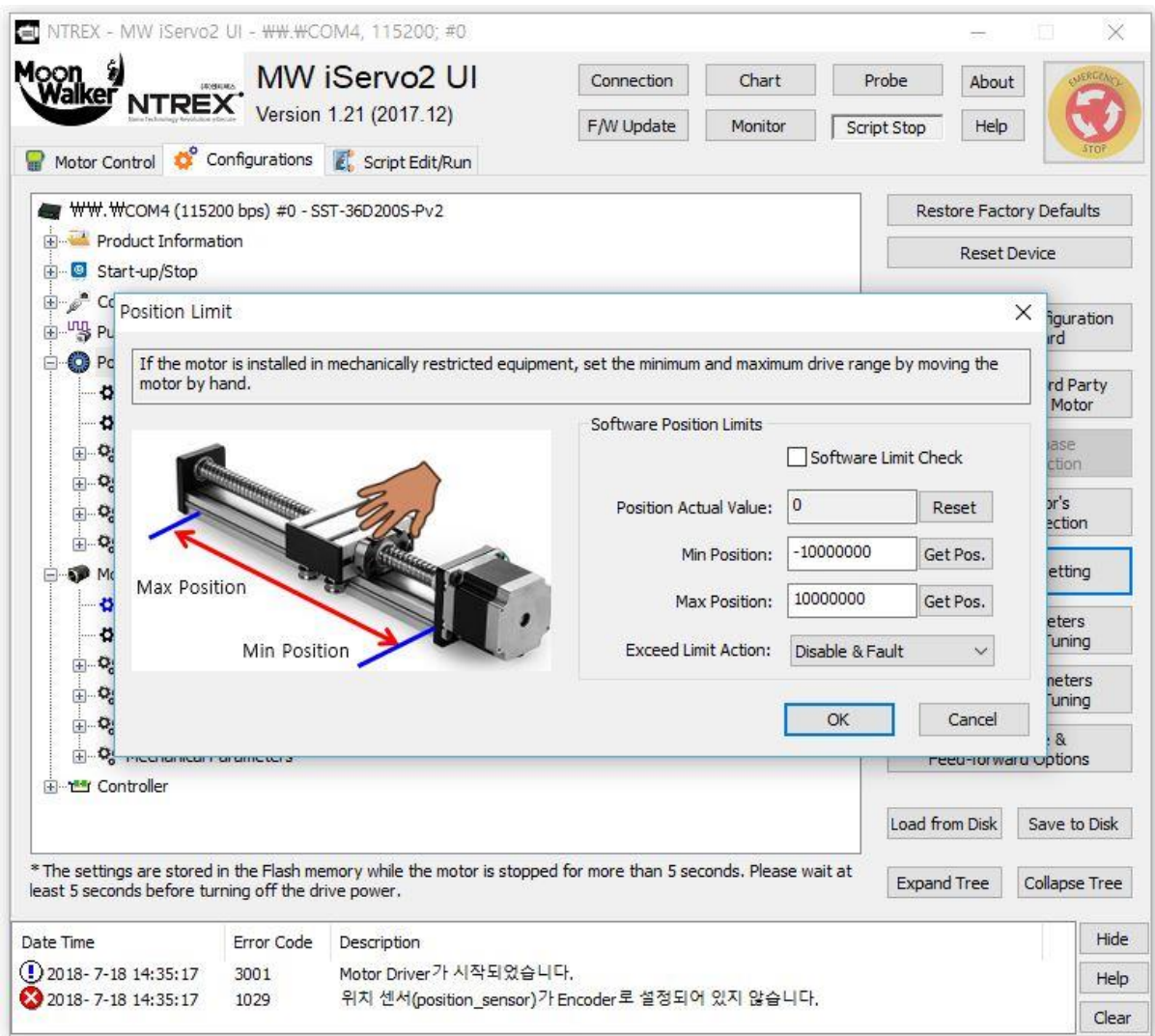
Linear 타입 모터인 경우, Encoder Resolution에 해당하는 길이만큼의 Linear Sensor Pitch 값을 사용자가 직접 측정하여 설정하거나 Encoder 데이터 시트를 참조하여 설정해야 합니다. 이 값을 측정할 때는 모터 제어를 Disable 상태로 만들어 손으로 모터를 직접 움직일 수 있도록 한 후, 손

으로 모터를 움직여 Position 값이 Encoder Resolution 만큼 움직였을 때의 이동 거리를자로 직접 측정하면 됩니다. 이 값이 잘못 설정되면, 모터의 위치나 속도가 올바르게 계산되지 않습니다.

***주의: Encoder에 Index 신호가 없는 경우, 통계적인 기법으로 Encoder와 Hallsensor 정보를 찾게 됩니다. 탐지 과정을 거쳐 찾은 값들은 모터의 데이터 시트를 참조하여 올바르게 설정되었는지 꼭 확인해야 한다. 만일 이 값들이 잘못 설정된 경우 모터의 회전 속도가 잘못 계산됩니다.**

7.8.1.7 Position Range Setting

[Position Range Setting] 버튼을 눌러 모터가 구동 가능한 최소/최대 범위를 설정합니다.



Linear 타입 모터 또는 모터가 리니어 모듈에 연결되어 모터의 구동 범위가 제한되는 경우, 모터가 구동 가능한 최소/최대 범위를 설정하여 이 범위를 벗어나지 못하도록 합니다. 만일 이동로봇의 바퀴와 같이 모터가 무한 회전 가능한 경우는 이 대화상자의 내용을 설정하지 않아도 됩니다.

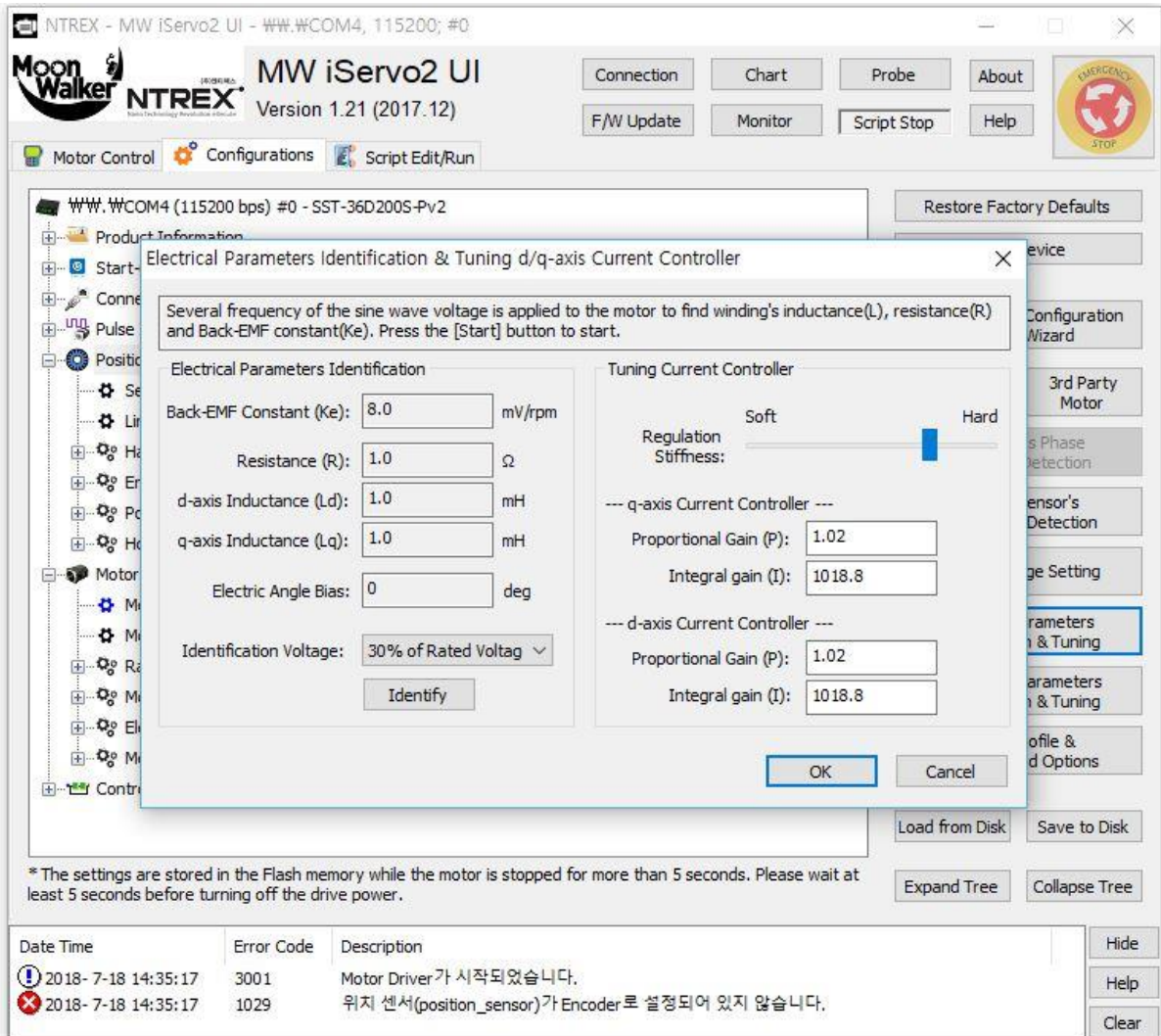
모터의 구동 범위를 제한해야 할 경우, Software Limit Check 체크박스를 체크하고 모터가 구동 가능한 Min Position 및 Max Position을 설정합니다. 이 때 손으로 구동부를 움직여 [Get Pos.] 버튼을 누름으로 현재 Encoder 값을 Min/Max Position 값으로 설정할 수 있습니다. 만일 [Reset] 버튼을 누르면 현재 위치를 Encoder 값 0으로 설정합니다.

Exceed Limit Action은 모터의 위치 값이 Min/Max Position을 벗어났을 때의 동작을 설정할 수 있습니다.

***주의:** 만일 Linear 타입 모터거나 모터가 리니어 모듈에 연결되어 이동 범위가 제한된 경우는 모터의 구동 범위를 꼭 올바르게 설정해야 합니다. 이 값이 올바르게 설정되지 않은 경우, 이후의 모터 파라미터 탐지 과정에서 모터가 스톱퍼나 기타 기구부에 충돌할 수 있습니다.

7.8.1.8 Electrical Parameters Identification & Tuning

[Electrical Parameters Identification & Tuning] 버튼을 눌러 모터의 전기적 파라미터를 찾고 이와 관련된 전류 제어기의 이득을 설정합니다.



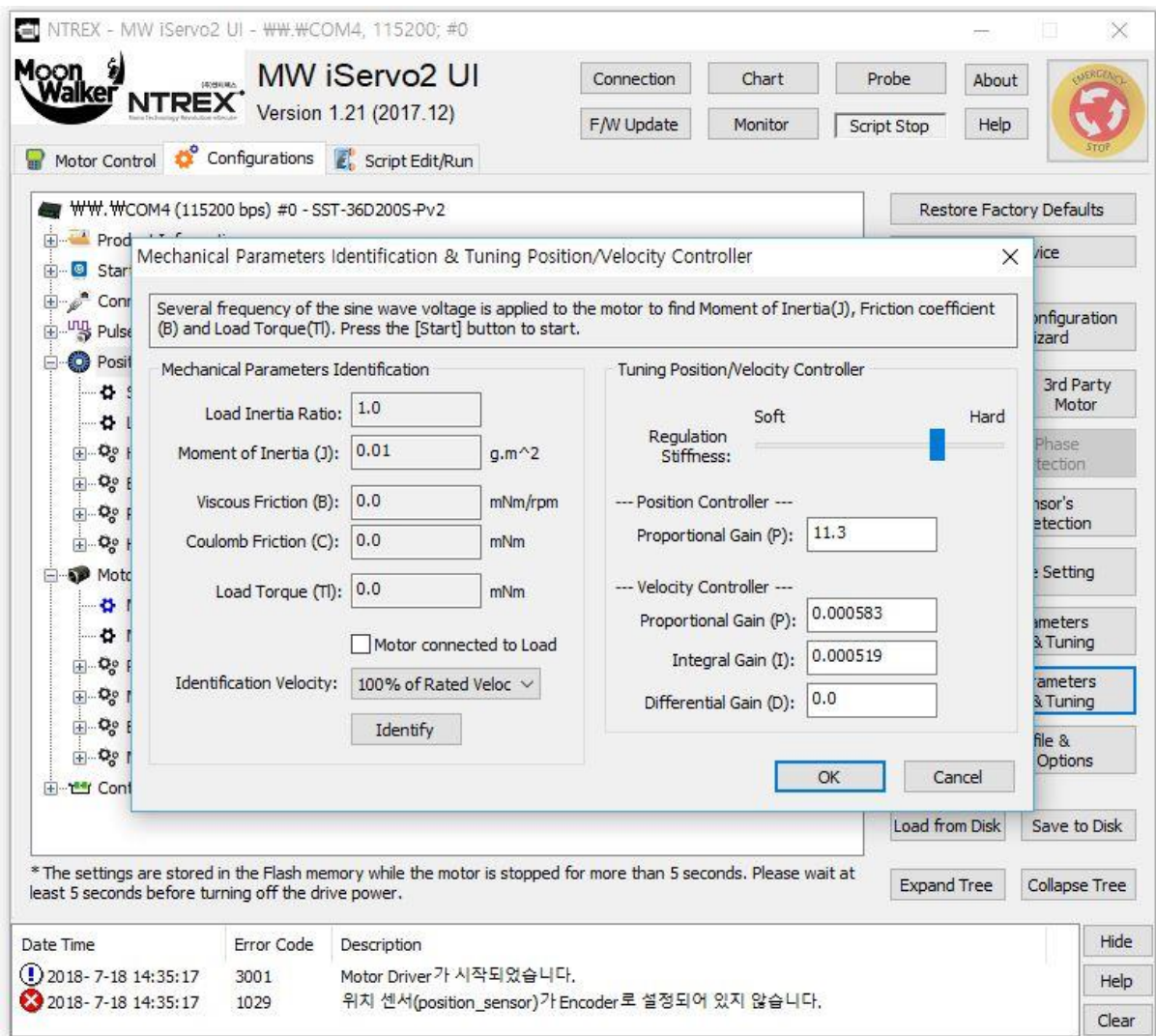
[Identify] 버튼을 누르면 모터가 좌우로 회전하면서 모터의 전기 파라미터인 Back-EMF Constant, Resistance, d/q-axis Inductance 값을 찾게 됩니다. 만일 모터가 좌우로 움직이지 못하였거나 식별에 실패하는 경우는 Identification Voltage를 높여서 다시 식별 과정을 수행해 보도록 합니다.

식별 과정을 성공적으로 수행한 경우, 오른 쪽의 Regulation Stiffness 슬라이드 바를 좌우로 움직여 전류 제어기의 이득을 설정합니다. Soft로 움직이면 이득이 낮게 설정되어 모터가 부드럽게 회전합니다. 반대로 Hard로 움직이면 이득이 높게 설정되어 모터의 전류 추종 성능이 좋아지지만 진동이 발생할 수 있습니다. 만일 전류제어기에서 이득을 너무 높게 설정하였다면, 모터는 가청 주파수 대(1kHz 근처)의 진동음을 내면서 진동하게 됩니다. 이 때는 소리가 나지 않을 때까지 슬라이드 바를 왼쪽으로 움직여 이득을 낮추도록 합니다.

***주의:** 모터의 전기 파라미터를 찾을 때는 모터가 기구부와 분리되어 있어야 합니다. 만약 모터가 액추에이터등 기구부와 연결되어있다면 탐지 과정에서 모터가 스톱퍼나 기타 기구부에 충돌할 수 있습니다. 또한 모터가 갑자기 움직일 수 있으므로 사용자가 다치지 않도록 주의해야 합니다.

7.8.1.9 Mechanical Parameters Identification & Tuning

[Mechanical Parameters Identification & Tuning] 버튼을 눌러 모터의 기계적 파라미터를 찾고 이와 관련된 위치/속도 제어기의 이득을 설정합니다.



모터의 기계 파라미터를 찾을 때는 모터가 기구부와 연결되어 있는 상태인지 또는 분리되어 있는 상태인지를 구분해야 합니다. 만일 Motor connected to Load 체크박스를 체크한 상태라면 모터가 기구부와 연결되어 있는 상태로 보고, 현재 설정된 Moment of Inertia 값은 모터 회전자의 관성으로 고려하여 기구부(Load) 관성과 비율을 구하게 됩니다.

$$\text{Load Inertia Ratio} = \frac{\text{Load Inertia} + \text{Motor Inertial}}{\text{Motor Inertial}}$$

만일 체크가 해제된 상태라면 Load Inertia Ratio는 1이 되고, Moment of Inertial 값은 모터 회전자의 관성이 됩니다.

[Identify] 버튼을 누르면 모터가 가감속을 반복하면서 좌우로 이동하여 모터의 기계 파라미터인 Moment of Inertia, Viscous Friction, Coulomb Friction 값을 찾게 됩니다. 만일 모터가 좌우로 움직이지 못하였거나 식별에 실패하는 경우는 Identification Velocity를 높여서 다시 식별 과정을 수행해 보도록 합니다.

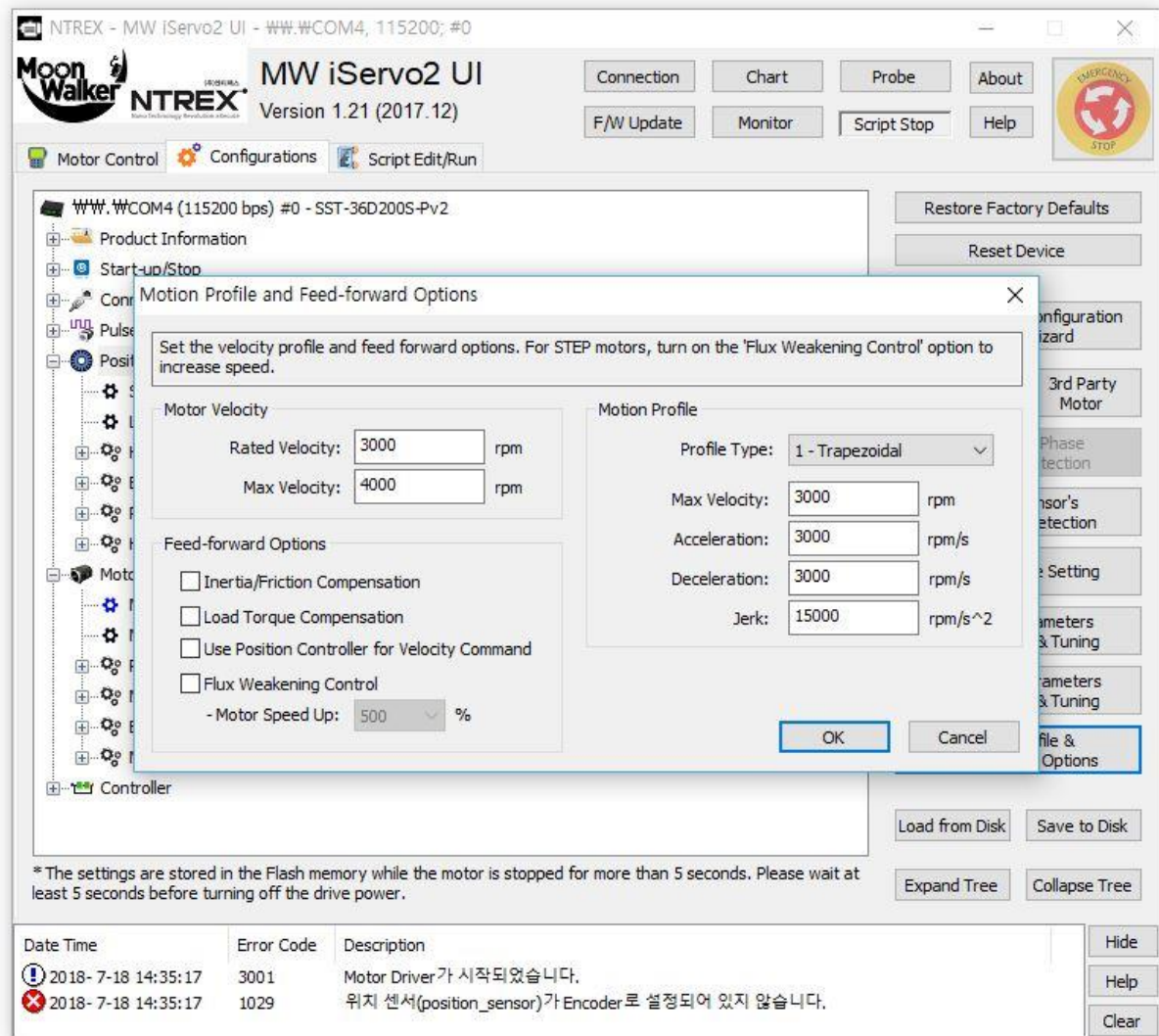
식별 과정을 성공적으로 수행한 경우, 오른 쪽의 Regulation Stiffness 슬라이드 바를 좌우로 움직여 위치/속도 제어기의 이득을 설정합니다. Soft로 움직이면 이득이 낮게 설정되어 모터가 부드럽게 회전합니다. 반대로 Hard로 움직이면 이득이 높게 설정되어 모터의 위치/속도 추종 성능이 좋아지지만 진동이 발생할 수 있습니다. 만일 위치/속도 제어기에서 이득을 너무 높게 설정하였다면, 모터는 저주파(수십~수백Hz)로 진동하게 됩니다. 이 때는 진동이 없어질 때까지 슬라이드 바를 왼쪽으로 움직여 이득을 낮추도록 합니다.

***주의: 이 값을 찾을 때 모터는 기구부와 연결되어 있어야 합니다. 모터는 진동하면서 값을 찾기 때문에 사람이 다치거나 기구부가 파손되지 않도록 주의해야 합니다.**

***주의: 위치 제어기와 속도 제어기의 이득을 설정할 때, PID값을 수동으로 각자 설정하는 것보다 Regulation Stiffness 슬라이드 바를 움직여 한꺼번에 설정하는 것을 권장합니다.**

7.8.1.10 Motion Profile & Feed-forward Options

[Motion Profile & Feed-forward Options] 버튼을 눌러 모터의 동작 프로파일을 설정하고 피드 포워드 옵션을 설정합니다.



Rated Velocity와 Max Velocity는 모터의 정격 전압과 Electrical Parameters Identification 과정에서 알아낸 Back-EMF constant에 의해 적절하게 계산된 값입니다.

Motion Profile 그룹에서, Profile Type을 설정하고 프로파일의 Max Velocity, Acceleration, Deceleration, Jerk 값을 설정합니다. [Electrical Parameters Identification & Tuning], [Mechanical Parameters Identification & Tuning] 과정을 올바르게 수행하였다면, 이 값들은 모터의 각종 파라미터를 기반으로 모터가 정격 범위 내에서 구동할 수 있도록 계산된 값으로 설정되어 있습니다.

※ Profile Type을 '1-Trapezoidal' 또는 '2-Sinusoidal' 중 하나로 설정하는 것을 권장하며, Max Velocity, Acceleration, Deceleration, Jerk 값들을 제어기가 계산한 값 이상으로 올리는 것은 모터에 무리를 가하게 되므로 권장하지 않지만, 이 값들을 낮추는 것은 문제가 되지 않습니다.

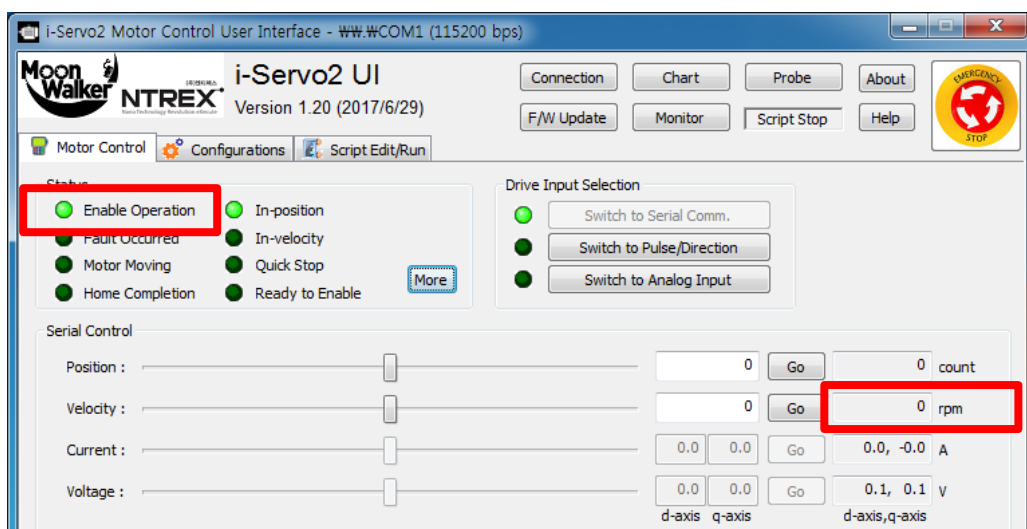
Feed-forward Options은 상기 그림에서와 같이 Inertia/Friction Compensation, Use Position Controller for Velocity Command 옵션을 켜서 사용합니다. 이 두 옵션은 제어기의 위치와 속도 추종 성능을 높일 수 있습니다. 만일 모터가 수직 부하를 구동하는 경우, 중력에 의한 부하를 상쇄시키기 위해 Load Torque Compensation를 켜서 사용하도록 합니다. 이 옵션을 켜게 되면 모터가 진동할 수도 있는데, 이 때는 옵션을 끄거나 Mechanical Parameters Identification & Tuning Position/Velocity Controller 대화상자에서 Regulation Stiffness를 좀더 낮추도록 합니다.

STEP 모터의 경우는 Flux Weakening Control을 켜고 모터의 속도를 정격 속도 대비 %단위로 하여 사용할 수 있습니다. 하지만 속도가 늘어난 것에 반비례로 모터의 토크는 감소하기 때문에 적절한 값으로 Motor Speed Up의 % 값을 설정해야 합니다. 예를 들자면, 모터의 구동 거리가 긴 경우, Motor Speed Up 값을 500%로 설정하여 빠른 속도로 이동할 수 있습니다. 반면, 모터의 구동 거리가 짧은 경우, 모터의 토크를 키워 빠르게 가감속 하여야 하기 때문에 Motor Speed Up 값을 100%로 설정하여 빠르게 가감속 하여 짧은 거리를 이동할 수 있습니다.

7.8.2 세부 튜닝

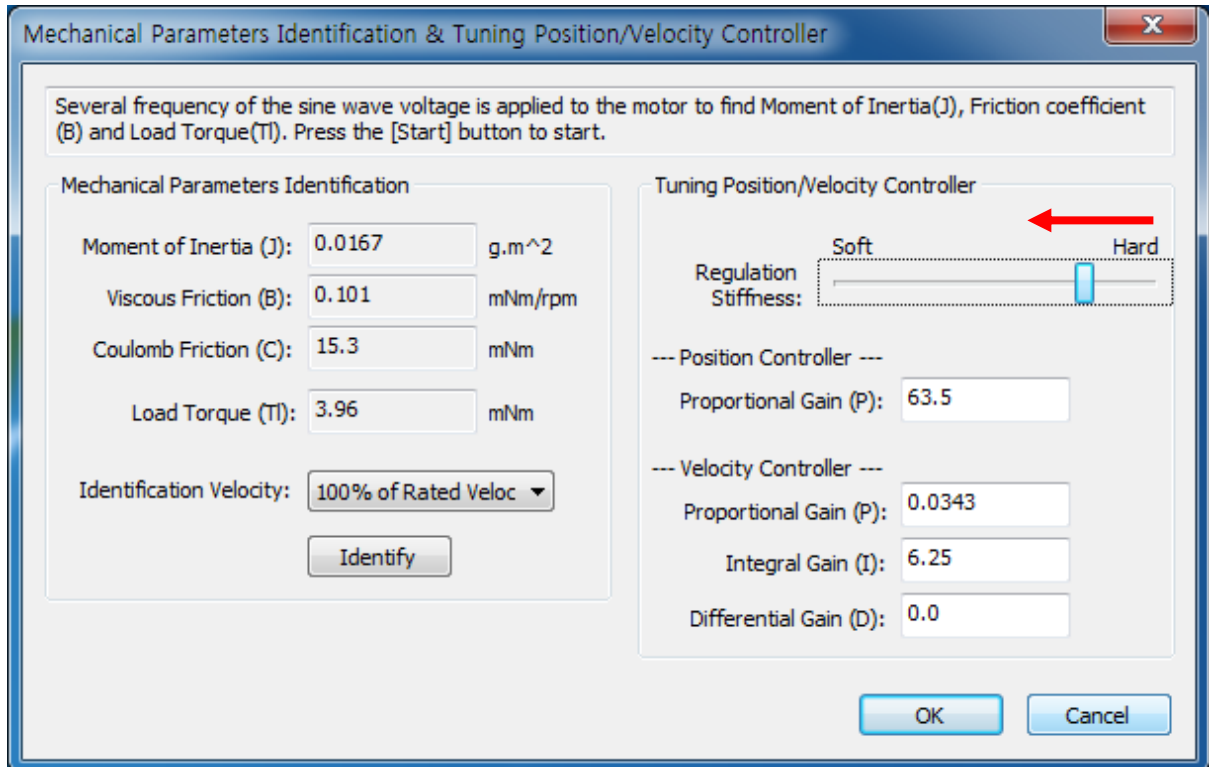
7.8.2.1 정지 상태에서 진동 발생하는 경우

UI의 Motor Control 탭에서 [Enable] 버튼을 눌러 모터를 Enable 하고 모터를 움직이지 않는 상태에서 진동이 발생하는 경우가 있을 수 있습니다.



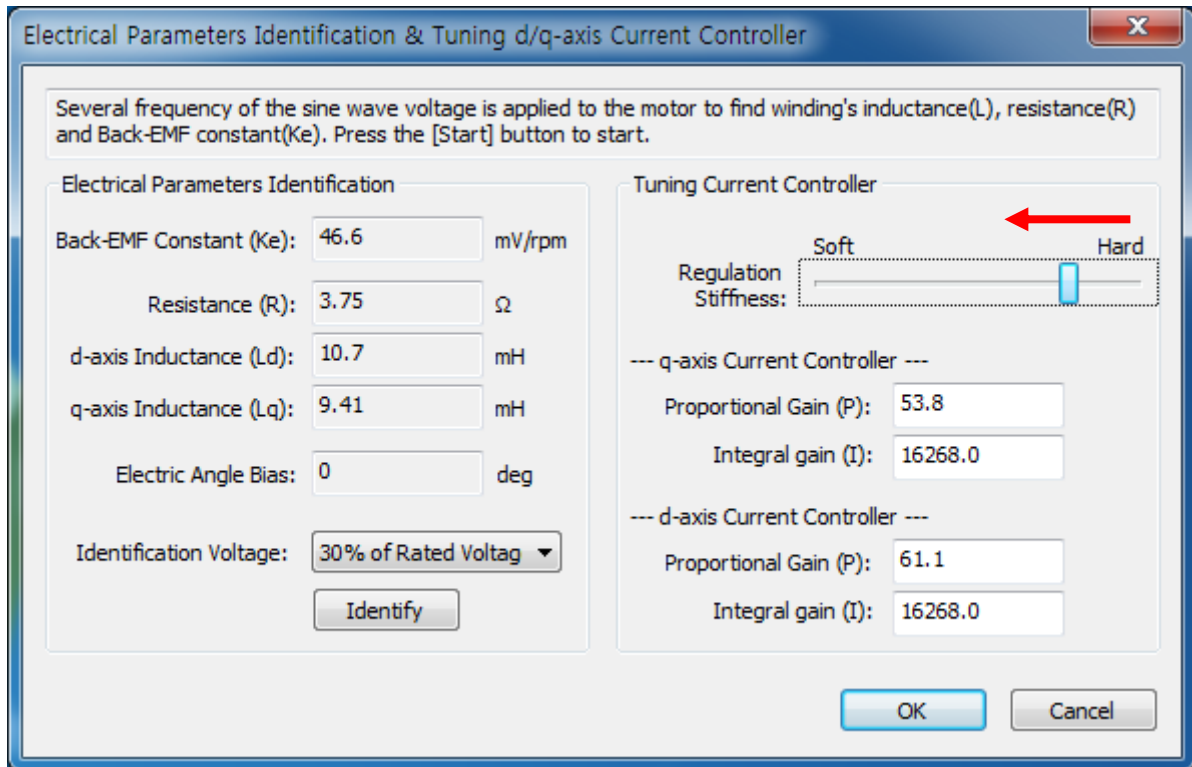
진동은 종종 벨트 타입의 리니어 모듈을 구동하는 모터에서 Current Controller의 Regulation Stiffness를 최대한 Hard로 설정하였거나 Position/Velocity Controller의 Regulation Stiffness를 최대한 Hard로 설정하였을 때 발생합니다. 이 때는 Tuning Current Controller와 Tuning Position/Velocity Controller의 Regulation Stiffness를 Soft 방향으로 진동이 없어질 때까지 조금씩 낮춰 보도록 합니다.

(1) Configurations 탭의 [Mechanical Parameters Identification & Tuning] 버튼을 누릅니다.



Tuning Position/Velocity Controller의 Regulation Stiffness를 Soft 방향으로 진동이 없어질 때까지 낮춰줍니다.

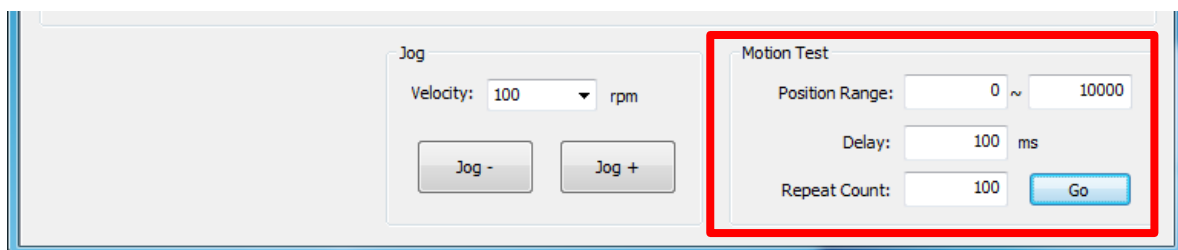
(2) Configurations 탭의 [Electrical Parameters Identification & Tuning] 버튼을 누릅니다.



Tuning Current Controller의 Regulation Stiffness를 (1)에서 낮춘 만큼 낮춰줍니다.

7.8.2.2 구동 상태에서 진동 발생하는 경우

UI의 Motor Control 탭에서 [Enable] 버튼을 눌러 모터를 Enable 하고 모터가 특정 구간을 반복하도록 Motion Test 그룹에서 Position Range, Delay, Repeat Count 값을 설정 합니다. 그리고 [Go] 버튼을 눌러 모터가 반복 구동하도록 합니다.



이후 “정지 상태에서 진동이 발생하는 경우” 절에서와 같은 방법으로 진동이 없어질 때까지 이득을 조절합니다.

7.8.3 PID 제어기 이해

PID(Proportional, Integrate, Derivative)제어기는 비례-적분-미분 제어기로서, 실제 산업현장에서 가장 많이 사용되는 제어기법이며, 본 모터 제어기에도 위치, 속도, 전류 제어기에는 PID 형태의 제

여기를 기본으로 채용하고 있습니다.

PID 제어기는 오차를 수학적으로 다음 식과 같이 계산하여 모터의 제어 값을 계산합니다.

$$u = K_p e + K_I \int e dt + K_D \frac{de}{dt}$$

여기서 K_p 는 오차신호에 곱해지는 비례이득이며, K_I 는 오차신호를 적분한 값에 곱해지는 적분이득이며, K_D 는 오차신호를 미분한 값에 곱해지는 미분이득입니다.

모터 제어기의 성능은 PID 제어기의 이득을 어떻게 정하느냐에 따라 달라지는데, 다음의 설명을 참고하여 위치, 속도, 전류 제어기의 이득을 설정하도록 합니다.

- 비례이득 K_p 의 값은 응답의 상승시간(rise time)을 줄일 수 있습니다. 이득을 높이면 사용자가 내린 명령을 모터가 더 빠르게 추종하게 됩니다. 하지만 너무 높이면 모터가 진동하거나 발산합니다.
- 적분이득 K_I 의 값은 정상상태 오차(steady state error)를 제거하는 효과를 가지고 있지만 과도응답 특성을 좋지 않게 만들 수 있습니다. 이득을 높이면 정상상태 오차를 더 빠르게 없애 주지만, 너무 높이면 모터가 진동하거나 발산합니다.
- 미분이득 K_D 의 값은 오버슈트(overshoot)를 줄이고 과도응답 특성을 향상시킬 수 있습니다. 하지만 측정 신호에 노이즈가 있는 경우, 이득을 높이면 노이즈에 민감하게 반응하여 제어기를 불안정하게 만들 수 있습니다.

상기 특성을 다음 표와 같이 정리할 수 있습니다.

응답성 이득	상승시간 (rise time)	오버슈트 (overshoot)	정착시간 (settling time)	정상상태오차 (steady state error)
비례이득 (K_p) 증가	감소	증가	-	감소
적분이득 (K_I) 증가	감소	증가	증가	빠르게 제거
미분이득 (K_D) 증가	-	감소	감소	-

7.8.4 수동 Gain 조정

본 모터 드라이버는 자동으로 Gain이 조정이 됩니다.

***주의: 수동으로 Gain을 조정하는 것은 권장 하지 않습니다.**

모터 제어기의 이득을 수동으로 조정하기 위해서는 다음의 절차를 따릅니다

7.8.4.1 q축 전류 제어기의 이득을 설정

1. 비례이득(K_P)과 적분이득(K_I)을 0으로 설정합니다.
2. 적분이득을 50정도의 값에서 시작하여 20%정도의 오버슈트(overshoot)가 발생할 때까지 올립니다. 올릴 때는 50, 100, 200, 400, 800 과 같이 2배씩 증가시키도록 합니다.
3. 비례이득을 0.1에서 시작하여 오버슈트가 없어질 때까지 올립니다. 올릴 때는 0.2, 0.5, 1, 2, 5 와 같이 2배씩 증가시킵니다.
4. 원하는 응답 특성이 나올 때까지 상기 과정을 반복합니다.

7.8.4.2 d축 전류 제어기의 이득을 설정

※ q축 전류 제어기의 이득 설정방법과 유사하게 설정하거나, q축 전류 제어기에서 설정한 이득을 복사하여 동일하게 사용합니다.

7.8.4.3 속도 제어기의 이득을 설정

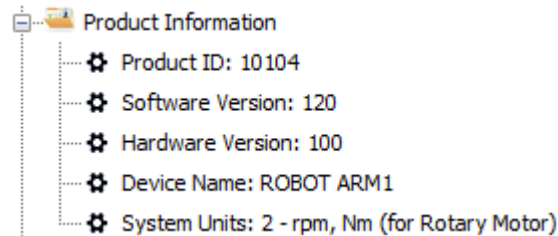
1. 비례이득(K_P)과 적분이득(K_I)을 미분이득(K_D)을 0으로 설정합니다.
2. 비례이득을 0.01정도의 값에서 시작하여 20%정도의 오버슈트(overshoot)가 발생할 때까지 올립니다. 올릴 때는 0.01, 0.02, 0.04, 0.08과 같이 2배씩 증가시키도록 합니다.
3. 적분이득을 0.1정도의 값에서 시작하여 정상상태 오차가 원하는 시간 내에 없어질 때까지 올립니다.
4. 미분이득을 0.00001에서 시작하여 오버슈트가 없어질 때까지 올립니다.

7.8.4.4 위치 제어기의 이득을 설정

- 비례이득(K_P)을 1정도의 값에서 시작하여 오버슈트가 발생하지 않을 때까지 올립니다. 올릴 때는 2, 4, 8, 16과 같이 2배씩 증가시키다가. 오버슈트가 발생하는 순간부터는 미세하게 조정하도록 합니다.

7.8.5 Product Information

Product Information 그룹은 현재 연결된 제품 ID, 모터 드라이버와 펌웨어 버전을 확인할 수 있습니다. 이 값들은 사용자가 임의로 변경할 수 없습니다. 그리고 장치의 이름과 모터 드라이버에서 사용하는 단위를 설정할 수 있습니다.



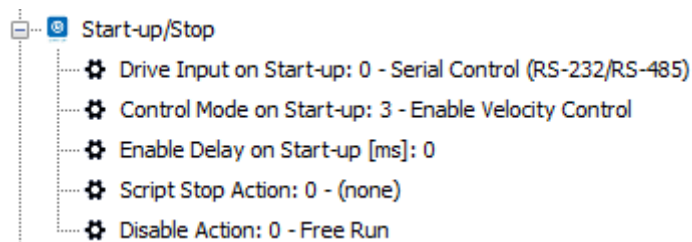
※ 자세한 설명은 오브젝트 항목에 있습니다.

이 그룹의 항목과 관련되는 오브젝트는 다음과 같습니다.

- Product ID - product_id
- Software Version - software_version
- Hardware Version - hardware_version
- Device Name - device_name
- System Units - units

7.8.6 Start-up/Stop

Start-up/Stop 그룹은 모터 드라이버에 전원이 투입되어 처음 시작할 때, 또는 스크립트가 시작되거나 종료될 때의 동작을 결정합니다.



※ 자세한 설명은 오브젝트 항목에 있습니다.

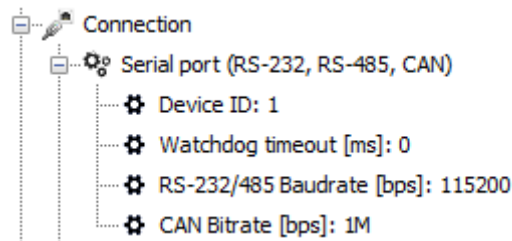
이 그룹의 항목과 관련되는 오브젝트는 다음과 같습니다.

- Drive Input on Start-up - startup_drive_input
- Control Mode on Start-up - startup_control_mode

- Enable Delay on Start-up - startup_enable_delay
- Script Stop Action - script_stop_action
- Disable Action - disable_oper_action

7.8.1 Connection – Serial port

Connection - Serial port 그룹은 모터 드라이버 ID, Serial 통신 속도, 와치독 등을 설정하고 확인할 수 있습니다. 이 그룹에서 Device ID, CAN Bitrate, RS-232/485 Baudrate 값을 변경한 경우에는 모터 드라이버를 재시작해야 변경된 값이 반영됩니다.



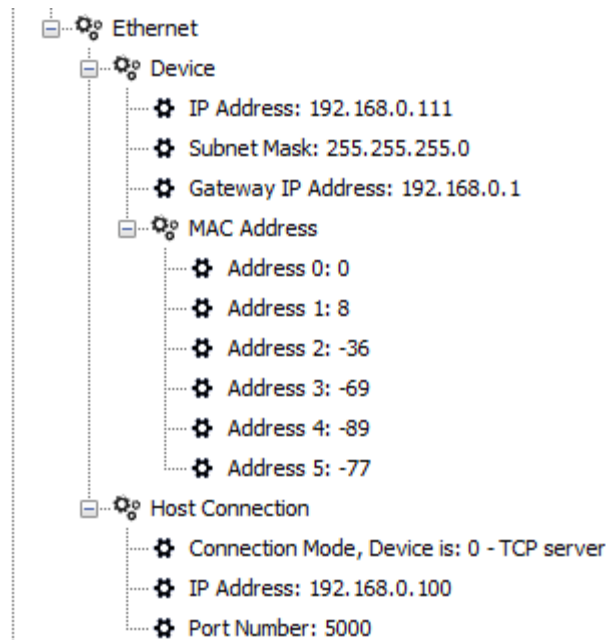
※ 자세한 설명은 오브젝트 항목에 있습니다.

이 그룹의 항목과 관련되는 오브젝트는 다음과 같습니다.

- Device ID - device_id
- Watchdog timeout - serial_watchdog
- RS-232/485 Baudrate - serial_baudrate[0]
- CAN Bitrate - serial_baudrate[2]

7.8.2 Connection – Ethernet

Connection – Ethernet 그룹은 이더넷으로 모터 드라이버에 연결하기 위한 IP 주소와 포트 번호 등을 설정할 수 있습니다. 이 그룹의 값을 변경한 경우에는 모터 드라이버를 재시작해야 변경된 값이 반영됩니다.



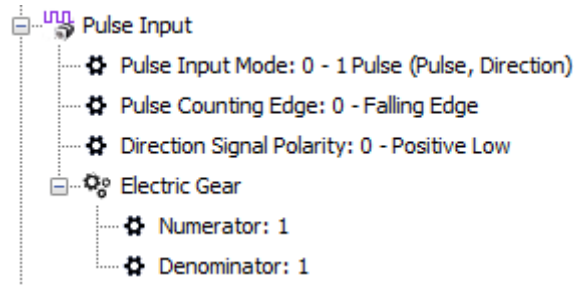
※ 자세한 설명은 연결 항목에 있습니다.

이 그룹의 항목과 관련되는 오브젝트는 다음과 같습니다.

- Device - IP Address - ip_address
- Device - Subnet Mask - subnet_mask
- Device - Gateway IP Address - gateway_ip
- Device - MAC Address - mac_address[0~5]
- Host Connection - Connection Mode - connection_mode
- Host Connection - IP Address - connection_ip
- Host Connection - Port Number - connection_port

7.8.3 Pulse Input

Pulse Input 그룹에는 모터의 위치 구동 명령을 디지털 입력 포트의 Pulse/Direction 신호로 입력 받기 위한 입력 모드와 신호의 레벨 등을 설정합니다.



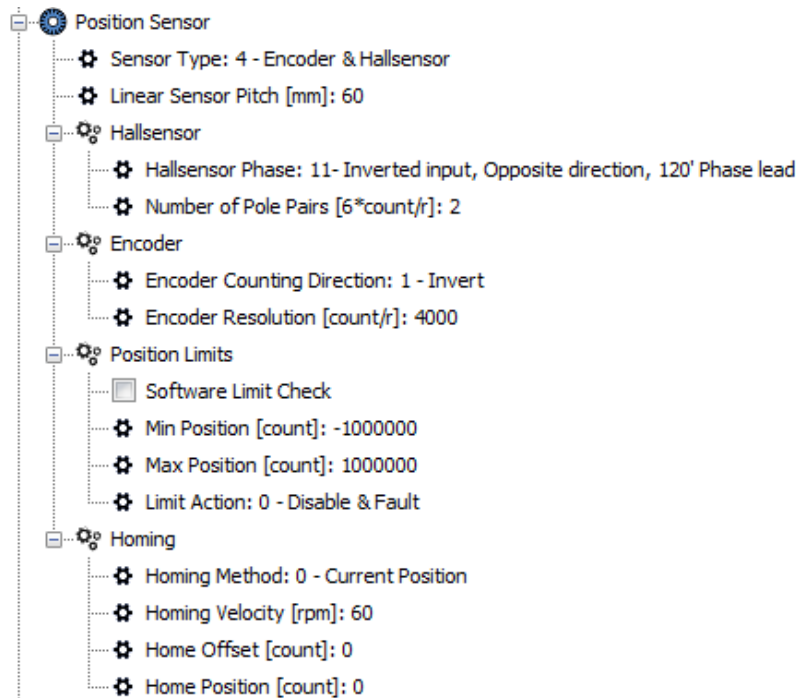
※ 자세한 설명은 오브젝트 항목에 있습니다.

이 그룹의 항목과 관련되는 오브젝트는 다음과 같습니다.

- Pulse Input Mode - pulse_input_mode
- Pulse Counting Edge - pulse_count_edge
- Direction Signal Polarity - dir_signal_polarity
- Electric Gear – Numerator - gear_numerator
- Electric Gear – Denominator - gear_denominator

7.8.4 Position Sensors

Position Sensor 그룹은 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.



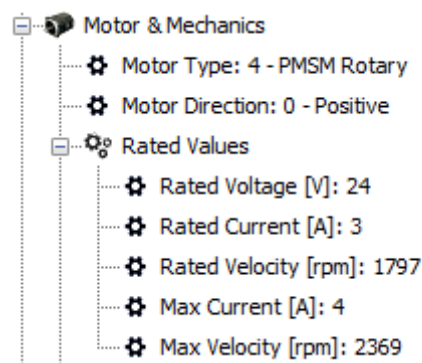
※ 자세한 설명은 모터드라이버 오토 튜닝 및 오브젝트 항목에 있습니다.

이 그룹의 항목과 관련되는 오브젝트는 다음과 같습니다.

- Sensor Type - position_sensor
- Linear Sensor Pitch - linear_sensor_pitch
- Hallsensor – Hallsensor Phase - hallsensor_phase
- Hallsensor – Number of Pole Pairs - no_pole_pairs
- Encoder – Encoder Counting Direction - encoder_direction
- Encoder - Encoder Resolution - encoder_resolution
- Position Limits – Software limit Check - soft_limit_check
- Position Limits – Min Position - min_position
- Position Limits – Max Position - max_position
- Position Limits – Limit Action - limit_action
- Homing – Homing Method - homing_method
- Homing – Homing Velocity - homing_velocity
- Homing – Home Offset - home_offset
- Homing – Home Position - home_position

7.8.5 Motors & Mechanics

Motor & Mechanics 그룹은 사용자가 사용하고자 하는 모터 자체의 특성에 대한 설정이다. 모터에 흐르는 최대 연속 전류, 모터를 구동하는 최대 전압, 모터의 최대 회전 속도, 모터의 회전 가속도, 모터의 회전 감속도를 설정할 수 있다.



※ 자세한 설명은 모터드라이버 오토 튜닝 및 오브젝트 항목에 있습니다.

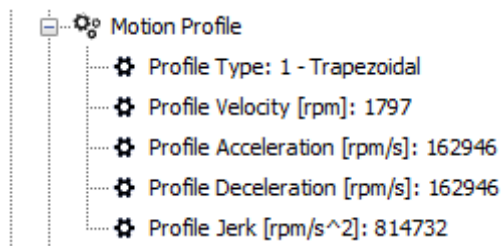
이 그룹의 항목과 관련되는 오브젝트는 다음과 같습니다.

- Motor Type - motor_type
- Motor Direction - motor_direction

- Rated Values – Rated Voltage - rated_voltage
- Rated Values – Rated Current - rated_current
- Rated Values – Rated Velocity - rated_velocity
- Rated Values – Max Current - max_current
- Rated Values – Max Velocity - max_velocity

7.8.6 Motors & Mechanics – Motion Profile

Motors & Mechanics – Motion Profile 그룹은 모터의 위치나 속도 제어시 사용하는 프로파일의 형식과 프로파일의 모양을 결정하는 최고 속도, 가속도, 감속도, 저크 값을 설정합니다.



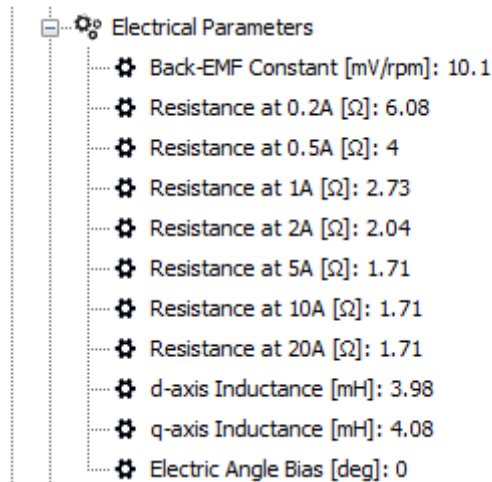
※ 자세한 설명은 오브젝트 항목에 있습니다.

이 그룹의 항목과 관련되는 오브젝트는 다음과 같습니다.

- Profile Type - profile_type
- Profile Velocity - profile_velocity
- Profile Acceleration - profile_acceleration
- Profile Deceleration - profile_deceleration
- Profile Jerk - profile_jerk

7.8.7 Motors & Mechanics – Electrical Parameters

Motors & Mechanics – Electrical Parameters 모터의 전기 파라미터인 역기전력 상수, 저항, 인덕턴스 값을 설정합니다.



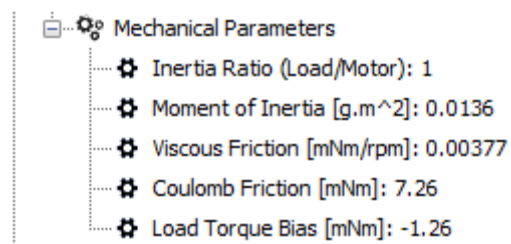
※ 자세한 설명은 오브젝트 항목에 있습니다.

이 그룹의 항목과 관련되는 오브젝트는 다음과 같습니다.

- Back-EMF Constant - bemf_constant
- Resistance at 0.2A ~ 20A - resistance[0 ~ 6]
- d-axis Inductance - inductance_d
- q0axis Inductance - inductance_q
- Electric Angle Bias - electric_angle_bias

7.8.8 Motors & Mechanics – Mechanical Parameters

Motors & Mechanics – Electrical Parameters 모터의 기계 파라미터인 관성모멘트, 정지 마찰계수, 운동마찰계수 값들을 설정합니다.



※ 자세한 설명은 모터드라이버 오토튜닝 및 오브젝트 항목에 있습니다.

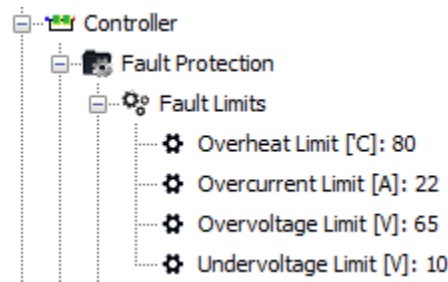
이 그룹의 항목과 관련되는 오브젝트는 다음과 같습니다.

- Inertia Ratio - inertia_ratio
- Moment of Inertia - moment_of_inertia

- Viscous Friction - viscous_friction
- Coulomb Friction - coulomb_friction
- Load Torque Bias - load_torque_bias

7.8.9 Controller – Fault Protection – Fault Limits

Controller – Fault Protection – Fault Limits 그룹에서는 모터 및 제어기를 보호하기 위한 과열, 과전류, 과전압 한계 값을 설정합니다.



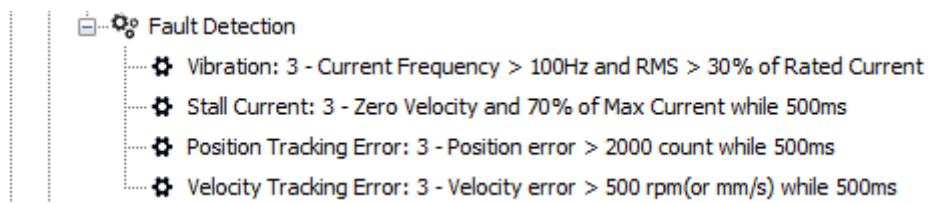
※ 자세한 설명은 오브젝트 항목에 있습니다.

이 그룹의 항목과 관련되는 오브젝트는 다음과 같습니다.

- Overheat Limit - overheat_limit
- Overcurrent Limit - overcurrent_limit
- Overvoltage Limit - overvoltage_limit
- Undervoltage Limit - undervoltage_limit

7.8.10 Controller – Fault Protection - Fault Detection

Controller – Fault Protection - Fault Detection 그룹에서는 모터 및 모터에 연결된 장비를 보호하기 위한 각종 폴트 감지 조건을 설정합니다.



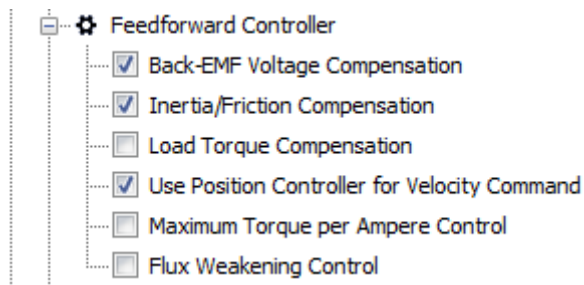
※ 자세한 설명은 오브젝트 항목에 있습니다.

이 그룹의 항목과 관련되는 오브젝트는 다음과 같습니다.

- Vibration - vibration_detect
- Stall Current - stall_current_detect
- Position Tracking Error - position_track_error
- Velocity Tracking Error - velocity_track_error

7.8.11 Controller – Feedforward Controller

Controller – Feedforward Controller 그룹은 피드 포워드 제어기와 관련된 설정들을 가지고 있습니다.



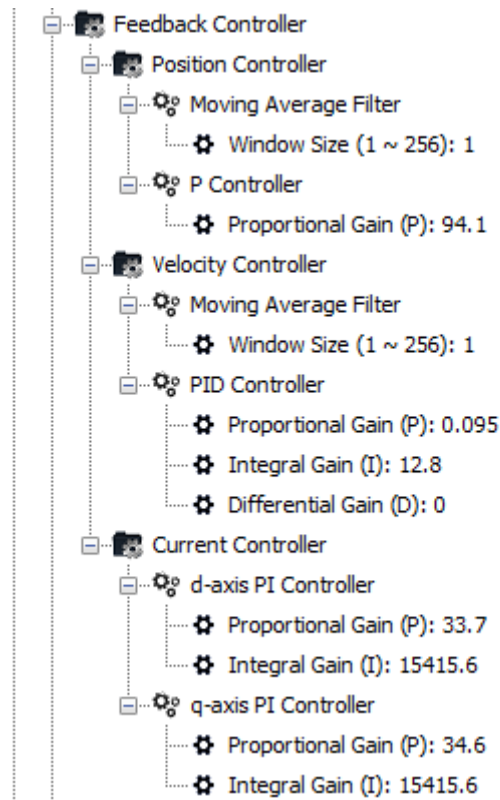
※ 자세한 설명은 모터드라이버 오토 튜닝 및 오브젝트 항목에 있습니다.

이 그룹의 항목과 관련되는 오브젝트는 다음과 같습니다.

- | | |
|--|-----------------------------|
| • Back-EMF Voltage Compensation | - feedforward_option & 0x01 |
| • Inertia/Friction Compensation | - feedforward_option & 0x02 |
| • Load Torque Compensation | - feedforward_option & 0x04 |
| • Use Position Controller for Velocity Command | - feedforward_option & 0x08 |
| • Maximum Torque per Ampere Control | - feedforward_option & 0x10 |
| • Flux Weakening Control | - feedforward_option & 0x20 |

7.8.12 Controller – Feedback Controller

Feedback Controller는 모터 드라이버 피드백 제어기의 PID 이득과 필터 계수를 설정할 수 있습니다.



PID Gain 값은 오토 튜닝을 통해 맞춰집니다.

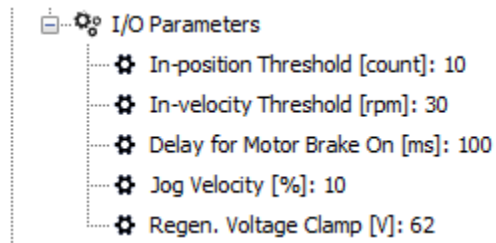
※ 자세한 설명은 [모터드라이버 오토튜닝 및 오브젝트 항목에 있습니다.](#)

이 그룹의 항목과 관련되는 오브젝트는 다음과 같습니다.

- Position Controller – Moving Average Filter – Window Size - pmaf_window_size
- Position Controller – P Controller – Proportional Gain - position_p_gain
- Velocity Controller – Moving Average Filter – Window Size - vmaf_window_size
- Velocity Controller – PID Controller – Proportional Gain - velocity_p_gain
- Velocity Controller – PID Controller – Integral Gain - velocity_i_gain
- Velocity Controller – PID Controller – Differential Gain - velocity_d_gain
- Current Controller – d-axis PI Controller – Proportional Gain - current_p_gain_d
- Current Controller – d-axis PI Controller – Integral Gain - current_i_gain_d
- Current Controller – q-axis PI Controller – Proportional Gain - current_p_gain_q
- Current Controller – q-axis PI Controller – Integral Gain - current_i_gain_q

7.8.13 Controller – I/O Parameters

Controller – I/O Parameters 그룹에서는 모터 드라이버에서 지원하는 입출력 포트의 동작과 관련된 파라미터를 설정 할 수 있습니다.



※ 자세한 설명은 오브젝트 항목에 있습니다.

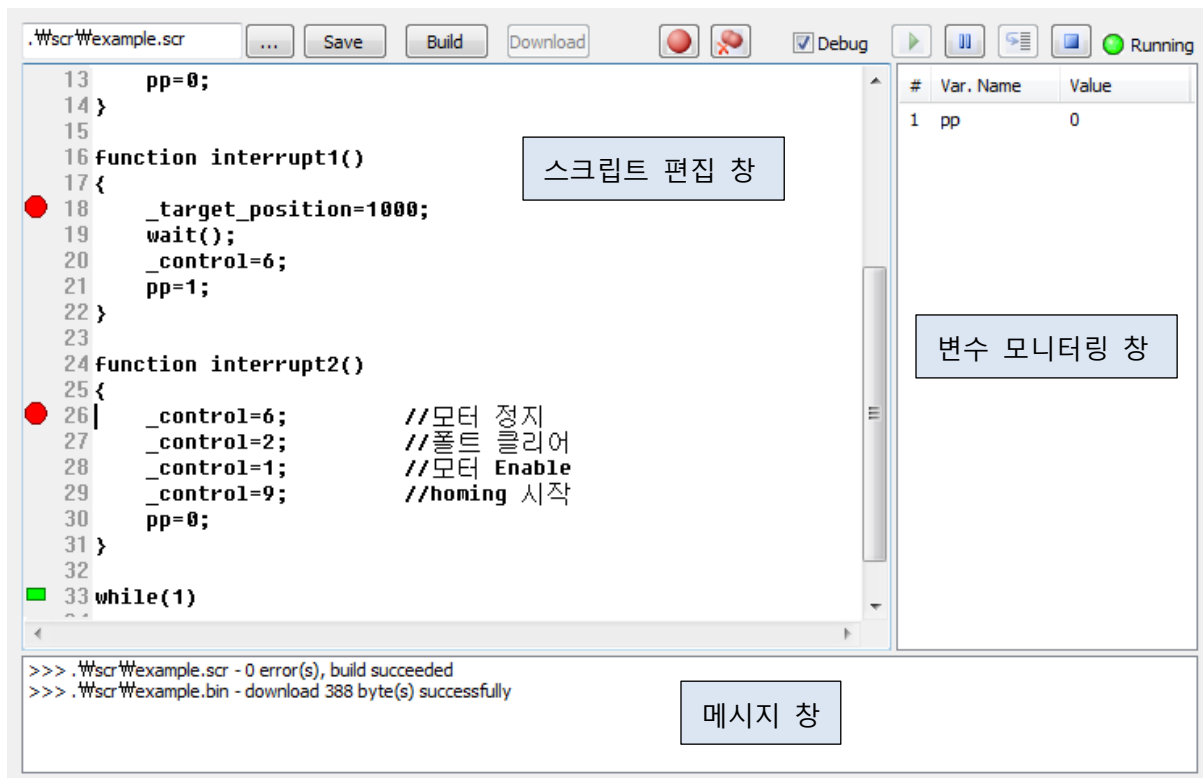
이 그룹의 항목과 관련되는 오브젝트는 다음과 같습니다.

- In-position Threshold - in_position_threshold
- In-velocity Threshold - in_velocity_threshold
- Delay for Motor Brake On - brake_on_delay
- Jog Velocity - jog_velocity
- Regen. Voltage Clamp - regen_voltage_clamp

7.9 Script Edit/Run 탭

모터 드라이버에서 스크립트를 작성하고 실행하는 것은 모터 드라이버 단독으로 사용자가 원하는 기능을 수행할 수 있도록 합니다.

스크립트 탭은 다음 그림과 같이 몇 개의 버튼과 좌측 스크립트 편집 창, 우측 변수 모니터링 창, 하단의 메시지 창으로 구성됩니다.



다음은 스크립트 탭에서 사용하는 버튼에 대한 간단한 설명입니다.

- [...] 버튼 - 스크립트 파일을 읽어 스크립트 편집 창에 표시
- [Save] 버튼 - 스크립트 편집 창의 코드를 파일로 저장
- [Build] 버튼 - 스크립트 편집 창의 코드를 빌드 하여 바이트코드 파일 생성
- [Download] 버튼 - 모터 드라이버에 바이트코드 파일을 다운로드
- [●] 버튼 - 스크립트 편집 창에서 커서가 위치한 라인에 브레이크 포인트 토글
- [🚫] 버튼 - 스크립트 편집 창에 지정된 모든 브레이크 포인트 해제
- Debug 체크박스 - 스크립트를 실행할 때 Debug 모드로 실행함, Debug 모드에서는 현재 실행되는 소스코드 행을 표시함
- [▶] 버튼 - 스크립트를 실행
- [⏏] 버튼 - 실행 중인 스크립트를 멈춤
- [⏮] 버튼 - 다음 라인 까지 실행하고 멈춤
- [⏹] 버튼 - 실행 중인 스크립트를 종료

7.9.1 Script 작성

Mini-C 스크립트 언어는 PLC나 PC와 같은 상위 제어기 없이 모터 드라이버 단독으로 특정 기능을 수행할 수 있도록 하며 C언어의 서브셋으로 만들어진 언어입니다. C언어의 제어문과 수식 연산구조를 일부 따오면서 구조체나 포인터 등 복잡한 부분을 제거하여 스크립트 언어를 처음 접하는 사용자가 쉽게 배우고 사용할 수 있습니다. 만일 C언어를 알고 있는 사용자라면 쉽게 사용 가능합니다.

스크립트 편집 창은 스크립트를 입력하는 데 사용됩니다. 편집기는 소스 코드를 작성하는 데 필요한 기본 텍스트 편집 기능을 가지고 있습니다.

***주의: Script 작성과 관련되는 자세한 내용은 “Mini-C Script.pdf” 파일을 참조하도록 합니다.**

작성한 코드를 저장하고 싶다면 [...] 버튼 좌측의 에디터 박스에 파일 이름을 지정하고 [Save] 버튼을 누르면 됩니다.

7.9.2 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 모터 드라이버로 다운로드 됩니다.

모터 드라이버에 다운로드 완료된 바이트코드는 플래시 메모리로 저장되어 영구적으로 실행될 수 있습니다.

빌드 중 하나 이상의 에러가 발생하면 빌드는 실패합니다. 그리고 asm 파일과 bin 파일은 생성되지 않습니다. 이때는 메시지 창의 에러 메시지를 참고하여 소스코드를 올바르게 수정해야 합니다. 메시지 창의 에러 메시지를 더블 클릭하면 스크립트 편집 창에서 커서는 에러가 발생한 행으로 이동합니다.

7.9.3 실행 및 디버깅

모터 드라이버에 다운로드 된 스크립트를 실행하기 위해서는 [▶], [⏏], [⏮], [⏭] 버튼 등이 사용됩니다. 스크립트를 다운로드 하고 [▶] 버튼을 누르면 스크립트가 시작됩니다. 스크립트가 계속 실행 중이라면 버튼 옆의 녹색 LED가 켜지면서 Running 상태가 표시됩니다. 이때 [⏏] 버튼을 누르면 스크립트의 실행이 종료됩니다.

다음 그림과 같이 Debug 체크박스를 체크하고 [▶] 버튼을 누르면 [⏏] 버튼이 활성화 되어 현재

수행 중인 스크립트를 잠시 멈출 수 있습니다. 스크립트가 멈춘 상태에서는 [] 버튼이 활성화되어 한 라인씩 단계적으로 수행할 수 있습니다.



실행 중 스크립트에서 사용된 전역 변수들의 값은 오른쪽 변수 모니터링 창에 실시간으로 업데이트됩니다. 이 기능은 스크립트 작성 및 디버깅을 편리하게 합니다.

만일 소스코드의 특정 위치에서 수행을 멈추어야 할 때는 커서를 위치 시키고 [] 버튼을 누릅니다. 그러면 스크립트 편집창 좌측의 라인넘버가 표시되는 부분에 브레이크포인트(●)가 표시되며, 스크립트의 실행이 브레이크포인트를 지날 때 멈추게 됩니다. 스크립트가 실행되는 위치는 ■로 표시됩니다. 모든 브레이크포인트를 해제할 경우에는 [] 버튼을 누릅니다.

```

24 function interrupt2()
25 {
26 |   _control=6;           //모터 정지
27   _control=2;           //폴트 클리어
28   _control=1;           //모터 Enable
29   _control=9;           //homing 시작
30   pp=0;
31 }
32
33 while(1)
  
```

7.10 모터 드라이버 펌웨어 업데이트

모터 드라이버의 버그를 수정하거나 기능이 향상에 의해 모터 드라이버의 펌웨어는 분기로 업데이트 될 수 있습니다. 사용자는 펌웨어 업데이트 기능으로 모터 드라이버에 항상 최신 펌웨어를 설치하고 사용할 수 있습니다.

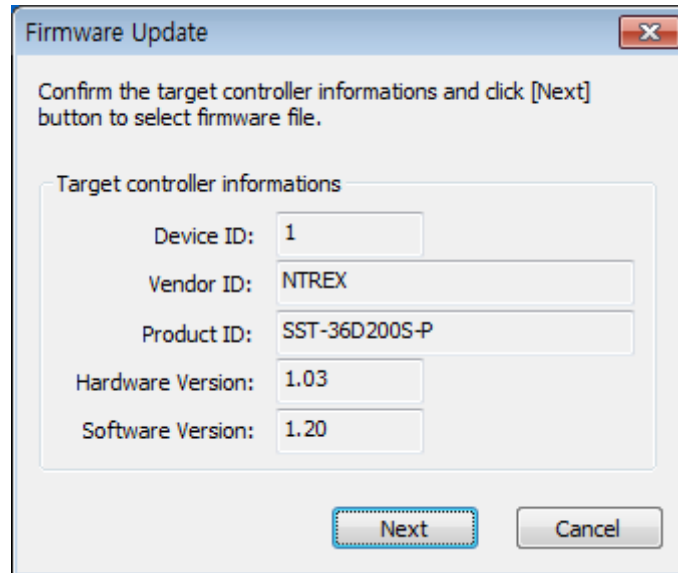
7.10.1 펌웨어 다운로드

펌웨어를 업데이트하기 위해서는 먼저 디바이스 마트 내 제품 관련 문서, 엔티렉스 연구소 홈페이지에서 최신 펌웨어를 다운로드 받아야 합니다.

7.10.2 펌웨어 업데이트

Firmware Update 대화상자는 UI 유틸리티 헤더의 [F/W Update] 버튼을 눌러 실행합니다. 만일 현재 모터 드라이버와 연결되어 있다면, 다음 그림과 같이 연결된 모터 드라이버의 Device ID,

Vender ID, Product ID, H/W Version, S/W Version 정보를 표시합니다. 모터 드라이버 정보를 확인하고 [Next] 버튼을 눌러 다음 스텝으로 이동하거나 [Cancel] 버튼을 눌러 펌웨어 업데이트를 취소할 수 있습니다.



Firmware Update

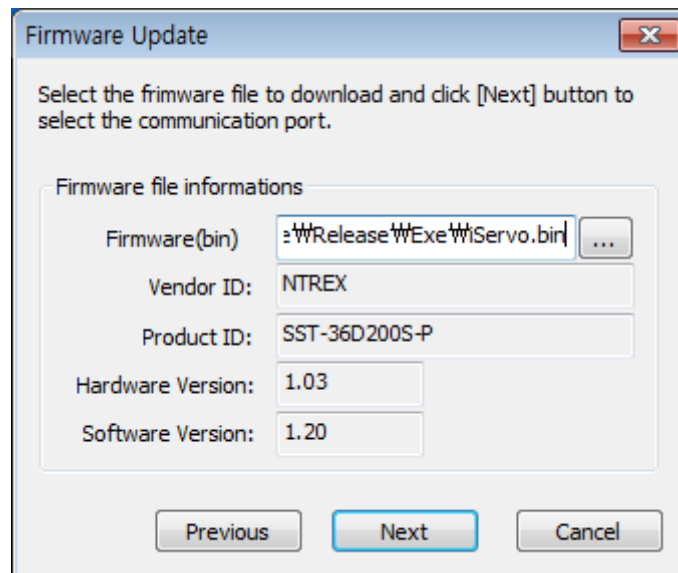
Confirm the target controller informations and click [Next] button to select firmware file.

Target controller informations

Device ID:	1
Vendor ID:	NTREX
Product ID:	SST-36D200S-P
Hardware Version:	1.03
Software Version:	1.20

Next Cancel

현재 모터 드라이버 버전을 확인한 후 [Next] 버튼을 누르면 다음 그림과 같이 업데이트할 펌웨어를 선택할 수 있는 대화상자가 표시됩니다.



Firmware Update

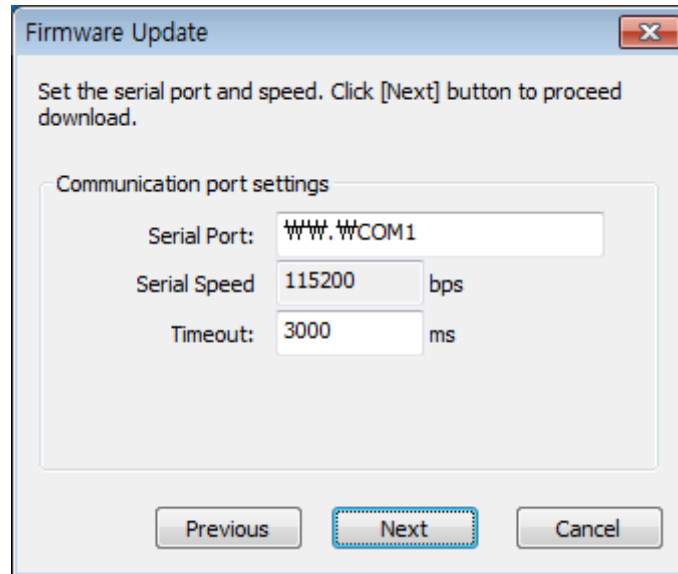
Select the firmware file to download and click [Next] button to select the communication port.

Firmware file informations

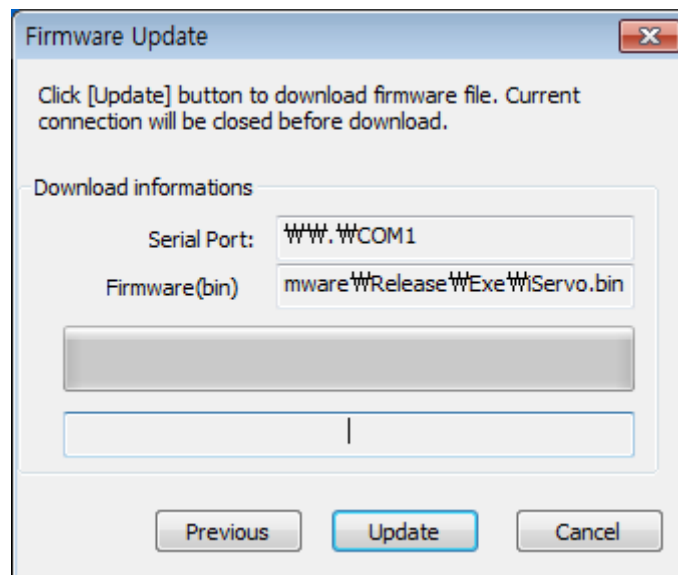
Firmware(bin)	Release\Exe\Servo.bin
Vendor ID:	NTREX
Product ID:	SST-36D200S-P
Hardware Version:	1.03
Software Version:	1.20

Previous Next Cancel

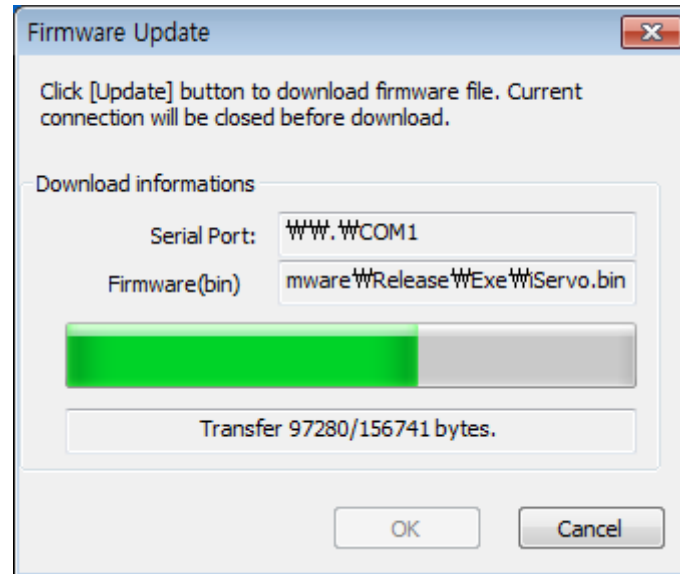
이 대화상자에서 Binary(bin) file 옆의 [...]버튼을 눌러서 엔티렉스 연구소 홈페이지, 혹은 디바이스 마트에서 다운받은 펌웨어 파일을 선택합니다. 그러면 펌웨어 파일의 Vender ID, Product ID, H/W Version, S/W Version을 확인할 수 있습니다. 이 정보와 대상 모터 드라이버의 모델 정보가 일치하는지 확인해야 합니다. 그 후 [Next] 버튼을 누르면 다음 그림과 같이 펌웨어를 다운로드 할 포트 정보가 표시됩니다.



여기서는 모터 드라이버에 펌웨어를 다운로드 할 시리얼 포트와 통신속도, 패킷 전송 Timeout 을 설정합니다. 그리고 [Next] 버튼을 누르면 다음과 같이 펌웨어를 다운로드 하기위한 대화상자가 표시됩니다.



여기서 다시한번 펌웨어를 다운로드 하기위한 시리얼 포트와 펌웨어 파일을 확인합니다. 마지막으로 [Update] 버튼을 누르면 기존의 모든 모터 드라이버 연결이 종료되고 다음 그림에서와 같이 선택한 펌웨어 다운로드가 시작됩니다.



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.

8 통신 프로토콜

8.1 Communication Protocol

이 장에서는 PC나 마이크로 컨트롤러에서 모터 드라이버의 오브젝트 값을 읽고 쓰기 위한 통신 프로토콜에 대해 설명합니다. 모터 드라이버에 동작을 명령하거나 모터 드라이버의 구성 파라미터를 설정하는 것은 해당 오브젝트에 특정 값을 쓰는 것을 의미하며, 모터 드라이버의 상태를 읽거나 모터 드라이버의 구성 파라미터를 읽는 것은 해당 오브젝트에서 특정 값을 읽는 것을 의미합니다.

통신 프로토콜은 크게 CAN 포트에서 사용되는 프로토콜과 시리얼(RS-232, RS-485) 포트에서 사용되는 프로토콜로 구분됩니다. 시리얼 포트 프로토콜은 다시 패킷의 구조에 따라 **바이너리(binary)** 형태와 **텍스트(text)** 형태로 구분됩니다.

8.1.1 용어의 정리

모터 드라이버의 객체를 통신 프로토콜로 읽고 쓰는데 사용되는 용어를 표 8-1에서 정의합니다.

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

표 8-1 통신 프로토콜에 사용되는 용어

상기 표에서 사용되는 용어 중 Index, Short Name, Long Name은 오브젝트를 표현하는 방식의 차이입니다. Index는 모터 드라이버 내부의 오브젝트를 참조하는 색인으로, CAN 및 시리얼 바이너리 패킷에 사용됩니다.

Sub-index는 오브젝트를 참조하는 부-색인으로, 통상 아날로그/디지털 I/O의 채널 번호를 의미합니다. 채널 1에 연결된 I/O에 대한 명령이나 설정이라면 Sub-index로 1을, 채널 2에 연결된 I/O의 경우는 2를 사용하면 됩니다. I/O가 아니라 모터 드라이버 자체에 대한 것이라면 통상 Sub-index는 0이 됩니다.

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

만약 사용자가 모터 드라이버에 인가된 주 전원의 전압을 체크하는 **power_source_voltage** 오브젝트의 내용을 읽고자 할 때, 시리얼 텍스트 기반으로 통신하는 경우, 해당 오브젝트의 Short Name인 **pv**나 Long Name인 **power_source_voltage**를 사용하면 됩니다. CAN이나 시리얼 바이너리 패킷 통신을 사용하는 경우는 해당 오브젝트의 Index인 **0x2008**을 사용하면 됩니다.

표 8-2에서는 오브젝트의 4가지 속성을 보여줍니다. 각각의 오브젝트 타입에 따라 패킷의 길이가 달라질 수 있습니다.

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

표 8-2 오브젝트의 타입

또한, 표 8-3에서는 오브젝트의 액세스 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.

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)

표 8-3 오브젝트의 액세스 속성

8.2 CAN 메시지

모터 드라이버의 구성 파라미터(Configuration Parameter)를 설정하고 명령(Command)을 내리거나 상태(Status)를 읽기 위해서 모터 드라이버에 존재하는 여러 오브젝트들을 CAN 통신으로 액세스할 수 있습니다. 이 절은 모터 드라이버에 존재하는 오브젝트들을 액세스하기 위한 CAN 통신의 메시지 구조에 대해 기술합니다.

※ 마스터 PC와 모터 드라이버가 CAN 버스에 연결되어 통신하기 위해서는, 네트워크에 연결된 모든 노드의 통신 속도가 일치해야 하고 각각의 장치 ID는 모두 달라야 합니다.

8.2.1 CAN 패킷의 기본 구조

마스터 PC와 모터 드라이버 간에 CAN 통신으로 주고받는 패킷의 주요 구성 요소는 Device ID와 메시지의 길이(Length) 그리고 전송되는 메시지(Message)가 됩니다(아래 그림 8-2-1 참조).

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

그림 8-2-1 CAN 통신 패킷 구성 요소

Device ID로는 최대 11bit를 설정할 수 있는데, 모터 드라이버의 **device_id**가 여기에 사용됩니다. 마스터에서 장치로 CAN 메시지를 보내거나 장치가 마스터로 회신 할 때, Device ID가 동일하게 지정되어야 합니다. Length는 메시지(Message)의 길이를 나타내는데, 메시지의 최대 크기가 8byte 이므로 0에서 8사이의 값이 됩니다. 메시지에는 전송되는 데이터를 담습니다.

그림 8-2-2는 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 Value가 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			

그림 8-2-2 CAN 통신 패킷 구성 요소

여기서 Command는 표 8-4과 같은 액세스 형식(Access Code)과 오브젝트 형식(Object Type)의 조합으로 1byte를 구성하게 됩니다.

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

표 8-4 액세스 형식(Access Code)과 오브젝트 형식(Object Type)

Index나 Value와 같이 한 바이트 이상의 데이터가 메시지에 저장될 때는 데이터의 하위 바이트부터 왼쪽에 저장되는 **리틀 인디언(Little-Endian)방식**을 따릅니다.

8.2.2 오브젝트 읽기 요청

마스터 PC가 모터 드라이버의 오브젝트를 읽기 위해, 마스터 PC가 모터 드라이버에 보내는 쿼리 패킷을 구성합니다.

오브젝트의 값을 읽을 때는 표 8-4의 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				

8.2.3 오브젝트 쓰기 요청

마스터 PC가 모터 드라이버의 오브젝트에 값을 쓰기 위해, 마스터 PC가 모터 드라이버에 보내는 쿼리를 구성합니다.

오브젝트에 값을 쓸 때는, 4th byte 지점 까지는 오브젝트 읽기 요청과 동일하게 구성합니다. 단, Command는 표 8-4의 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)			

8.2.4 오브젝트 읽기/쓰기 요청에 대한 성공 응답

모터 드라이버가 마스터 PC의 오브젝트 읽기/쓰기 요청에 응답하기 위해, 모터 드라이버가 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 8-4의 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)			

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

8.2.5 오브젝트 읽기/쓰기 요청에 대한 실패 응답

모터 드라이버가 마스터 PC의 오브젝트 읽기/쓰기 요청에 실패를 알리기 위해, 모터 드라이버가 마스터 PC에 보내는 실패 응답 패킷을 구성합니다.

오브젝트의 내용을 읽거나 쓰기 위한 패킷을 모터 드라이버에 요청했을 때, 오류가 발생하는 경우는 다음 그림 8-2-3과 같은 형식의 오류 패킷이 돌아옵니다.

1st byte	2nd byte	3rd byte	4th byte	5th byte	6th byte	7th byte	8th byte
Command	Error Code						

그림 8-2-3 CAN 오류 패킷

오류 메시지는 표 8-5에 나타나 있습니다. CAN 패킷에서 오류 메시지 내용은 전달되지 않으며 Error Code만 반환됩니다. 그리고 Command는 0x80이 반환됩니다.

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

표 8-5 오브젝트의 읽기/쓰기 과정에서 발생하는 오류

※ 본 절의 오류 메시지는 패킷의 구성에 관련된 오류로 모터 드라이버의 오류 상황이나 오작동에 대한 부분이 아닙니다.

8.3 시리얼 바이너리 패킷

모터 드라이버의 오브젝트들은 시리얼(RS-232, RS-485) 통신으로도 읽고 쓰기가 가능합니다. 시리얼 통신은 텍스트 기반의 패킷과 함께 바이너리 기반의 패킷 통신을 함께 지원합니다.

이 절은 모터 드라이버에 존재하는 오브젝트들을 액세스하기 위한 시리얼 통신의 바이너리 패킷 구조에 관해 설명합니다.

8.3.1 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)

그림 8-3-1 바이너리 패킷 구성 요소

상기 그림 8-3-1에서 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, 남은 공간에 0을 채우면 됩니다.

시리얼 바이너리 패킷에서도 CAN 패킷에서와 같이 리틀 인디언(Little-Endian)방식으로 패킷을 구성합니다.

8.3.2 오브젝트 읽기 요청

오브젝트의 값을 읽을 때는, 패킷의 4th ~ 11th byte 지점에 위치하는 Message를 아래와 같이 구성하면 됩니다. Command는 표 8-4의 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

8.3.3 오브젝트 쓰기 요청

오브젝트에 값을 쓸 때는, 7th byte 지점까지는 오브젝트 읽기 요청과 동일하게 구성합니다. 단 Command는 표 8-4의 Access Code에 의해 0x10과 오브젝트 형에 맞는 조합으로 작성되어야 하며, 그 형식에 맞춰 오브젝트의 값(Value)을 작성합니다.

4th byte	5th byte	6th byte	7th byte	8th byte	9th byte	10th byte	11th byte
Command	Index		Sub-index	Value(INT8)	0	0	0
				Value(INT16)		0	0
				Value(INT32)			
				Value(FLOAT)			

8.3.4 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 8-4의 Access Code에 제시된 것처럼 0x20이나 0x40중 하나와 Object Type에 맞는 조합으로 작성됩니다.

4th byte	5th byte	6th byte	7th byte	8th byte	9th byte	10th byte	11th byte
Command	Index		Sub-index	Value(INT8)	0	0	0
				Value(INT16)		0	0
				Value(INT32)			
				Value(FLOAT)			

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

8.3.5 오브젝트 읽기/쓰기 요청에 대한 실패 응답

오브젝트의 내용을 읽거나 쓰기 위한 패킷을 모터 드라이버에 요청했을 때, 오류가 발생하는 경우는 다음 그림 8-3-2와 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

그림 8-3-2 오류 패킷

오류 메시지는 표 8-5에 나타나 있습니다. 시리얼 바이너리 패킷에서 오류 메시지 내용은 전달되지 않으며 Error Code만 반환됩니다. 그리고 Command는 0x80이 반환됩니다.

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

8.4 시리얼 텍스트 패킷

시리얼(RS-232, RS-485) 통신에서는 텍스트(text) 기반으로 모터 드라이버의 오브젝트를 읽고 쓸 수 있도록 합니다. 이는 사용자가 통신을 위한 전용 프로그램을 사용하지 않더라도 Hyperterminal과 같은 유틸리티를 사용하여 모터 드라이버의 오브젝트들을 쉽게 액세스 할 수 있도록 합니다.

시리얼 텍스트 기반 패킷에서는 Index를 직접적으로 사용하지 않고 표 8-1의 Short Name이나 Long Name과 Sub-index를 사용합니다.

8.4.1 오브젝트 읽기 요청

오브젝트 값을 읽을 때는 텍스트 패킷을 다음과 같이 구성합니다.

Short/Long Name	CR/LF
-----------------	-------

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

Short/Long Name	[Sub-index]	CR/LF
-----------------	---	-----------	---	-------

Short/Long Name은 오브젝트의 Index에 해당하는 이름이며, Sub-index는 해당 Index에 따라 필요하면 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)

예제) 모터 드라이버의 **user_variable**은 sub-index가 0~255 범위의 배열형 오브젝트입니다. 이 오브젝트의 sub-index 가 1인 곳의 값을 읽기 위해서는 다음과 같이 명령을 입력하고 엔터 키를 입력합니다.

```
user_variable[1]↵
user_variable1↵
```

8.4.2 오브젝트 쓰기 요청

오브젝트 값을 쓸 때는 텍스트 패킷을 다음과 같이 구성합니다.

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

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

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

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

예제) 모터 버에 연결된 모터를 1000rpm 속도로 회전하기 위해서는 다음과 같이 명령을 입력하고 엔터 키를 입력합니다.

```
tv=1000↵
target_velocity=1000↵
```


8.4.3 오브젝트 읽기/쓰기 요청에 대한 성공 응답

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

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

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

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

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

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

8.4.4 오브젝트 읽기/쓰기 요청에 대한 실패 응답

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

!	Error Code	,	Error Message	CR/LF
---	------------	---	---------------	-------

오류 메시지는 표 8-5에 나타나 있습니다.

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

8.4.5 Device ID의 부여

만일 둘 이상의 모터 드라이버가 RS-485 버스에 연결된 경우, 시리얼 텍스트 패킷에는 각각의 모터 드라이버를 구분하는 장치의 ID를 부여해야 합니다.

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

Device ID	;	Serial Text Packet
-----------	---	--------------------

예제) 만약 **device_id** 1번을 가지는 모터 드라이버의 모터를 10rpm 속도 구동시키고자 할 때 아래와 같이 입력합니다.

```
1;tv=10↓
1;target_velocity=10↓
```

모터 드라이버가 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 모터 드라이버를 연결할 때는, 각각의 모터 드라이버 간에 Device ID가 서로 충돌하지 않도록 설정하여야 합니다.**

8.4.6 여러 오브젝트의 읽기/쓰기

시리얼 텍스트 패킷 통신에서 하나의 요청으로 여러 오브젝트를 동시에 읽고 쓸 수 있습니다. 각각의 읽기와 쓰기 요청을 구분하는 문자는 ';' 입니다.

다음은 3개의 읽기/쓰기 요청을 구성하는 패킷의 모양입니다. 여기서 텍스트 패킷의 길이가 256 byte를 넘지 않도록 구성하여야 합니다.

읽기/쓰기 요청 1	;	읽기/쓰기 요청 2	;	읽기/쓰기 요청 3
------------	---	------------	---	------------

모터 드라이버가 둘 이상의 읽기/쓰기 요청을 받아들이고 정상적으로 처리하였다면 다음과 같이 각각의 요청에 대한 응답을 구성하여 회신합니다. 각각의 응답을 구분하는 문자는 ';' 입니다.

읽기/쓰기 응답 1	;	읽기/쓰기 응답 2	;	읽기/쓰기 응답 3
------------	---	------------	---	------------

만일 모터 드라이버가 구성한 응답 패킷의 길이가 256 byte보다 클 때는 256 byte를 초과하는 부분이 잘려나간 상태에서 오류를 표시하지 않고 응답하기 때문에 되도록 요청과 응답 패킷을 너무 길게 만들지 않는 것이 좋습니다.

예제) 모터 드라이버의 **user_variable** 오브젝트의 sub-index 1,2,3 값을 동시에 쓰고자 할 때 아래와 같이 입력합니다.

```
u1=10;u2=20;u3=30↓
```

예제) 1번 채널의 디지털 입력과 2번 채널의 디지털 입력, 3번 채널의 디지털 입력을 동시에 읽고자 할 때는 다음과 같이 입력합니다.

```
div1;div2;div3↓
```

9 오브젝트

9.1 Object 설명

오브젝트는 모터 제어를 외부에서 또는 내부의 Virtual Machine에서 액세스 하는데 사용하도록 외부로 공개된 일종의 변수 목록입니다.

시리얼 통신(RS-232, RS-485)을 통해 오브젝트를 다음과 같이 TEXT 방식으로 액세스 할 수 있습니다.

```
Ex) brake_on_delay = 100↓ // Brake on Delay 값을 100ms로 설정한다.
    brake_on_delay↓       // Brake on Delay 값을 읽어온다.
```

Script 언어에서는 오브젝트 이름 앞에 언더스코어('_')를 붙여 사용한다.

```
Ex) _brake_on_delay = 100 // Brake on Delay 값을 100ms로 설정한다.
    d = _brake_on_delay   // Brake on Delay 값을 변수 d로 읽어온다.
```

9.2 오브젝트의 기본 단위

오브젝트 종류	시리얼 통신, Script	Related Objects
RS232 Baudrate	bps	serial_baudrate
Time	ms	serial_watchdog, brake_on_delay
Distance	mm	linear_sensor_pitch
Position	counts	target_position, position_demand, position_actual, position_error, min_position, max_position, in_position_threshold
Velocity	Rotary: rpm, Linear: mm/s	target_velocity, velocity_demand, velocity_actual, velocity_error, rated_velocity, max_velocity, profile_velocity, jog_velocity,

		in_velocity_threshold
Current	A	power_source_current, target_current, current_demand_d, current_demand_q, current_actual_d, current_actual_q, current_error_d, current_error_q, rated_current, max_current, overcurrent_limit
Voltage	V	power_source_voltage, target_voltage, voltage_demand_d, voltage_demand_q, rated_voltage, overvoltage_limit, undervoltage_limit
Temperature	°C	temperature, overheat_limit
Torque Limit	%	torque_limit
Torque, Force	Rotary: mNm, Linear: N	load_torque
Acceleration	Rotary: rpm/s, Linear: mm/s ²	acceleration_demand, acceleration_actual, profile_acceleration, profile_deceleration
Jerk	Rotary: rpm/s ² Linear: mm/s ³	profile_jerk
Encoder Resolution	cpr	encoder_resolution
# of Pole Pairs	6*cpr	no_pole_pairs
Back-EMF constant	Rotary: mV/rpm, Linear: V/(m/s)	bemf_constant
Resistance	Ω	resistance
Inductance	mH	inductance_d, inductance_q
Torque constant, Force constant	Rotary: mNm/A, Linear: N/A	torque_constant

Moment of Inertia, Weight	Rotary: gm^2 Linear: kg	moment_of_inertia
Viscous Friction	Rotary: mNm/rpm, Linear: Nm/(m/s)	viscous_friction

※ bps – bits per second

※ cpr – counts per revolute

※ rpm – Revolute per minute

9.3 오브젝트 테이블

Index	Sub-index	Type	Access	Short name	Long name
0x2001	0	I16	RO	vid	vendor_id
0x2002	0	I16	RO	pid	product_id
0x2003	0	I16	RO	swv	software_version
0x2004	0	I16	RO	hwv	hardware_version
0x2005	0	I32	RO	ss	system_status
0x2006	0	I32	WO	sc	system_control
0x2007	0	I32	RW	sa	system_argument
0x2008	0	F32	RO	pv	power_source_voltage
0x2009	0	F32	RO	pc	power_source_current
0x2010	0~7	S32	RW	na	device_name
0x2011	0	I16	RW	id	device_id
0x2012	0	I16	RW	sw	serial_watchdog
0x2014	0~2	I32	RW	sb	serial_baudrate
0x201A	0	I08	RW	un	units
0x201B	0	I08	RW	idi	startup_drive_input
0x201C	0	I08	RW	scm	startup_control_mode
0x201D	0	I16	RW	sed	startup_enable_delay
0x201E	0	I08	RW	ssa	script_stop_action
0x201F	0	I08	RW	doa	disable_oper_action
0x2041	0	I08	RW	pi	pulse_input_mode
0x2042	0	I08	RW	pce	pulse_count_edge
0x2043	0	I08	RW	dp	dir_signal_polarity
0x2048	0	I32	RW	gn	gear_numerator
0x2049	0	I32	RW	gd	gear_denominator
0x2060	0~255	I32	RW	u	user_variable

0x2061	0~255	I32	RW	t	temp_variable
0x2080	0~15	I16	RW	ec	error_code
0x2101	0	I32	WO	c	control
0x2102	0	I32	RO	s	status
0x2103	0	I32	RO	f	fault
0x2105	0	F32	RO	te	temperature
0x2106	0	F32	RW	l	load_torque
0x2111	0	I32	RW	tp	target_position
0x2112	0	I32	RW	tv	target_velocity
0x2113	0	F32	RW	tcd	target_current_d
0x2114	0	F32	RW	tcq	target_current_q
0x2115	0	F32	RW	tvd	target_voltage_d
0x2116	0	F32	RW	tvq	target_voltage_q
0x2119	0	F32	RW	tl	torque_limit
0x2121	0	I32	RO	pd	position_demand
0x2122	0	I32	RO	vd	velocity_demand
0x2123	0	F32	RO	cdd	current_demand_d
0x2124	0	F32	RO	cdq	current_demand_q
0x2125	0	F32	RO	vdd	voltage_demand_d
0x2126	0	F32	RO	vdq	voltage_demand_q
0x2127	0	I32	RO	ad	acceleration_demand
0x2131	0	I32	RW	p	position_actual
0x2132	0	I32	RO	v	velocity_actual
0x2133	0	F32	RO	cd	current_actual_d
0x2134	0	F32	RO	cq	current_actual_q
0x2137	0	F32	RO	aa	acceleration_actual
0x2141	0	I32	RO	pe	position_error
0x2142	0	I32	RO	ve	velocity_error
0x2143	0	F32	RO	ced	current_error_d
0x2144	0	F32	RO	ceq	current_error_q
0x2151	0	I08	RW	sl	soft_limit_check
0x2152	0	I32	RW	n	min_position
0x2153	0	I32	RW	x	max_position
0x2154	0	I08	RW	la	limit_action
0x2158	0	I08	RW	hm	homing_method
0x2159	0	I32	RW	hv	homing_velocity
0x215A	0	I32	RW	ho	home_offset
0x215B	0	I32	RW	hp	home_position

0x2161	0	F32	RW	rvo	rated_voltage
0x2162	0	F32	RW	rc	rated_current
0x2163	0	I32	RW	rv	rated_velocity
0x2164	0	F32	RW	mc	max_current
0x2165	0	I32	RW	mv	max_velocity
0x2167	0	I08	RW	pty	profile_type
0x2168	0	I32	RW	pve	profile_velocity
0x2169	0	I32	RW	pa	profile_acceleration
0x216A	0	I32	RW	pde	profile_deceleration
0x216B	0	I32	RW	pj	profile_jerk
0x2171	0	F32	RW	oh	overheat_limit
0x2172	0	F32	RW	oc	overcurrent_limit
0x2173	0	F32	RW	ov	overvoltage_limit
0x2174	0	F32	RW	uv	undervoltage_limit
0x2176	0	I08	RW	vb	vibration_detect
0x2177	0	I08	RW	sd	stall_current_detect
0x2178	0	I08	RW	pt	position_track_error
0x2179	0	I08	RW	vt	velocity_track_error
0x2181	0	I08	RW	m	motor_type
0x2182	0	I08	RW	d	motor_direction
0x2183	0	I08	RW	ff	feedforward_option
0x2188	0	I08	RW	ps	position_sensor
0x2189	0	I08	RW	ed	encoder_direction
0x218A	0	I32	RW	er	encoder_resolution
0x218B	0	I16	RW	np	no_pole_pairs
0x218C	0	I08	RW	hs	hallsensor_phase
0x218D	0	I32	RW	lsp	linear_sensor_pitch
0x2191	0	F32	RW	b	bemf_constant
0x2192	0~6	F32	RW	r	resistance
0x2193	0	F32	RW	ld	inductance_d
0x2194	0	F32	RW	lq	inductance_q
0x2195	0	I32	RW	eb	electric_angle_bias
0x2198	0	F32	RW	to	torque_constant
0x2199	0	F32	RW	i	moment_of_inertia
0x219A	0	F32	RW	vf	viscous_friction
0x219B	0	F32	RW	cf	coulomb_friction
0x219C	0	F32	RW	tb	load_torque_bias
0x21A1	0	F32	RW	pp	position_p_gain

0x21A4	0	F32	RW	vp	velocity_p_gain
0x21A5	0	F32	RW	vi	velocity_i_gain
0x21A6	0	F32	RW	vdg	velocity_d_gain
0x21A7	0	F32	RW	cpd	current_p_gain_d
0x21A8	0	F32	RW	cid	current_i_gain_d
0x21A9	0	F32	RW	cpq	current_p_gain_q
0x21AA	0	F32	RW	ciq	current_i_gain_q
0x21B1	0	I16	RW	pw	pmaf_window_size
0x21B2	0	I16	RW	vw	vmaf_window_size
0x21D1	0	I32	RW	ip	in_position_threshold
0x21D2	0	I32	RW	iv	in_velocity_threshold
0x21D3	0	I16	RW	bo	brake_on_delay
0x21D4	0	F32	RW	jv	jog_velocity
0x21D5	0	F32	RW	rvc	regen_voltage_cutoff
0x2201	0	I08	RO	nid	no_digital_input
0x2202	0	I08	RO	ndo	no_digital_output
0x2203	0	I08	RO	nai	no_analog_input
0x2204	0	I08	RO	npi	no_pulse_input
0x2210	0	I32	RO	di	digital_inputs
0x2220	0	I32	RW	do	digital_outputs
0x2221	0	I32	RW	dom	digital_outputs_mask
0x2230	1~N	I08	RW	dio	di_option
0x2231	1~N	I08	RO	div	di_value
0x2232	1~N	I16	RW	dif	di_function
0x2240	1~M	I08	RW	doo	do_option
0x2241	1~M	I08	RW	dov	do_value
0x2242	1~M	I16	RW	dof	do_function
0x2250	1~O	I08	RW	aio	ai_option
0x2251	1~O	F32	RO	aiv	ai_value
0x2252	1~O	I16	RW	aif	ai_function
0x2253	1~O	I32	RO	air	ai_raw_value
0x2255	1~O	I16	RW	acf	ai_cutoff_freq
0x2256	1~O	I32	RW	ain	ai_cal_min
0x2257	1~O	I32	RW	aic	ai_cal_center
0x2258	1~O	I32	RW	aix	ai_cal_max
0x2259	1~O	I32	RW	aidb	ai_cal_deadband
0x2260	1~P	I08	RW	pio	pi_option
0x2261	1~P	F32	RO	piv	pi_value

0x2262	1~P	I16	RW	pif	pi_function
0x2263	1~P	I32	RO	pir	pi_raw_value
0x2264	1~P	I08	RW	pit	pi_capture_type
0x2265	1~P	I16	RW	pcf	pi_cutoff_freq
0x2266	1~P	I32	RW	pin	pi_cal_min
0x2267	1~P	I32	RW	pic	pi_cal_center
0x2268	1~P	I32	RW	pix	pi_cal_max
0x2269	1~P	I32	RW	pidb	pi_cal_deadband

※ N – Number of Digital Inputs

※ M – Number of Digital Outputs

※ O – Number of Analog Inputs

※ P – Number of Pulse Inputs

9.4 Product Information

9.4.1 vendor_id

0으로 고정된 값입니다.

ex)

```
vendor_id^ // vendor_id를 입력하고 [Enter] 키를 입력
vendor_id=0 // 제어기에서 vendor_id=0으로 응답
```

9.4.2 product_id

모터 제어기 제품 ID입니다.

- 10091 - MW-SPM36D200S
- 10094 - MW-SSEH224D5S

ex)

```
product_id^
product_id=10094
```

9.4.3 software_version

모터 제어기 펌웨어 버전에 100을 곱한 값입니다.

펌웨어 버전은 다음과 같이 표시됩니다: XX.YY

- XX: major version
- YY: minor version

ex)

```
software_versiond
software_version=111          // Version 1.11 임을 나타냄
```

9.4.4 hardware_version

모터 제어기 하드웨어 버전에 100을 곱한 값입니다.

하드웨어 버전은 다음과 같이 표시됩니다: XX.YY

- XX: major version
- YY: minor version

9.4.5 units

모터 제어기에서 사용할 단위의 종류를 선택합니다.

값	대상 모터 군	길이, 속도, 가속도 (velocity, acceleration, jerk)	각종 상수 (bemf_constant, viscous_friction)	힘, 토크 (torque, torque_constant, coulomb_friction)	관성, 무게 (moment_of_ inertia)
0	Rotary Motor	count/s, count/s ² , count/ ³	mV/(count/s), mNm/(count/s)	mNm, mNm/A	g.m ²
1	Linear Motor	count/s, count/s ² , count/ ³	mV/(count/s), mN/(count/s)	N, N/A	kg
2	Rotary Motor	rpm, rpm/s, rpm/s ²	mV/rpm, mNm/rpm	mNm, mNm/A	g.m ²
3	Linear Motor	mm/s, mm/s ² ,	V/(m/s), N/(m/s)	N, N/A	kg

		mms^3			
--	--	-------	--	--	--

```
ex)
units^4
units=2
```

9.5 Startup/Stop

9.5.1 startup_drive_input

제어기가 시작될 때 제어기 구동 명령이 입력되는 종류를 선택합니다.

- 0 – Serial Drive Mode (RS-232, RS-485, CAN, Ethernet 등)
- 1 – Pulse/Direction Input Mode
- 2 – Analog Input Mode

9.5.2 startup_control_mode

제어기가 시작될 때 초기 제어 모드를 설정합니다.

- 0 – Disable
- 1 – Enable and Voltage Output Mode
- 2 – Enable and Current Control Mode
- 3 – Enable and Velocity Control Mode
- 4 – Enable and Position Control Mode
- 5 - Enable and Position Control Mode, Home Search
- 6 - Enable and Position Control Mode, Script Run

이 설정 값은 모터 제어를 Enable 할 때 모터의 구동 모드를 결정할 때도 사용됩니다.

- 0번이나 1번으로 설정된 경우 Voltage Output 모드로 Enable 되고,
- 2번으로 설정된 경우 Current Control 모드가 됩니다.
- 3번으로 설정된 경우는 Velocity Control 모드가 되고, 4로 설정된 경우 Position Control 모드가 됩니다.
- 5번으로 설정된 경우, 제어를 Enable한 후 Home Search를 시작합니다.
- 6번으로 설정된 경우, 제어를 Enable한 후 Script를 시작합니다.

9.5.3 startup_enable_delay

startup_control_mode가 0이 아닌 값으로 설정된 경우, 제어기가 시작될 때 Enable 할 때까지의 지연 시간을 [ms] 단위로 설정합니다.

***주의: 이 설정은 STEP 모터를 구동할 때 사용됩니다. STEP 모터는 Enable 될 때 max_current에 설정된 전류를 코일에 흘려 회전자를 정렬합니다. 이 때 여러 모터 드라이버가 하나의 전원 소스에 연결된 경우, 전원 소스에서 충분한 전류를 공급하지 못할 수도 있습니다. 이러한 경우, 모터 드라이버의 startup_enable_delay를 각기 다른 값으로 설정하여 한번에 다량의 전류가 흐르는 것을 방지할 수 있습니다.**

9.5.4 script_stop_action

실행중인 스크립트를 사용자가 중단할 때 혹은 스크립트의 실행이 종료되었을 때의 행동을 선택합니다.

- 0 - (none)
- 1 - Stop
- 2 - Quick Stop
- 3 - Disable

9.5.5 disable_oper_action

사용자가 모터를 Disable 할 때 또는 Fault가 발생하여 모터가 Disable 될 때의 행동을 선택합니다.

- 0 - Free Run
- 1 - Stop & Free Run
- 2 - Stop & Dynamic Brake

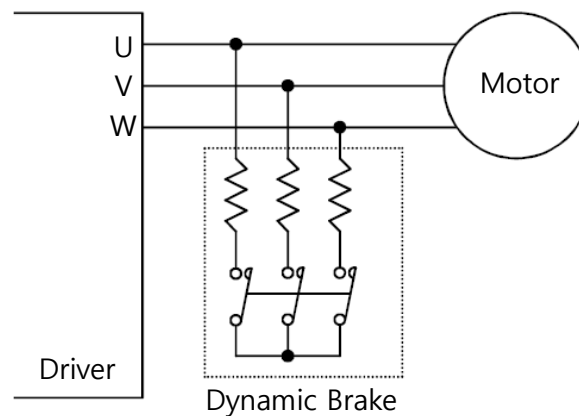
만일 이 값이 0으로 설정된 경우, 모터가 회전 중일 때 Disable 명령을 내린다면 모터는 관성에 의해 모터는 멈추지 않고 회전 속도를 유지합니다. 하지만 마찰력이 작용하여 천천히 속도가 줄어 멈추게 됩니다.

만일 이 값이 1로 설정된 경우, 모터가 회전 중일 때 Disable 명령을 내린다면 제어기는 모터를 멈추도록 제어하여 모터의 속도가 30rpm 이하가 되었을 때 Free Run 상태가 됩니다.

만일 이 값이 2로 설정되었고 제어기가 Dynamic Brake 기능을 지원하는 경우, 모터가 회전 중일 때 Disable 명령을 내린다면 제어기는 모터를 멈추도록 제어하여 모터의 속도가 30rpm 이하가 되었을 때 Dynamic Brake를 작동합니다.

Fault가 발생하여 모터가 Disable 될 때는 Stop 과정을 거치지 않고 Free Run 상태나 Dynamic Brake가 즉각 작동하게 됩니다.

※ **Dynamic Brake:** 모터가 회전 중 모터가 Disable 되었다면 모터는 관성에 의해 속도를 유지하려는 Free run 상태가 되며 이 때는 마찰력에 의해 서서히 속도를 줄여 정지하게 됩니다. 그러나 모터의 정지 시간을 단축하기 위해 다음 그림과 같은 Dynamic Brake를 사용하게 됩니다. Free run 상태에서는 발전기가 되어 있기 때문에, Dynamic Brake를 작동하여 모터의 U, V, W 단자를 단락 함으로 모터의 운동에너지를 저항을 통해 열로 소비시켜 정지 시간을 단축시킵니다.



9.6 System Parameters

9.6.1 system_status

모터 제어기의 전반적인 상태를 표시합니다. 모터 구동과 직접 관련된 상태는 status 오브젝트에서 표시합니다.

definition	code	description
SS_HS_PHASE_DETECT	0x0001	SC_HS_PHASE_DETECT 명령이 실행 중이거나 실행 되었음을 표시합니다.
SS_POS_SENSOR_PARAM	0x0002	SC_POS_SENSOR_PARAM 명령이 실행 중이거나 실행 되었음을 표시합니다.
SS_ELECTRIC_PARAM_EST	0x0003	SC_ELECTRIC_PARAM_EST 명령이 실행 중이거나 실행 되었음을 표시합니다.
SS_MECHANIC_PARAM_EST	0x0004	SC_MECHANIC_PARAM_EST 명령이 실행 중이거나 실행 되었음을 표시합니다.
SS_COMMAND_SUCCESS	0x0010	상기 SC_HS_PHASE_DETECT, SC_POS_SENSOR_PARAM, SC_ELECTRIC_PARAM_EST, SC_MECHANIC_PARAM_EST 명령이 성공적으로 실행

		행 종료 되었음을 표시합니다.
SS_COMMAND_FAILED	0x0020	상기 SC_HS_PHASE_DETECT, SC_POS_SENSOR_PARAM, SC_ELECTRIC_PARAM_EST, SC_MECHANIC_PARAM_EST 명령이 수행 도중 또는 수행 완료 후 실패하였음을 표시합니다.
SS_COMMAND_ABORTED	0x0040	상기 SC_HS_PHASE_DETECT, SC_POS_SENSOR_PARAM, SC_ELECTRIC_PARAM_EST, SC_MECHANIC_PARAM_EST 명령이 사용자 요청에 의해 중단되었음을 표시합니다.
SS_SCRIPT_DEBUG	0x08000000	Script가 디버깅 모드로 실행 중인 상태를 표시합니 다..
SS_SCRIPT_RUN	0x10000000	Script가 실행 중인 상태를 표시합니다..
SS_SCRIPT_PAUSE	0x20000000	Script가 실행을 멈춘 상태를 표시합니다..
SS_SCRIPT_STEP	0x40000000	Script가 한 스텝(다음 줄까지) 실행 중인 상태를 표시합니다.
SS_SCRIPT_STOP	0x80000000	Script가 종료 된 상태를 표시합니다.

9.6.2 system_control

모터 제어기에 운용 명령을 내립니다. 모터 구동과 직접 관련된 명령은 control 오브젝트로 내립니다.

definition	code	description
SC_SCRIPT_RUN	1	Script를 실행합니다.
SC_SCRIPT_RUN_DEBUG	2	Script를 Debug 모드로 실행합니다.
SC_SCRIPT_PAUSE	3	Script 실행을 멈춥니다.
SC_SCRIPT_STEP	4	Script를 한 줄 실행하고 멈춥니다.
SC_SCRIPT_STOP	5	Script 실행을 종료합니다.
SC_COMMAND_ABORT	10	현재 수행중인 SC_HS_PHASE_DETECT, SC_POS_SENSOR_PARAM, SC_ELECTRIC_PARAM_EST, SC_MECHANIC_PARAM_EST 명령을 중단합니다. system_status 오브젝트에는 SS_COMMAND_ABORTED 가 표시됩니다.
SC_HS_PHASE_DETECT	11	모터를 60도 각도 단위로 Micro-stepping 구동하여 Hallsensor의 연결 순서와 위상을 찾습니다. 찾은 값은 hallsensor_phase 오브젝트에 저장됩니다. 이 명령은 BLDC와 PMSM 모터에만 해당하며, 명령을 실행

		<p>행하였을 때 모터는 1바퀴 정도 회전하게 됩니다. 명령을 실행하기 전에 모터와 부하를 분리하여 모터만 구동할 것을 권장합니다.</p> <p>Hallsensor가 올바르게 연결되었는데도 탐지를 실패하는 경우가 발생하는데, 이 때는 몇 번 재시도 하면 찾게 됩니다.</p> <p>이 명령을 실행하기 전, system_argument에 모터를 구동하기 위한 전류를 rated_current의 %값(10~100)으로 설정해야 합니다.</p>
SC_POS_SENSOR_PARAM	12	<p>모터를 Micro-stepping 구동 또는 Voltage 구동하여 위치 센서 (Hallsensor and/or Encoder) 정보를 찾습니다. 이 명령을 실행하면 position_sensor, encoder_direction, encoder_resolution, no_pole_pairs 오브젝트 값이 변경됩니다.</p> <p>이 명령을 실행하면 모터는 일정 속도로 수~수십 바퀴 회전하게 되는데, 모터와 부하를 분리시킨 상태에서 구동할 것을 권장합니다.</p> <p>Linear motor에서는 soft_limit_check 옵션을 켜고 움직이는 최소 최대 범위(min_position, max_position)를 지정하여 모터가 스톱퍼에 충돌하지 않도록 합니다.</p> <p>이 명령을 실행하기 전, system_argument에 모터를 구동하기 위한 전압을rated_voltage의 %값(2~20)으로 설정해야 합니다.</p>
SC_ELECTRIC_PARAM_EST	13	<p>모터의 전기 파라미터(역기전력 상수(Ke), 권선 저항(R), 권선 d/q축 인덕턴스(Ld/Lq)) 추정을 시작합니다. 이 명령을 실행하면 bemf_constant, resistance, inductance_d, inductance_q, electric_angle_bias, torque_constant 오브젝트 값이 변경됩니다.</p> <p>명령이 실행 중일 때 모터는 전기 파라미터를 찾기 위해 움직이게 되며, 모터와 부하를 분리시킨 상태에서 구동할 것을 권장합니다.</p> <p>Linear motor에서는 soft_limit_check 옵션을 켜고 움직이는 최소 최대 범위(min_position, max_position)를 지정하여 모터가 스톱퍼에 충돌하지 않도록 합니다.</p> <p>이 명령을 실행하기 전, system_argument에 모터를 구동하기 위한 전압을rated_voltage의 %값(10~30)으로 설정해야 합니다.</p>
SC_MECHANIC_PARAM_EST	14	<p>모터의 기계 파라미터(기구부의 관성 모멘트(J), 속도에 따른 마찰 계수(B, C)) 추정을 시작합니다.</p>

		<p>이 명령을 실행하면 <code>moment_of_inertia</code>, <code>viscous_friction</code>, <code>coulomb_friction</code> 오브젝트 값이 변경됩니다.</p> <p>Linear motor에서는 <code>soft_limit_check</code> 옵션을 켜고 움직이는 최소 최대 범위(<code>min_position</code>, <code>max_position</code>)를 지정해야 합니다.</p> <p>이 명령을 실행하기 전에 <code>system_argument</code>에 기구부를 움직이는 속도를 <code>rated_velocity</code>의 %값(10~100)으로 설정해야 합니다.</p> <p>이 명령은 모터와 기구부에 대한 기계 파라미터를 찾기 때문에 모터와 기구부가 연결되어 있어야 합니다. 명령이 실행 중일 때 모터가 급격하게 가감속 운동하게 됨으로 기계 시스템의 강성을 고려하여 %값을 순차적으로 증가시키면서 수행할 것을 권장합니다.</p>
SC_CC_GAIN_STIFFNESS	41	<p>d축과 q축 전류 제어기의 이득에 대한 <code>stiffness</code>를 0에서 100% 사이의 값으로 설정 합니다. 이 값은 d축과 q축 전류 제어기의 P, I 이득을 동시에 조정하게 되며, 높은 값을 설정하면 전류제어 응답성이 좋아지지만 모터가 진동하거나 발산할 수 있습니다.</p> <p>이 명령을 실행하면 <code>current_p_gain_d</code>, <code>current_i_gain_d</code>, <code>current_p_gain_q</code>, <code>current_i_gain_q</code> 오브젝트의 값이 변경 됩니다.</p> <p>명령을 실행하기 전에 <code>stiffness</code> 값을 <code>system_argument</code>에 설정해야 합니다.</p> <p>*주의: 전류 제어기의 이득이 높으면 가청 주파수 대의 '빠~~'하는 소음이 들립니다. 전류 제어기의 이득은 소음이 들리지 않는 정도에서 설정합니다.</p>
SC_PV_GAIN_STIFFNESS	42	<p>위치와 속도 제어기의 이득에 대한 <code>stiffness</code>를 0에서 100% 사이의 값으로 설정 합니다. 이 값은 위치 제어기의 P 이득과 속도 제어기의 P, I 이득을 동시에 조정하게 되며, 높은 값을 설정하면 전류제어 응답성이 좋아지지만 모터가 진동하거나 발산할 수 있습니다.</p> <p>이 명령을 실행하면 <code>position_p_gain</code>, <code>velocity_p_gain</code>, <code>velocity_i_gain</code> 오브젝트의 값이 변경됩니다.</p> <p>명령을 실행하기 전에 <code>stiffness</code> 값을 <code>system_argument</code>에 설정해야 합니다.</p> <p>*주의: 위치와 속도 제어기의 이득이 높으면 모터가 낮은 주파수로 진동합니다. 모터의 진동이 발생하지 않도록 설정해야 합니다.</p>

SC_FWC_SPEED_UP	43	<p>스텝 모터의 속도를 정격 이상으로 높여 구동하기 위해 Flux Weakening Control할 때, 정격 속도에 비해 증가하는 속도의 비율을 %로 설정합니다. 설정 범위는 100에서 500 사이의 값이 됩니다.</p> <p>명령을 실행하기 전에 speed up 값을 system_argument에 설정해야 합니다.</p> <p>*주의: 속도를 정격 이상으로 올리게 되면, 속도가 증가된 비율에 반비례하여 모터의 토크가 감소하게 됩니다.</p>
SC_CC_GAIN_STIF_READ	81	<p>제어기에 설정된 전류 제어기의 이득에 대한 stiffness를 읽어옵니다.</p> <p>읽은 값은 system_argument에 저장됩니다.</p>
SC_PV_GAIN_STIF_READ	82	<p>제어기에 설정된 위치와 속도 제어기의 이득에 대한 stiffness를 읽어옵니다.</p> <p>읽은 값은 system_argument에 저장됩니다.</p>
SC_FWC_SPEED_UP_READ	83	<p>제어기에 설정된 speed up 값을 읽어옵니다.</p> <p>읽은 값은 system_argument에 저장됩니다.</p>
SC_CLEAR_ERROR_CODE	90	<p>error_code 오브젝트 배열의 에러 코드를 모두 삭제합니다.</p>
SC_FACTORY_DEFAULT	98	<p>모터 제어기의 설정과 관련된 모든 오브젝트들을 공장 출하시 기본 값으로 초기화 합니다.</p>
SC_SYSTEM_RESET	99	<p>모터 제어기를 소프트웨어 리셋 합니다.</p> <p>*주의: 모터 파라미터를 변경한 경우 모터 구동을 멈추고 5초 정도 지나야 변경된 파라미터가 Flash Memory에 저장됩니다.</p>
SC_FIRMWARE_DOWNLOAD	100	<p>제어기를 Firmware Download 모드로 진입합니다. Firmware는 Y-mode 프로토콜로 전송할 수 있으며, 전송 도중 실패하거나 중단한 경우 제어기는 계속해서 Firmware Download 모드에 남아있게 됩니다. 이 상태는 제어기에 올바른 펌웨어 다운로드를 완료하였을 때 빠져나올 수 있습니다.</p>

9.6.3 system_argument

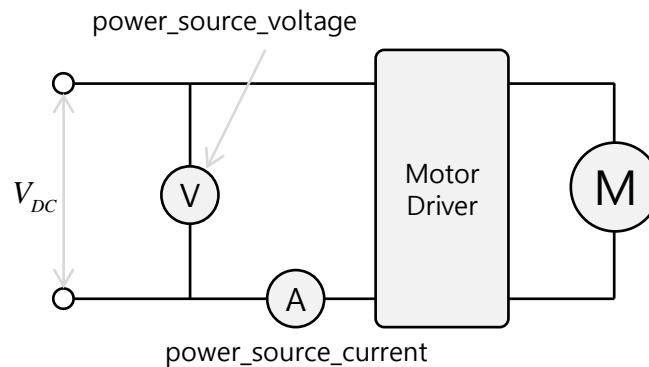
system_control 명령어 중 argument를 필요로 하는 명령어가 있습니다. SC_HS_PHASE_DETECT 명령어의 경우, 명령어를 실행하기 전에 홀센서 탐지 전류 값을 지정해야 하는데, system_argument에 홀센서 탐지 전류 값을 %로 지정하고 명령을 실행하게 됩니다.

ex)

```
system_argument = 40↵
system_control = 11↵
```

9.6.4 power_source_voltage

전원단에 공급되는 전압을 측정합니다.



9.6.5 power_source_current

전원단에 흐르는 전류를 계산합니다. 이 값은 실제 측정된 값이 아니라 모터에 흐르는 전류로부터 계산된 값입니다.

$$\text{power_source_current} = [\text{제어기의 기본 소비 전력}] / \text{power_source_voltage} + [\text{PWM Duty ratio}] * [\text{모터에 흐르는 전류}]$$

9.7 Connection

9.7.1 device_name

모터 제어기의 이름을 설정합니다. 설정 가능한 문자열은 최대 32byte입니다.

ex)

```
device_name=STEP_2P↵           // Device Name을 STEP_2P로 설정함
device_name↵                   // Device Name을 읽어옴
```

device_name은 모터의 구동이나 연결에는 영향을 미치지 않으며, 단지 사용자가 모터 제어를 쉽게 구분하도록 이름을 부여하는 기능입니다.

9.7.2 device_id

RS-485, CAN과 같이 여러 개의 제어기가 하나의 통신 bus에 연결된 상태에서 호스트와 제어기와 통신하는 경우, 각각의 제어기는 서로 다른 device_id를 가지도록 설정해야 합니다. device_id는 0에서 255사이 값으로 설정합니다. 기본 설정 값은 1입니다.

RS-485 통신으로 연결된 제어기에 명령을 내리는 경우, 다음과 같이 device_id를 사용하지 않고 명령을 내릴 수 있습니다. 하지만 RS-485 버스에 한 개 이상의 제어기가 연결되어 있다면 연결된 모든 제어기가 응답하게 됨으로 올바른 응답을 받을 수 없게 됩니다.

ex)

```
software_version#           // 소프트웨어 버전 읽기 명령을 내림  
????????????????????      // 여러 제어기가 명령에 응답하여 수신 문자열이 깨짐
```

상기와 같은 경우, 다음과 같이 device_id를 명령 앞에 사용하여 해당 제어기에만 명령을 내리고 응답 받을 수 있습니다.

ex)

```
1:software_version#         // device_id가 1인 제어기에 명령을 내림  
1:software_version=111      // device_id가 1인 제어기가 응답 함
```

9.7.3 serial_watchdog

RS-232, RS-485 통신에서 모터를 구동하는 명령에 대한 와치독 타임아웃 값을 설정합니다.

만일 설정된 시간 이내에 모터 구동 명령이 수신되지 않으면 모터를 정지하게 됩니다.

모터가 정지된 상태에서 새로운 구동 명령이 내려지면, 제어기는 구동 명령을 수행합니다.

이 값이 0으로 설정된 경우, 와치독 타이머는 작동하지 않습니다.

9.7.4 serial_baudrate

시리얼(RS-232, RS-485, CAN) 통신의 속도를 설정합니다.

RS-232, RS-485의 경우 다음과 같은 통신 속도를 설정 가능합니다.

- 9600
- 19200
- 38400
- 57600

- 115200 (기본값)
- 230400
- 460800
- 921600

CAN의 경우 다음과 같은 통신 속도를 설정 가능합니다.

- 10 - 10K
- 25 - 25K
- 50 - 50K
- 100 - 100K
- 125 - 125K
- 250 - 250K
- 500 - 500K
- 750 - 750K
- 1000 - 1M (기본값)

serial_baudrate 오브젝트는 3개의 원소를 가지는 배열과 같으며(sub-index가 0, 1, 2가 됩니다), 다음과 같이 구분하여 사용할 수 있습니다.

ex)

```
serial_baudrate[0]=115200↵ // 첫 번째 통신 포트(RS-232)의 속도를 설정
serial_baudrate[1]=115200↵ // 두 번째 통신 포트(RS-485)의 속도를 설정
serial_baudrate[2]=1000↵   // 세 번째 통신 포트(CAN)의 속도를 설정
```

만일 배열에서 첫 번째 요소를 액세스 하는 경우에는 다음과 같이 [0]을 생략할 수 있다.

ex)

```
serial_baudrate[0]↵
serial_baudrate[0]=115200
serial_baudrate ↵
serial_baudrate=115200
```

9.8 Pulse/Direction Input

Pulse/Direction 입력으로 제어기에 위치 구동 명령을 내릴수 있습니다.

※ Digital Input 포트의 채널 1과 2는 Pulse/Direction 입력과 채널을 공유하기 때문에, 제어기가 제어 입력을 Pulse/Direction 입력 모드로 전환한 경우 Digital Input1과 Digital Input2의 Enable 상태는 해제됩니다.

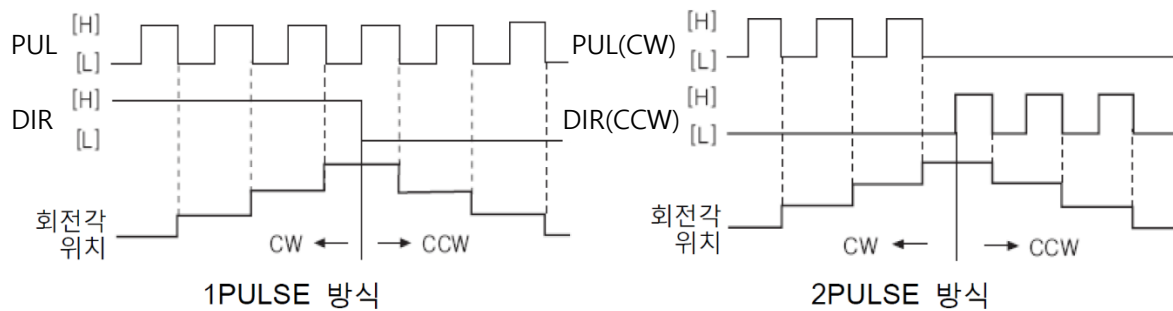
9.8.1 pulse_input_mode

펄스 입력 신호의 펄스 방식을 선택합니다.

- 0 – One Pulse mode (기본값)
- 1 – Two Pulse mode

(1) One pulse mode인 경우는 제어 입력으로 Pulse와 Direction 입력을 받아들입니다. Pulse는 구동 펄스로 카운트 되고 Direction은 카운터의 방향이 됩니다.

(2) Two Pulse mode인 경우는 제어 입력으로 CW pulse와 CCW pulse 입력을 받아들입니다. CW pulse는 정방향 구동 펄스로 카운트 되고 CCW pulse는 역방향 구동 펄스로 카운트 됩니다.

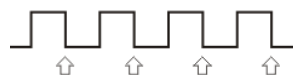


9.8.2 pulse_count_edge

Pulse 입력 신호의 카운트 시점을 결정합니다.

- 0 - Falling Edge: Pulse 입력의 하강 모서리에서 펄스 카운트 (기본값)
- 1 - Rising Edge: Pulse 입력의 상승 모서리에서 펄스 카운트

이 값을 0으로 설정한 경우, 아래 그림과 같이 펄스가 카운트 되는 시점은 Pulse 입력의 하강 모서리에서 카운트 됩니다.



9.8.3 dir_signal_polarity

Direction 입력 신호의 상태에 따라 카운트 방향에 대한 논리 레벨을 결정합니다.

- 0 - Positive Low: Direction 입력이 0일 때 정방향 펄스 카운트 (기본값)
- 1 - Positive High: Direction 입력이 1일 때 정방향 펄스 카운트

9.8.4 gear_numerator

펄스 입력에 대해 전자 기어비를 적용합니다.

이 값은 전자 기어의 분자 값이 됩니다. 다음 식을 이용하여 펄스 입력 신호의 펄스를 카운트 하게 됩니다.

$$PulseCounter = \frac{gear_numerator}{gear_denominator} \times InputPulse$$

9.8.5 gear_denominator

펄스 입력에 대해 전자 기어비를 적용합니다.

이 값은 전자 기어의 분모 값입니다.

9.9 Variables

9.9.1 user_variable

user_variable 오브젝트는 제어기의 운용과 관련된 사용자 파라미터 값들을 저장하는데 사용됩니다. 256개의 사용자 파라미터들을 저장할 수 있으며, 각각의 파라미터는 Sub-index에 의해 구분됩니다. Sub-index의 범위는 0에서 255까지 사용됩니다.

ex)

```
user_variable[0]=1000ㄹ  
user_variable[1]=2000ㄹ  
user_variable[255]=3000ㄹ
```

이 값들은 제어기의 Flash Memory에 저장되어 제어기의 전원이 꺼지더라도 유지됩니다.

9.9.2 temp_variable

temp_variable 오브젝트는 호스트 PC와 제어기에서 실행되는 스크립트간에 명령/상태를 주고받기 위해 사용할 수 있습니다. 256개의 사용자 파라미터들을 저장할 수 있으며, 각각의 파라미터는 Sub-index에 의해 구분됩니다. Sub-index의 범위는 0에서 255까지 사용합니다.

ex)

```
temp_variable[0]=1000d  
temp_variable[1]=2000d
```

이 값들은 제어기 전원이 꺼지면 소실됩니다.

9.10 Error Code

9.10.1 error_code

error_code의 Sub-index 0에는 모터 제어가 시작되고 현재까지 발생한 Error Code의 개수가 설정됩니다. Sub-index 1~15 까지는 Error Code가 발생한 순서대로 기록됩니다.

만일 error_code[0]의 값이 20이라면 error_code[1 ~ 15]에는 최근에 발생한 에러코드 15개만 기록되며, 나머지 5개는 뒤로 밀려나 지워집니다.

다음은 Error Code의 각 코드 값의 범위에 대한 메시지의 종류입니다.

- 1000 ~ 1999: Error
- 2000 ~ 2999: Warning
- 3000 ~ 3999: Information

Error는 보통 모터 구동 중 모터를 더 이상 정상적으로 구동할 수 없는 상황에서 발생되며, Error가 발생하면 fault 플래그가 설정되고 제어기는 Disable 상태가 됩니다. 반면 Warning은 각종 제약이나 제어기 상태에 의해 사용자가 내린 명령을 수행할 수 없을 때 발생합니다. Information은 단지 사용자에게 정보를 전달하기 위해 사용됩니다.

다음 표는 각 코드 값에 대한 설명입니다.

Error Code	Description
1001	DC 모터에서는 Hallsensor Phase 찾기를 수행할 수 없습니다.
1002	Hallsensor의 Sector 정보를 읽어올 수 없습니다.
1003	Encoder Resolution이 설정되지 않았습니다.
1004	Hallsensor Phase가 설정되지 않았습니다.
1005	Hallsensor의 Number of Pole Pairs가 설정되지 않았습니다.
1006	Stall Current(모터가 힘을 발생하여도 회전하지 않음)가 탐지되었습니다.
1007	Position Controller에서 Position Error가 설정된 값을 초과하였습니다.
1008	Velocity Controller에서 Velocity Error가 설정된 값을 초과하였습니다.
1009	Encoder를 읽을 수 없습니다. Encoder 연결을 확인하십시오.
1010	Motor에 전압을 가하여도 전류가 흐르지 않습니다. Motor의 결선을 확인하십시오.
1011	Motor의 단자가 전원과 단락 되었을 가능성이 있습니다.
1012	Motor의 q축 전류가 Overcurrent limit을 초과하였습니다.
1013	Motor의 d축 전류가 Overcurrent limit을 초과하였습니다.
1014	전원의 전압이 Undervoltage limit 아래로 떨어졌습니다.
1015	전원의 전압이 Overvoltage limit을 초과하였습니다.
1016	FET와 Heatsink의 온도가 Overheta limit을 초과하였습니다.
1017	모터에 흐르는 전류에서 진동이 탐지되었습니다.
1018	모터의 속도가 Max Velocity의 150%를 초과하였습니다.
1019	Hallsensor의 Number of Pole Pairs에 비해 Encoder Resolution이 잘못 설정되었습니다.
1020	Script Code 수행도중 Stack overflow가 발생하였습니다.
1021	잘못된 Script Code가 실행되었습니다.
1022	모터의 위치가 Positive Limit 위치를 벗어났습니다.
1023	모터의 위치가 Negative Limit 위치를 벗어났습니다.
1024	Emergency Switch가 켜졌습니다.
1025	모터의 전기 파라미터(R, Ld, Lq, Ke)에 잘못된 값이 설정되었습니다.
1026	모터의 기계 파라미터(J, B, C, TI)에 잘못된 값이 설정되었습니다.

Error Code	Description
2001	System Command가 수행중인 상태에서 Motor 제어 명령을 수행할 수 없습니다.
2002	Motor가 Disable 상태에서 제어 명령을 수행할 수 없습니다.
2003	Digital Input이나 Serial Command에 의한 Motor Stop 상태에서 제어 명령을 수행할 수 없습니다.
2004	Forward Limit 상태에서 제어 명령을 수행할 수 없습니다.

2005	Reverse Limit 상태에서 제어 명령을 수행할 수 없습니다.
2006	Serial 포트의 watchdog timer가 타임아웃 되었습니다. Motor에 Stop 명령을 내립니다.
2007	Factory default 값들을 제어기에 적용합니다.
2008	Motor의 position이 Software forward limit에 도달하였습니다. Motor를 stop 합니다.
2009	Motor의 position이 Software reverse limit에 도달하였습니다. Motor를 stop 합니다.
2010	Digital Input에 의한 Motor Disable(Emergency Stop) 상태에서 제어 명령을 수행할 수 없습니다.
2012	Motor의 q축 전류가 Overcurrent limit을 초과한 상태에서 제어 명령을 수행할 수 없습니다.
2013	Motor의 d축 전류가 Overcurrent limit을 초과한 상태에서 제어 명령을 수행할 수 없습니다.
2014	전원의 전압이 Undervoltage limit 미만인 상태에서 제어 명령을 수행할 수 없습니다.
2015	전원의 전압이 Overvoltage limit을 초과한 상태에서 제어 명령을 수행할 수 없습니다.
2016	FET와 Heatsink의 온도가 Overheta limit을 초과한 상태에서 제어 명령을 수행할 수 없습니다.
2017	Quick stop 상태에서 제어 명령을 수행할 수 없습니다.
2019	Motor가 움직이고 있을 때는 Script를 다운로드 할 수 없습니다.
2020	Script가 실행중일 때는 Script를 다운로드 할 수 없습니다.
2021	Script의 다운로드 size가 잘못 설정되었습니다.
2022	Script의 address가 Script의 size를 벗어나 설정되었습니다.
2023	Script의 code가 size를 벗어나 다운로드 되었습니다.
2024	실행할 수 없는 Script가 적재되어 있습니다.
2030	Stall Current Detection 옵션이 설정되지 않았습니다. Home Search를 시작할 수 없습니다.
2031	Digital Input에서 Home Switch가 설정되지 않았습니다. Home Search를 시작할 수 없습니다.
2032	Digital Input에서 Forward Limit Switch가 설정되지 않았습니다. Home Search를 시작할 수 없습니다.
2033	Digital Input에서 Reverse Limit Switch가 설정되지 않았습니다. Home Search를 시작할 수 없습니다.
2034	Encoder가 Index 신호를 가지고 있지 않습니다. Home Search를 시작할 수 없습니다.
2035	DC 모터에서는 Hallsensor Phase 찾기를 수행할 수 없습니다.
2036	Hallsensor의 Sector 정보를 읽어올 수 없습니다.

Error Code	Description
3001	Motor Driver가 시작되었습니다.
3002	Motor가 Home position 탐색을 완료하였습니다.
3003	Motor의 position을 Motion table에 저장하였습니다.
3004	Motion table을 시작합니다.
3005	Motion table을 종료합니다.
3006	Script를 시작합니다.
3007	Script를 종료합니다.
3008	Motor의 position을 새로운 값으로 설정 합니다.
3010	Notch filter 1의 파라미터를 설정하였습니다.
3011	Notch filter 2의 파라미터를 설정하였습니다.
3012	Notch filter 3의 파라미터를 설정하였습니다.

9.11 Motor Status

9.11.1 control

제어기에 모터의 구동과 관련된 명령을 내립니다.

definition	code	description
MC_DISABLE	0	모터에 공급되는 전원을 차단합니다. disable_oper_action 에 설정한 기능에 따라 모터는 Free run 상태가 되거나 Dynamic Brake가 작동합니다.
MC_ENABLE	1	모터에 전원을 공급하여 구동 준비상태가 됩니다. Position/Velocity/Current/Voltage 명령을 수신하고 모터를 명령에 따라 구동합니다. Enable 될 때 startup_control_mode 의 설정에 따라 Position, Velocity, Current, Voltage 모드 중 하나의 제어 모드가 됩니다.
MC_CLEAR_FAULT	2	제어기에 발생한 fault 오브젝트 비트를 모두 지웁니다.
MC_RESET_POSITION	3	모터의 현재 위치(position_actual 오브젝트)를 0으로 설정합니다.
MC_STOP	6	모터를 감속하여 정지합니다.
MC_QUICK_STOP	7	모터를 긴급 정지합니다. 그리고 ST_QUICK_STOP 상태가 됩니다. Quick stop 상태에서는 모터가 Enable 상태

		<p>에 머무르지만 구동 명령은 실행되지 않습니다. Quick stop 상태를 해제하기 위해서는 MC_ENABLE 명령을 내려야 합니다.</p> <p>Quick stop에 의한 정지 방법은 제어 모드에 따라 다음과 같습니다.</p> <ul style="list-style-type: none"> - Position 모드: Velocity 모드로 바뀜, 속도가 0이 되도록 제어 - Velocity 모드: 속도가 0이 되도록 제어 - Current 모드: 전압 출력 모드로 바뀜, 전압 0을 출력 - 전압 출력 모드: 목표 전압 0을 출력
MC_HOME_SEARCH	9	홈 위치 탐색을 시작합니다.
MC_JOG_POSITIVE	10	정방향 Jog 운전을 시작합니다.
MC_JOG_NEGATIVE	11	역방향 Jog 운전을 시작합니다.
MC_ANALOG_DRIVE	100	Analog Input 또는 Pulse Input 포트의 입력에 의해 구동 명령을 수행하도록 명령 입력 모드를 전환합니다. 모터가 구동 중일 때는 모드 전환이 불가능합니다.
MC_PULSE_DRIVE	101	Pulse/Direction 신호에 의해 구동 명령을 수행하도록 명령 입력 모드를 전환합니다. 모터가 구동 중일 때는 모드 전환이 불가능합니다.
MC_SERIAL_DRIVE	102	Serial Control(RS-232, RS-485, CAN, Ethernet) 구동 명령을 수행하도록 명령 입력 모드를 전환합니다. 모터가 구동 중일 때는 모드 전환이 불가능합니다.

9.11.2 status

제어기의 상태를 표시합니다.

definition	code	description
ST_ENABLE	0x0001	모터에 전원이 공급되고 명령을 수신하여 모터를 구동 가능한 상태입니다.
ST_FAULT	0x0002	Fault가 발생하였으며, 모터는 Disable 됩니다. 발생한 폴트의 상세한 정보는 fault 오브젝트에서 확인 가능합니다.
ST_MOVING	0x0004	모터가 구동 중인 상태입니다.
ST_HOMING	0x0008	모터가 홈 위치로 이동 중인 상태입니다.
ST_HOME_COMP	0x0010	홈 위치 이동이 완료된 상태입니다.
ST_POSITION_CTL	0x0020	위치 구동 모드: 위치 제어 중인 상태입니다.
ST_VELOCITY_CTL	0x0040	속도 구동 모드: 속도 제어 중인 상태입니다.

ST_CURRENT_CTL	0x0080	전류 구동 모드: 전류 제어 중인 상태입니다.
ST_POS_LIMIT	0x0100	모터의 위치가 정방향 또는 역방향 리미트에 도달했습니다.
ST_CURRENT_LIM	0x0200	모터의 전류가 Max Current 근처에 도달했습니다
ST_IN_POSITION	0x0400	모터의 위치가 목적 위치에 도달했습니다.
ST_IN_VELOCITY	0x0800	모터가 목표 속도에 도달했습니다.
ST_DI_STOP	0x1000	Digital Input에서 Stop, Quick Stop 스위치가 작동함
ST_DI_DISABLE	0x2000	Emergency Stop 스위치가 작동했습니다.
ST_QUICK_STOP	0x4000	Digital Input에서 Quick Stop 스위치가 작동하였거나 UI에서 Quick Stop 명령이 내려짐, Quick stop 상태에서 복귀하기 위해서는 MC_ENABLE 명령을 수행해야 합니다.
ST_ANALOG_DRIVE	0x00020000	Analog input, Pulse input에 의해 제어기에 명령이 내려지는 중입니다.
ST_PULSE_DRIVE	0x00040000	Pulse/Direction 신호에 의해 제어기에 명령이 내려지는 중입니다.
ST_SERIAL_DRIVE	0x00080000	시리얼 포트(RS-232, RS-485, CAN, Ethernet)로부터 제어기에 명령이 내려지는 중입니다.
ST_READY	0x20000000	모터 Disable 상태에서 Enable 준비가 되었음을 표시합니다.
ST_DC_STEP	0x40000000	모터의 종류가 DC or STEP 모터이며, Hallsensor를 사용하지 않는 모터임을 표시합니다.
ST_VM_RUN	0x80000000	VM에서 Script가 수행 중입니다.

9.11.3 fault

제어기에 발생한 폴트 상태를 표시합니다.

definition	code	description	error_code
FF_OVER_CURRENT	0x0001	모터에 허용된 한계전류 이상이 흐름 (관련 오브젝트: overcurrent_limit)	1012, 1013
FF_OVER_VOLTAGE	0x0002	제어기의 전원으로부터 허용된 한계전압 이상이 가해짐 (관련 오브젝트: overvoltage_limit)	1015
FF_UNDER_VOLTAGE	0x0004	제어기의 전원으로부터 허용된 한계전압 이하로 떨어짐 (관련 오브젝트: undervoltage_limit)	1014
FF_OVER_HEAT	0x0008	제어기의 FET와 방열판에서 허용된 한계온도 이	1016

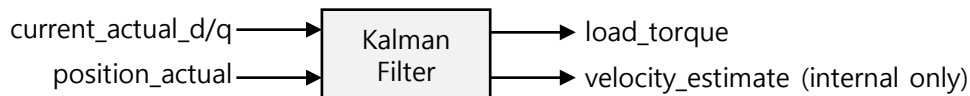
		상으로 온도가 올라감 (관련 오브젝트: overheat_limit)	
FF_SHORT_CIRCUIT	0x0010	모터 단자에 과전류가 흐름, 모터 단자와 전원 단자간 단락이 의심됨	1011
FF_VIBRATION	0x0020	모터 구동 중 진동이 탐지 됨 (관련 오브젝트: vibration_detect)	1017
FF_STALL_CURRENT	0x0040	모터가 멈춘 상태에서 설정된 시간 동안 설정된 값 이상의 전류가 흐름 (관련 오브젝트: stall_current_detect)	1006
FF_POSITION_TRACKING	0x0080	위치 제어기에서 설정된 시간 동안 위치 오차가 설정된 값을 초과 함 (관련 오브젝트: position_track_error)	1007
FF_VELOCITY_TRACKING	0x0100	속도 제어기에서 설정된 시간 동안 속도 오차가 설정된 값을 초과 함 (관련 오브젝트: velocity_track_error)	1008
FF_HALLSENSOR	0x0200	Hallsensor 입력이 들어오지 않거나 비정상적인 조합의 Hallsensor 입력이 들어옴	1002, 1004, 1005
FF_ENCODER	0x0400	Encoder 입력이 들어오지 않거나, 한 바퀴당 Hallsensor 카운트 값과 Encoder 카운트 값이 설 정 값과 일치하지 않음	1003, 1009
FF_MOTOR	0x0800	모터 단자에 전류가 흐르지 않음, 모터 연결이 끊어진 상황이 의심됨	1001, 1010
FF_OVER_SPEED	0x1000	모터의 속도가 정격 속도(rated_velocity)의 200% 이상으로 회전함	1018
FF_POSITIVE_LIMIT	0x2000	모터의 위치가 정방향 리미트를 벗어남	1022
FF_NEGATIVE_LIMIT	0x4000	모터의 위치가 역방향 리미트를 벗어남	1023
FF_POWER	0x8000	인버터 전원이 차단됨 (AC 모터 드라이버에 해당 됨)	
FF_EMERGENCY_SW	0x1000 0	Emergency Switch가 켜짐	
FF_ELECTRIC_PARAM	0x2000 0	Electric Parameter (R, Ld, Lq, Ke) 설정 오류	1025
FF_MECHANIC_PARAM	0x4000 0	Mechanic Parameter (J, B, C, Tl) 설정 오류	1026

9.11.4 temperature

FET와 방열판의 현재 온도 값입니다. 이 값이 `overheat_limit` 설정 값 이상으로 올라가면 제어기를 보호하기 위해 폴트를 발생하고 모터는 Disable 상태가 됩니다.

9.11.5 load_torque

모터에 흐르는 전류와 모터의 속도로부터 추정된 부하 토크 값입니다.



9.12 Motor Drive

모터 구동 명령은 다음 중 하나의 입력 포트를 통해 입력될 수 있습니다.

- Analog Input mode - Analog input, Pulse input에 의해 제어기에 명령이 내려짐
- Pulse Input mode - Pulse/Direction 신호에 의해 제어기에 명령이 내려짐
- Serial Input mode - 시리얼 포트(RS-232, RS-485, CAN, Ethernet)로부터 제어기에 명령이 내려짐

9.12.1 target_position

모터가 도달해야 할 목표 위치를 지정합니다. 제어기가 Enable 상태에 있다면, 이 값은 변경 즉시 적용됩니다.

속도 프로파일을 사용하지 않는 경우(`profile_type = 0`), Moving Average Filter를 거쳐 위치 명령이 완만하게 변하도록 세이핑 할 수 있습니다. 속도 프로파일을 사용하는 경우, Velocity Profile Generation을 거쳐 속도 프로파일의 형태를 사다리꼴(Trapezoidal profile) 또는 사인파(Sinusoidal profile) 형태가 되도록 할 수 있습니다.

만일 제어기가 속도 제어나 전류 제어 상태에서 구동 중일 때 이 값을 변경하면, 제어기는 위치 제어 상태로 변경되고 현재 속도와 위치를 고려하여 `target_position`에 지정된 위치로 이동을 시작합니다.

9.12.2 target_velocity

모터가 도달해야 할 목표 속도를 지정합니다. 제어기가 Enable 상태에 있다면, 이 값은 변경 즉시 적용됩니다.

속도 프로파일을 사용하지 않는 경우(profile_type = 0), Moving Average Filter를 거쳐 속도 명령이 완만하게 변하도록 세이핑 할 수 있습니다. 속도 프로파일을 사용하는 경우, Velocity Profile Generation을 거쳐 속도 프로파일의 형태를 사다리꼴(Trapezoidal profile) 또는 사인파(Sinusoidal profile) 형태가 되도록 할 수 있습니다.

만일 제어기가 위치 제어나 전류 제어 상태에서 구동 중일 때 이 값을 변경하면, 제어기는 속도 제어 상태로 변경되고 현재 속도를 고려하여 target_velocity에 지정된 속도로 가속이나 감속하여 목표 속도에 도달하게 됩니다.

9.12.3 target_current_d/q

모터가 도달해야 할 d축/q축 목표 전류를 지정합니다. 제어기가 Enable 상태에 있다면, 이 값은 변경 즉시 적용됩니다.

만일 제어기가 위치 제어나 속도 제어 상태에서 구동 중일 때 이 값을 변경하면, 제어기는 전류 제어 상태로 변경되고 target_current_d/q에 지정된 전류를 모터에 흐르도록 제어합니다.

9.12.4 target_voltage_d/q

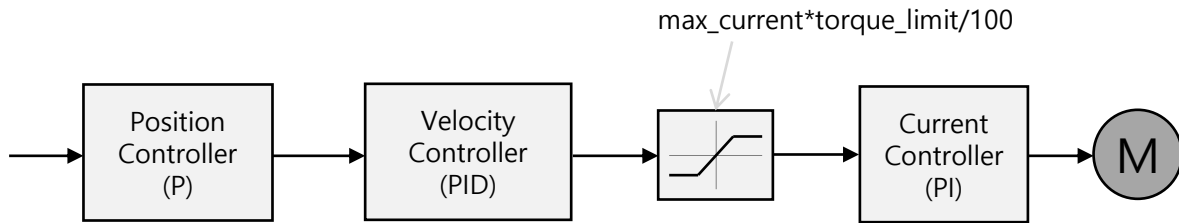
모터에 가해지는 d축/q축 전압을 지정합니다. 제어기가 Enable 상태에 있다면, 이 값은 변경 즉시 적용됩니다.

만일 제어기가 위치 제어나 속도 제어, 전류 제어 상태에서 구동 중일 때 이 값을 변경하면, 제어기는 모든 제어 상태를 해제하고 target_voltage_d/q에 지정된 전압을 모터에 가합니다.

q축 전압이 직접적으로 출력되는 경우, d축 전압은 0으로 출력됩니다.

9.12.5 torque_limit

위치 제어나 속도 제어시 전류의 최대 값을 제한함으로 제어기의 토크 리미트를 %단위로 설정합니다.



torque_limit 오브젝트 값은 모터가 Enable 되면서 100%로 재설정 됩니다.

9.12.6 position_demand

위치 제어기가 추종해야 할 demand 값입니다. position_demand 값은 목표 위치(target_position)에 도달하기 위한 위치 경로 값으로 속도 프로파일 생성기에 의해 만들어집니다.

$$\text{position_error} = \text{position_demand} - \text{position_actual}$$

9.12.7 velocity_demand

속도 제어기가 추종해야 할 demand 값입니다. velocity_demand 값은 목표 속도(target_velocity)에 도달하기 위한 속도 경로 값으로 속도 프로파일 생성기에 의해 만들어지고, 여기에 위치 제어기의 출력 값이 더해집니다.

$$\text{velocity_error} = \text{velocity_demand} - \text{velocity_actual}$$

9.12.8 current_demand_d/q

d축과 q축 전류 제어기가 추종해야 할 demand 값입니다. 만일 위치 제어기나 속도 제어기가 동작 중이라면 속도 제어기의 출력 값이 되고, 전류 제어기만 동작 중이라면 목표 전류(target_current_d/q) 값으로 설정됩니다.

$$\text{current_error_d} = \text{current_demand_d} - \text{current_actual_d}$$

$$\text{current_error_q} = \text{current_demand_q} - \text{current_actual_q}$$

9.12.9 voltage_demand_d/q

모터에 가해지는 d축과 q축 전압 값입니다. 만일 전류 제어기가 동작 중이라면 전류 제어기의 출력 값이 되고, 목표 전압(target_voltage_d/q)이 직접 설정되었다면 목표 전압 값이 됩니다.

9.12.10 acceleration_demand

속도 프로파일 생성기가 만들어내는 속도 프로파일을 미분한 값입니다. Feed-forward 제어기에서 토크 값을 전향보상할 때 사용합니다.

9.12.11 position_actual

Encoder나 Hallsensor로부터 읽은 카운트 값으로 모터의 현재 위치에 해당하는 값입니다.

9.12.12 velocity_actual

position_actual을 미분한 값입니다.

***주의: 이 값은 Encoder나 Hallsensor로부터 읽은 위치 값으로부터 Kalman Filter를 거쳐 추정된 속도라서 모터 제어기에 설정한 Mechanical Parameter가 실제 기구부의 파라미터와 다를 경우 바이어스가 생길 수 있습니다.**

velocity_actual의 절대값이 max_velocity의 150%를 초과하면 FF_OVER_SPEED 폴트를 발생시킵니다.

9.12.13 current_actual_d/q

모터의 U, V, W 단자의 전류센서로부터 읽은 현재 전류 값을 d축과 q축 전류로 분해한 값입니다.

두 값 중 하나라도 overcurrent_limit 설정 값을 초과하면 FF_OVER_CURRENT 폴트를 발생시킵니다.

9.12.14 acceleration_actual

velocity_actual 을 미분한 값입니다.

***주의:** 이 값은 Kalman Filter에 의해 추정된 속도를 미분한 값으로, 위치의 양자화 잡음 및 기타 왜곡으로 인해 추정한 값에 상당한 오차를 포함하고 있다.

9.12.15 position_error

위치 제어기에 입력되는 값입니다.

$\text{position_error} = \text{position_demand} - \text{position_actual}$

위치 제어 모드에서 position_error 값이 position_track_error 오브젝트로 설정한 조건을 벗어나게 되면 FF_POSITION_TRACKING 폴트를 발생시킵니다.

9.12.16 velocity_error

속도 제어기에 입력되는 값입니다.

$\text{velocity_error} = \text{velocity_demand} - \text{velocity_actual}$

속도 제어 모드에서 velocity_error 값이 velocity_track_error 오브젝트로 설정한 조건을 벗어나게 되면 FF_VELOCITY_TRACKING 폴트를 발생시킵니다.

9.12.17 current_error_d/q

d축과 q축 전류 제어기에 입력되는 값입니다.

$\text{current_error_d} = \text{current_demand_d} - \text{current_actual_d}$

$\text{current_error_q} = \text{current_demand_q} - \text{current_actual_q}$

9.13 Position Limit

모터 구동 중 사용자의 Position/Velocity/Current/Voltage 명령에 의해 모터의 위치가 최소/최대 리미트를 벗어나는 경우 모터는 limit_action에 설정된 동작을 실행하게 됩니다.

9.13.1 soft_limit_check

소프트웨어 위치 리미트 사용 여부를 결정합니다.

- 0 – 소프트웨어 위치 리미트 사용하지 않음
- 1 – 소프트웨어 위치 리미트 사용 함

***주의: 소프트웨어 위치 리미트는 Pulse 입력 모드(ST_PULSE_DRIVE)에서는 적용되지 않습니다.**

9.13.2 min_position

소프트웨어 위치 리미트의 최소 값입니다. soft_limit_check가 1로 설정되고 position_actual 값이 min_position보다 작아진 경우 limit_action에 설정된 동작을 수행합니다.

이 값은 Pulse/Direction Input 모드에서는 적용되지 않습니다.

9.13.3 max_position

소프트웨어 위치 리미트의 최대 값입니다. soft_limit_check가 1로 설정되고 position_actual 값이 max_position보다 커진 경우 limit_action에 설정된 동작을 수행합니다.

이 값은 Pulse/Direction Input 모드에서는 적용되지 않습니다.

9.13.4 limit_action

소프트웨어 위치 리미트를 벗어나거나 Digital Input 포트의 Limit +, Limit - 입력이 들어오면 limit_action에 설정된 동작을 실행하게 됩니다. 설정 가능한 동작은 다음과 같습니다.

- 0 - Disable & Fault
- 1 - Quick Stop
- 2 - Stop
- 3 - Hold Current Position

Disable & Fault 상태가 된 경우에는 모터가 Disable 상태로 바뀌면서 fault에 FF_POSITIVE_LIMIT 또는 FF_NEGATIVE_LIMIT 플래그를 표시합니다.

Quick Stop 상태가 된 경우에는 status에 ST_QUICK_STOP 플래그가 표시되며 모터가 Enable 상태에서 새로운 구동 명령을 수행하지 않게 됩니다. 이 때는 Enable 명령을 제어기에 내림으로 새로운 구동 명령을 수행할 수 있게 됩니다.

Quick Stop 되었다가 Quick Stop 상태가 해제된 경우와 Stop 상태가 된 경우, Hold Current Position 상태가 된 경우에는 다음과 같이 동작합니다.

(1) 모터의 위치가 min_position을 벗어난 경우

- 위치 구동: 위치 명령이 min_position보다 작은 명령은 무시되고 min_position보다 큰 명령은 실행됩니다.
- 속도 구동: 속도 명령이 0보다 작은 명령은 무시되고 0보다 큰 명령은 실행됩니다.
- 전류 구동: 전류 명령이 0보다 작은 명령은 무시되고 0보다 큰 명령은 실행됩니다.
- 전압 구동: 전압 명령이 0보다 작은 명령은 무시되고 0보다 큰 명령은 실행됩니다.

(2) 모터의 위치가 max_position을 벗어난 경우

- 위치 구동: 위치 명령이 max_position보다 큰 명령은 무시되고 max_position보다 작은 명령은 실행됩니다.
- 속도 구동: 속도 명령이 0보다 큰 명령은 무시되고 0보다 작은 명령은 실행됩니다.
- 전류 구동: 전류 명령이 0보다 큰 명령은 무시되고 0보다 작은 명령은 실행됩니다.
- 전압 구동: 전압 명령이 0보다 큰 명령은 무시되고 0보다 작은 명령은 실행됩니다.

9.14 Homing

9.14.1 homing_method

제어기에서 제공하는 홈잉 방법은 다음과 같습니다.

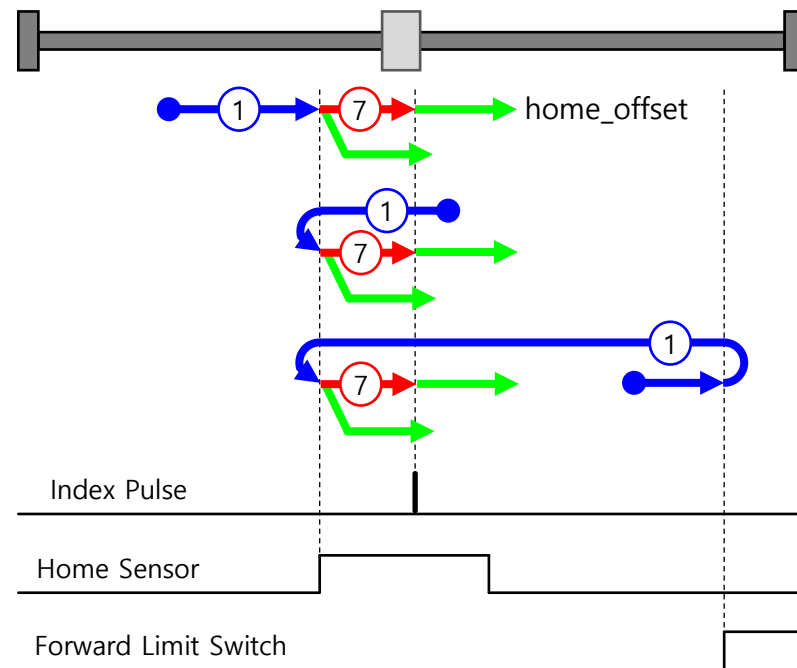
- 0 - Current Position
- 1 - Home Switch Positive Speed
- 2 - Home Switch Negative Speed
- 3 - Forward Limit Switch
- 4 - Reverse Limit Switch
- 5 - Forward Stall Detect
- 6 - Reverse Stall Detect
- 7 - Home Switch Positive Speed & Encoder Index
- 8 - Home Switch Negative Speed & Encoder Index
- 9 - Forward Limit Switch & Encoder Index
- 10 - Reverse Limit Switch & Encoder Index
- 11- Forward Stall Detect & Encoder Index
- 12- Reverse Stall Detect & Encoder Index
- 13 – Encoder Index Positive Speed
- 14 – Encoder Index Negative Speed

(1) Current position

현재 위치를 홈 위치로 설정합니다. 이 때는 home_offset 이동이 적용되지 않습니다.

(2) Home Switch Positive Speed, Home Switch Positive Speed & Encoder index

Digital input 포트에 연결된 Home switch를 사용하여 정방향 홈 위치 이동을 수행합니다(빨간색 화살표). Encoder index 펄스를 사용하는 홈잉 방법에서는 Index 펄스가 입력될 때까지 정방향으로 탐색을 계속합니다(파란색 화살표). Home switch나 Index pulse 탐색이 끝나면 home_offset에 설정된 변위만큼 이동합니다.



Home switch를 사용한 정방향 홈잉에서는 슬라이더가 홈잉을 시작하는 위치에 따라 3가지 경우가 있을 수 있습니다.

상기 그림의 처음과 같이 슬라이더의 위치가 Home sensor의 좌측에 있어 정방향 이동할 때 Home sensor를 감지 가능한 경우입니다.

두 번째에서는 Home sensor 신호가 켜진 경우인데, 이 때는 Home sensor 신호가 꺼질 때까지 역방향으로 이동하다가 다시 Home sensor 신호가 켜질 때까지 정방향 이동합니다.

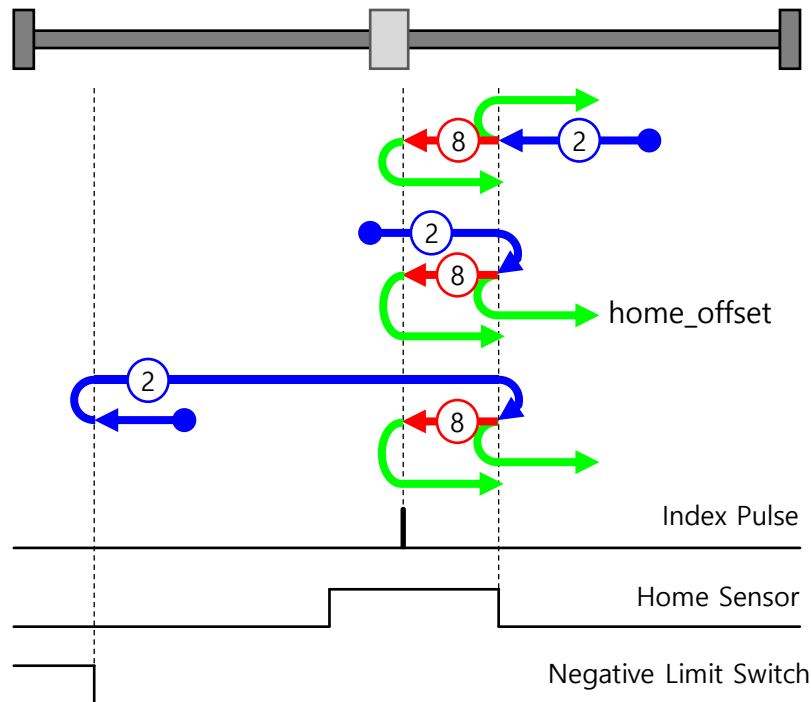
마지막으로는 슬라이더의 위치가 Home sensor의 우측에 있어 이미 Home sensor를 지나친 경우입니다. 이 때는 Forward Limit switch 입력이 켜질 때까지 이동하다가 방향을 바꾸어 Home sensor가 한 번 켜지고 꺼질 때까지 이동하다가 다시 Home sensor 신호가 켜질 때까지 정방향 이동합니다.

(3) Home Switch Negative Speed, Home Switch Negative Speed & Encoder index

Digital input 포트에 연결된 Home switch를 사용하여 역방향 홈 위치 이동을 수행합니다(빨간색 화살표).

Encoder index 펄스를 사용하는 홈잉 방법에서는 Index 펄스가 입력될 때까지 역방향으로 탐색을 계속합니다(파란색 화살표).

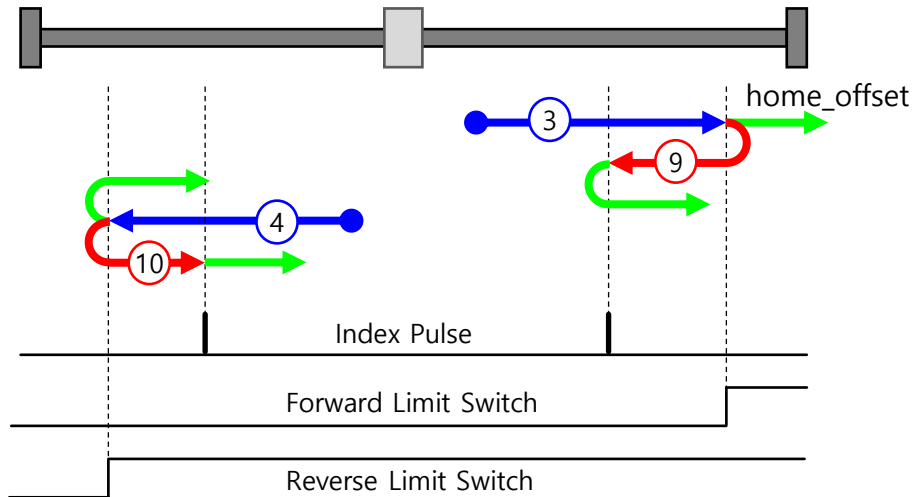
Home switch나 Index pulse 탐색이 끝나면 home_offset에 설정된 변위만큼 이동합니다.



Home switch를 사용한 역방향 홈잉에서는 슬라이더가 홈잉을 시작하는 위치에 따라 3가지 경우가 있을 수 있습니다. 상기 그림의 처음과 같이 슬라이더의 위치가 Home sensor의 우측에 있어 역방향 이동할 때 Home sensor를 감지 가능한 경우입니다. 두 번째에서는 Home sensor 신호가 켜진 경우인데, 이 때는 Home sensor 신호가 꺼질 때까지 정방향으로 이동하다가 다시 Home sensor 신호가 켜질 때까지 역방향 이동합니다. 마지막으로 슬라이더의 위치가 Home sensor의 좌측에 있어 이미 Home sensor를 지나친 경우입니다. 이 때는 Reverse Limit switch 입력이 켜질 때까지 이동하다가 방향을 바꾸어 Home sensor가 한 번 켜지고 꺼질 때까지 이동하다가 다시 Home sensor 신호가 켜질 때까지 역방향 이동합니다.

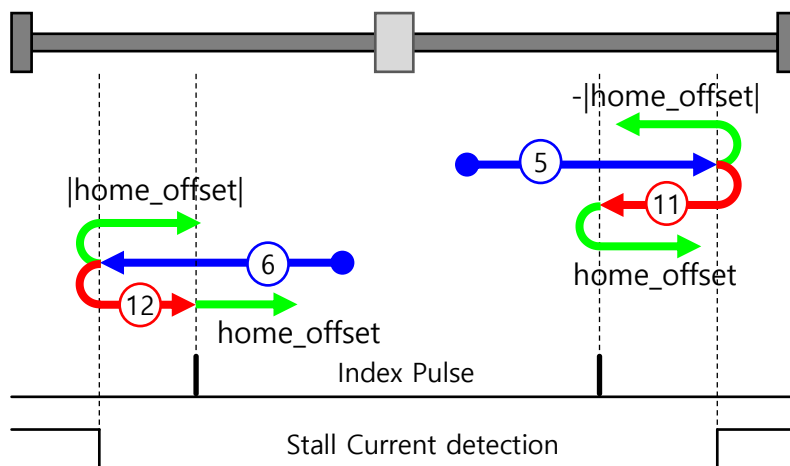
(4) Forward/Reverse Limit switch, Forward/Reverse Limit switch and Encoder Index

Digital input 포트에 연결된 Forward/Reverse Limit switch를 사용하여 홈 위치 이동을 수행합니다(빨간색 화살표). Encoder index 펄스를 사용하는 홈잉 방법에서는 Index 펄스가 입력될 때까지 탐색을 계속합니다(파란색 화살표). Forward/Reverse Limit switch나 Index pulse 탐색이 끝나면 home_offset에 설정된 변위만큼 이동합니다.



(5) Forward/Reverse Stall Current detect, Forward/Reverse Stall Current detect and Encoder Index

슬라이더가 좌우의 스톱퍼에 부딪혔을 때 stall current가 탐지되는 것을 사용하여 홈 위치 이동을 수행합니다(빨간색 화살표). Encoder index 펄스를 사용하는 홈잉 방법에서는 Index 펄스가 입력될 때까지 탐색을 계속합니다(파란색 화살표). Forward/Reverse Stall current나 Index pulse 탐색이 끝나면 home_offset에 설정된 변위만큼 이동합니다.



상기 과정에서 home_offset 이동 방법이 일반적이지 않게 작동합니다. Forward Stall current가 감지된 후 홈 이동을 완료하는 경우에는 home_offset의 절대값에 역방향 이동합니다. 반면 Reverse Stall current가 감지된 후 홈 이동을 완료하는 경우에는 home_offset의 절대값에 정방향 이동합니다.

(6) Encoder Index Positive/Negative Speed

엔코더의 Index pulse를 탐지하는 정방향 또는 역방향 홈 위치 이동을 수행합니다. Index pulse 탐색이 끝나면 home_offset에 설정된 변위만큼 이동합니다.

9.14.2 homing_velocity

홈 위치 탐색을 위해 이동하는 속도를 설정합니다. 이 속도는 home_offset 이동에는 적용되지 않습니다.

9.14.3 home_offset

Home sensor, Positive Limit switch, Negative Limit switch를 찾았거나 Encoder의 Index pulse를 찾았을 때의 위치로부터 오프셋 위치를 설정합니다. home_offset이 설정된 경우, 모터는 홈 위치이동을 완료하기 전에 home_offset에 설정된 위치만큼 이동하게 됩니다.

home_offset 이동에는 Velocity Profile에 설정된 속도와 가속도로 이동하게 됩니다.

9.14.4 home_position

홈 위치이동을 완료한 후, position_actual에 적재되는 값입니다.

9.15 Rated Values

9.15.1 rated_voltage

모터의 정격 전압을 설정합니다. rated_voltage는 모터 구동을 위해 모터에 가하는 최대 전압입니다. 모터에 내려지는 전압 명령의 절대값은 rated_voltage 값보다 커질 수 없습니다.

※ Power supply에서 공급하는 전압은 rated_voltage 설정 값보다 약 20% 정도 높은 전압을 공급하는 것이 좋습니다. 만일 정격 24V 모터를 최고 속도에서 최대 토크로 구동하고자 할 때 전압 여유가 없다면 모터가 최고 속도에서 토크를 제대로 내지 못하게 됩니다.

9.15.2 rated_current

모터의 정격 전류를 설정합니다. 부하가 있는 상태에서 모터를 가감속 하여 구동할 때 모터에 연속으로 흘릴 수 있는 전류입니다. 모터에 내려지는 전류 명령의 절대값은 rated_current 값보다 커질 수 없습니다.

※ rated_current는 Velocity Profile의 profile_acceleration과 profile_deceleration을 계산하는데 사용됩니다. rated_current 값에 비례하여 가속도와 감속도 값이 설정됩니다.

9.15.3 rated_velocity

모터의 정격 회전 속도를 설정합니다. 부하가 있는 상태에서 모터를 연속 구동할 때의 속도가 됩니다. 모터에 내려지는 속도 명령의 절대값은 rated_velocity 값보다 커질 수 없습니다.

9.15.4 max_current

모터에 순간적으로 흘릴 수 있는 최대 전류입니다. 전류 제어기는 모터에 흐르는 d축과 q축 전류가 max_current를 넘지 않도록 제한합니다.

※ 일반적으로 max_current는 rated_current의 150% ~ 300% 값으로 설정합니다.

9.15.5 max_velocity

무부하 상태에서 모터에 rated_voltage가 가해질 때의 속도입니다.

모터의 회전 속도가 max_velocity의 150%를 초과하면 FF_OVER_SPEED fault를 발생시킵니다.

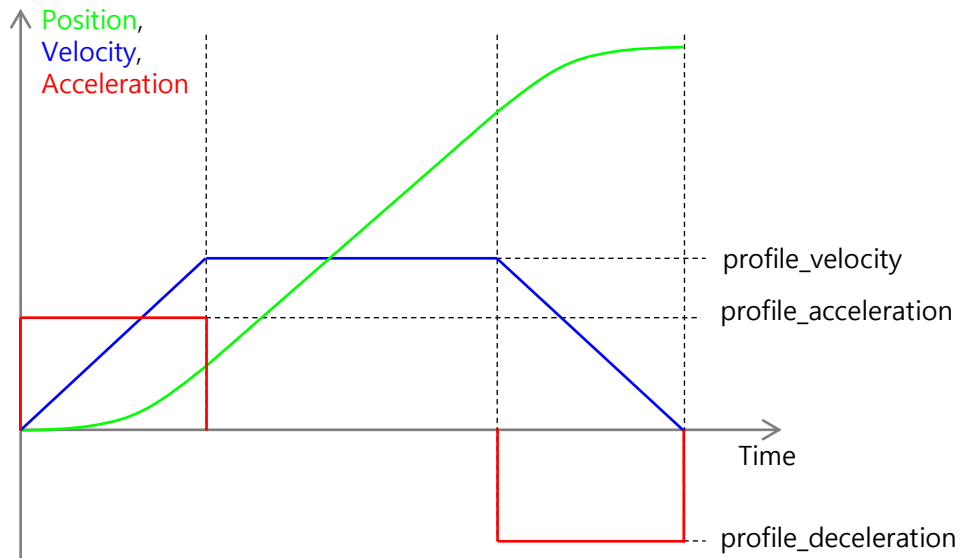
9.16 Velocity Profile

9.16.1 profile_type

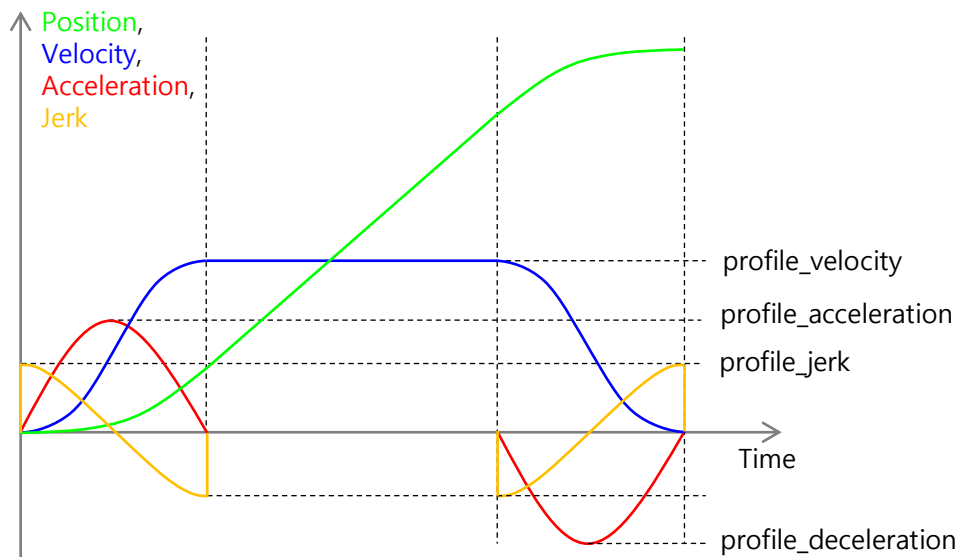
제어기의 위치와 속도 명령이 내려질 때 목적 위치나 속도로 이동하기 위한 프로파일 형상을 결정합니다.

- 0 - None
- 1 - Trapezoidal (기본값)
- 2 - Sinusoidal

다음은 사다리꼴 형태의 속도 프로파일을 사용할 때의 위치와 속도, 가속도 그래프입니다.



다음은 사인파 형태의 속도 프로파일을 사용할 때의 위치와 속도, 가속도, 저크 그래프입니다.



9.16.2 `profile_velocity`

속도 프로파일을 사용할 때 적용되는 최고 속도 값입니다.

이 값을 `rated_velocity`와 같거나 작게 설정해야 합니다.

9.16.3 profile_acceleration

속도 프로파일을 사용할 때 적용되는 가속도 값입니다.

이 값은 다음 식으로부터 계산된 값으로 설정하는 것을 권장합니다.

$$\text{profile_acceleration} = \frac{\text{torque_constant} \times \text{rated_current}}{\text{moment_of_inertia}}$$

하지만 가감속 구간의 시간이 짧고 큰 토크가 필요하다면 다음과 같이 설정합니다.

$$\text{profile_acceleration} = \frac{\text{torque_constant} \times \text{max_current}}{\text{moment_of_inertia}}$$

※ 계산시 단위 변환에 유의해야 합니다.

※ 상기 식은 다음 식으로부터 유도됩니다: $J\alpha = K_t i$

9.16.4 profile_deceleration

속도 프로파일을 사용할 때 적용되는 감속도 값입니다.

이 값도 profile_acceleration 과 마찬가지로 설정되어야 합니다.

9.16.5 profile_jerk

속도 프로파일을 사용할 때 적용되는 저크 값입니다.

이 값은 Sinusoidal 프로파일에서만 사용됩니다.

9.17 Fault Limits

9.17.1 overheat_limit

FET와 방열판의 온도(temperature)가 overheat_limit에 설정된 값을 넘어가면 FF_OVER_HEAT fault를 발생합니다. 보통 이 값은 제어기의 구동부 소자(FET)가 정상 동작 가능한 온도 범위 내에서 설정되며, FET와 방열판의 온도가 overheat_limit 전류 이상으로 올라가면 제어기를 보호하기 위하여 fault를 발생하고 모터에 전력 공급을 차단합니다.

※ 원인

- 모터에 다량의 전류를 흘림으로 FET에서 열이 발생하고, 발생한 열을 방열판에서 효과적으로 배출하지 못함

※ 해결책

- 모터를 구동하는 가속도와 감속도 프로파일 값을 낮추어 모터에 흐르는 전류의 양을 줄임.
- 제어기 방열판에 팬을 달거나 환풍을 하여 발생한 열을 효과적으로 배출해야 함.

9.17.2 overcurrent_limit

모터에 흐르는 전류(current_actual_d/q)가 overcurrent_limit에 설정된 값을 넘어가면 FF_OVER_CURRENT Fault를 발생합니다. 보통 이 값은 제어기의 구동부 소자(FET)를 통해 흘릴 수 있는 최대 전류로 설정되며, 모터에 흐르는 전류가 overcurrent_limit 전류 이상으로 올라가면 제어기를 보호하기 위하여 fault를 발생하고 모터에 전력 공급을 차단합니다.

※ 원인

- 위치 제어기 또는 속도 제어기의 이득이 너무 높게 설정됨
- 전원이나 모터에서 발생하는 노이즈가 제어기로 유입
- 모터 단자의 단락(short circuit)을 점검

※ 해결책

- 제어기의 전류 제어 이득을 낮춤
- 전원 입력단에 노이즈 필터 장착

9.17.3 overvoltage_limit

전원단에서 제어기로 입력되는 전압(power_source_voltage)이 overvoltage_limit에 설정된 값을 넘어가면 FF_OVER_VOLTAGE fault를 발생합니다. 보통 이 값은 제어기를 구성하는 각종 전자 부품들이 견딜 수 있는 최대 전압으로 설정되며, 전원 전압이 overvoltage_limit 전압 위로 올라가면 제어기를 보호하기 위하여 fault를 발생하고 모터에 전력 공급을 차단합니다.

※ 원인

- 전원 입력단에서의 과전압 유입 또는 전원의 고장
- 모터가 관성이나 외력에 의해 발전기 상태로 구동 함:
 - . 정지한 모터를 강제로 회전할 때
 - . 관성이 큰 부하가 연결된 모터를 정격 이상의 감속도로 감속할 때

※ 해결책

- 속도 프로파일의 감속도(profile_deceleration)를 낮은 값으로 설정
- 모터의 감속 시 기구적인 감속 브레이크 사용
- 전원 입력단에 dynamic brake 회로 사용
- 전원 입력단과 병렬로 대용량 커패시터 연결
- 전원 입력단과 병렬로 충방전이 가능한 배터리 연결 (Ex: 납 축전지0)

9.17.4 undervoltage_limit

전원단에서 제어기로 입력되는 전압(power_source_voltage)이 undervoltage_limit에 설정된 값 아래로 떨어지면 FF_UNDER_VOLTAGE fault를 발생합니다. 보통 이 값은 제어기가 정상적으로 동작하기 위한 최소 전압으로 설정되며, 전원 전압이 undervoltage_limit 전압 아래로 떨어지면 제어기를 보호하기 위하여 fault를 발생하고 모터에 전력 공급을 차단합니다.

※ 원인

- 전원 입력단에서의 전압 강하 또는 전원의 고장
- 전원이 모터를 가속하는데 필요한 충분한 전류를 공급하지 못함

※ 해결책

- 모터의 정격 전류보다 큰 용량을 가지는 파워 서플라이 사용
- 속도 프로파일의 가속도(profile_acceleration)를 낮은 값으로 설정
- 전원 입력단과 병렬로 대용량 커패시터 연결
- 전원 입력단과 병렬로 충방전이 가능한 배터리(ex. 납축전지) 연결

9.18 Fault Detection

9.18.1 vibration_detect

전류 제어기에서 발생하는 진동을 탐지하여 설정된 조건에 부합되는 경우 FF_VIBRATION fault를 발생합니다.

- 0 – Disable
- 1 - Current Frequency > 20Hz and RMS > 30% of Rated Current
- 2 - Current Frequency > 50Hz and RMS > 30% of Rated Current
- 3 - Current Frequency > 100Hz and RMS > 30% of Rated Current
- 4 - Current Frequency > 20Hz and RMS > 60% of Rated Current
- 5 - Current Frequency > 50Hz and RMS > 60% of Rated Current
- 6 - Current Frequency > 100Hz and RMS > 60% of Rated Current

- 7 - Current Frequency > 20Hz and RMS > 90% of Rated Current
- 8 - Current Frequency > 50Hz and RMS > 90% of Rated Current
- 9 - Current Frequency > 100Hz and RMS > 90% of Rated Current

※ 원인

- 위치 제어기 또는 속도 제어기의 이득이 너무 높게 설정됨
- 구동부에 기구적으로 구성된 벨트, 기어, 커플링 등의 탄성체에 의한 진동

※ 해결책

- 위치 제어기 또는 속도 제어기의 이득을 낮춤

9.18.2 stall_current_detect

모터가 스톱퍼에 부딪혀 멈춘 상태에서 설정된 시간 동안 설정된 값 이상의 전류가 흐르면 FF_STALL_CURRENT fault를 발생합니다.

- 0 - Disable
- 1 - Zero Velocity and 50% of Max Current while 100ms
- 2 - Zero Velocity and 60% of Max Current while 200ms
- 3 - Zero Velocity and 70% of Max Current while 500ms
- 4 - Zero Velocity and 80% of Max Current while 1000ms
- 5 - Zero Velocity and 90% of Max Current while 2000ms
- 6 - Zero Velocity and 90% of Max Current while 1000ms
- 7 - Zero Velocity and 90% of Max Current while 500ms
- 8 - Zero Velocity and 90% of Max Current while 200ms
- 9 - Zero Velocity and 90% of Max Current while 100ms

※ 원인

- 모터가 스톱퍼에 부딪혀 더 이상 움직일 수 없는 경우
- 구동부의 마찰이 커 모터가 부하를 움직이지 못하는 경우

※ 해결책

- 더 큰 용량의 모터를 사용하거나 감속기의 감속비를 높여 출력 토크를 늘려야 함

9.18.3 position_track_error

위치 제어기에서 설정된 시간 동안 위치 오차(position_error)가 설정 값 이상을 초과하면 FF_POSITION_TRACKING fault를 발생합니다.

- 0 - Disable
- 1 - Position error > 100 counts while 100ms

- 2 - Position error > 500 counts while 200ms
- 3 - Position error > 2000 counts while 500ms
- 4 - Position error > 5000 counts while 1000ms
- 5 - Position error > 20000 counts while 2000ms

※ 원인

- 위치제어 모드에서 목표 위치 제대로 추종하지 못함

※ 해결책

- `profile_type`을 '1 - Trapezoidal' 또는 '2 - Sinusoidal'로 설정
- `profile_velocity`를 모터가 구동 가능한 최대 속도 이하로 낮춤
- `profile_acceleration`, `profile_deceleration`을 모터가 구동 가능한 최대 가속도 이하로 낮춤

9.18.4 velocity_track_error

속도 제어기에서 설정된 시간 동안 속도 오차(`velocity_error`)가 설정 값 이상을 초과하면 FF_VELOCITY_TRACKING fault를 발생합니다.

- 0 - Disable
- 1 - Velocity error > 100 rpm(or mm/s) while 100ms
- 2 - Velocity error > 200 rpm(or mm/s) while 200ms
- 3 - Velocity error > 500 rpm(or mm/s) while 500ms
- 4 - Velocity error > 1000 rpm(or mm/s) while 1000ms
- 5 - Velocity error > 2000 rpm(or mm/s) while 2000ms

※ 원인

- 속도제어 모드에서 목표 속도 제대로 추종하지 못함

※ 해결책

- `profile_type`을 '1 - Trapezoidal' 또는 '2 - Sinusoidal'로 설정
- `profile_velocity`를 모터가 구동 가능한 최대 속도 이하로 낮춤
- `profile_acceleration`, `profile_deceleration`을 모터가 구동 가능한 최대 가속도 이하로 낮춤

9.19 Motor Properties

9.19.1 motor_type

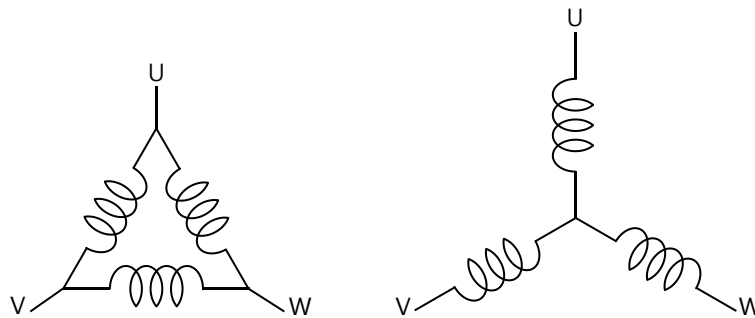
제어기에 연결된 모터의 종류를 설정합니다.

각 모터드라이버에 맞는 모터 종류를 설정하셔야 정상 동작이 가능합니다.

- 0 - DC Rotary
- 1 - DC Linear
- 2 - BLDC Rotary
- 3 - BLDC Linear
- 4 - PMSM Rotary
- 5 - PMSM Linear
- 6 - STEP Rotary (3 phase)
- 7 - STEP Linear (3 phase)
- 8 - STEP Rotary (2 phase)
- 9 - STEP Linear (2 phase)

모터의 종류를 변경할 때는 모터가 Disable 상태에서 수행하도록 합니다. 만일 Enable 상태에서 모터의 종류를 변경하면 제어기는 Disable 상태로 전환됩니다.

3 phase 스텝 모터는 Delta 결선과 Star 결선 두 가지 방식을 사용할 수 있습니다.



2 phase 스텝 모터는 Unipolar 방식과 Bipolar 방식 둘 다 사용 가능합니다. Unipolar 방식에서 코일中间的의 탭은 사용하지 않도록 결선합니다.



9.19.2 motor_direction

모터의 구동 방향을 설정합니다. motor_direction을 1로 설정할 경우, 엔코더 카운트 값이나 속도

의 부호는 바뀌지 않고 모터의 회전 방향만 반대로 바뀝니다.

- 0 - Normal
- 1 - Invert

***주의:** 모터의 구동 중 방향을 변경하는 용도로 사용하지 말아야 합니다. 모터의 구동 방향을 변경할 때는 모터가 **Disable** 상태에 있거나 정지한 상태에서 수행하도록 합니다.

9.20 Feed-forward Controller

9.20.1 feedforward_option

Feed-forward 제어기의 선택 사항을 설정합니다.

- bit 0 – Back-EMF Voltage Compensation
- bit 1 – Inertia/Friction Compensation
- bit 2 – Load Torque Compensation
- bit 3 – Use Position Controller for Velocity Control
- bit 4 – Maximum Torque per Ampere Control
- bit 5 – Flux Weakening Control (STEP only)

(1) Back-EMF Voltage Compensation

모터의 구동에 의해 발생하는 역기전력과 d축과 q축간 간섭 전류를 보상합니다.

이 옵션은 항상 켜져 있어야 합니다.

(2) Inertia/Friction Compensation

모터 및 구동부 기계장치에 의한 관성과 마찰력을 보상합니다.

관성과 마찰력이 일정한 경우, 이 옵션을 켜게 되면 모터의 위치결정력을 향상시킵니다.

(3) Load Torque Compensation

구동부 부하에 의한 토크 변화를 감지하고 보상합니다.

부하 토크가 느리게 변하는 경우, 이 옵션을 켜게 되면 모터의 위치결정력을 향상시킵니다.

하지만 진동을 발생할 수 있으며, 모터가 진동할 경우 이 옵션을 끄거나 위치 제어기와 속도 제어기의 이득을 낮추어 진동을 줄일 수 있습니다.

(4) Use Position Controller for Velocity Control

속도 명령에 대해서는 속도 제어기만 동작하게 되는데, 저속에서 모터의 속도 추종 성능을 높이기 위해 위치 제어기를 사용할 수 있게 합니다. 이 때 제어기는 `position_demand` 값을 자동으로 생성하여 위치 제어기를 구동합니다.

BLDC모터를 Hallsensor 만으로 구동할 때나 PMSM 모터에서 해상도가 낮은 Encoder를 사용할 때 위치 제어기와 속도 제어기의 이득을 낮추고 이 옵션을 켜서 속도 제어 성능을 높일 수 있습니다.

(5) Maximum Torque per Ampere Control

단위 전류당 최대 토크가 되도록 d축과 q축 전류를 동시에 내보냅니다. MTPA는 주로 IPMSM 모터에서 사용하며 SPMSM 모터에서는 효과를 기대하기 어렵습니다. 그리고 DC나 BLDC 모터에서는 사용할 수 없습니다.

(6) Flux Weakening Control (STEP only)

약자속제어를 하여 STEP 모터의 최대 속도보다 더 높은 속도에서 운전할 수 있도록 합니다. 하지만 약자속제어로 고속 운전 중에는 모터에 흐르는 전류가 증가하여 모터의 발열이 심하며, 모터의 토크는 반비례하여 낮아집니다.

만일 24V에서 1000rpm으로 회전하는 STEP모터를 약자속제어하여 속도를 2000rpm으로 2배 높인 경우 모터의 토크는 1/2로 감소합니다.

9.21 Sensor Properties

9.21.1 position_sensor

모터의 위치를 파악하기 위해 사용하는 센서를 선택한다.

- 0 – (none)
- 1 – Encoder
- 2 – Encoder with Index
- 3 – Hallsensor
- 4 – Encoder & Hallsensor
- 5 – Encoder with Index & Hallsensor
-

각 모터별 사용 가능한 센서는 다음과 같다:

	DC	BLDC	PMSM	STEP
(none)	X	X	X	X
Encoder	O	X	O	O
Encoder w. Index	O	X	O	O
Hallsensor	X	O	X	X
Encoder & Hallsensor	-	O	O	-
Encoder w. Index & Hallsensor	-	O	O	-

※ O – 사용 가능, X – 사용 불가, - - 고려안함

9.21.2 encoder_direction

엔코더의 카운트 방향을 모터의 회전 방향과 일치하도록 설정합니다.

- 0 – Normal (기본값)
- 1 – Invert.

이 값이 올바르게 설정되었는지 확인하기 위해서, 모터를 전압 출력으로 구동합니다. 이때 모터의 회전 방향과 반대로 엔코더 값이 카운트 되는 경우, 현재의 encoder_direction 설정 값을 다른 값으로 변경해야 합니다. (0인 경우 1로, 1인 경우 0으로)

***주의: 모터의 회전 방향을 바꾸기 위해 엔코더의 카운트 방향 설정을 바꾸면 안됩니다. 이 값은 모터의 회전 방향과 엔코더의 카운트 방향이 불일치할 때 사용합니다. 즉, 모터에 양의 전압을 가**

했는데 엔코더가 거꾸로 카운트 되는 경우 이 값을 1-Invert로 설정합니다.

이 값이 모터의 회전 방향과 반대 방향일 경우 제어가 모터를 컨트롤 할 수 없는 상태가 됩니다.

9.21.3 encoder_resolution

엔코더의 한 회전당 펄스 카운트 수로 엔코더의 펄스 수를 4배한 값입니다. 즉, 모터 축을 한 바퀴 돌렸을 때 카운트 되는 엔코더 펄스 수가 됩니다.

이 값을 직접 찾기 위해서는, position_sensor를 '1 - Encoder'로 설정하고 모터 축을 손으로 한 바퀴 돌립니다. 이때 위치 카운트 값의 변화량이 encoder_resolution 값이 됩니다. 이 값은 4의 배수가 되어야 합니다. 한 바퀴로 값을 찾기 어려운 경우, 여러 바퀴 돌린 후 평균을 내면 어느정도 정확한 값을 얻을 수 있습니다.

일례로, 모터 축을 손으로 10바퀴 회전하였을 때 카운트 되는 위치 변화량이 1012라고 가정하겠습니다. 그러면 1회전당 101.2 펄스가 되고, 이 값은 4의 배수가 되어야 하기때문에 100으로 보면 됩니다.

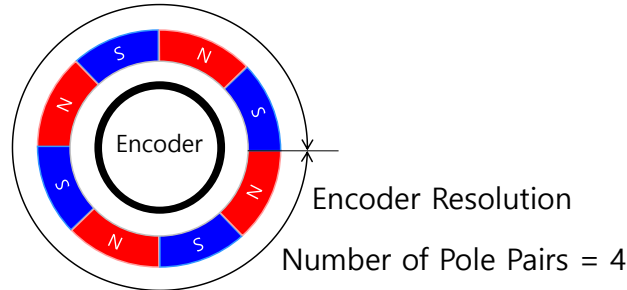
***주의: 이 값이 잘못 설정되면, 모터의 회전속도와 자동으로 탐지되는 모터의 각종 파라미터들이 잘못 계산됩니다. 데이터 시트를 참조하여 올바른 값을 입력해야합니다.**

***주의: DC 모터에 Index 신호가 없는 엔코더가 장착된 경우, system_control의 SC_POS_SENSOR_PARAM 명령으로 자동 탐지한 encoder_resolution은 어림 값입니다. 데이터 시트를 참조하여 올바른 값을 입력해야합니다.**

9.21.4 no_pole_pairs

홀센서의 극쌍 수(number of pole pairs)를 설정합니다. no_pole_pairs 값은 모터 축을 한 바퀴 돌렸을 때 카운트 되는 홀센서의 펄스 수를 6으로 나눈 값입니다.

일례로, 모터를 한 바퀴 돌렸을 홀센서의 카운트 값은 24가 됩니다. 이때 no_pole_pairs 값은 4가 됩니다 (24/6)



no_pole_pairs 값을 직접 찾기 위해서는, position_sensor를 '3 - Hallsensor'로 설정하고 모터 축을 손으로 한 바퀴 돌립니다. 이때 위치 카운트 값의 변화량을 Δ 라고 하면, Δ 는 6의 배수가 되어야 합니다. 만일 Δ 가 11, 12, 13 과 같다면 no_pole_pairs는 2라고 보면 됩니다.

9.21.5 hallsensor_phase

홀센서 입력이 들어오는 순서에 따른 홀센서 위상을 선택합니다.

- 0 - Normal,
- 1 - 60' Phase lead,
- 2 - 120' Phase lead,
- 3 - Opposite direction (Hall2 <--> Hall3),
- 4 - Opposite direction, 60' Phase lead,
- 5 - Opposite direction, 120' Phase lead,
- 6 - Inverted input,
- 7 - Inverted input, 60' Phase lead,
- 8 - Inverted input, 120' Phase lead,
- 9 - Inverted input, Opposite direction,
- 10 - Inverted input, Opposite direction, 60' Phase lead,
- 11 - Inverted input, Opposite direction, 120' Phase lead.

BLDC 또는 PMSM 모터에서 홀센서의 U, V, W 입력 신호가 들어오는 순서 및 위상에 따라 상기와 같이 12가지 조합이 있을 수 있습니다. 이 값을 수동으로 설정해야할 때는, position_sensor를 '3 - Hallsensor'로 설정하고 0 부터 11까지 hallsensor_phase 값을 바꾸어 가며 모터를 전압 출력으로 구동해 봅니다. 여기서 모터가 가장 매끄럽게 회전하는 경우의 값을 사용하면 됩니다.

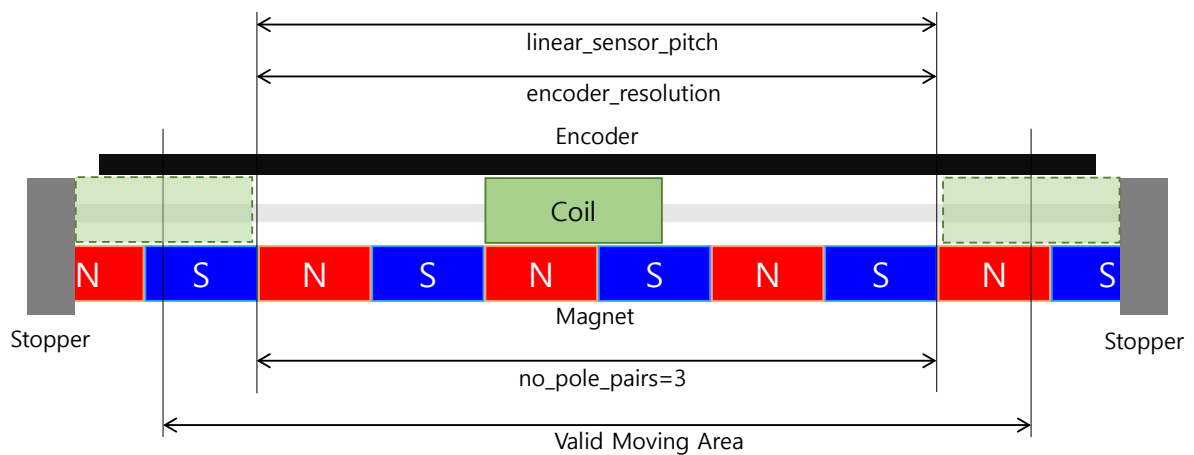
이 값을 직접 설정하기 보다는 system_control의 SC_HS_PHASE_DETECT 명령으로 자동 탐지할 것을 권장합니다.

9.21.6 linear_sensor_pitch

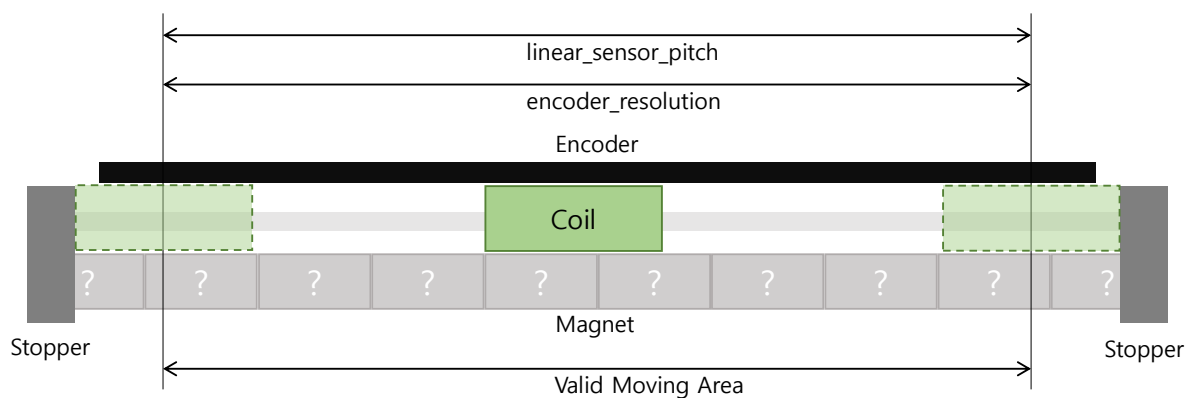
Linear Motor에서 사용되는 값으로, 모터가 encoder_resolution 만큼 이동하였을 때의 거리가 됩니다.

BLDC/PMSM/STEP Linear Motor에서 linear_sensor_pitch 오브젝트 값의 범위는 encoder_resolution 과 no_pole_pairs 값의 범위와 일치합니다. (다음 그림 참조)

Linear motor의 coil이 움직일 수 있는 Valid Moving Area 내에서 유효한 자석의 수를 센 것이 no_pole_pairs 값으로 그림에서는 3입니다. 그리고 유효한 자석의 총 길이가 linear_sensor_pitch가 되고, 이 길이만큼 이동하였을 때의 엔코더 카운트 값이 encoder_resolution이 됩니다.



하지만 DC Linear Motor에서는 제어기가 모터에 장착된 자석의 수를 알 수 없기 때문에, 다음 그림과 같이 Coil이 움직일 수 있는 유효한 영역의 길이가 linear_sensor_pitch가 되고, 이 길이만큼 이동하였을 때의 엔코더 카운트 값이 encoder_resolution이 됩니다.



쉽게 linear_sensor_pitch를 측정하는 방법은, 모터를 Disable 하여 free run 상태로 만들고 손으로

Coil을 encoder_resolution 값만큼 이동시키면서 이동거리를자로 재는 것입니다.

9.22 Electrical Parameters

9.22.1 bemf_constant

모터의 역기전력 상수는 모터에서 발생하는 역기전력을 전향보상하기 위해 Feed-forward 제어기에서 사용되며, 속도 제어기의 이득을 계산하는데 사용됩니다. 그리고 모터의 max_velocity, rated_velocity도 bemf_constant로부터 계산됩니다. 이 값은 Electric parameter estimation에서 계산됩니다.

※ Bemf_constant를 설정하면 torque_constant 값도 동일한 값으로 변경됩니다.

9.22.2 resistance

모터의 권선 저항이며, 권선 저항은 0.2A, 0.5A, 1A, 2A, 5A, 10A, 20A의 전류를 흘릴 때 측정된 7개의 값입니다. 각각의 전류에 대한 저항 값은 Sub-index 0에서 6사이의 resistance 오브젝트에 저장합니다.

모터의 권선 저항은 d축과 q축 전류 제어기의 이득을 계산하는데 사용됩니다. 이 값은 Electric parameter estimation에서 계산됩니다.

9.22.3 inductance_d/q

모터 권선의 d-축/q-축 인덕턴스는 d축과 q축 전류 제어기의 이득을 설정하는데 사용됩니다. 이 값은 Electric parameter estimation에서 계산됩니다.

사용자가 LRC미터로 모터의 권선으로부터 직접 값을 측정하여 입력할 수도 있지만, 가능하다면 제어기가 추정한 값을 사용할 것을 권장합니다.

9.22.4 electric_angle_bias

회전자의 전기각 바이어스 값입니다. PMSM 모터는 Hallsensor 신호가 바뀔 때 회전자의 전기각을 계산하게 되는데, 모터에 장착되는 Hallsensor의 위치나 기타 여러 조건들로 인해 회전자의 전기각이 어긋나 있는 경우가 있습니다. 이런 경우 electric_angle_bias를 조정하여 회전자의 전기각을 올바르게 계산할 수 있도록 합니다.

PMSM 모터와 STEP 모터에서는 구동 중 회전자의 전기각 바이어스 값을 실시간으로 추정하여 업

데이트 하게 된다. 하지만 `electric_angle_bias` 값은 변하지 않습니다.

이 값은 Electric parameter estimation에서 계산 됩니다. 사용자가 임의로 값을 수정하는 것을 권장하지 않습니다.

9.23 Mechanical Parameters

9.23.1 torque_constant

모터의 토크 상수는 부하 토크(`load_torque`)를 추정하는 Kalman Filter에서 사용되며, 속도 제어기의 이득을 계산하는데 사용됩니다. 모터의 토크 상수는 `bemf_constant`와 동일한 값으로 설정되며, Electric parameter estimation에서 계산됩니다.

`torque_constant` 값과 `bemf_constant` 값은 동일한 값으로 설정되는데, 모터 제어기에 설정된 값을 읽어보면 서로 다른 값으로 읽힙니다. 이는 두 값의 단위가 서로 다른 것이 원인이며, 두 값을 MKS 단위로 변환하면 동일한 값이 됩니다.

`bemf_constant`의 경우: $[mV/rpm] \rightarrow [V/(rad/s)]$

`torque_constant`의 경우: $[mNm/A] \rightarrow [Nm/A]$

9.23.2 moment_of_inertia

모터 회전자 및 기구부의 관성 모멘트는 부하 토크(`load_torque`)를 추정하는 Kalman Filter에서 사용되며, 속도 제어기의 이득을 계산하는데 사용됩니다. 또한 속도 프로파일의 가속도와 감속도를 결정하는 중요한 파라미터입니다. 이 값이 클수록 가속도와 감속도를 낮추어야 하고 작을 수록 가속도와 감속도를 높일 수 있습니다. 이 값은 Mechanic parameter estimation에서 계산됩니다.

9.23.3 viscous_friction

모터의 점성 마찰계수는 모멘트는 부하 토크(`load_torque`)를 추정하는 Kalman Filter에서 사용되며, 속도 제어기의 이득을 계산하는데 사용됩니다. 이 값은 Mechanic parameter estimation에서 계산 됩니다.

9.23.4 coulomb_friction

모터의 정지 마찰계수는 부하 토크(`load_torque`)를 추정하는 Kalman Filter에서 사용됩니다. 이 값은 Mechanic parameter estimation에서 계산됩니다.

9.23.5 load_torque_bias

모터의 부하 토크의 편향을 표시합니다. 이 값은 Mechanic parameter estimation에서 계산되어지거나 실제 제어기에서 전향보상에 사용할 때는 Kalman filter에서 실시간으로 추정된 값이 사용됩니다.

9.24 Controller's Gain

9.24.1 position_p_gain

위치 제어기의 비례 이득으로, 이 값을 높이면 목표 위치를 더 빠르게 추종하게 됩니다. 하지만 과도하게 높은 경우 모터에 진동이 발생하게 됩니다.

이 값을 직접 조정하기 보다는 system_control의 SC_PV_GAIN_STIFFNESS 명령으로 조정할 것을 권장합니다.

9.24.2 velocity_p_gain

속도 제어기의 비례 이득으로, 이 값을 높이면 목표 속도를 더 빠르게 추종하게 됩니다. 하지만 과도하게 높은 경우 모터에 진동이 발생하게 됩니다.

이 값을 직접 조정하기 보다는 system_control의 SC_PV_GAIN_STIFFNESS 명령으로 조정할 것을 권장합니다.

***주의: SC_PV_GAIN_STIFFNESS 명령으로 속도 제어기의 이득을 설정하는 경우 모터의 올바른 Mechanic Parameter를 미리 입력해 두어야 합니다.**

9.24.3 velocity_i_gain

속도 제어기의 적분 이득으로, 이 값을 높이면 정상상태 오차를 빠르게 줄일 수 있습니다. 하지만 과도하게 높은 경우 모터에 저주파 진동이 발생하게 됩니다.

이 값을 직접 조정하기 보다는 system_control의 SC_PV_GAIN_STIFFNESS 명령으로 조정할 것을 권장합니다.

9.24.4 velocity_d_gain

속도 제어기의 미분 이득으로, 이 값을 높이면 오버슈트를 감소할 수 있습니다. 하지만 Encoder의

해상도가 낮을 때 이 값을 과도하게 높인 경우 모터에 고주파 진동이 발생하게 됩니다.

***주의: 이 값은 system_control의 SC_PV_GAIN_STIFFNESS 명령으로 설정되지 않으며, 필요에 따라 사용자가 직접 입력하여야 합니다.**

9.24.5 current_p_gain_d/q

d/q축 전류 제어기의 비례 이득으로, 이 값을 높이면 목표 전류를 더 빠르게 추종하게 됩니다. 하지만 과도하게 높인 경우 모터에 소음이 발생하게 됩니다.

이 값을 직접 조정하기 보다는 system_control의 SC_CC_GAIN_STIFFNESS 명령으로 조정할 것을 권장합니다.

***주의: SC_CC_GAIN_STIFFNESS 명령으로 전류 제어기의 이득을 설정하기 전에 모터의 올바른 Electric Parameter를 미리 설정해야 합니다.**

9.24.6 current_i_gain_d/q

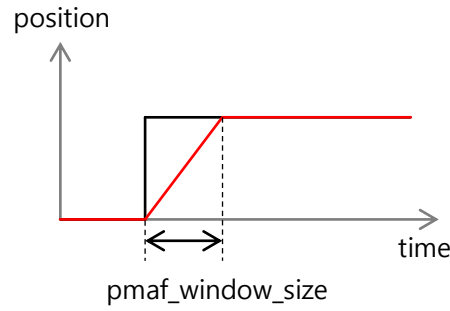
d/q축 전류 제어기의 적분 이득으로, 이 값을 높이면 정상상태 오차를 빠르게 줄일 수 있습니다. 하지만 과도하게 높인 경우 모터에 소음과 진동이 발생하게 됩니다.

이 값을 직접 조정하기 보다는 system_control의 SC_CC_GAIN_STIFFNESS 명령으로 조정할 것을 권장합니다.

9.25 Moving Average Filter

9.25.1 pmaf_window_size

위치제어기 입력단에 배치된 Moving Average 필터의 윈도우 사이즈를 설정합니다. 윈도우 사이즈는 1 ~ 256 사이의 값으로 설정합니다. 윈도우 사이즈가 1로 설정된 경우는 필터가 동작하지 않습니다.

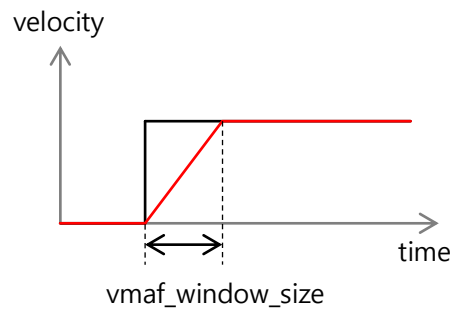


$$\text{position_demand} = \frac{1}{n} \sum_{i=0}^{n-1} \text{target_position}[M-i]$$

시간 축에서 검은색 그래프가 필터를 통과하기 전의 위치 값이고, 붉은색 그래프가 필터를 통과한 후의 위치 값입니다. 급격한 위치 이동 명령이 Moving Average 필터를 통과함으로 부드럽게 변하는 위치 이동 명령으로 바뀌게 됩니다.

9.25.2 vmaf_window_size

속도제어기 입력단에 배치된 Moving Average 필터의 윈도우 사이즈를 설정합니다. 윈도우 사이즈는 1 ~ 256 사이의 값으로 설정합니다. 윈도우 사이즈가 1로 설정된 경우는 필터가 동작하지 않습니다.



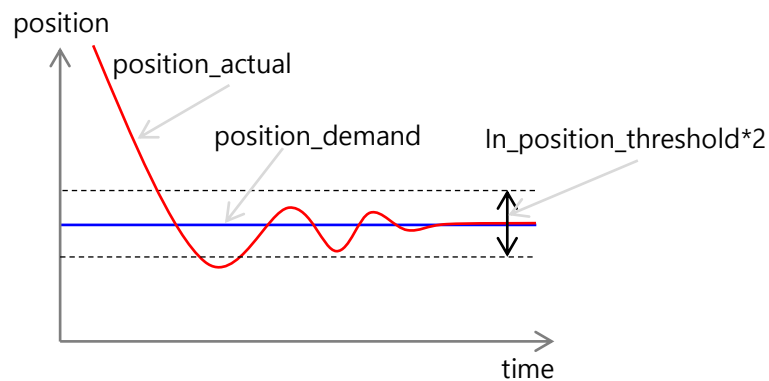
$$\text{velocity_demand} = \frac{1}{n} \sum_{i=0}^{n-1} \text{target_velocity}[M-i]$$

시간 축에서 검은색 그래프가 필터를 통과하기 전의 속도 값이고, 붉은색 그래프가 필터를 통과한 후의 속도 값입니다. 급격한 속도 명령이 Moving Average 필터를 통과함으로 부드럽게 변하는 속도 명령으로 바뀌게 됩니다.

9.26 I/O Parameters

9.26.1 in_position_threshold

목표 위치 도달을 판단하기 위한 threshold 값을 설정합니다. $\text{position_demand} - \text{position_actual}$ 값이 $\text{in_position_threshold}$ 이하로 1ms 이상 유지되면 모터가 목표 위치에 도달하였다고 판단합니다.

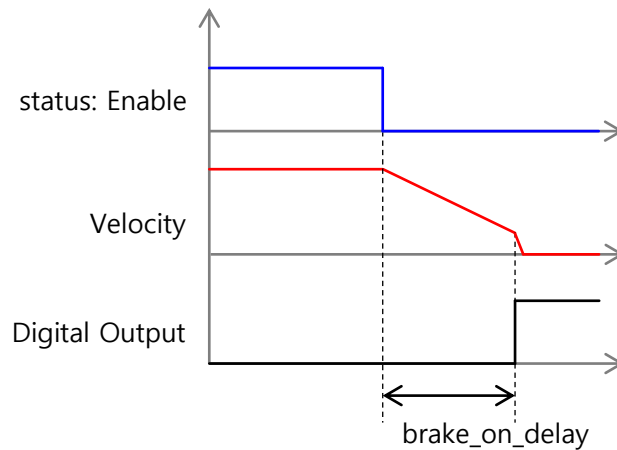


9.26.2 in_velocity_threshold

목표 속도 도달을 판단하기 위한 threshold 값을 설정합니다. $\text{velocity_demand} - \text{velocity_actual}$ 값이 $\text{in_velocity_threshold}$ 이하로 1ms 이상 유지되면 모터가 목표 속도에 도달하였다고 판단합니다.

9.26.3 brake_on_delay

모터가 Disable 상태로 되었을 때, 모터에 장착된 브레이크를 작동하는데 필요한 시간 지연을 설정합니다. 만일 모터가 정지한 상태라면 브레이크는 Disable 되었을 때 brake_on_delay 시간 설정에 관계 없이 즉시 작동합니다. 하지만 회전하고 있다면 회전이 멈추거나 brake_on_delay 시간이 지날 때까지 브레이크 작동을 대기합니다.



보통 모터가 Disable 상태가 되면 일정 시간 후(보통 모터가 멈출 때까지 기다렸다가) 브레이크를 작동하여 모터를 고정함으로 수직 부하의 낙하를 방지하게 됩니다.

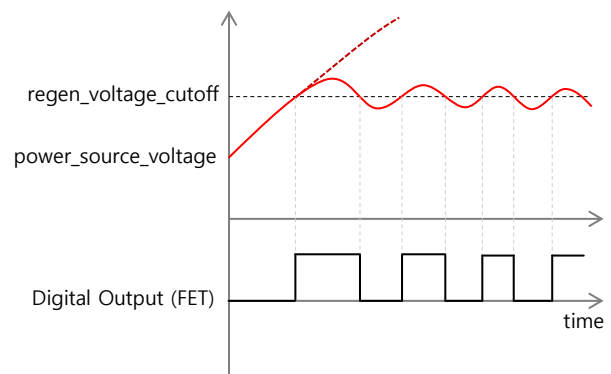
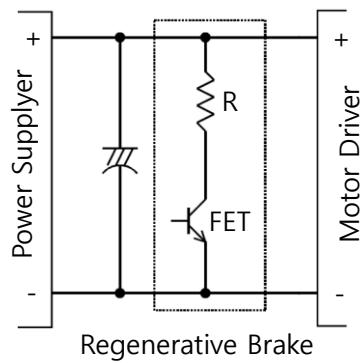
9.26.4 jog_velocity

Digital Input에 연결된 Jog+ 스위치나 Jog- 스위치가 작동할 경우, jog_velocity에 지정된 속도로 모터를 구동하게 됩니다.

9.26.5 regen_voltage_cutoff

전원단의 전압이 regen_voltage_cutoff 설정 값 이상으로 올라가면 Digital Output 출력으로 구동 되는FET나 릴레이를 통해 연결된 저항으로 전원단을 단락 시켜 회생 에너지를 저항으로 소모하게 됩니다.

회생 브레이크: 관성이 큰 부하가 장착된 모터를 감속하거나 정지한 모터를 강제로 회전시킬 경우, 모터는 발전기 상태가 되어 전기 에너지를 전원단으로 되돌리게 됩니다. 충방전이 가능한 배터리와 같이 되돌아온 에너지를 수용할 수 있는 경우는 문제가 없지만, 파워 서플라이와 같이 되돌아온 에너지를 다시 수용하지 못할 경우 전원단의 전압이 상승하여 제어기를 파손시키게 됩니다. 이것을 방지하기 위해서 다음 그림과 같이 전원단의 +, - 단자 사이에 저항과 FET를 직렬로 연결하여, 전원 전압이 일정 값이 상으로 상승할 경우 FET를 켜 저항으로 전류를 흘림으로 회생 에너지를 열로 소비시킵니다.



9.27 I/O Common

9.27.1 no_digital_input

제어기가 가진 디지털 입력 채널의 수를 나타냅니다. 디지털 입력 채널의 수는 제어기 모델에 따라 다릅니다.

9.27.2 no_digital_output

제어기가 가진 디지털 출력 채널의 수를 나타냅니다. 디지털 출력 채널의 수는 제어기 모델에 따라 다릅니다.

9.27.3 no_analog_input

제어기가 가진 아날로그 입력 채널의 수를 나타냅니다. 디지털 입력 채널의 수는 제어기 모델에 따라 다릅니다.

9.27.4 no_pulse_input

제어기가 가진 펄스 입력 채널의 수를 나타냅니다. 디지털 입력 채널의 수는 제어기 모델에 따라 다릅니다.

9.27.5 digital_inputs

제어기에서 지원하는 모든 디지털 입력을 32bit 정수로 모은 값입니다.

- bit 0 – Digital Input Channel 1
- bit 1 – Digital Input Channel 2

- bit 2 – Digital Input Channel 3
- bit 3 - ...

9.27.6 digital_outputs

제어기에서 지원하는 모든 디지털 출력을 32bit 정수로 모은 값입니다.

- bit 0 – Digital Output Channel 1
- bit 1 – Digital Output Channel 2
- bit 2 – Digital Output Channel 3
- bit 3 - ...

9.27.7 digital_outputs_mask

디지털 출력에 사용자가 지정한 값을 내보낼 때, 내보내는 채널을 선택합니다.

ex)

```
digital_outputs_mask=0x07 ㄹ // 마지막 3개 채널에 대해 mask 지정  
digital_outputs=0x05 ㄹ // 1번과 3번 채널은 켜고 2번 채널은 끄
```

9.28 Digital Input

9.28.1 di_option

해당 디지털 입력 채널의 사용여부와 반전 여부를 결정합니다.

- bit 0 - 디지털 입력 채널의 사용 여부 (0 – Disable, 1 – Enable)
- bit 1 - 디지털 입력 채널의 반전 여부 (0 – Normal, 1 – Invert)

9.28.2 di_value

해당 디지털 입력 채널의 입력 값을 나타낸다. 값으로는 1bit의 0이나 1을 가집니다.

9.28.3 di_function

해당 디지털 입력 채널을 제어기 기능으로 연결합니다. 연결 가능한 기능은 다음과 같습니다.

- 0 - (none)
- 1 - Disable (Emergency Stop)
- 2 - Enable
- 3 - Clear Fault
- 4 - Stop
- 5 - Quick Stop
- 6 - Forward Limit Switch
- 7 - Reverse Limit Switch
- 8 - Home Switch
- 9 - Reset Position
- 10- Jog +
- 11- Jog -
- 12- Script Run
- 13- Script: interrupt0
- 14- Script: interrupt1
- 15- Script: interrupt2
- 16- Script: interrupt3
- 17- Script: interrupt4
- 18- Script: interrupt5
- 19- Script: interrupt6
- 20- Script: interrupt7

9.29 Digital Output

9.29.1 do_option

해당 디지털 출력 채널의 사용여부와 반전 여부를 결정합니다.

- bit 0 - 디지털 출력 채널의 사용 여부 (0 - Disable, 1 - Enable)
- bit 1 - 디지털 출력 채널의 반전 여부 (0 - Normal, 1 - Invert)

9.29.2 do_value

해당 디지털 출력 채널의 출력 값을 나타냅니다. 값으로는 1bit의 0이나 1을 가집니다.

디지털 출력 채널의 do_function으로 상태가 연결되지 않은 경우, 사용자가 직접 do_value에 값을

섬으로 디지털 출력 채널로 원하는 값을 내보낼 수 있습니다.

9.29.3 do_function

해당 디지털 출력 채널을 제어기 상태로 연결합니다. 연결 가능한 상태는 다음과 같습니다.

- 0 - (none)
- 1 - Enable
- 2 - Fault
- 3 - Moving
- 4 - Homing
- 5 - Home Complete
- 6 - Position Limit
- 7 - Current Limit
- 8 - * Regenerative Voltage Cutoff
- 9 - * Motor Brake Release
- 10- In-position
- 11- In-Velocity
- 12- Ready
- 13- Script Run

9.30 Analog Input

9.30.1 ai_option

해당 아날로그 입력 채널의 사용여부와 반전 여부를 결정합니다.

- bit 0 - 아날로그 입력 채널의 사용 여부 (0 - Disable, 1 - Enable)
- bit 1 - 아날로그 입력 채널의 반전 여부 (0 - Normal, 1 - Invert)

bit1의 값이 1인 경우, 아날로그 입력 값이 변환 과정을 거쳐 정규화 된 ai_value 값의 극성이 반전됩니다. 즉, -1은 1로 반전되고 1은 -1로 반전된다. 0의 값은 그대로 유지됩니다.

9.30.2 ai_value

ai_value 값은 아날로그 입력 채널에서 읽은 원시 값(ai_raw_value)을 변환 과정을 거쳐 -1과 1 사이의 정규화 된 값으로 나타냅니다.

9.30.3 ai_function

해당 디지털 입력 채널을 제어기 구동 명령으로 연결합니다. 연결 가능한 구동 명령은 다음과 같습니다.

- 0 - (none)
- 1 - Target Voltage
- 2 - Target Current
- 3 - Target Velocity
- 4 - Target Position
- 5 - Torque Limit

9.30.4 ai_raw_value

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

ai_raw_value은 아날로그 입력 채널에서 읽은 값으로, 변환이 이루어지기 전의 0과 4095 사이의 12bits 디지털 값입니다.

9.30.5 ai_cutoff_freq

정규화 된 아날로그 입력 채널의 값을 1차 로우 패스 필터를 통과시키는데, ai_cutoff_freq는 1차 로우 패스 필터의 차단 주파수입니다.

9.30.6 ai_cal_min, ai_cal_center, ai_cal_max

ai_cal_min, ai_cal_center, ai_cal_max는 아날로그 입력 값을 정규화하는데 사용되는 캘리브레이션 파라미터 들입니다.

ai_cal_min은 아날로그 입력의 최소값을 나타낸다. 이 값이 정규화되면 -1이 됩니다.

ai_cal_center은 아날로그 입력의 중앙값을 나타낸다. 이 값이 정규화되면 0이 됩니다.

ai_cal_max은 아날로그 입력의 최대값을 나타낸다. 이 값이 정규화되면 1이 됩니다.

ai_cal_min 보다 ai_cal_center 값이 같거나 커야 하고, ai_cal_center 보다 ai_cal_max 값이 같거나 커야 합니다.

9.30.7 ai_cal_deadband

ai_cal_deadband는 아날로그 입력의 중앙값에서 0으로 인식하는 입력 범위를 나타냅니다.

9.31 Pulse Input

9.31.1 pi_option

해당 펄스 입력 채널의 사용여부와 반전 여부를 결정합니다.

- bit 0 - 펄스 입력 채널의 사용 여부 (0 – Disable, 1 – Enable)
- bit 1 - 펄스 입력 채널의 반전 여부 (0 – Normal, 1 – Invert)

bit1의 값이 1인 경우, 펄스 입력 값이 변환 과정을 거쳐 정규화 된 pi_value 값의 극성이 반전됩니다. 즉, -1은 1로 반전되고 1은 -1로 반전됩니다. 0의 값은 그대로 유지됩니다.

9.31.2 pi_value

pi_value 값은 펄스 입력 채널에서 읽은 원시 값(pi_raw_value)을 변환 과정을 거쳐 -1과 1 사이의 정규화 된 값으로 나타냅니다.

9.31.3 pi_function

해당 펄스 입력 채널을 제어기 구동 명령으로 연결합니다. 연결 가능한 구동 명령은 다음과 같습니다.

- 0 - (none)
- 1 - Target Voltage
- 2 - Target Current
- 3 - Target Velocity
- 4 - Target Position
- 5 - Torque Limit

9.31.4 pi_raw_value

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

pi_raw_value 값은 설정된 pi_capture_type에 따라 Pulse Width, Frequency, Duty Cycle 중 하나의 값을 캡처 한 원시 값입니다.

9.31.5 pi_capture_type

pi_capture_type은 펄스 입력의 데이터 수집 형식(Capture Type)을 나타냅니다. 이 오브젝트 값으로 다음 중 하나를 선택합니다.

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

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

9.31.6 pi_cutoff_freq

정규화 된 펄스 입력 채널의 값을 1차 로우 패스 필터를 통과시키는데, pi_cutoff_freq는 1차 로우 패스 필터의 차단 주파수입니다.

9.31.7 pi_cal_min, pi_cal_center, pi_cal_max

pi_cal_min, pi_cal_center, pi_cal_max는 펄스 입력 값을 정규화하는데 사용되는 캘리브레이션 파라미터 들입니다.

pi_cal_min은 펄스 입력의 최소값을 나타냅니다. 이 값이 정규화되면 -1이 됩니다.

pi_cal_center은 펄스 입력의 중앙값을 나타냅니다. 이 값이 정규화되면 0이 됩니다.

pi_cal_max은 펄스 입력의 최대값을 나타냅니다. 이 값이 정규화되면 1이 됩니다.

pi_cal_min 보다 pi_cal_center 값이 같거나 커야 하고, pi_cal_center 보다 pi_cal_max 값이 같거나 커야 합니다.

9.31.8 pi_cal_deadband

pi_cal_deadband는 펄스 입력의 중앙값에서 0으로 인식하는 입력 범위를 나타냅니다.

10 Mini-C 스크립트 언어

Mini-C 스크립트 언어는 제어기 운용을 프로그래밍 할 수 있도록 C언어의 서브셋으로 만들어진 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 쉽게 사용 가능합니다.

10.1 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.

10.2 C언어와의 차이

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

- `#ifdef`, `#if` 등의 전처리 문
- `int`, `float`, `long` 등 자료형 키워드
- 문자열 상수 (Ex: `"abc"`, `"ABC"`)와 문자 상수 (Ex: `'a'`, `'A'`)
- 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 `a`를 선언하면서 값 할당
- `b=4.567` - 실수형 변수 `b`를 선언하면서 값 할당

1차원 배열을 지원하며, 'array'라는 키워드를 사용하여 다음과 같이 선언합니다:

- `array ar[5]` - 5개의 원소를 가지는 배열을 만들고 모두 0으로 초기화
- `array br[5] = {1, 2, 3 }` - 배열을 1, 2, 3 으로 초기화, 나머지는 0으로 초기화

함수를 선언할 때는 'function' 키워드를 사용합니다:

- `function fu(a, b) { return a + b; }`

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
- `return` - 호출된 함수로부터 빠져나감

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, function, array, return`

Mini-C 스크립트 언어에서는 다음 상수들이 정의되어 있습니다:

- `M_E, M_PI, VI_VENDOR_ID, VI_PRODUCT_ID, ...`

10.3 문장

스크립트로 작성된 프로그램은 문장(Statement)들로 구성됩니다. 문장은 프로그램을 실행하기 위한 기본적인 실행 단위라고 볼 수 있습니다.

10.3.1 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;
```

10.3.2 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;
}
```

10.3.3 제어문

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

제어문에 대해서는 "10.8 제어문"에서 상세하게 설명합니다.

10.4 주석

스크립트 언어에서 주석문(Comment)은 `'/*'` 와 `'*/'`사이 또는 `'//'`뒤에 기술하며, 컴파일 대상에

서 제외됩니다. 주석문은 주로 프로그램의 이해를 증진시키기 위해 사용됩니다. 스크립트 프로그램 내에 주석문을 나타내기 위해, C언어에서 사용하는 것과 동일한 두 가지 방법을 제공합니다.

10.4.1 블록 주석문

블록 주석은 `/*`와 `*/`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
    ...
*/
```

10.4.2 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
```

10.5 상수

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

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

10.5.1 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 옵니다.
- 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 옵니다.

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.

10.5.2 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.

10.5.3 미리 정의된 상수

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

표 10-1 정의된 수학 상수

상수	값	설명
M_E	2.7182818284590452354	자연상수

M_PI	3.14159265358979323846	원주율
M_EULER	0.57721566490153286061	오일러-마스케로니 상수

수학 상수는 다음과 같이 사용할 수 있습니다.

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

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

```
#define VI_VENDOR_ID                0x2001
#define VI_PRODUCT_ID               0x2002
#define VI_SOFTWARE_VERSION         0x2003
#define VI_HARDWARE_VERSION         0x2004

#define VI_SYSTEM_STATUS             0x2005
#define      SS_HS_PHASE_DETECT      0x0001
#define      SS_POS_SENSOR_PARAM     0x0002
#define      SS_ELECTRIC_PARAM_EST   0x0003
#define      SS_MECHANIC_PARAM_EST   0x0004
#define      SS_COMMAND_ID_MASK      0x000F
#define      SS_COMMAND_RESULT_MASK  0x00F0
#define      SS_COMMAND_SUCCESS      0x0010
#define      SS_COMMAND_FAILED       0x0020
#define      SS_COMMAND_ABORTED      0x0040
#define      SS_AUTO_GAIN_TUN        0x4000
#define      SS_AUTO_VIBR_SUP        0x8000
#define      SS_SCRIPT_DEBUG         0x08000000
#define      SS_SCRIPT_RUN           0x10000000
#define      SS_SCRIPT_PAUSE         0x20000000
#define      SS_SCRIPT_STEP          0x40000000
#define      SS_SCRIPT_STOP          0x80000000

#define VI_SYSTEM_CONTROL            0x2006
#define      SC_SCRIPT_RUN            1
#define      SC_SCRIPT_RUN_DEBUG      2
#define      SC_SCRIPT_PAUSE          3
#define      SC_SCRIPT_STEP           4
#define      SC_SCRIPT_STOP           5
#define      SC_COMMAND_ABORT         10
#define      SC_HS_PHASE_DETECT       11
#define      SC_POS_SENSOR_PARAM      12
#define      SC_ELECTRIC_PARAM_EST    13
#define      SC_MECHANIC_PARAM_EST    14
#define      SC_AUTO_OP_STOP          20
#define      SC_AUTO_GAIN_TUN         21
#define      SC_AUTO_VIBR_SUP         22
#define      SC_CC_GAIN_STIFFNESS     41
#define      SC_PV_GAIN_STIFFNESS     42
#define      SC_FWC_SPEED_UP          43
#define      SC_CC_GAIN_STIF_READ     81
#define      SC_PV_GAIN_STIF_READ     82
#define      SC_FWC_SPEED_UP_READ     83
#define      SC_CLEAR_ERROR_CODE      90
#define      SC_FACTORY_DEFAULT       98
```

```
#define SC_SYSTEM_RESET 99
#define SC_FIRMWARE_DOWNLOAD 100

#define VI_SYSTEM_ARGUMENT 0x2007
#define VF_POWER_SOURCE_VOLTAGE 0x2008
#define VF_POWER_SOURCE_CURRENT 0x2009

#define VS_DEVICE_NAME 0x2010
#define VI_DEVICE_ID 0x2011
#define VI_SERIAL_WATCHDOG 0x2012
#define VI_SERIAL_BAUDRATE 0x2014

#define VI_UNITS 0x201A
#define VI_STARTUP_DRIVE_INPUT 0x201B
#define VI_STARTUP_CONTROL_MODE 0x201C
#define VI_STARTUP_ENABLE_DELAY 0x201D
#define VI_SCRIPT_STOP_ACTION 0x201E
#define VI_DISABLE_OPER_ACTION 0x201F

#define VI_PULSE_INPUT_MODE 0x2041
#define VI_PULSE_COUNT_EDGE 0x2042
#define VI_DIR_SIGNAL_POLARITY 0x2043

#define VI_GEAR_NUMERATOR 0x2048
#define VI_GEAR_DENOMINATOR 0x2049

#define VI_SCRIPT_ADDRESS 0x2050
#define VI_SCRIPT_SIZE 0x2051
#define VI_SCRIPT_CODE 0x2052
#define VF_SCRIPT_VARIABLE 0x2055
#define VI_SCRIPT_BREAKPOINT 0x2056
#define VI_SCRIPT_DEBUG_LINE 0x2057

#define VI_USER_VARIABLE 0x2060
#define VI_TEMP_VARIABLE 0x2061

#define VI_ERROR_CODE 0x2080
#define VI_BUFFER_A 0x2091
#define VI_BUFFER_B 0x2092
#define VI_BUFFER_C 0x2093

#define VI_CONTROL 0x2101
#define MC_DISABLE 0
#define MC_ENABLE 1
#define MC_CLEAR_FAULT 2
#define MC_RESET_POSITION 3
#define MC_STOP 6
#define MC_QUICK_STOP 7
#define MC_HOME_SEARCH 9
#define MC_JOG_POSITIVE 10
#define MC_JOG_NEGATIVE 11
#define MC_ANALOG_DRIVE 100
#define MC_PULSE_DRIVE 101
#define MC_SERIAL_DRIVE 102

#define VI_STATUS 0x2102
#define ST_ENABLE 0x0001
#define ST_FAULT 0x0002
#define ST_MOVING 0x0004
#define ST_HOMING 0x0008
```

```
#define ST_HOME_COMP 0x0010
#define ST_POSITION_CTL 0x0020
#define ST_VELOCITY_CTL 0x0040
#define ST_CURRENT_CTL 0x0080
#define ST_POS_LIMIT 0x0100
#define ST_CURRENT_LIM 0x0200
#define ST_IN_POSITION 0x0400
#define ST_IN_VELOCITY 0x0800
#define ST_DI_STOP 0x1000
#define ST_DI_DISABLE 0x2000
#define ST_QUICK_STOP 0x4000
#define ST_ANALOG_DRIVE 0x00020000
#define ST_PULSE_DRIVE 0x00040000
#define ST_SERIAL_DRIVE 0x00080000
#define ST_READY 0x20000000
#define ST_DC_STEP 0x40000000
#define ST_VM_RUN 0x80000000

#define VI_FAULT 0x2103
#define FF_OVER_CURRENT 0x0001
#define FF_OVER_VOLTAGE 0x0002
#define FF_UNDER_VOLTAGE 0x0004
#define FF_OVER_HEAT 0x0008
#define FF_SHORT_CIRCUIT 0x0010
#define FF_VIBRATION 0x0020
#define FF_STALL_CURRENT 0x0040
#define FF_POSITION_TRACKING 0x0080
#define FF_VELOCITY_TRACKING 0x0100
#define FF_HALLSENSOR 0x0200
#define FF_ENCODER 0x0400
#define FF_MOTOR 0x0800
#define FF_OVER_SPEED 0x1000
#define FF_POSITIVE_LIMIT 0x2000
#define FF_NEGATIVE_LIMIT 0x4000
#define FF_POWER 0x8000
#define FF_EMERGENCY_SW 0x00010000
#define FF_ELECTRIC_PARAM 0x00020000
#define FF_MECHANIC_PARAM 0x00040000

#define VF_TEMPERATURE 0x2105
#define VF_LOAD_TORQUE 0x2106

#define VI_TARGET_POSITION 0x2111
#define VI_TARGET_VELOCITY 0x2112
#define VF_TARGET_CURRENT_D 0x2113
#define VF_TARGET_CURRENT_Q 0x2114
#define VF_TARGET_VOLTAGE_D 0x2115
#define VF_TARGET_VOLTAGE_Q 0x2116

#define VF_TORQUE_LIMIT 0x2119

#define VI_POSITION_DEMAND 0x2121
#define VI_VELOCITY_DEMAND 0x2122
#define VF_CURRENT_DEMAND_D 0x2123
#define VF_CURRENT_DEMAND_Q 0x2124
#define VF_VOLTAGE_DEMAND_D 0x2125
#define VF_VOLTAGE_DEMAND_Q 0x2126
#define VF_ACCELERATION_DEMAND 0x2127

#define VI_POSITION_ACTUAL 0x2131
```

```
#define VI_VELOCITY_ACTUAL          0x2132
#define VF_CURRENT_ACTUAL_D         0x2133
#define VF_CURRENT_ACTUAL_Q         0x2134
#define VF_ACCELERATION_ACTUAL      0x2135

#define VF_POSITION_ERROR           0x2141
#define VF_VELOCITY_ERROR           0x2142
#define VF_CURRENT_ERROR_D          0x2143
#define VF_CURRENT_ERROR_Q          0x2144

#define VI_SOFT_LIMIT_CHECK         0x2151
#define VI_MIN_POSITION             0x2152
#define VI_MAX_POSITION             0x2153
#define VI_LIMIT_ACTION             0x2154

#define VI_HOMING_METHOD            0x2158
#define VI_HOMING_VELOCITY          0x2159
#define VI_HOME_OFFSET              0x215A
#define VI_HOME_POSITION            0x215B
#define HS_CURRENT_POS              0
#define HS_HOME_POSITIVE             1
#define HS_HOME_NEGATIVE            2
#define HS_FORWARD_LIMIT            3
#define HS_REVERSE_LIMIT            4
#define HS_FORWARD_STALL            5
#define HS_REVERSE_STALL            6
#define HS_HOME_POSITIVE_INDEX      7
#define HS_HOME_NEGATIVE_INDEX      8
#define HS_FORWARD_LIMIT_INDEX      9
#define HS_REVERSE_LIMIT_INDEX     10
#define HS_FORWARD_STALL_INDEX     11
#define HS_REVERSE_STALL_INDEX     12
#define HS_INDEX_POSITIVE           13
#define HS_INDEX_NEGATIVE           14

#define VF_RATED_VOLTAGE            0x2161
#define VF_RATED_CURRENT            0x2162
#define VI_RATED_VELOCITY           0x2163
#define VF_MAX_CURRENT              0x2164
#define VI_MAX_VELOCITY             0x2165
#define VI_PROFILE_TYPE             0x2167
#define VI_PROFILE_VELOCITY         0x2168
#define VI_PROFILE_ACCEL            0x2169
#define VI_PROFILE_DECEL            0x216A
#define VI_PROFILE_JERK             0x216B

#define VF_OVERHEAT_LIMIT           0x2171
#define VF_OVERCURRENT_LIMIT        0x2172
#define VF_OVERVOLTAGE_LIMIT        0x2173
#define VF_UNDERVOLTAGE_LIMIT       0x2174
#define VI_VIBRATION_DET            0x2176
#define VI_STALL_CURRENT_DET         0x2177
#define VI_POSITION_TRACK_ERROR     0x2178
#define VI_VELOCITY_TRACK_ERROR     0x2179

#define VI_MOTOR_TYPE               0x2181
#define MT_DC_ROTARY                 0
#define MT_DC_LINEAR                 1
#define MT_BLDC_ROTARY               2
#define MT_BLDC_LINEAR               3
```

```
#define MT_PMSM_ROTARY 4
#define MT_PMSM_LINEAR 5
#define MT_STEP3_ROTARY 6
#define MT_STEP3_LINEAR 7
#define MT_STEP2_ROTARY 8
#define MT_STEP2_LINEAR 9
#define NO_MOTOR_TYPE 10
#define VI_MOTOR_DIRECTION 0x2182
#define VI_FEEDFORWARD_OPTIONS 0x2183
#define FF_BACK_EMF_COMP 0x01
#define FF_INERTIA_COMP 0x02
#define FF_LOAD_TORQUE_COMP 0x04
#define FF_POS_CTL_FOR_VEL 0x08
#define FF_MAXIMUM_TORQUE 0x10
#define FF_FLUX_WEAKENING 0x20

#define VI_POSITION_SENSOR 0x2188
#define PS_NONE 0
#define PS_ENCODER 1
#define PS_ENCODER_WI 2
#define PS_HALLSENSOR 3
#define PS_ENCODER_HALLSENSOR 4
#define PS_ENCODER_WI_HALLSENSOR 5

#define VI_ENCODER_DIRECTION 0x2189
#define VI_ENCODER_RESOLUTION 0x218A
#define VI_NO_POLE_PAIRS 0x218B
#define VI_HALLSENSOR_PHASE 0x218C
#define VI_LINEAR_SENSOR_PITCH 0x218D

#define VF_BEMF_CONSTANT 0x2191
#define VF_RESISTANCE 0x2192
#define VF_INDUCTANCE_D 0x2193
#define VF_INDUCTANCE_Q 0x2194
#define VI_ELECTRIC_ANGLE_BIAS 0x2195
#define VF_TORQUE_CONSTANT 0x2198
#define VF_MOMENT_OF_INERTIA 0x2199
#define VF_VISCOUS_FRICTION 0x219A
#define VF_COULOMB_FRICTION 0x219B
#define VF_LOAD_TORQUE_BIAS 0x219C

#define VF_POSITION_P_GAIN 0x21A1
#define VF_VELOCITY_P_GAIN 0x21A4
#define VF_VELOCITY_I_GAIN 0x21A5
#define VF_VELOCITY_D_GAIN 0x21A6
#define VF_CURRENT_P_GAIN_D 0x21A7
#define VF_CURRENT_I_GAIN_D 0x21A8
#define VF_CURRENT_P_GAIN_Q 0x21A9
#define VF_CURRENT_I_GAIN_Q 0x21AA

#define VI_PMAF_WINDOW_SIZE 0x21B1
#define VI_SMAF_WINDOW_SIZE 0x21B2

#define VI_IN_POSITION_TH 0x21D1
#define VI_IN_VELOCITY_TH 0x21D2
#define VI_BRAKE_ON_DELAY 0x21D3
#define VF_JOG_VELOCITY 0x21D4
#define VF_REGEN_VOLTAGE_CUTOFF 0x21D5

#define VI_NO_DI 0x2201
```

```
#define VI_NO_DO 0x2202
#define VI_NO_AI 0x2203
#define VI_NO_PI 0x2204
#define VI_DIGITAL_INPUTS 0x2210
#define VI_DIGITAL_OUTPUTS 0x2220
#define VI_DIGITAL_OUTPUTS_MASK 0x2221

#define VI_DI_OPTIONS 0x2230
#define IO_ENABLE 0x01
#define IO_INVERT 0x02
#define VI_DI_VALUE 0x2231
#define VI_DI_FUNCTION 0x2232
#define DI_NONE 0
#define DI_DISABLE 1
#define DI_ENABLE 2
#define DI_CLEAR_FAULT 3
#define DI_STOP 4
#define DI_QUICK_STOP 5
#define DI_FWD_LIMIT_SW 6
#define DI_REV_LIMIT_SW 7
#define DI_HOME_SWITCH 8
#define DI_RESET_POSITION 9
#define DI_JOG_POSITIVE 10
#define DI_JOG_NEGATIVE 11
#define DI_SCRIPT_RUN 12
#define DI_SCRIPT_INTERRUPT0 13
#define DI_SCRIPT_INTERRUPT1 14
#define DI_SCRIPT_INTERRUPT2 15
#define DI_SCRIPT_INTERRUPT3 16
#define DI_SCRIPT_INTERRUPT4 17
#define DI_SCRIPT_INTERRUPT5 18
#define DI_SCRIPT_INTERRUPT6 19
#define DI_SCRIPT_INTERRUPT7 20
#define DI_FUNCTION_END 21

#define VI_DO_OPTIONS 0x2240
#define VI_DO_VALUE 0x2241
#define VI_DO_FUNCTION 0x2242
#define DO_NONE 0
#define DO_ENABLE 1
#define DO_FAULT 2
#define DO_MOVING 3
#define DO_HOMING 4
#define DO_HOME_COMP 5
#define DO_POS_LIMIT 6
#define DO_CURRENT_LIM 7
#define DO_REGEN_CUTOFF 8
#define DO_MOTOR_BRAKE 9
#define DO_IN_POSITION 10
#define DO_IN_VELOCITY 11
#define DO_READY 12
#define DO_SCRIPT_RUN 13
#define DO_FUNCTION_END 14

#define VI_AI_OPTIONS 0x2250
#define VF_AI_VALUE 0x2251
#define VI_AI_FUNCTION 0x2252
#define AI_NONE 0
#define AI_MC_VOLTAGE 1
```

```
#define      AI_MC_CURRENT                2
#define      AI_MC_VELOCITY              3
#define      AI_MC_POSITION              4
#define      AI_MC_TORQUE_LIMIT          5
#define      AI_FUNCTION_END             6

#define VI_AI_RAW_VALUE                   0x2253
#define VI_AI_CUTOFF_FREQ                 0x2255
#define VI_AI_CAL_MIN                     0x2256
#define VI_AI_CAL_CENTER                  0x2257
#define VI_AI_CAL_MAX                     0x2258
#define VI_AI_CAL_DEADBAND                0x2259

#define VI_PI_OPTIONS                     0x2260
#define VF_PI_VALUE                       0x2261
#define VI_PI_FUNCTION                    0x2262
#define VI_PI_RAW_VALUE                   0x2263
#define VI_PI_CAPTURE_TYPE                0x2264
#define      CT_PULSE_WIDTH               0
#define      CT_FREQUENCY                  1
#define      CT_DUTY_CYCLE                 2
#define      CT_END                       3

#define VI_PI_CUTOFF_FREQ                 0x2265
#define VI_PI_CAL_MIN                     0x2266
#define VI_PI_CAL_CENTER                  0x2267
#define VI_PI_CAL_MAX                     0x2268
#define VI_PI_CAL_DEADBAND                0x2269
```

상기의 상수들은 제어기 오브젝트와 함께 다음과 같이 사용할 수 있습니다.

```
_control = MC_ENABLE;

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

10.6 변수

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

10.6.1 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 또

는 밑줄문자 '_'가 되어야 함

- 대문자와 소문자는 서로 다른 문자로 인식함
- 예약어(for, while, do, ...)와 미리 정의된 상수(M_PI, M_E, ...)는 변수명으로 사용할 수 없음

변수 명은 다음과 다음과 같이 만들 수 있습니다.

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

10.6.2 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
```

배열형 변수는 'array' 키워드를 사용하여 다음과 같이 선언하고 초기화 합니다. 만일 초기값이 할당되지 않으면 배열의 모든 요소가 0으로 초기화 됩니다.

```
array a[5];
array b[5] = { 1, 2, 3 };
```

10.6.3 지역변수와 전역변수

지역변수는 함수 내부에서 선언되는 변수입니다. 함수를 벗어나면 더 이상 사용이 불가능해집니다.

```
function func()
{
    a = 1;           // func() 함수 내부에서만 사용되는 변수
    b = 2;           // func() 함수 내부에서만 사용되는 변수
    return a + b;
}
```

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

10.6.4 시스템 변수

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

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

상기 예제는 시스템 변수를 사용하여 다음과 같이 작성됩니다.

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

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

```
_device_id;                    // 제어기의 장치 ID, sub-index = 0
_user_variable[0];             // _user_variable의 첫 번째 요소, sub-index = 0
_user_variable[1];             // _user_variable의 두 번째 요소, sub-index = 1
```

제어기에서 제공하는 시스템 변수는 다음 표와 같습니다.

표 10-2 시스템 변수 목록

Index	Sub-index	System Variable	
		Short name	Long name
VI_VENDOR_ID	0	_vid	_vendor_id
VI_PRODUCT_ID	0	_pid	_product_id
VI_SOFTWARE_VERSION	0	_swv	_software_version
VI_HARDWARE_VERSION	0	_hwv	_hardware_version
VI_SYSTEM_STATUS	0	_ss	_system_status
VI_SYSTEM_CONTROL	0	_sc	_system_control
VI_SYSTEM_ARGUMENT	0	_sa	_system_argument
VF_POWER_SOURCE_VOLTAGE	0	_pv	_power_source_voltage
VF_POWER_SOURCE_CURRENT	0	_pc	_power_source_current
VS_DEVICE_NAME	0	_na	_device_name
VI_DEVICE_ID	0	_id	_device_id

VI_SERIAL_WATCHDOG	0	_sw	_serial_watchdog
VI_SERIAL_BAUDRATE	0	_sb	_serial_baudrate
VI_UNITS	0	_un	_units
VI_STARTUP_DRIVE_INPUT	0	_sdi	_startup_drive_input
VI_STARTUP_CONTROL_MODE	0	_scm	_startup_control_mode
VI_STARTUP_ENABLE_DELAY	0	_sed	_startup_enable_delay
VI_SCRIPT_STOP_ACTION	0	_ssa	_script_stop_action
VI_DISABLE_OPER_ACTION	0	_doa	_disable_oper_action
VI_PULSE_INPUT_MODE	0	_pi	_pulse_input_mode
VI_PULSE_COUNT_EDGE	0	_pce	_pulse_count_edge
VI_DIR_SIGNAL_POLARITY	0	_dp	_dir_signal_polarity
VI_GEAR_NUMERATOR	0	_gn	_gear_numerator
VI_GEAR_DENOMINATOR	0	_gd	_gear_denominator
VI_USER_VARIABLE	0~255	_u	_user_variable
VI_TEMP_VARIABLE	0~255	_t	_temp_variable
VI_ERROR_CODE	0~15	_ec	_error_code
VI_CONTROL	0	_c	_control
VI_STATUS	0	_s	_status
VI_FAULT	0	_f	_fault
VF_TEMPERATURE	0	_te	_temperature
VF_LOAD_TORQUE	0	_l	_load_torque
VI_TARGET_POSITION	0	_tp	_target_position
VI_TARGET_VELOCITY	0	_tv	_target_velocity
VF_TARGET_CURRENT_D	0	_tcd	_target_current_d
VF_TARGET_CURRENT_Q	0	_tcq	_target_current_q
VF_TARGET_VOLTAGE_D	0	_tvd	_target_voltage_d
VF_TARGET_VOLTAGE_Q	0	_tvq	_target_voltage_q
VF_TORQUE_LIMIT	0	_tl	_torque_limit

VI_POSITION_DEMAND	0	_pd	_position_demand
VI_VELOCITY_DEMAND	0	_vd	_velocity_demand
VF_CURRENT_DEMAND_D	0	_cdd	_current_demand_d
VF_CURRENT_DEMAND_Q	0	_cdq	_current_demand_q
VF_VOLTAGE_DEMAND_D	0	_vdd	_voltage_demand_d
VF_VOLTAGE_DEMAND_Q	0	_vdq	_voltage_demand_q
VF_ACCELERATION_DEMAND	0	_ad	_acceleration_demand
VI_POSITION_ACTUAL	0	_p	_position_actual
VI_VELOCITY_ACTUAL	0	_v	_velocity_actual
VF_CURRENT_ACTUAL_D	0	_cd	_current_actual_d
VF_CURRENT_ACTUAL_Q	0	_cq	_current_actual_q
VF_ACCELERATION_ACTUAL	0	_aa	_acceleration_actual
VF_POSITION_ERROR	0	_pe	_position_error
VF_VELOCITY_ERROR	0	_ve	_velocity_error
VF_CURRENT_ERROR_D	0	_ced	_current_error_d
VF_CURRENT_ERROR_Q	0	_ceq	_current_error_q
VI_SOFT_LIMIT_CHECK	0	_sl	_soft_limit_check
VI_MIN_POSITION	0	_n	_min_position
VI_MAX_POSITION	0	_x	_max_position
VI_LIMIT_ACTION	0	_la	_limit_action
VI_HOMING_METHOD	0	_hm	_homing_method
VI_HOMING_VELOCITY	0	_hv	_homing_velocity
VI_HOME_OFFSET	0	_ho	_home_offset
VI_HOME_POSITION	0	_hp	_home_position
VF_RATED_VOLTAGE	0	_rvo	_rated_voltage
VF_RATED_CURRENT	0	_rc	_rated_current
VI_RATED_VELOCITY	0	_rv	_rated_velocity
VF_MAX_CURRENT	0	_mc	_max_current
VI_MAX_VELOCITY	0	_mv	_max_velocity
VI_PROFILE_TYPE	0	_pty	_profile_type
VI_PROFILE_VELOCITY	0	_pve	_profile_velocity
VI_PROFILE_ACCEL	0	_pa	_profile_acceleration
VI_PROFILE_DECEL	0	_pde	_profile_deceleration

VI_PROFILE_JERK	0	_pj	_profile_jerk
VF_OVERHEAT_LIMIT	0	_oh	_overheat_limit
VF_OVERCURRENT_LIMIT	0	_oc	_overcurrent_limit
VF_OVERVOLTAGE_LIMIT	0	_ov	_overvoltage_limit
VF_UNDERVOLTAGE_LIMIT	0	_uv	_undervoltage_limit
VI_VIBRATION_DET	0	_vb	_vibration_detect
VI_STALL_CURRENT_DET	0	_sd	_stall_current_detect
VI_POSITION_TRACK_ERROR	0	_pt	_position_track_error
VI_VELOCITY_TRACK_ERROR	0	_vt	_velocity_track_error
VI_MOTOR_TYPE	0	_m	_motor_type
VI_MOTOR_DIRECTION	0	_d	_motor_direction
VI_FEEDFORWARD_OPTIONS	0	_ff	_feedforward_option
VI_POSITION_SENSOR	0	_ps	_position_sensor
VI_ENCODER_DIRECTION	0	_ed	_encoder_direction
VI_ENCODER_RESOLUTION	0	_er	_encoder_resolution
VI_NO_POLE_PAIRS	0	_np	_no_pole_pairs
VI_HALLSENSOR_PHASE	0	_hs	_hallsensor_phase
VI_LINEAR_SENSOR_PITCH	0	_lsp	_linear_sensor_pitch
VF_BEMF_CONSTANT	0	_b	_bemf_constant
VF_RESISTANCE	0	_r	_resistance
VF_INDUCTANCE_D	0	_ld	_inductance_d
VF_INDUCTANCE_Q	0	_lq	_inductance_q
VI_ELECTRIC_ANGLE_BIAS	0	_eb	_electric_angle_bias
VF_TORQUE_CONSTANT	0	_to	_torque_constant
VF_MOMENT_OF_INERTIA	0	_i	_moment_of_inertia
VF_VISCOUS_FRICTION	0	_vf	_viscous_friction
VF_COULOMB_FRICTION	0	_cf	_coulomb_friction
VF_LOAD_TORQUE_BIAS	0	_tb	_load_torque_bias
VF_POSITION_P_GAIN	0	_pp	_position_p_gain
VF_VELOCITY_P_GAIN	0	_vp	_velocity_p_gain
VF_VELOCITY_I_GAIN	0	_vi	_velocity_i_gain
VF_VELOCITY_D_GAIN	0	_vdg	_velocity_d_gain
VF_CURRENT_P_GAIN_D	0	_cpd	_current_p_gain_d

VF_CURRENT_I_GAIN_D	0	_cid	_current_i_gain_d
VF_CURRENT_P_GAIN_Q	0	_cpq	_current_p_gain_q
VF_CURRENT_I_GAIN_Q	0	_ciq	_current_i_gain_q
VI_PMAF_WINDOW_SIZE	0	_pw	_pmaf_window_size
VI_SMAF_WINDOW_SIZE	0	_vw	_vmaf_window_size
VI_IN_POSITION_TH	0	_ip	_in_position_threshold
VI_IN_VELOCITY_TH	0	_iv	_in_velocity_threshold
VI_BRAKE_ON_DELAY	0	_bo	_brake_on_delay
VF_JOG_VELOCITY	0	_jv	_jog_velocity
VF_REGEN_VOLTAGE_CUTOFF	0	_rv	_regen_voltage_cutoff
VI_NO_DI	0	_ndi	_no_digital_input
VI_NO_DO	0	_ndo	_no_digital_output
VI_NO_AI	0	_nai	_no_analog_input
VI_NO_PI	0	_npi	_no_pulse_input
VI_DIGITAL_INPUTS	0	_di	_digital_inputs
VI_DIGITAL_OUTPUTS	0	_do	_digital_outputs
VI_DIGITAL_OUTPUTS_MASK	0	_dom	_digital_outputs_mask
VI_DI_OPTIONS	1 ~ N	_dio	_di_option
VI_DI_VALUE	1 ~ N	_div	_di_value
VI_DI_FUNCTION	1 ~ N	_dif	_di_function
VI_DO_OPTIONS	1 ~ M	_doo	_do_option
VI_DO_VALUE	1 ~ M	_dov	_do_value
VI_DO_FUNCTION	1 ~ M	_dof	_do_function
VI_AI_OPTIONS	1 ~ O	_aio	_ai_option
VF_AI_VALUE	1 ~ O	_aiv	_ai_value
VI_AI_FUNCTION	1 ~ O	_aif	_ai_function
VI_AI_RAW_VALUE	1 ~ O	_air	_ai_raw_value
VI_AI_CUTOFF_FREQ	1 ~ O	_acf	_ai_cutoff_freq
VI_AI_CAL_MIN	1 ~ O	_ain	_ai_cal_min
VI_AI_CAL_CENTER	1 ~ O	_aic	_ai_cal_center
VI_AI_CAL_MAX	1 ~ O	_aix	_ai_cal_max
VI_AI_CAL_DEADBAND	1 ~ O	_aidb	_ai_cal_deadband

VI_PI_OPTIONS	1 ~ P	_pio	_pi_option
VF_PI_VALUE	1 ~ P	_piv	_pi_value
VI_PI_FUNCTION	1 ~ P	_pif	_pi_function
VI_PI_RAW_VALUE	1 ~ P	_pir	_pi_raw_value
VI_PI_CAPTURE_TYPE	1 ~ P	_pit	_pi_capture_type
VI_PI_CUTOFF_FREQ	1 ~ P	_pcf	_pi_cutoff_freq
VI_PI_CAL_MIN	1 ~ P	_pin	_pi_cal_min
VI_PI_CAL_CENTER	1 ~ P	_pic	_pi_cal_center
VI_PI_CAL_MAX	1 ~ P	_pix	_pi_cal_max
VI_PI_CAL_DEADBAND	1 ~ P	_pidb	_pi_cal_deadband

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

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

10.6.5 자료형 변환

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

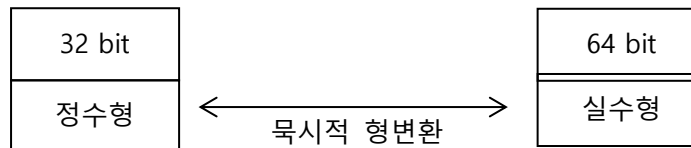


그림 10-1 스크립트에서의 자료 형변환

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
```

10.7 연산자

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

10.7.1 산술, 부호 연산자

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

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

산술 연산자와 부호 연산자에 대해 정리하면, 다음과 같습니다.

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

10.7.2 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. 전위

형으로 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.

10.7.3 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

10.7.4 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 논

리값을 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	a거나 b 둘 중 하나라도 참이면 참

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

10.7.5 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

\wedge	$a \wedge b$	10111011 \wedge 10101000	00010011	a와 b의 비트단위의 XOR (배타적 논리합)
\sim	$\sim a$	~ 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.

10.7.6 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
+=	$a += b$	$n = n + 1; \rightarrow n += 1;$	$a = a + b$
-=	$a -= b$	$n = n - 1; \rightarrow n -= 1;$	$a = a - b$
*=	$a *= b$	$n = n * 1; \rightarrow n *= 1;$	$a = a * b$
/=	$a /= b$	$n = n / 1; \rightarrow n /= 1;$	$a = a / b$
%=	$a \% = b$	$n = n \% 1; \rightarrow n \% = 1;$	$a = a \% b$
&=	$a \& = b$	$n = n \& 1; \rightarrow n \& = 1;$	$a = a \& b$
=	$a = b$	$n = n 1; \rightarrow n = 1;$	$a = a b$
^=	$a \wedge = b$	$n = n \wedge 1; \rightarrow n \wedge = 1;$	$a = a \wedge b$
<<=	$a << = b$	$n = n << 1; \rightarrow n << = 1;$	$a = a << b$
>>=	$a >> = b$	$n = n >> 1; \rightarrow n >> = 1;$	$a = a >> b$

10.7.7 연산자 우선순위

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

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

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

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.

10.8 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. 스

크립트 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.

10.8.1 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) {
    ...
}
```

10.8.2 if-else 문

`if-else` 문은 조건식이 참 혹은 거짓에 따라 문장을 양자택일할 경우 사용합니다. 조건식이 참이면 `if`문 다음의 문장만을 실행하고, 거짓이면 `else`문 다음의 문장만을 실행합니다. 여기서 `else` 문은 `if`문에 종속적입니다. `if`문 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 else 를 순서대로 짝지어 주는 것입니다. 이러한 혼동을 피하기 위하여 '{'와 '}'를 이용하여 if-else 문을 올바르게 묶어 주어야 합니다.

If-else 문
<pre> if (조건식) { if (조건식) 문장 또는 블록 else 문장 또는 블록 } </pre>

10.8.3 if-else-if-else 문

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

If-else-if-else 문
<pre> if (조건식) 문장 또는 블록 else if (조건식) 문장 또는 블록 else if (조건식) 문장 또는 블록 else 문장 또는 블록 </pre>

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
}
    
```

10.8.4 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++;
}
```

10.8.5 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;
}
```

10.8.6 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
<pre>do { 문장 또는 블록 } while (조건식);</pre>

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

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

10.8.7 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
<pre>while (i<10) { ... break; ... }</pre>

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

```
i = 1;
sum = 0;
while (1) {
```



```

        if (i <= 100) sum += i;
        else break;
        i++;
    }

```

10.8.8 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
<pre> while (i<10) { ... continue; ... } </pre>

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

```

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

```

10.8.9 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
<pre> 레이블: ... </pre>

```
goto 레이블;
```

다음은 1에서 100까지 더하는 연산을 goto문으로 작성한 예입니다.

```
i = 1;
sum = 0;

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

10.9 함수

사용자 정의 함수는 모듈형 프로그래밍을 가능하게 하는 중요한 요소입니다. 사용자는 function 키워드를 사용하여 직접 함수를 만들 수 있으며 함수의 이름으로 함수를 호출할 수 있습니다.

10.9.1 함수 선언

function 키워드를 사용하여 다음과 같이 함수를 선언합니다. 함수의 이름은 변수명과 동일한 규칙으로 작성합니다. 함수의 인자로는 배열을 받을 수 없고 정수형이나 실수형 값을 받습니다. 인자는 변수명과 동일한 규칙으로 만듭니다.

```
function add(a, b)
{
    c = a + b;
    return c;
}
```

함수는 return 키워드를 사용하여 하나의 정수형이나 실수형 값을 반환할 수 있습니다. 만일 함수의 끝에서 return 키워드가 사용되지 않은 경우에는 함수는 강제로 0을 반환하도록 합니다.

10.9.2 인터럽트 함수

인터럽트 함수도 function 키워드를 사용하여 정의됩니다. 일반 함수와 달리 인터럽트 함수의 이름은 interrupt0, interrupt1, ... interrupt7와 같이 0에서 7까지 8개의 인터럽트 플래그에 대응하는 함수를 만들 수 있습니다.

```
function interrupt0()
{
    ...
}
```

10.9.3 함수 호출

함수 이름으로 함수를 호출합니다.

```
x = add(1,2);
```

10.10 내장 함수

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

스크립트 언어에서는 다음과 같은 내장 함수들을 제공합니다:

- `rand()` - 0과 32767 사이의 의사-난수를 반환
- `int(x)` - `x`를 소숫점 첫째 자리에서 반올림하여 정수로 변환
- `sin(x)` - 라디안으로 주어진 `x`의 sine 값을 반환
- `cos(x)` - 라디안으로 주어진 `x`의 cosine 값을 반환
- `tan(x)` - 라디안으로 주어진 `x`의 tangent 값을 반환
- `asin(x)` - `x`(sine `x`의 결과 값)의 arc sine의 값을 반환
- `acos(x)` - `x`(cosine `x`의 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, `x`의 쌍곡선 sine을 반환
- `cosh(x)` - 수학적으로 $\exp(x) + \exp(-x)/2$ 로 정의된, `x`의 쌍곡선 cosine을 반환
- `tanh(x)` - $\sinh(x)/\cosh(x)$ 이라는 수학적 정의를 가진, `x`의 쌍곡선 tangent `x`를 반환
- `fabs(x)` - `x`의 절대값을 반환
- `floor(x)` - `x`보다 작거나 같은 정수 중에서 최대 값을 반환
- `ceil(x)` - `x`보다 크거나 같은 정수 중에서 최소 값을 반환
- `sqrt(x)` - `x`의 음이 아닌 루트 값을 반환
- `exp(x)` - 자연대수 `e`의 `x`승 값을 반환
- `log(x)` - `x`의 자연로그를 반환
- `log10(x)` - 10을 밑으로 하는 `x`의 로그 값을 반환
- `atan2(x,y)` - 두 개의 인수를 가진 arc tangent 함수
- `pow(x,y)` - `x`의 `y`승을 반환하는 일반적 지수 함수
- `min(x,y)` - 주어진 두 인자의 최소값을 반환
- `max(x,y)` - 주어진 두 인자의 최대값을 반환
- `ei()` - 인터럽트 발생을 가능하도록 함
- `di()` - 인터럽트 발생을 불가능하도록 함
- `timer0(x)` - `x` 밀리초 이후 `interrupt0()` 함수 실행
- `timer1(x)` - `x` 밀리초 이후 `interrupt1()` 함수 실행
- `timer2(x)` - `x` 밀리초 이후 `interrupt2()` 함수 실행

- clock() - 현재 시간을 밀리초 단위로 반환
- wait() - 모터가 목표 위치에 도달할 때까지 대기
- 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

10.10.1 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;
```

10.10.2 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
```

10.10.3 sin

Declaration: sin(x)

라디안 각 x의 sine 값을 계산합니다.

인수 x에는 모든 값을 사용할 수 있습니다.

반환 값은 -1과 1 사이의 값입니다.

10.10.4 cos

Declaration: $\cos(x)$

라디안 각 x 의 cosine 값을 계산합니다.

인수 x 에는 모든 값을 사용할 수 있습니다.

반환 값은 -1과 1 사이의 값입니다.

10.10.5 tan

Declaration: $\tan(x)$

라디안 각 x 의 tangent 값을 계산합니다.

인수 x 에는 모든 값을 사용할 수 있습니다.

반환 값은 모든 값이 될 수 있습니다.

10.10.6 asin

Declaration: $\text{asin}(x)$

x 의 arc sine 값을 계산합니다. $\sin()$ 함수의 역함수입니다.

인수 x 의 값은 -1과 1 사이의 값입니다.

반환 값은 $-\pi/2$ 와 $\pi/2$ 사이의 라디안 각입니다.

10.10.7 acos

Declaration: $\text{acos}(x)$

x 의 arc cosine 값을 계산합니다. 이 함수는 $\cos()$ 함수의 역함수입니다.

인수 x 의 값은 -1과 1 사이의 값입니다.

반환 값은 0과 π 사이의 라디안 각입니다.

10.10.8 atan

Declaration: atan(x)

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

인수 x에는 모든 값을 사용할 수 있습니다.

반환 값은 $-\pi/2$ 와 $\pi/2$ 사이의 라디안 각입니다.

10.10.9 sinh

Declaration: sinh(x)

x의 쌍곡선 sine을 계산합니다.

인수 x에는 모든 값을 사용할 수 있습니다.

반환 값은 모든 값이 될 수 있습니다.

10.10.10 cosh

Declaration: cosh(x)

x의 쌍곡선 cosine을 계산합니다.

인수 x에는 모든 값을 사용할 수 있습니다.

반환 값은 모든 값이 될 수 있습니다.

10.10.11 tanh

Declaration: tanh(x)

x의 쌍곡선 tangent를 계산합니다.

인수 x에는 모든 값을 사용할 수 있습니다.

반환 값은 모든 값이 될 수 있습니다.

10.10.12 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
```

10.10.13 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
```

10.10.14 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
```

10.10.15 **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
```

10.10.16 **exp**

Declaration: exp(x)

자연상수 e의 x 승 값을 계산합니다.

인수 x에는 모든 값을 사용할 수 있습니다.

반환 값은 모든 값이 될 수 있습니다.

Examples:

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

10.10.17 **log**

Declaration: log(x)

x 의 자연로그를 계산합니다.

인수 x에는 모든 값을 사용할 수 있습니다.

반환 값은 모든 값이 될 수 있습니다.

Examples:

```
a = log(M_E);          // a는 1
```



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

10.10.18 **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
```

10.10.19 **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 $-\pi$ 와 π 사이의 라디안 각입니다.

Examples:

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

10.10.20 **pow**

Declaration: pow (x, y)

x의 y승을 계산합니다.

만일 인수 y 가 분수 값이면 인수 x 는 음수가 될 수 없습니다. 그리고 y 가 0보다 작거나 같으면 x 는 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...
```

10.10.21 min

Declaration: min (x , y)

주어진 두 인수 x , y 의 최소값을 선택합니다.

인수 x , y 에는 모든 값을 사용할 수 있습니다.

10.10.22 max

Declaration: max (x , y)

주어진 두 인수 x , y 의 최대값을 선택합니다.

인수 x , y 에는 모든 값을 사용할 수 있습니다.

10.10.23 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;
```

10.10.24 wait

Declaration: wait()

모터가 목표 위치에 도달할 때까지 대기합니다.

Remarks: 제어기는 Position Control 모드에서 In-Position 상태가 아니라면 wait() 함수에서 빠져나오지 않고 대기합니다. 모터가 목표 위치에 도달하여 In-Position 상태가 되면 wait() 함수를 빠져나오게 됩니다.

Examples:

```
_target_position = 1000; // 1000 위치로 이동
wait ();              // 1000 위치에 도달할 때까지 대기
_target_position = 2000; // 2000 위치로 이동
wait ();              // 2000 위치에 도달할 때까지 대기
```

10.10.25 ei

Declaration: ei()

인터럽트 발생을 가능하도록 합니다.

Remarks: Digital Input 채널에서 인터럽트 기능이 연결되어 있다면, 입력 값이 0에서 1로 변할 때 가상머신의 해당 인터럽트 플래그를 세팅 합니다. 그리고 ei() 함수를 사용하여 인터럽트를 사용 가능하도록 하면 인터럽트 플래그에 해당하는 함수가 실행됩니다. 가상머신이 시작되면서 인터럽트 발생 가능한 상태가 됩니다.

Examples:

```
di ();
...
ei ();
```

10.10.26 di

Declaration: di()

인터럽트 발생을 불가능하도록 합니다.

Remarks: Digital Input 채널에 의해 해당 인터럽트 플래그가 세팅 되더라도 인터럽트 사용이 불가능하도록 설정되어 있다면, 인터럽트 함수는 실행되지 않습니다. di() 함수로 가상머신에서 인터럽트 사용을 불가능하도록 설정합니다. 만일 인터럽트 사용이 불가능하도록 설정된 상태에서 인

터럽트 플래그가 세팅 된다면, ei() 함수에 의해 인터럽트를 사용가능하도록 설정하면 인터럽트 플래그에 해당하는 함수가 실행됩니다.

Examples:

```
di();  
timer(10);  
...  
ei();
```

10.10.27 timer0

Declaration: timer0(x)

x 밀리초 후에 interrupt0() 함수가 실행 되도록 합니다.

Examples:

```
function interrupt0()  
{  
    _target_velocity = 0;  
}  
  
_go_position = 100000;    // 100000 위치로 이동 명령  
timer0 (1000);           // 1초 후 목표 위치에 도달하지 못하더라도  
                           // interrupt0() 함수에 의해 모터가 정지함  
wait ();                 // 모터가 목표 위치에 도달할 때까지 대기
```

10.10.28 timer1

Declaration: timer1(x)

x 밀리초 후에 interrupt1() 함수가 실행 되도록 합니다.

10.10.29 timer2

Declaration: timer2(x)

x 밀리초 후에 interrupt2() 함수가 실행 되도록 합니다.

10.10.30 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:

```
// 모터는 1초 마다 위치 -5000과 5000 사이를 이동합니다.
// 모터가 목표 위치에 도달하지 못하더라도 1초 후에는 새로운 명령을 수행합니다.
while (1) {
    _target_position = -5000;
    sleep (1000);
    _target_position = 5000;
    sleep (1000);
}
```

10.10.31 getv

Declaration: getv (index, sub_index)

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

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

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

Examples:

```
a = _position_actual;           // 모터의 현재 위치를 읽어 옴
a = getv (VI_POSITION_ACTUAL, 0); // a = _position; 과 동일
a = getv (0x2131, 0);           // a = _position; 과 동일
```

10.10.32 setv

Declaration: setv (index, sub_index, x)

인수 index와 sub_index로 지정된 제어기의 오브젝트에 x를 씁니다. setv() 함수로 제어기의 모

든 오브젝트에 값을 변경할 수 있습니다. 변경된 값은 제어기에 즉시 적용됩니다. 하지만 변경된 값이 제어기의 플래시 메모리에 저장 되려면, 모터가 멈춘 상태에서 약 5초 정도 지나야 합니다.

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

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

Examples:

```
_target_position = 5000;           // 모터를 5000 위치로 이동합니다.  
setv (VI_TARGET_POSITION, 0, 5000); // _go_position = 5000; 과 동일  
setv (0x2111, 0, 5000);           // _go_position = 5000; 과 동일
```

11 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 수행 하던 일부 혹은 모든 기능을 제어기에서 구현할 수 있기 때문에, 전체 시스템 구성을 단순화 할 수 있습니다.

11.1 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, "**10.2 C언어와의 차이**"만 살펴보더라도 바로 프로그램을 작성할 수 있습니다.

11.1.1 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.

11.1.2 소스코드 구조

스크립트 언어로 프로그램을 작성할 때 소스코드의 구조는 다른 종류의 프로그램을 작성하는 것과 비슷합니다. 단순히, 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.

```
// 모터를 움직일 때 사다리꼴 속도 프로파일을 사용하며,  
// 프로파일의 최대 속도, 가속도, 감속도 설정  
_profile_type = 1;           // 1 - 사다리꼴형, 2 - 사인파형  
_profile_velocity = 3000;    // 최대속도 3000rpm 설정  
_profile_acceleration = 1500; // 가속도 1500rpm/s 설정  
_profile_deceleration = 1500; // 감속도 1500rpm/s 설정
```

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.

```
// 모터를 움직일 때 사다리꼴 속도 프로파일을 사용하며,  
// 프로파일의 최대 속도, 가속도, 감속도 설정  
_profile_type = 1;           // 1 - 사다리꼴형, 2 - 사인파형  
_profile_velocity = 3000;    // 최대속도 3000rpm 설정  
_profile_acceleration = 1500; // 가속도 1500rpm/s 설정  
_profile_deceleration = 1500; // 감속도 1500rpm/s 설정  
  
// 디지털 입력 채널 1의 값이 1일 때 모터를 1000rpm 속도로 회전  
while (1) {  
    speed = 0;  
    if (_di_value[1]) speed = 1000;  
  
    _target_velocity = speed;  
    sleep (100);  
}
```

반복 실행구조는 보통 `while(1) { ... }` 과 같이 만듭니다. `while` 문의 조건은 항상 참이기 때문에, 이 프로그램은 사용자가 스크립트의 실행을 중단하거나 제어가 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 제어가 스크립트의 실행에 불필요한 시간을 할당하는 것을 방지해야 합니다.

11.2 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.

11.2.1 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 "11.2.2 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.

11.2.2 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'

11.3 다운로드 및 실행

11.3.1 다운로드

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

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

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

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

11.3.2 실행

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

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

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

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

```
system_control=1↵ // 스크립트 시작 명령
system_status↵    // 스크립트의 실행 상태 읽기
system_control=5↵ // 스크립트 종료 명령
system_status↵
```

12 관련 자료

아래 홈페이지에서 MoonWalker 관련 제품과 자료 그리고 예제, 동영상을 확인하실 수 있습니다.

- 엔티렉스:
<http://www.ntrexgo.com/>
- 디바이스마트:
<http://www.devicemart.co.kr/>
- MoonWalker 제품:
<http://mwbot.co.kr/>
- MoonWalker 액추에이터:
<http://www.devicemart.co.kr/goods/list.php?category=006011008011>
- MoonWalker 판매페이지:
<http://www.devicemart.co.kr/goods/list.php?category=006011008>

문서 변경 이력

Data	Version	Charges
2018. 07. 17	1.00	- 첫 출시

제품의 보증

1. 본 제품은 엄정한 품질관리 및 검사과정을 거쳐서 만들어 진 제품입니다.
2. 제품 구입 후 1개월 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