User Manual

AxisLink Motion User Manual

AxisLink Motion User Manual (Rev0.2)

2015.01.07

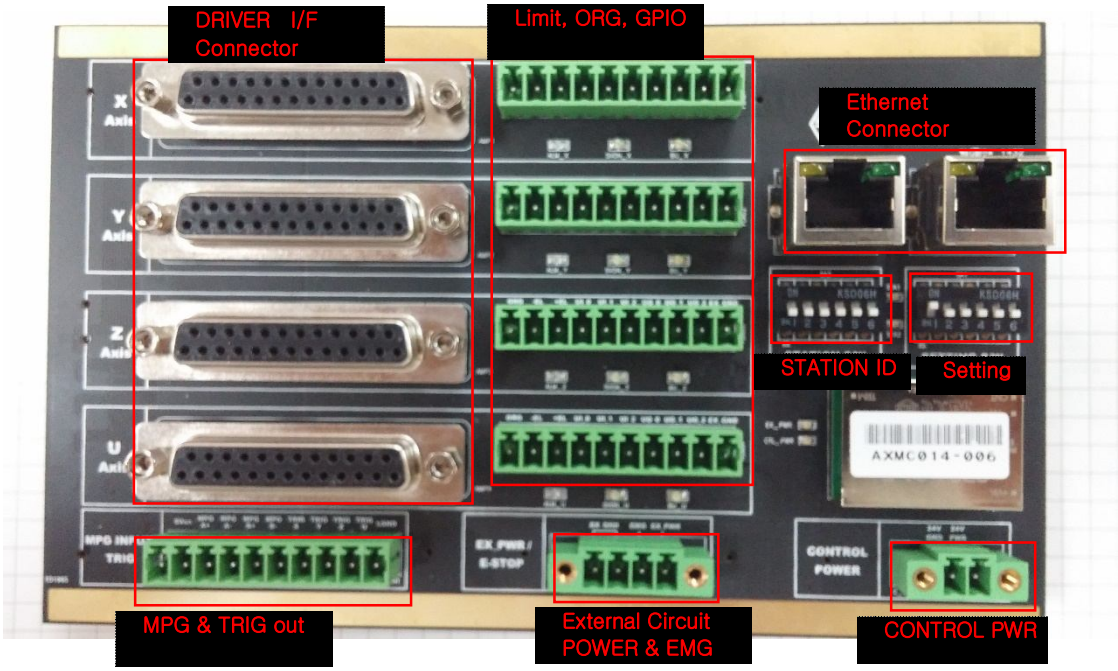
DELTA TAU Data Systems, Inc.
NEW IDEAS IN MOTION ...

REVISION HISTORY			
버전	수정 내용	일자	수정자
1	최초작성	2014.10.02	이재혁
2	가속도 최대값 설정 관련 추가	2014.10.02	이재혁
3	HOME 관련 설명 추가	2015.01.07	이재혁

제품개요

본 매뉴얼은 AxisLink-Motion 보드를 사용하는데 필요한 내용을 포함하고 있습니다.
AxisLink-Motion 보드는 당사의 모션 컨트롤러인 UMAC과 함께 사용되는 Slave 보드입니다.
적용 가능한 모델은 UMAC, Clipper, Cruiser가 있으며 연결되는 모델에는 반드시 AxisLink-Master 보드(OPT-2, Memory-shared)가 장착되어 있어야 합니다.
AxisLink-Motion은 UMAC or Cruiser의 Memory share 기능을 통하여 연결이 되며 각각 할당된 메모리 주소에 값을 쓰고/읽어 Data를 전달 합니다.
AxisLink-Motion 보드는 최대 6개까지 추가가 가능 하며, 각각 4개의 축씩 최대 24축을 위치모드로 구동 할 수 있습니다.
각각의 4 축은 X축 Y축 Z축 U축으로 고정 되어 사용 합니다.
본 매뉴얼은 YASKAWA SERVO 에서 TEST 되었습니다.

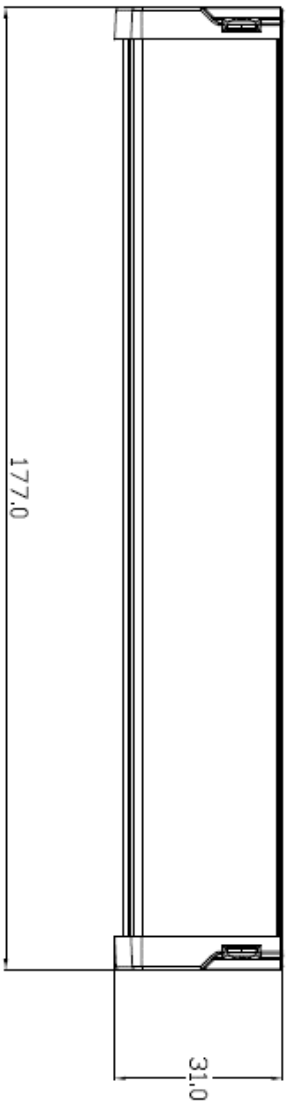
보드구성

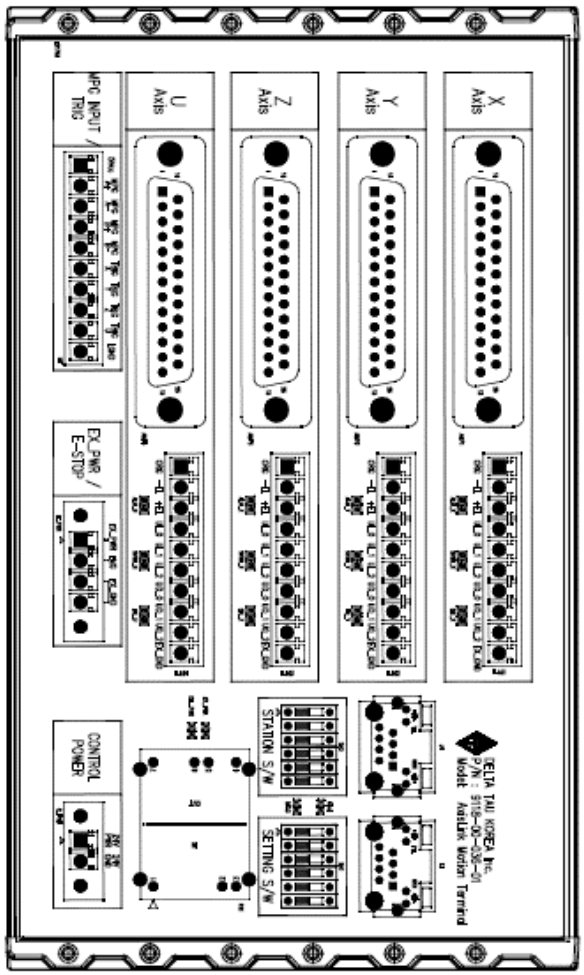


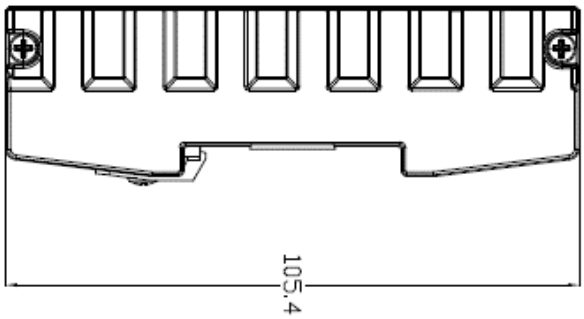
전원 사양

- 24[V]
- 300[mA]

AxisLink_Motion Board Dimension

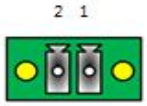




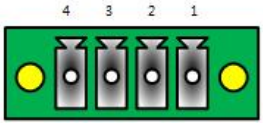


I/O 구성

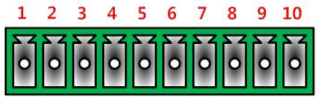
Control Power Input Connector (2 PIN TERMINAL BLOCK)

PIN	Symbol	Function	Description	Note
1	24V PWR	Input	DC 24V Control Power Input	
2	24V GND	GND	DC 24V Control Power GND	

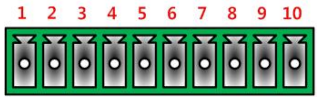
External Circuit Power & EMG Signal Input Connector (4 PIN TERMINAL BLOCK)

PIN	Symbol	Function	Description	Note
1	EX_PWR	Input	External Circuit 24V Power Input	
2	EMG	Input	Emergency Signal Input	
3	EX_GND	GND	External Circuit 24V Ground	
4	EX_GND	GND	External Circuit 24V Ground	

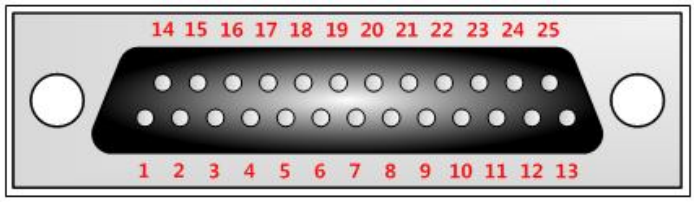
MPG and Trig Signal I/F Connector

PIN	Symbol	Function	Description	Note
1	5Vcc	Output	DC 5V Output	
2	MPG A+	Input	Manual Pulse Generator A+ Input (Linedriver)	
3	MPG A-	Input	Manual Pulse Generator A- Input (Linedriver)	
4	MPG B+	Input	Manual Pulse Generator B+ Input (Linedriver)	
5	MPG B-	Input	Manual Pulse Generator B- Input (Linedriver)	
6	TRIG X	Output	1st Axis Trig Output (TTL)	
7	TRIG Y	Output	2nd Axis Trig Output (TTL)	
8	TRIG Z	Output	3rd Axis Trig Output (TTL)	
9	TRIG U	Output	4th Axis Trig Output (TTL)	
10	LGND	GND	DC 5V Ground	

Mechanical Signal I/F Connector

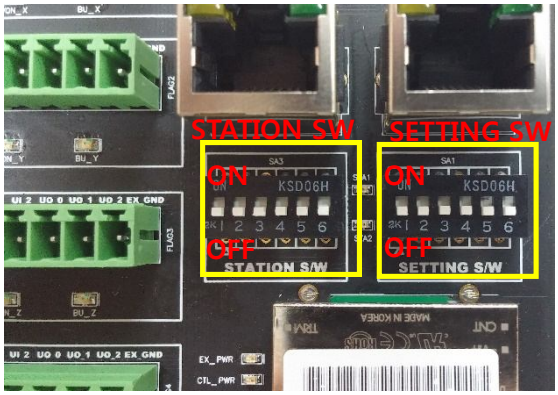
PIN	Symbol	Function	Description	Note
1	ORG	Input	Origin Input	
2	-EL	Input	Minus End Limit Input	
3	+EL	Input	Plus End Limit Input	
4	UI_0	Input	1st Universal Input	
5	UI_1	Input	2nd Universal Input	
6	UI_2	Input	3rd Universal Input	
7	UO_0	Output	1st Universal Output	
8	UO_1	Output	2nd Universal Output	
9	UO_2	Output	3rd Universal Output	
10	EX_GND	GND	External Circuit 24V Ground	

Driver I/F Connector (DSUB25Pin Connector-Male)



PIN	Symbol	Function	Description
1	PULSE +	Output	지령 Pulse + Output (Linedriver)
2	DIR+	Output	지령 Direction + Output (Linedriver)
3	EXT5V	Output	DC 5V Output
4	ENC-A+	Input	Encoder A+ Input (Linedriver)
5	ENC-B+	Input	Encoder B+ Input (Linedriver)
6	ENC-Z+	Input	Encoder Z+ Input (Linedriver)
7	EXT24V	In/Out	External Circuit 24V Power (In/Out)
8	SVON	Output	Servo On Output
9	CLR	Output	Deflection counter Clear Output
10	RST	Output	Alarm Reset Output
11	EXT24GND	GND	External Circuit 24V Ground
12	EXT24GND	GND	External Circuit 24V Ground
13	EXT24GND	GND	External Circuit 24V Ground
14	PULSE -	Output	지령 Pulse - Output (Linedriver)
15	DIR-	Output	지령 Direction - Output (Linedriver)
16	EXTGND	GND	DC 5V Ground
17	ENC-A-	Input	Encoder A- Input (Linedriver)
18	ENC-B-	Input	Encoder B- Input (Linedriver)
19	ENC-Z-	Input	Encoder Z- Input (Linedriver)
20	EXT24V	In/Out	External Circuit 24V Power (In/Out)
21	INP	Input	In-position Input
22	RDY	Input	Ready Input
23	ALM	Input	Alarm Input
24	EXT24GND	GND	External Circuit 24V Ground
25	EXT24GND	GND	External Circuit 24V Ground

DIP SWITCH 설정



STATION SWITCH : Slave address 설정

- 여러 개의 AxisLink-Motion 보드를 연결할 경우 Station Switch 을 이용해 각 Slave 보드의 ID를 설정합니다. Slave 보드의 ID에 따라 Slave Address가 달라집니다.

.

- Link 내에 중복되는 Slave Address가 존재할 경우 같은 주소를 가지는 Slave 보드들은 통신이 이루어지지 않습니다.

Board No	SA3.1	SA3.2	SA3.3	SA3.4	SLAVE
1번 Board	OFF	OFF	OFF	OFF	**Motion @A (Default)**
2번 Board	ON	OFF	OFF	OFF	**Motion @B**
3번 Board	OFF	ON	OFF	OFF	**Motion @C**
4번 Board	ON	ON	OFF	OFF	**Motion @D**
5번 Board	OFF	OFF	ON	OFF	**Motion @E**
6번 Board	ON	OFF	ON	OFF	**Motion @F**

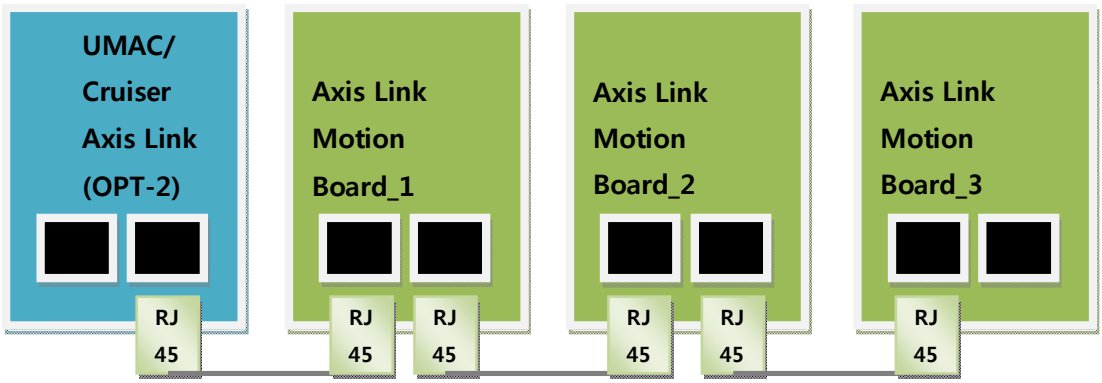
SETTING SW : FACTORY RESET

SA1.1	SA1.2	SA1.3	SA1.4	SA1.5	SA1.6	Description	DEFAULT
OFF	OFF	OFF	OFF	OFF	OFF	일반 부팅	Default
ON	OFF	OFF	OFF	OFF	OFF	초기화 부팅	

CONNECTION

MASTER-SLAVE 연결

AxisLink는 Master와 Slave간에 "RJ45" Connector를 통해 "RS422" 통신을 합니다. 여러 개의 Slave 보드를 연결 시 아래와 같이 Daisy Chain을 구성하여야 합니다.(최대 6개)



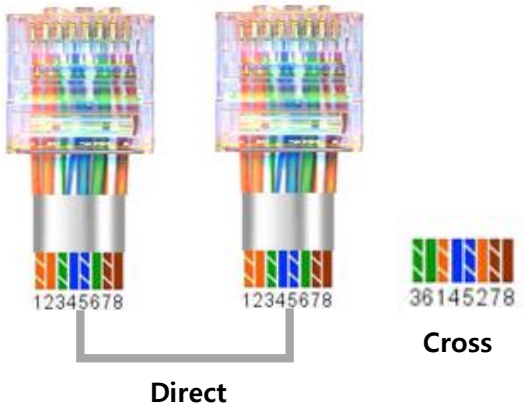
- 여러 개의 Slave는 서로 다른 Address를 가져야 하며 Slave Board의 Address는 **STATION SWITCH**를 사용하여 설정합니다.

- 연결을 위해 사용되는 Ethernet Cable은 일반적으로 사용되는 Cross Type이 아닌 **Direct Type**을 사용하여야 합니다.

  Direct Type의 경우 양단의 RJ45 Connector 선 배열이 일치합니다.

  Cross Type의 경우 양단의 배열이 다릅니다.

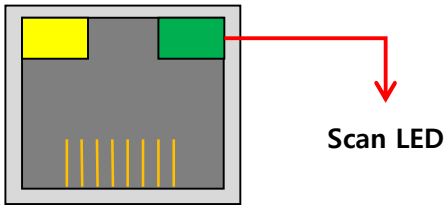
  1번(주황 줄) ↔ 3번(녹색 줄)
  2번(주황) ↔ 6번(녹색)



(위의 선 색깔은 일반적으로 사용되는 경우이며 제조사에 따라 색깔은 다를 수 있습니다)

연결상태 확인

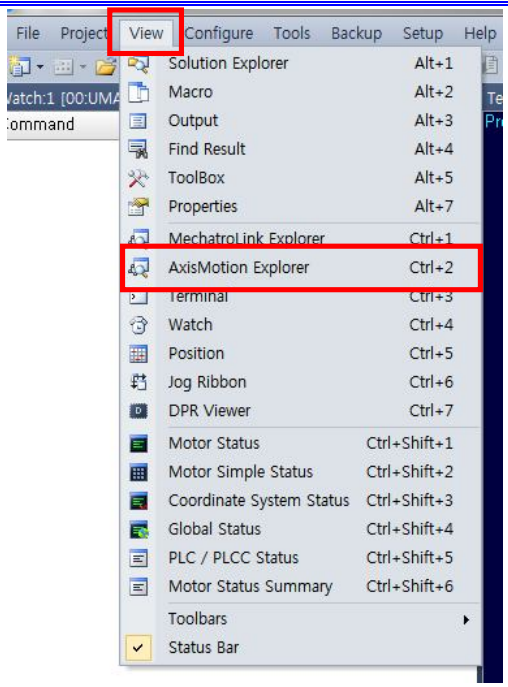
AxisLink 연결포트

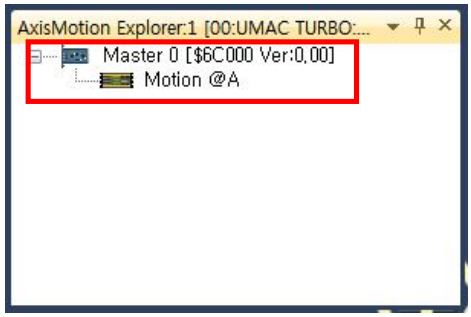


Scan Status LED (GREEN)

- AxisLink-Master 보드가 AxisLink-Motion 보드를 Scan 시 켜지게 됩니다. Master 보드와 Slave 보드가 연결되면 켜져야 정상입니다.

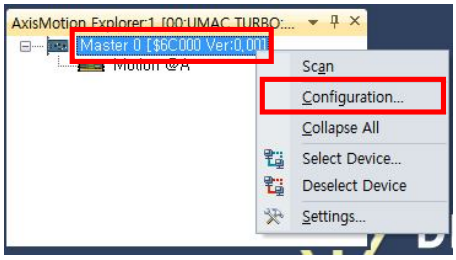
AXISMOTION EXPLORER 를 통한 연결 및 설정

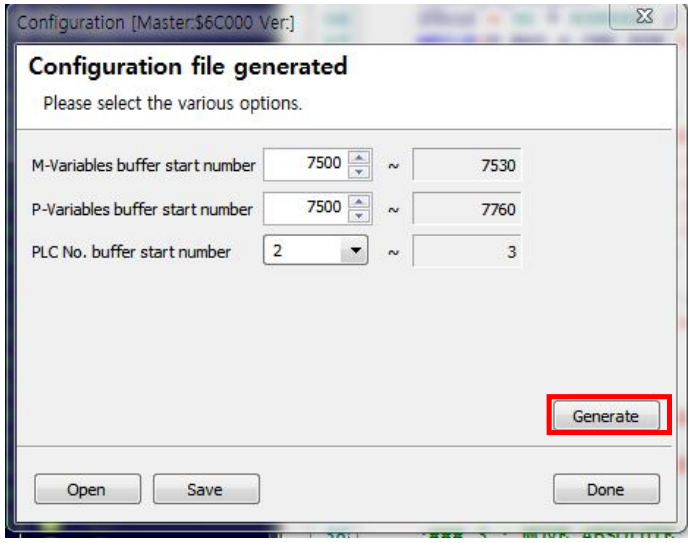


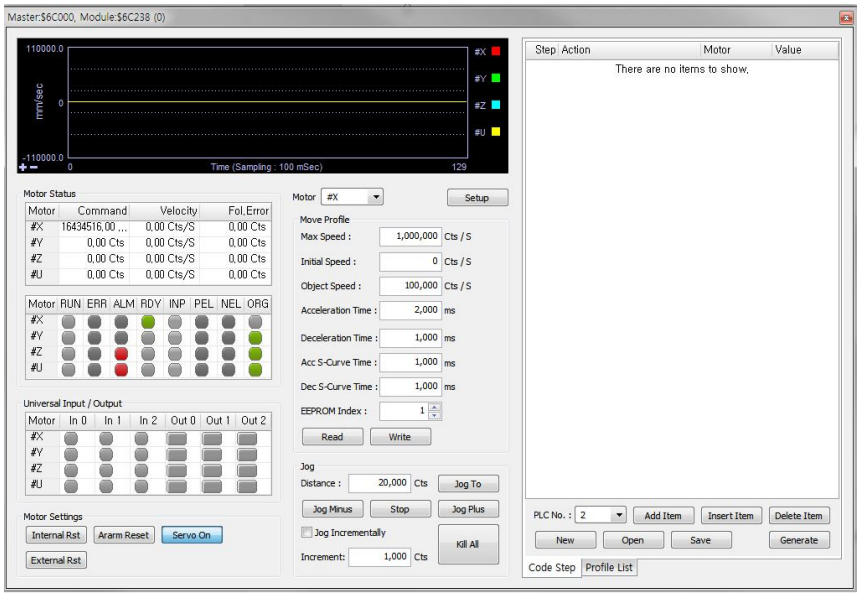


PMAC Suite "View" ➔ "AxisMotion Explorer"를 실행 하면 위와 같은 창이 생성 됩니다.
위 화면 에서 Master 0에 Motion@A가 Scan 된 것을 확인 할 수 있습니다.
AxisMotion이 여러 개 일 경우 Motion@A, Motion@B...Motion@F까지 확인 할 수 있습니다.

최초 1회 아래그림처럼 AxisMotion Explorer의 Configuration에서 사용할 M변수 및 P변수 범위를 설정하고, 사용할 PLC 번호를 지정한 후 "Generate"해야 합니다.
파일 저장 경로 : .
C:₩Program Files (x86)₩Delta Tau₩PMAC Executive Pro Suite₩PMAC Suite₩AxisLink

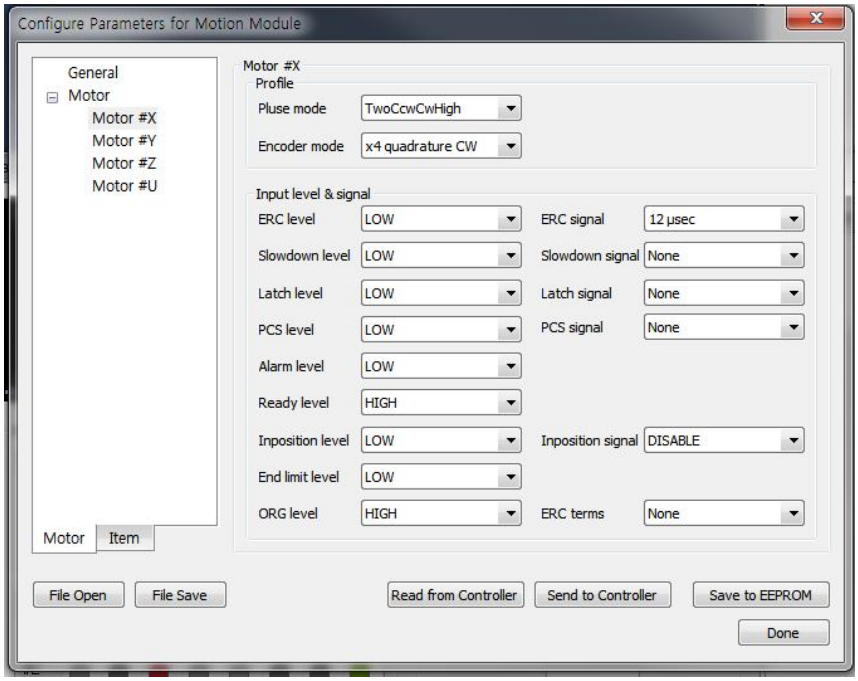






위 화면은 Motion@A를 더블클릭하면 나오는 최초 화면 입니다.
위 화면에서 현재 모터의 속도 및 위치 Flag 상태, IO 상태 속도 프로파일 등을 확인 가능 하며, JOG 및 간단한 동작의 확인이 가능 합니다.

가장 먼저 SETUP 버튼을 눌러 모터 구동 및 FLAG Active level 등을 설정 합니다.
개별 축 설정이 가능 합니다.



*Pulse Mode 설정 : Pulse 출력 방식을 설정 합니다.

Default 설정은 Two-wire CcwCwHigh 방식 입니다.

*위 설정은 AMP의 설정에 따라 변경 되어야 합니다.

When feeding in a positive direction		When feeding in a negative direction	
OUT output	DIR output	OUT output	DIR output
	High		Low
	High		Low
	Low		High
	Low		High
	High	High	
OUT DIR		OUT DIR	
OUT DIR		OUT DIR	
	Low	Low	

*ERC level : 잔여 펄스 Clear신호의 Active level 설정

*ERC signal : ERC 신호의 pulse width 설정

*ERC term : ERC 신호가 출력되는 상태 설정(Limit , Alarm , ORG ,EMG)

*SlowDown level : SD signal의 Active level 설정

*SlowDown signal : SD signal input source 설정

*Latch level : Latch signal의 Active level 설정

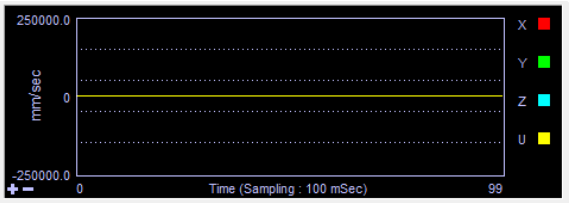
*Latch signal : Latch signal input source 설정

*Alarm , Ready , In-position , Limit , ORG level 설정

*Read from Controller : 현재 AxisLink Motion의 설정을 읽습니다.

*Send to Controller : 현재의 설정을 AxisLink Motion으로 Write 합니다. (Reset시 EEPROM의 설정을 읽어 오기 때문에 저장은 되지 않습니다.

*Save to EEPROM : 현재의 설정을 EEPROM 에 저장 합니다.



현재 Slave의 각 모터의 속도를 실시간 그래프로 표시 합니다. "+","-"버튼으로 Y축 Scale 조절이 가능 합니다.

Motor Status

Motor	Command	Velocity	Fol.Error
X	0.00 Cts	0.00 Cts/S	0.00 Cts
Y	0.00 Cts	0.00 Cts/S	0.00 Cts
Z	0.00 Cts	0.00 Cts/S	0.00 Cts
U	0.00 Cts	0.00 Cts/S	0.00 Cts

현재 Slave 의 각 위치 및 속도 FE 를 cts 단위로 표시 합니다.

Motor	RUN	ERR	ALM	RDY	INP	PEL	NEL	ORG
X								
Y								
Z								
U								

각 축의 Status 를 표기 합니다.

Universal Input / Output

Motor	In 0	In 1	In 2	Out 0	Out 1	Out 2
X						
Y						
Z						
U						

각 축의 IO Level 을 표기 합니다. 축력의 경우 클릭으로 On/Off 제어가 가능합니다.

Motor #X Setup

Move Profile

Max Speed :	1,000,000	Cts / S
Initial Speed :	0	Cts / S
Object Speed :	100,000	Cts / S
Acceleration Time :	2,000	ms
Deceleration Time :	1,000	ms
Acc S-Curve Time :	1,000	ms
Dec S-Curve Time :	1,000	ms
EEPROM Index :	1	

Read Write

각 축의 속도 프로파일을 설정 합니다.
현재 구동중인 속도 프로파일은 SETUP➔ Save to EEPROM 시 저장 됩니다.

Jog

20000 cts Jog To Kill All

Jog Minus Stop Jog Plus

Jog Incrementally Increment: 1000 cts

각 축의 기본적인 JOG 기능을 제공 합니다. 속도는 위에서 설정한 속도 프로파일로 움직입니다.

Motor Settings

Internal Rst Arrarm Reset Servo On

External Rst Error Clear

각 축의 Encoder Reset. Alarm 및 Error Clear , Servo On/Off 를 설정 합니다.

PLC 를 통한 연결 순서

AxisMotion Explorer를 사용 하지 않고, AxisMotion 과 연결 하는 방법은 아래와 같습니다.

#1. 기본 Define 된 Setup File을 Download 합니다.
#2. Save 후 Reset 을 실행 합니다. 이때, Reset 시 plc1번이 실행 되면서 Memory share할 공간의 data 값을 초기화 합니다.

COMMAND 종류

설정 및 구동을 위한 COMMAND는 크게 MAIN COMMAND와 SUB COMMAND가 있습니다.
MAIN COMMAND는 메모리 관련, 환경 설정 관련, 구동 관련으로 나뉘어져 있습니다.
메모리 관련 – MAIN CMD: 0x01
환경 설정 관련 - MAIN CMD: 0x02
구동 관련 – MAIN CMD: 0x03
SUB COMMAND는 아래와 같이 구성 되어 있습니다.

*Axis Sel- 해당 축의 BIT를 '1'로 활성화 합니다.
X – Bit0, Y – Bit1, Z – Bit2, U – Bit3

MAIN CMD 0x01 메모리 관련

MAIN CMD	SUB CMD	Description	Axis Sel	Data_0	Data_1	Data_2	Data_3
0x01	0x00	RAM Clear	-	-	-	-	-
	0x01	Write data to RAM From EEPROM(copy)	-	-	-	-	-
	0x10	Write data to EEPROM	축 1개씩 '1'	Data Type	Data Number	Data Value	-
	0x11	Write data to RAM	축 1개씩 '1'	Data Type	Data Number	Data Value	
	0x20	Write environment data to PCL from RAM	다축 '1'로 설정	X-Data Number	Y-Data Number	Z-Data Number	U-Data Number
	0x21	Write velocity profile data to PCL from RAM	다축 '1'로 설정	X-Data Number	Y-Data Number	Z-Data Number	U-Data Number
	0x22	Write operating velocity data to PCL from RAM	다축 '1'로 설정	X-Data Number	Y-Data Number	Z-Data Number	U-Data Number
	0x23	Write operating position data to PCL from RAM(상대위치)	다축 '1'로 설정	X-Data Number	Y-Data Number	Z-Data Number	U-Data Number
	0x24	Write operating position data to PCL from RAM(절대위치)	다축 '1'로 설정	X-Data Number	Y-Data Number	Z-Data Number	U-Data Number
	0x25	Write Compare Position List to PCL from RAM (RCMP1)	다축 '1'로 설정	X-Data Number	Y-Data Number	Z-Data Number	U-Data Number
	0x26	Write Compare Position List to PCL from RAM (RCMP2)	다축 '1'로 설정	X-Data Number	Y-Data Number	Z-Data Number	U-Data Number
	0x40	EEPROM Clear Command	-	-	-	-	-
	0x41	Write Data to EEPROM from RAM (copy)	-	-	-	-	-
	0x50	Read Date from EEPROM	축 1개씩 '1'	Data Type	Data Number	Data Value	-
	0x51	Read Date from RAM	축 1개씩 '1'	Data Type	Data Number	Data Value	-
	0x60	Write environment data to RAM from PCL	다축 '1'로 설정	X-Data Number	Y-Data Number	Z-Data Number	U-Data Number
	0x61	Write velocity profile data to RAM from PCL	다축 '1'로 설정	X-Data Number	Y-Data Number	Z-Data Number	U-Data Number
	0x62	Write operating velocity data to RAM from PCL	다축 '1'로 설정	X-Data Number	Y-Data Number	Z-Data Number	U-Data Number
	0x63	Write operating position data to RAM from PCL (상대위치)	다축 '1'로 설정	X-Data Number	Y-Data Number	Z-Data Number	U-Data Number
	0x64	Write operating position data to RAM from PCL (절대위치)	다축 '1'로 설정	X-Data Number	Y-Data Number	Z-Data Number	U-Data Number
	0x65	Write Compare Position List to RAM from PCL (RCMP1)	다축 '1'로 설정	X-Data Number	Y-Data Number	Z-Data Number	U-Data Number
	0x66	Write Compare Position List to RAM from PCL (RCMP2)	다축 '1'로 설정	X-Data Number	Y-Data Number	Z-Data Number	U-Data Number

MAIN CMD 0x02 환경설정 관련

MAIN CMD	SUB CMD	Description	Axis Sel	Data_0	Data_1	Data_2	Data_3
0x02	0x00	Non Command	-	-	-	-	-
	0x01	Init_PCL6143(공장초기화)	-	-	-	-	-
	0x10	Counter 1(Internal Counter) 쓰기	다축 '1'로 설정	X-Internal CNT Data	Y-Internal CNT Data	Z-Internal CNT Data	U-Internal CNT Data
	0x11	Counter 2(External Counter) 쓰기	다축 '1'로 설정	X-External CNT Data	Y-External CNT Data	Z-External CNT Data	U-External CNT Data
	0x40	Read Main Status and Sub Status	축 1개씩 '1'	-	-	-	-
	0x41	Read Error Status	축 1개씩 '1'	-	-	-	-
	0x42	Read Error Code	축 1개씩 '1'	-	-	-	-
	0x80	MAX Speed 설정 쓰기	축 1개씩 '1'	Max Speed Data(cts/s)			
	0x81	속도 Profile 설정 쓰기 (Max speed 비교)	축 1개씩 '1'	초기속도 (24bit)	목표속도 (24bit)	S커브시간/가속 시간 (각각16bit)	S커브시간/감속 시간 (각각16bit)
	0x86	환경설정1 쓰기 (From Host)	다축 '1'로 설정	X-RENV1 data	Y-RENV1 data	Z-RENV1 data	U-RENV1 data
	0x87	환경설정2 쓰기 (From Host)	다축 '1'로 설정	X-RENV2 data	Y-RENV2 data	Z-RENV2 data	U-RENV2 data
	0x88	환경설정3 쓰기 (From Host)	다축 '1'로 설정	X-RENV3 data	Y-RENV3 data	Z-RENV3 data	U-RENV3 data
	0x89	환경설정4 쓰기 (From Host)	다축 '1'로 설정	X-RENV4 data	Y-RENV4 data	Z-RENV4 data	U-RENV4 data
	0x8A	환경설정1,2,3,4 쓰기 (From Host)	축 1개씩 '1'	RENV1 data	RENV2 data	RENV3 data	RENV4 data
	0xC0	MAX Speed 설정 읽기	축 1개씩 '1'	-	-	-	-
	0xC1	현재 프로파일 속도 데이터1 읽기	축 1개씩 '1'	-	-	-	-
	0xC2	현재 프로파일 속도 데이터2 읽기	축 1개씩 '1'	-	-	-	-
	0xC3	현재 프로파일 가속도 데이터 읽기	축 1개씩 '1'	-	-	-	-
	0xC4	현재 프로파일 감속도 데이터 읽기	축 1개씩 '1'	-	-	-	-
	0xC5	현재 속도 읽기	축 1개씩 '1'	-	-	-	-
	0xC6	환경설정1 읽기	축 1개씩 '1'	-	-	-	-
	0xC7	환경설정2 읽기	축 1개씩 '1'	-	-	-	-
	0xC8	환경설정3 읽기	축 1개씩 '1'	-	-	-	-
	0xC9	환경설정4 읽기	축 1개씩 '1'	-	-	-	-

MAIN CMD 0x03 동작 관련

MAIN CMD	SUB CMD	Description	Axis Sel	Data_0	Data_1	Data_2	Data_3
0x03	0x01	다축 상대 좌표 설정	다축 '1'로 설정	X-상대위치값 (28bits)	Y-상대위치값 (28bits)	Z-상대위치값 (28bits)	U-상대위치값 (28bits)
	0x02	다축 절대 좌표 설정	다축 '1'로 설정	X-절대위치값 (28bits)	Y-절대위치값 (28bits)	Z-절대위치값 (28bits)	U-절대위치값 (28bits)
	0x20	Servo ON 명령	다축 '1'로 설정	-	-	-	-
	0x21	Servo Off 명령	다축 '1'로 설정	-	-	-	-
	0x22	Alarm Reset 명령	다축 '1'로 설정	-	-	-	-
	0x30	범용출력0 Set 명령	축 1개씩 '1'	-	-	-	-
	0x31	범용출력0 Reset 명령	축 1개씩 '1'	-	-	-	-
	0x32	범용출력1 Set 명령	축 1개씩 '1'	-	-	-	-
	0x33	범용출력1 Reset 명령	축 1개씩 '1'	-	-	-	-
	0x34	범용출력2 Set 명령	축 1개씩 '1'	-	-	-	-
	0x35	범용출력2 Reset 명령	축 1개씩 '1'	-	-	-	-
	0x41	다축 구동 명령	다축 '1'로 설정	-	-	-	-
	0x42	다축 상대위치결정 구동 명령	다축 '1'로 설정	X-상대위치값 (28bits)	Y-상대위치값 (28bits)	Z-상대위치값 (28bits)	U-상대위치값 (28bits)
	0x43	다축 절대위치결정 구동 명령	다축 '1'로 설정	X-절대위치값 (28bits)	Y-절대위치값 (28bits)	Z-절대위치값 (28bits)	U-절대위치값 (28bits)
	0x44	비상정지 명령	모든축	-	-	-	-
	0x45	다축 정속 구동 명령	다축 '1'로 설정	X-방향 0: positive 1: negative	Y-방향 0: positive 1: negative	Z-방향 0: positive 1: negative	U-방향 0: positive 1: negative
	0x49	다축 급정지 명령	다축 '1'로 설정	-	-	-	-
	0x4A	다축 감속정지 명령	다축 '1'로 설정	-	-	-	-
	0x60	HOME	축 1개씩 '1'	Home Condition	Home Flag Select	Home 구동 속도(방향)	Home Offset

데이터 설정				
데이터 종류	데이터 번호 범위	데이터 정의		
0x100	0 ~ 31	RENV1 (PCL6143)		
0x101	0 ~ 31	RENV2 (PCL6143)		
0x102	0 ~ 31	RENV3 (PCL6143)		
0x103	0 ~ 31	RENV4 (FPGA)		
0x200	0 ~ 63	속도프로파일 위치 데이터(28bits)		cts
0x201	0 ~ 63	속도프로파일 최고 속도 (24bits)		cts/s
0x202	0 ~ 63	속도프로파일 초기 속도 (24bits)		cts/s
0x203	0 ~ 63	속도프로파일 목표 속도 (24bits)		cts/s
0x204	0 ~ 63	속도프로파일 가속도정보 (32bits)		가속시 Scurve 시간(msec, 16b)+ 가속시간(msec, 16b)
0x205	0 ~ 63	속도프로파일 감속도정보 (32bits)		감속시 Scurve 시간(msec, 16b)+ 감속시간(msec, 16b)
0x206	0 ~ 63	속도프로파일 구동모드(30bits)		
0x300	0 ~ 255	구동속도 리스트(24bits)		cts/s
0x301	0 ~ 511	구동좌표 리스트(28bits)		cts
0x302	0 ~ 511	비교좌표 리스트(28bits)		cts

기본적인 Command Code Sample

전체적인 CODE는 아래의 Command Code의 조합으로 이루어 집니다.
Command의 종류에 따라 Data0..3의 값을 필요로 합니다. XX는 각 SLAVE 명칭 입니다.

P_B00_xx_CMD_CODE : 원하는 Command 명령 값 기입
Main CMD와 Sub CMD를 함께 적습니다. (ex: Main :0x01 Sub:0x01 ➔ 0x101)
P_B00_xx_CMD_DATA0 : Command에 따른 Data값 기입
P_B00_xx_CMD_DATA1 : Command에 따른 Data값 기입
P_B00_xx_CMD_DATA2 : Command에 따른 Data값 기입
P_B00_xx_CMD_DATA3 : Command에 따른 Data값 기입
P_B00_xx_MOTOR_SEL : 적용을 원하는 축 Bit '1'로 설정
P_B00_xx_CMD_RUN : 위 command를 실행하는 변수 입니다. Slave 별로 아래와 같습니다.
**
Motion@A : P_B00_A_CMD_RUN
Motion@B : P_B00_B_CMD_RUN
Motion@C : P_B00_C_CMD_RUN
Motion@D : P_B00_D_CMD_RUN
Motion@E : P_B00_E_CMD_RUN
Motion@F : P_B00_F_CMD_RUN

COMMAND 예제

```
P_B00_A_CMD_CODE = $320      // COMMAND 번호 ex)$320 : Servo On
P_B00_A_CMD_DATA0 = 0        // COMMAND에 해당하는 Data0의 값
P_B00_A_CMD_DATA1 = 0        // COMMAND에 해당하는 Data1의 값
P_B00_A_CMD_DATA2 = 0        // COMMAND에 해당하는 Data2의 값
P_B00_A_CMD_DATA3 = 0        // COMMAND에 해당하는 Data3의 값
P_B00_A_MOTOR_SEL = $F           // COMMAND에 해당하는 축 선택
P_B00_A_CMD_RUN =1          // Motion@A
        While(P_B00_A_CMD_RUN!=0)       // 명령이 끝날 때 까지 대기 합니다.
        ENDWHILE

P_B00_A_CMD_CODE = $342      // 다축 상대좌표 설정 & 동작
P_B00_A_CMD_DATA0 = 1000      // 축별로 다른  POSITION
P_B00_A_CMD_DATA1 = 10000
P_B00_A_CMD_DATA2 = 20000
P_B00_A_CMD_DATA3 = 40000
P_B00_A_MOTOR_SEL = $F
P_B00_A_CMD_RUN =1
        While(P_B00_A_CMD_RUN!=0)
        ENDWHILE
```

STATE 확인하기

Status는 크게 3가지로 구분 할 수 있습니다.

1. Main Status – 구동 및 Interrupt 등에 대한 Status
2. Sub Status – 각 축의 상태에 대한 Status
3. Axis Status – Main 및 Sub Status 조합 및 FPGA를 통한 User 확인 용 Status

Main Status

bit	Bit Name	Details
0	SSCM	Start Commnad 기입하면 1 이 되고, 동작정지에서 0 이 됩니다.
1	SRUN	Pulse 출력을 개시에서 1 이 되고 동작정지에서 0 이 됩니다
2	SENI	정지 Interrupt flag RENV2 의 IEND=1 의 때에 동작 중→정지로의 변화에 의해 INT 출력을 ON 으로 하고 이 bit 가 1 이 됩니다.(main status 의 읽기 후, 0 으로 되돌아 갑니다.) 또한 IEND=0 의 때는 늘 0 이 됩니다.
3	SEND	Start Command 를 기입 하면 0 이 되고, 동작 정지에서 1 이 됩니다
4	SERR	Error Interrupt 발생에 의해 1 이 되고, REST 읽기에서 0 이 됩니다.
5	SINT	Event Interrupt 발생에 의해 1 이 되고, RIST 읽기에서 0 이 됩니다.
6~7	SSC0 ~ 1	실행 중 또는 정지 때의 Sequence 번호 입니다.
8	SCP1	Comparator1 의 비교 조건 성립 때에 1 이 됩니다.
9	SCP2	Comparator2 의 비교 조건 성립 때에 1 이 됩니다.
10~12	-	-
13	SEOR	위치의 override 가 실행 할 수 없을 때(정지 상태에서의 RMV Register 기입 때)에 1 이 되고, main status 의 읽기 후에 0 으로 되돌아갑니다.
14	SPRF	다음 동작 data 용 pre-Register 가 full 때에 1 이 됩니다
15	-	-

Sub Status

bit	Bit Name	Details
0 to 7	UIO0~7	GPIO 상태
8	SFU	가속중에 1 이 됩니다.
9	SFD	감속중에 1 이 됩니다
10	SFC	정속중에 1 이 됩니다
11	SALM	ALM 입력 ON 때에 1 이 됩니다
12	SPEL	+EL 입력 ON 때에 1 이 됩니다
13	SMEL	-EL 입력 ON 때에 1 이 됩니다
14	SORG	ORG 입력 ON 때에 1 이 됩니다
15	SSD	SD 입력 ON 때에 1 이 됩니다. (SD 입력의 Latch 신호)

Axis Status

bit	Bit Name	Details	
0	**SRUN**	**펄스 출력중**	Main Status 1bit
1	**SEND**	**펄스 출력종료**	Main Status 3bit
2	**SERR**	**에러 발생시 1 로 됨 상태 읽어야 해지**	Main Status 4bit
3	**SALM**	**알람 입력시**	Sub Status 11bit
4	**SRDY**	**Servo Drive Ready 신호 입력상태**	Sub Status 5bit (IOP5)
5	**SVON**	**SERVO ON 출력 신호 상태**	Sub Status 6bit (IOP6)
6	**SINP**	**INP 입력상태**	RSTS 15bit
7	**SPEL**	**양방향 리미트시**	Sub Status 12bit
8	**SMEL**	**음방향 리미트시**	Sub Status 13bit
9	**SORG**	**원점 입력**	Sub Status 14bit
10	**UI0**	**범용입력 0**	FPGA
11	**UI1**	**범용입력 1**	FPGA
12	**UI2**	**범용입력 2**	FPGA
13	**UO0**	**범용출력 0**	Sub Status 0bit (IOP0)
14	**UO1**	**범용출력 1**	Sub Status 1bit (IOP1)
15	**UO2**	**범용출력 2**	Sub Status 2bit (IOP2)

Main & Sub Status 확인 CMD

Main CMD : 0x02 Sub CMD : 0x40 ➔ Read Main Status and Sub Status

Response Data ➔ Main STS(Bit 32~16) + SUB STS(Bit 15~0)

P_B00_A_RCMD_DATA0 로 해당 Data 값이 Return 됩니다.

예제)

```
P_B00_A_CMD_CODE = $240   //Read Main Status and Sub Status
P_B00_A_CMD_DATA0 = $0
P_B00_A_CMD_DATA1 = $0
P_B00_A_CMD_DATA2 = $0
P_B00_A_CMD_DATA3 = $0
P_B00_A_MOTOR_SEL = $1      //  해당 축 1로 설정 // X축에 대한
P_B00_A_CMD_RUN =1
        While(P_B00_A_CMD_RUN!=0)
        ENDWHILE
P_B00_A_RCMD_DATA0_TEMP = P_B00_A_RCMD_DATA0    //Return 값 TEMP 저장
```

Axis Status Bit Masking 예제

(Axis Status는 별도의 CMD 없이 PLC에서 주기적으로 업데이트 됩니다)

본 예제는 여러 축 이동 명령 후 모든 축이 이동이 끝날 때까지 기다리는 예제 입니다. 최초 이동 명령 후 모든 축이 이동을 시작하는지 우선 확인 후, 모든 축이 In-position 및 Pulse 출력 종료 되었는지 확인 후 다음 명령으로 넘어가게 됩니다.

// P_B00_A_X_STS (Bit 0) ➔ SALVE@A_X축 펄스 출력 중 (정지 상태: 0, 펄스 출력 중: 1)
// P_B00_A_X_STS (Bit 6) ➔ SALVE@A_X축 In-position (이동 완료: 0, 이동 중: 1)
// P_B00_A_Y_STS (Bit 0) ➔ SALVE@A_Y축 펄스 출력 중 (정지 상태: 0, 펄스 출력 중: 1)
// P_B00_A_Y_STS (Bit 6) ➔ SALVE@A_Y축 In-position (이동 완료: 0, 이동 중: 1)
// P_B00_A_Z_STS (Bit 0) ➔ SALVE@A_Z축 펄스 출력 중 (정지 상태: 0, 펄스 출력 중: 1)
// P_B00_A_Z_STS (Bit 6) ➔ SALVE@A_Z축 In-position (이동 완료: 0, 이동 중: 1)
// P_B00_A_U_STS (Bit 0) ➔ SALVE@A_U축 펄스 출력 중 (정지 상태: 0, 펄스 출력 중: 1)
// P_B00_A_U_STS (Bit 6) ➔ SALVE@A_U축 In-position (이동 완료: 0, 이동 중: 1)

```
P_B00_A_CMD_CODE = $342   // 다축 상대좌표 설정
P_B00_A_CMD_DATA0   = 1000                  // 축별로 다른 POSITION
P_B00_A_CMD_DATA1   = 10000
P_B00_A_CMD_DATA2   = 20000
P_B00_A_CMD_DATA3   = 40000
P_B00_A_MOTOR_SEL = $F            // X,Y,Z,U 축 선택
P_B00_A_CMD_RUN =1
            While(P_B00_A_CMD_RUN!=0)
            ENDWHILE                  // 구동 Command

WHILE(P_B00_A_X_STS & $1 =0 OR P_B00_A_Y_STS & $1 =0 OR P_B00_A_Z_STS & $1 =0 OR P_B00_A_U_STS & $1
=0)
ENDWHILE   // 이동 시작 확인

WHILE(P_B00_A_X_STS & $1 !=0 OR P_B00_A_Y_STS & $41 !=0 OR P_B00_A_Z_STS & $41 !=0 OR P_B00_A_U_STS
& $41 !=0 OR
P_B00_A_X_STS & $40 =0 OR P_B00_A_Y_STS & $40 =0 OR P_B00_A_Z_STS & $40 =0 OR P_B00_A_U_STS & $40
=0)
ENDWHILE   // INP 및 PULSE 출력상태 BIT 검사 // 이동 완료 확인
```

GPIO ON/OFF 및 입력에 따른 조건 설정

AxisLink-Motion 보드에는 각 축마다 각각 3개의 입력 및 출력이 있습니다.
입력 상태 및 출력 상태는 각 축의 Axis Status에서 확인 가능 합니다.(변수 : P_B00_A_X_STS)
입력 상태 확인

P_B00_A_X_STS (Bit10 ~ Bit15)

Bit 10 : 입력 0 Bit 11 : 입력 1 Bit 12 : 입력 2

Bit 13 : 출력 0 Bit 14 : 출력 1 Bit 15 : 출력 2

출력 Command

Main CMD : 0x03 Sub CMD : 0x30 ➔ 범용출력 0 SET

Sub CMD : 0x31 ➔ 범용출력 0 RESET

Sub CMD : 0x32 ➔ 범용출력 1 SET

Sub CMD : 0x33 ➔ 범용출력 1 RESET

Sub CMD : 0x34 ➔ 범용출력 2 SET

Sub CMD : 0x35 ➔ 범용출력 2 RESET

각 보드 구분 및 출력 Axis를 구분하여 CMD를 사용 하시면 됩니다.

Motion@A : P_B00_A_CMD_RUN

Motion@B : P_B00_B_CMD_RUN

Motion@C : P_B00_C_CMD_RUN

Motion@D : P_B00_D_CMD_RUN

Motion@E : P_B00_E_CMD_RUN

Motion@F : P_B00_F_CMD_RUN

P_B00_A_MOTOR_SEL = $1 or $2 or $4 or $8

```
IF(P_B00_A_X_STS_TEMP != P_B00_A_X_STS) // 상태 변경 감시

        P_B00_A_X_STS_TEMP = P_B00_A_X_STS     // 상태 TEMP 값 최신화

        IF(P_B00_A_X_STS_TEMP & $400 !=0)              //범용 입력 UI0 On 시

                P_B00_A_CMD_CODE = $330  //범용 출력 UO0 CLOSE
                P_B00_A_CMD_DATA0 = 0
                P_B00_A_CMD_DATA1 = 0
                P_B00_A_CMD_DATA2 = 0
                P_B00_A_CMD_DATA3 = 0
                P_B00_A_MOTOR_SEL = $1              // 출력 축 선택 (X축)
                P_B00_A_CMD_RUN =1          // CMD Device 선택(Motion@A)
                        While(P_B00_A_CMD_RUN!=0)
```

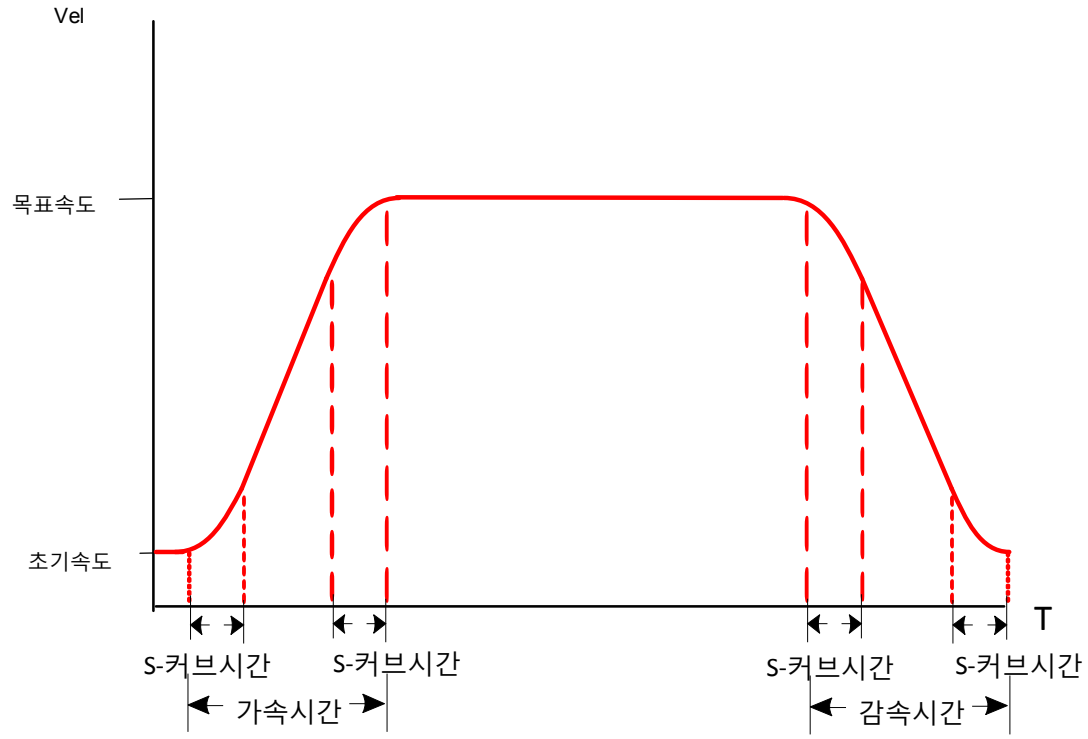
```
                    ENDWHILE
                P_B00_A_CMD_CODE = $332   //범용 출력 UO1 CLOSE
                P_B00_A_CMD_DATA0 = 0
                P_B00_A_CMD_DATA1 = 0
                P_B00_A_CMD_DATA2 = 0
                P_B00_A_CMD_DATA3 = 0
                P_B00_A_MOTOR_SEL = $1
                P_B00_A_CMD_RUN =1
                    While(P_B00_A_CMD_RUN!=0)
                    ENDWHILE

        ELSE                                        //범용 입력 UI0 Off 시
                P_B00_A_CMD_CODE   = $331   //범용 출력 UO0 OPEN
                P_B00_A_CMD_DATA0 = 0
                P_B00_A_CMD_DATA1 = 0
                P_B00_A_CMD_DATA2 = 0
                P_B00_A_CMD_DATA3 = 0
                P_B00_A_MOTOR_SEL = $1
                P_B00_A_CMD_RUN =1
                    While(P_B00_A_CMD_RUN!=0)
                    ENDWHILE
                P_B00_A_CMD_CODE = $333   //범용 출력 UO1 OPEN
                P_B00_A_CMD_DATA0 = 0
                P_B00_A_CMD_DATA1 = 0
                P_B00_A_CMD_DATA2 = 0
                P_B00_A_CMD_DATA3 = 0
                P_B00_A_MOTOR_SEL = $1
                P_B00_A_CMD_RUN =1
                    While(P_B00_A_CMD_RUN!=0)
                    ENDWHILE
        ENDIF
ENDIF
```

모터 구동 관련 COMMAND

모터 구동은 크게 3가지로 구분이 됩니다.

1. 정속 구동
2. 상대좌표 이동
3. 절대 좌표 이동

모터 구동 관련된 Command는 Main CMD : 0x03 입니다.
모터를 구동 하기 위해서는 기본적으로 속도 프로파일의 값이 모두 유효한 값이 있어야 구동이 가능 합니다.
기본적으로 속도 프로파일의 값은 미리 설정해 두고 사용 합니다.



MAX SPEED의 설정에 따른 Resolution 변화

Max Speed와 Initial Speed 사이에서 총 14bit의 속도 지령 값을 내릴 수 있습니다.
다시 말해 Max와 Initial 사이를 16383개로 나누어 속도 조절이 가능 합니다.
예를 들어 낮은 Max Speed를 설정 할 경우 높은 Max Speed보다 좀더 세밀하게 속도를 조정 할 수 있습니다. 따라서 장비의 최대 사용 속도를 Max Speed로 설정하는 것이 가장 효율적 입니다.
Max Speed를 구동 중에 변경 한다면 반드시 속도 프로파일을 다시 Write 하여야 합니다.
AxisLink Explorer가 아닌 CMD를 통하여 Max Speed를 수정한 경우에도(0x280) 반드시 프로파일 설정 쓰기(0x281)을 다시 수행 하여야 합니다.

초기 속도와 목표 속도 그리고 최대속도에 따른 최대 가감속 시간의 변화

$$최대\ 가감속\ 시간(s) = \frac{(목표속도 - 초기속도) * (가속관련\ Reg+1) * 2}{19660800}$$ (S커브 가감속 없는 경우)

$$최대\ 가감속\ 시간(s) = \frac{(목표속도 - 초기속도 + 2 * S커브관련Reg) * (가속관련\ Reg+1) * 2}{19660800}$$ (S커브 있는 경우)

위 수식을 보면 목표 속도와 초기 속도의 차이에 따라서 가질 수 있는 최대 가감속 시간이 달라집니다. 이때 최대 속도와 목표 속도의 비율 역시 관련이 있습니다.
복잡한 계산을 피하기 위하여 가감속 값이 MAX를 넘는 경우 최대 가감속 값으로 자동 적용 됩니다.

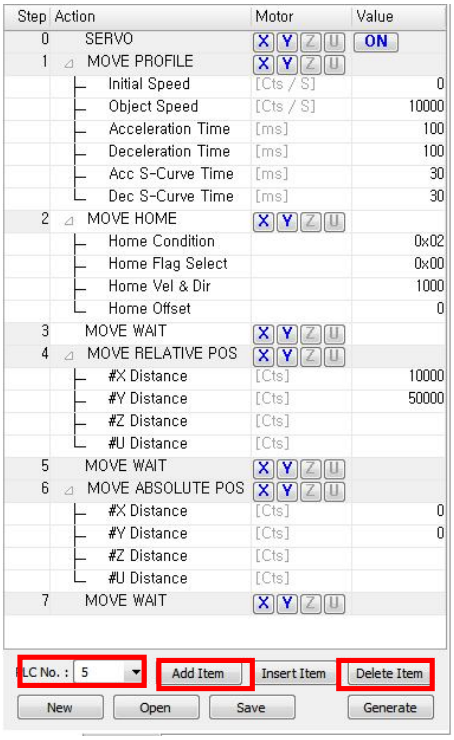
AxisLink Explorer를 통한 모션 PLC 생성

아래 그림처럼 모션CMD를 순서대로 생성하여 사용 할 수 있습니다.

제일 먼저 생성하고자 하는 PLC 번호를 선택하고 "Add Item"으로 명령을 추가 할 수 있습니다. 선택한 CMD에서 Motor 선택 및 해당 항목을 기입하여 사용 가능 합니다.

**ACTION

Servo ON /OFF - Servo On/Off
MOVE PROFILE - 속도 프로파일 설정
MOVE HOME - HOME
MOVE RELATIVE - 설정된 POS로 이동
MOVE RELATIVE POS - 상대 POS 설정 및 이동
MOVE ABSOLUT POS - 절대 POS 설정 및 이동
MOVE CONTINUOUS - 연속 구동
MOVE WAIT - 이동 완료 대기
EMERGENCY STOP - 긴급 정지
SLOWDOWN STOP - 감속 정지
SET RELATIVE POS - 상대 좌표 입력
SET ABSOLUTE POS - 절대 좌표 입력
UNIVERSAL OUTPUT - GPIO 설정

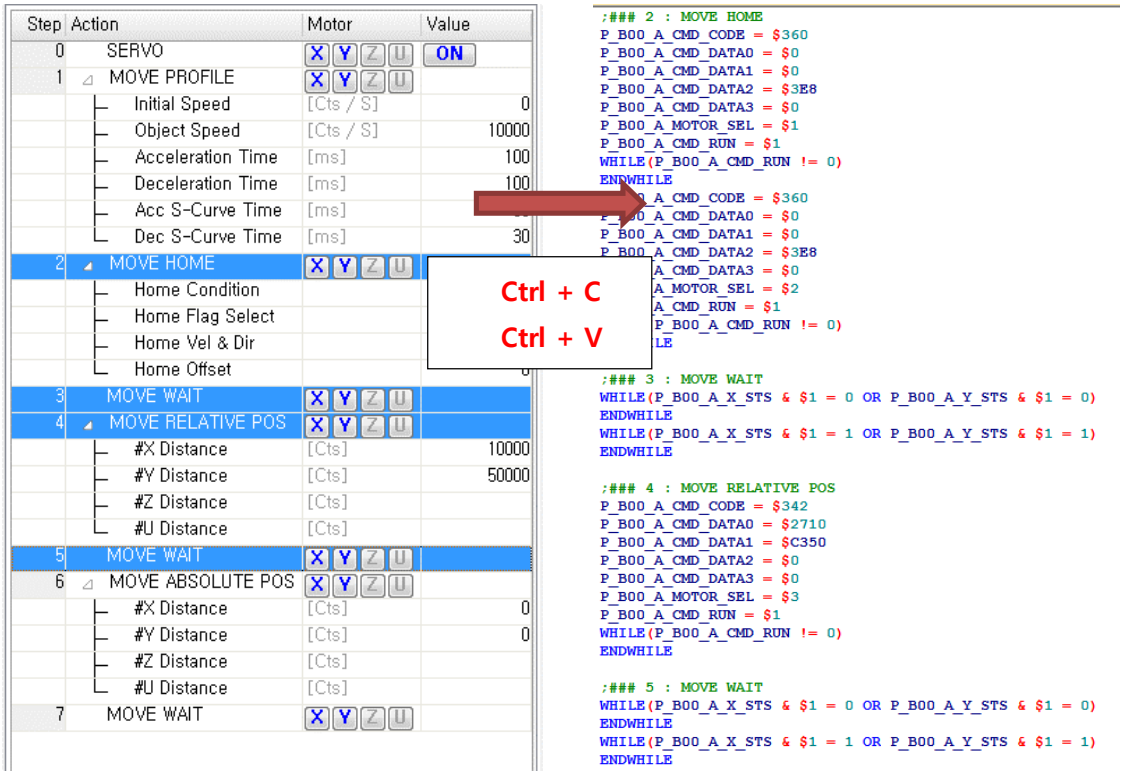


0) Servo On (X축,Y축)
1) X,Y축 속도 프로파일 설정
2) X,Y HOME
3) X,Y 완료 대기
4) X,Y 10000,50000 상대좌표이동
5) X,Y 완료 대기
6) X,Y 절대좌표 0 으로 이동
7) X,Y 완료 대기
종료

"Generate"버튼을 누르면 PLC가 생성 됩니다.

원하는 STEP 을 클릭하고(여러 개 가능) Ctrl +C 로 복사 후 새 파일에 Ctrl +V 붙여 넣기를 하면

원하는 STEP 의 CODE 를 바로 불러 올 수 있습니다.



생성 파일)

```
CLOSE
END GAT
DEL GAT

#include "C:₩Program Files (x86)₩Delta Tau₩PMAC Executive Pro Suite₩PMAC Suite₩AxisLink₩AxisLink_Master_00.pmc"

OPEN PLC 2 CLEAR

        WHILE(P_B00_A_CMD_RUN != 0)
        ENDWHILE

        ;### 0 : SERVO      // Servo ON
        P_B00_A_CMD_CODE = $320
        P_B00_A_CMD_DATA0 = $0
        P_B00_A_CMD_DATA1 = $0
        P_B00_A_CMD_DATA2 = $0
        P_B00_A_CMD_DATA3 = $0
        P_B00_A_MOTOR_SEL = $3
        P_B00_A_CMD_RUN = $1
        WHILE(P_B00_A_CMD_RUN != 0)
        ENDWHILE
```

```
;### 1 : MOVE PROFILE     // 속도 프로파일 설정
P_B00_A_CMD_CODE = $281
P_B00_A_CMD_DATA0 = $0
P_B00_A_CMD_DATA1 = $2710
P_B00_A_CMD_DATA2 = $140064
P_B00_A_CMD_DATA3 = $140064
P_B00_A_MOTOR_SEL = $1
P_B00_A_CMD_RUN = $1
WHILE(P_B00_A_CMD_RUN != 0)
ENDWHILE
P_B00_A_CMD_CODE = $281
P_B00_A_CMD_DATA0 = $0
P_B00_A_CMD_DATA1 = $2710
P_B00_A_CMD_DATA2 = $140064
P_B00_A_CMD_DATA3 = $140064
P_B00_A_MOTOR_SEL = $2
P_B00_A_CMD_RUN = $1
WHILE(P_B00_A_CMD_RUN != 0)
ENDWHILE

;### 2 : MOVE HOME    // HOME
P_B00_A_CMD_CODE = $360
P_B00_A_CMD_DATA0 = $0
P_B00_A_CMD_DATA1 = $0
P_B00_A_CMD_DATA2 = $3E8
P_B00_A_CMD_DATA3 = $0
P_B00_A_MOTOR_SEL = $1
P_B00_A_CMD_RUN = $1
WHILE(P_B00_A_CMD_RUN != 0)
ENDWHILE
P_B00_A_CMD_CODE = $360
P_B00_A_CMD_DATA0 = $0
P_B00_A_CMD_DATA1 = $0
P_B00_A_CMD_DATA2 = $3E8
P_B00_A_CMD_DATA3 = $0
P_B00_A_MOTOR_SEL = $2
P_B00_A_CMD_RUN = $1
WHILE(P_B00_A_CMD_RUN != 0)
ENDWHILE

;### 3 : MOVE WAIT     // X, Y 완료 대기
WHILE(P_B00_A_X_STS & $1 = 0 OR P_B00_A_Y_STS & $1 = 0)
ENDWHILE  // 출발 대기

WHILE(P_B00_A_X_STS & $40= 0 OR P_B00_A_Y_STS & $40= 0 OR P_B00_A_X_STS & $1= 1 OR P_B00_A_Y_STS & $1= 1)
ENDWHILE  // 완료 대기( In-position bit 및 Pulse 출력 검사)

WHILE(P_B00_A_CMD_RUN != 0)
ENDWHILE
```

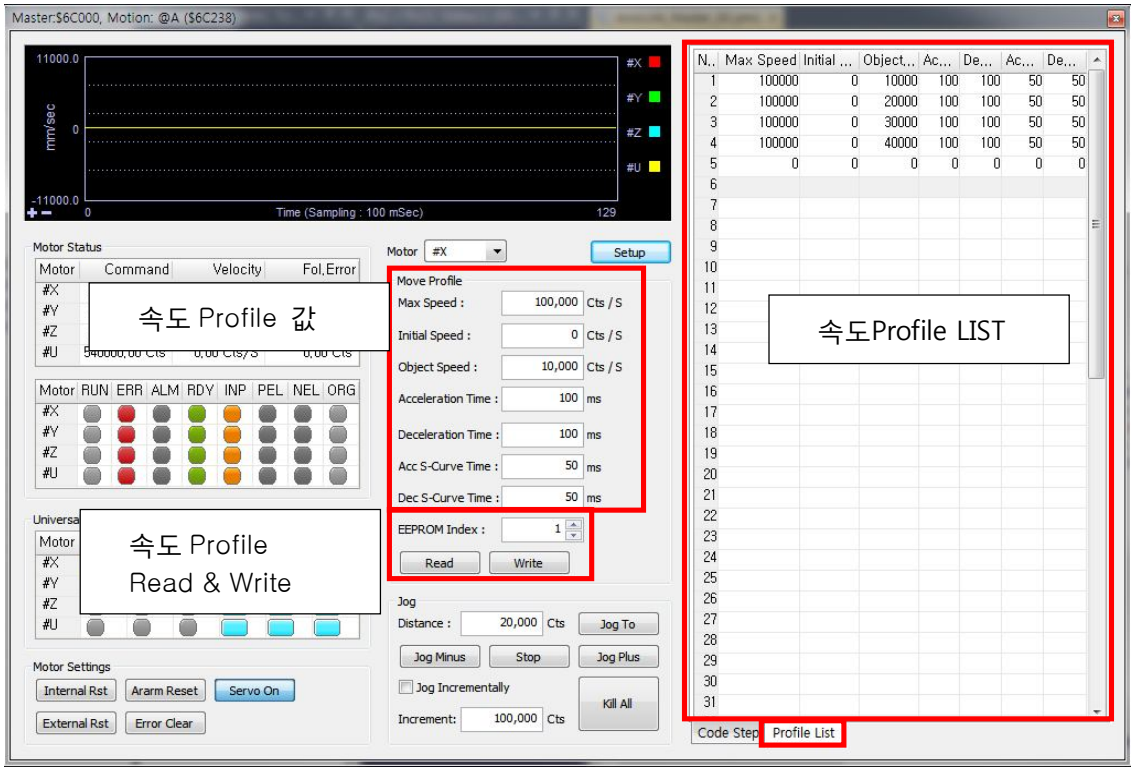
```
            ;### 4 : MOVE RELATIVE POS      // X, Y 상대좌표 이동
            P_B00_A_CMD_CODE = $342
            P_B00_A_CMD_DATA0 = $2710
            P_B00_A_CMD_DATA1 = $C350
            P_B00_A_CMD_DATA2 = $0
            P_B00_A_CMD_DATA3 = $0
            P_B00_A_MOTOR_SEL = $3
            P_B00_A_CMD_RUN = $1
            WHILE(P_B00_A_CMD_RUN != 0)
            ENDWHILE

            ;### 5 : MOVE WAIT       // X, Y 완료 대기
            WHILE(P_B00_A_X_STS & $1 = 0 OR P_B00_A_Y_STS & $1 = 0)
            ENDWHILE
            WHILE(P_B00_A_X_STS & $1 = 1 OR P_B00_A_Y_STS & $1 = 1)
            ENDWHILE

            WHILE(P_B00_A_CMD_RUN != 0)
            ENDWHILE

            ;### 6 : MOVE ABSOLUTE POS      // X, Y 절대좌표 이동
            P_B00_A_CMD_CODE = $343
            P_B00_A_CMD_DATA0 = $0
            P_B00_A_CMD_DATA1 = $0
            P_B00_A_CMD_DATA2 = $0
            P_B00_A_CMD_DATA3 = $0
            P_B00_A_MOTOR_SEL = $3
            P_B00_A_CMD_RUN = $1
            WHILE(P_B00_A_CMD_RUN != 0)
            ENDWHILE

            ;### 7 : MOVE WAIT       // X, Y 완료 대기
            WHILE(P_B00_A_X_STS & $1 = 0 OR P_B00_A_Y_STS & $1 = 0)
            ENDWHILE
            WHILE(P_B00_A_X_STS & $1 = 1 OR P_B00_A_Y_STS & $1 = 1)
            ENDWHILE

            WHILE(P_B00_A_CMD_RUN != 0)
            ENDWHILE
DISABLE PLC 2
CLOSE
```

AxisLink Explorer를 통한 모션 속도 프로파일 설정

AxisLink Motion은 각 축당 최대 63개의 속도 Profile data를 저장하여 필요 시 이를 읽어와서 사용이 가능 합니다.

속도 프로파일은 아래 화면에서 볼 수 있듯이 쉽게 저장이 가능 합니다.

참고로 속도 프로파일은 1번부터 가능 하며 0번은 현재 가지고 있는 속도 프로파일입니다.



PLC에서 속도 프로파일 변경 하기 예제

```
P_B00_A_CMD_CODE = $121
P_B00_A_CMD_DATA0 = X축 읽어올 프로파일 번호
P_B00_A_CMD_DATA1 = Y축 읽어올 프로파일 번호
P_B00_A_CMD_DATA2 = Z축 읽어올 프로파일 번호
P_B00_A_CMD_DATA3 = U축 읽어올 프로파일 번호
P_B00_A_MOTOR_SEL = 읽어 오고자 하는 축 설정 (ex. Y축,Z축 ➔ $6 )
P_B00_A_CMD_RUN = $1
    WHILE(P_B00_A_CMD_RUN != 0)
    ENDWHILE
```

전체 구동 예제

구동 예제)

```
P_B00_A_CMD_CODE = $320   //SERVO ON
P_B00_ A_CMD_DATA0 = 0
P_B00_ A_CMD_DATA1 = 0
P_B00_ A_CMD_DATA2 = 0
P_B00_ A_CMD_DATA3 = 0
P_B00_A_MOTOR_SEL = $F
            P_B00_A_CMD_RUN =1
            While(P_B00_A_CMD_RUN!=0)
            ENDWHILE

P_B00_ A_CMD_CODE = $345 // 다축 정속구동 명령
P_B00_ A_CMD_DATA0 = 0                    // X-Positive 방향 정속구동
P_B00_ A_CMD_DATA1 = 1                    // Y-Negative 방향 정속구동
P_B00_ A_CMD_DATA2 = 0                    // Z-Positive 방향 정속구동
P_B00_ A_CMD_DATA3 = 1                    // U-Negative 방향 정속구동
P_B00_A_MOTOR_SEL = $F
            P_B00_A_CMD_RUN =1
            I5112=5000*8388608/I10    //5 초간 정속 구동 유지
            While(P_B00_A_CMD_RUN!=0 or I5112>0)
            ENDWHILE

P_B00_ A_CMD_CODE = $34A       // 다축 감속 정지 명령
P_B00_ A_CMD_DATA0 = 0
P_B00_ A_CMD_DATA1 = 0
P_B00_ A_CMD_DATA2 = 0
P_B00_ A_CMD_DATA3 = 0
P_B00_A_MOTOR_SEL = $F              // 모든 축 감속 정지
            P_B00_A_CMD_RUN =1
            While(P_B00_A_CMD_RUN!=0 )
            ENDWHILE

P_B00_ A_CMD_CODE = $210             //Write INTERNAL CNT
P_B00_ A_CMD_DATA0 = 0                  //X축 내부 카운터 0의 값 쓰기
P_B00_ A_CMD_DATA1 = 0                  //Y축 내부 카운터 0의 값 쓰기
P_B00_ A_CMD_DATA2 = 0                  //Z축 내부 카운터 0의 값 쓰기
P_B00_ A_CMD_DATA3 = 0                  //U축 내부 카운터 0의 값 쓰기
P_B00_A_MOTOR_SEL = $F
            P_B00_A_CMD_RUN =1
            While(P_B00_A_CMD_RUN!=0)
            ENDWHILE
```

```
P_B00_ A_CMD_CODE = $211            //Write EXTERNAL CNT
P_B00_ A_CMD_DATA0 = 0                  //X축 외부 카운터 0의 값 쓰기
P_B00_ A_CMD_DATA1 = 0                  //Y축 외부 카운터 0의 값 쓰기
P_B00_ A_CMD_DATA2 = 0                  //Z축 외부 카운터 0의 값 쓰기
P_B00_ A_CMD_DATA3 = 0                  //U축 외부 카운터 0의 값 쓰기
P_B00_A_MOTOR_SEL = $F
        P_B00_A_CMD_RUN =1
        While(P_B00_A_CMD_RUN!=0)
        ENDWHILE

P_B00_ A_CMD_CODE = $342  // 다축 상대좌표 설정 및 구동 명령
P_B00_ A_CMD_DATA0 = 1000                  // 축별로 다른  POSITION
P_B00_ A_CMD_DATA1 = 10000
P_B00_ A_CMD_DATA2 = 20000
P_B00_ A_CMD_DATA3 = 40000
P_B00_A_MOTOR_SEL = $F
        P_B00_A_CMD_RUN =1
        While(P_B00_A_CMD_RUN!=0)
        ENDWHILE

WHILE(P_B00_A_X_STS & $1 =0 OR P_B00_A_Y_STS & $1 =0 OR P_B00_A_Z_STS & $1 =0 OR P_B00_A_U_STS & $1
=0)
ENDWHILE  // PULSE 출력 상태 BIT 검사
WHILE(P_B00_A_X_STS & $41 !=0 OR P_B00_A_Y_STS & $41 !=0 OR P_B00_A_Z_STS & $41 !=0 OR P_B00_A_U_STS
& $41 !=0)
ENDWHILE  // INP 및 PULSE 출력상태 BIT 검사

WHILE(P_B00_A_CMD_RUN != 0)
ENDWHILE

P_B00_ A_CMD_CODE = $343  // 다축 절대좌표 설정 및 구동
P_B00_ A_CMD_DATA0 = 10                  // 모든축 최종 POSITION 10
P_B00_ A_CMD_DATA1 = 10
P_B00_ A_CMD_DATA2 = 10
P_B00_ A_CMD_DATA3 = 10
P_B00_A_MOTOR_SEL = $F
        P_B00_A_CMD_RUN =1
        While(P_B00_A_CMD_RUN!=0)
        ENDWHILE

WHILE(P_B00_A_X_STS & $1 =0 OR P_B00_A_Y_STS & $1 =0 OR P_B00_A_Z_STS & $1 =0 OR P_B00_A_U_STS & $1
=0)
ENDWHILE  // PULSE 출력 상태 BIT 검사

WHILE(P_B00_A_X_STS & $41 !=0 OR P_B00_A_Y_STS & $41 !=0 OR P_B00_A_Z_STS & $41 !=0 OR P_B00_A_U_STS
& $41 !=0)
```

```
ENDWHILE   // INP 및 PULSE 출력상태 BIT 검사

WHILE(P_B00_A_CMD_RUN != 0)
ENDWHILE

            I5112=2000*8388608/I10      // 동작 완료 후 2초간 대기
            While((I5112>0)
            ENDWHILE

P_B00_ A_CMD_CODE = $16   //SERVO OFF
P_B00_ A_CMD_DATA0 = 0
P_B00_ A_CMD_DATA1 = 0
P_B00_ A_CMD_DATA2 = 0
P_B00_ A_CMD_DATA3 = 0
P_B00_A_MOTOR_SEL = $F
            P_B00_A_CMD_RUN =1
            While(P_B00_A_CMD_RUN!=0)
            ENDWHILE
```

HOME SEARCH

원점을 찾기 위해서는 반드시 HOME SENSOR가 각 축마다 있어야 합니다.

Main CMD : 0x03 Sub CMD : 0x60

P_B00_ A_CMD_CODE = $360 //단축 HOME search

P_B00_ A_CMD_DATA0 = $02 // Home Condition

P_B00_ A_CMD_DATA1 = $00 //Home Flag Select (반드시 HOME SENSOR가 있어야 함)

P_B00_ A_CMD_DATA2 = 1000 // Home 구동 속도 부호에 따른 방향 설정

P_B00_ A_CMD_DATA3 = 1000 // home Offset 설정

P_B00_A_MOTOR_SEL = $1 // 축 설정

//** OFFSET 설정 시 원점 Serach 후 이전 속도 프로파일로

// OFFSET 만큼 이동 합니다.

Home Condition

- ⇨ 0x00 Immediate capture
- ⇨ 0x02 Capture on Flag High
- ⇨ 0x03 Capture on Index(Z상) High and Flag High
- ⇨ 0x07 Capture on Index(Z상) Low and Flag High

HOME SEARCH 시 ERROR CODE 관련 확인 사항

HOME 진행 시 ERROR가 발생되면 안전을 위하여 Run bit (ex= P_B00_A_X_STS, bit0) 가 1로 강제로 setting 됩니다. 이는 ERROR CODE를 읽어서 확인 및 Clear 가능 합니다. 각 축별로 가능 하며 0x241로 확인 합니다. (P_B00_xx_RCMD_DATA0에 저장됨)

상위 8BIT는 두 가지를 구분 합니다.

0xF2xxxx ➔ HOME 도중에 Error 발생

0xF4xxxx ➔ OFFSET 적용 도중에 Error 발생

하위 9BIT는 아래와 같이 해당 Error를 표기 합니다.

Bit	Bit name	Description
0	ESPL	Stopped by the +EL input being turned ON.
1	ESML	Stopped by the -EL input being turned ON.
2	ESAL	Stopped by turning the ALM input ON, or when an ALM input occurs while stopping.
3	ESSP	Stopped by the $\overline{\text{CSTP}}$ input being turned ON.
4	ESEM	Stopped by the $\overline{\text{CEMG}}$ input being turned ON, or when an ALM input occurs while stopping.
5	ESSD	Decelerated and stopped by the SD input being turned ON.
6	ESPO	An overflow occurred in the PA/PB input buffer counter.
7	ESEE	An EA/EB input error occurred. (Does not stop)
8	ESPE	A PA/PB input error occurred. (Does not stop)

Ex) 0xF20002 ➔ HOME 도중에 + Limit 발생

0xF40001 ➔ OFFSET 적용 도중에 - Limit 발생

```
// Home Search 예제
P_B00_ A_CMD_CODE = $320  //SERVO ON
P_B00_ A_CMD_DATA0 = 0
P_B00_ A_CMD_DATA1 = 0
P_B00_ A_CMD_DATA2 = 0
P_B00_ A_CMD_DATA3 = 0
P_B00_A_MOTOR_SEL = $1
        P_B00_A_CMD_RUN =1
        While(P_B00_A_CMD_RUN!=0)
        ENDWHILE

P_B00_ A_CMD_CODE = $345  // 다축 정속구동 명령 //-리미트까지 모터 구동
P_B00_ A_CMD_DATA0 = $01      //$00: Positive  $01: Negative X-Axis
P_B00_ A_CMD_DATA1 = $01      //Y-Axis
P_B00_ A_CMD_DATA2 = $01      //Z-Axis
P_B00_ A_CMD_DATA3 = $01      //U-Axis
P_B00_A_MOTOR_SEL = $1
        P_B00_A_CMD_RUN =1
        While(P_B00_A_CMD_RUN!=0)
        ENDWHILE

WHILE(P_B00_A_X_STS & $1 =0 )
ENDWHILE
WHILE(P_B00_A_X_STS & $41 !=0 or P_B00_A_X_STS & $100 =0 )
ENDWHILE // INP 및 PULSE 종료 및 (-)리미트 확인
WHILE(P_B00_A_CMD_RUN != 0)
ENDWHILE

P_B00_ A_CMD_CODE = $241  //ERROR CLEAR
P_B00_ A_CMD_DATA0 = 0
P_B00_ A_CMD_DATA1 = 0
P_B00_ A_CMD_DATA2 = 0
P_B00_ A_CMD_DATA3 = 0
P_B00_A_MOTOR_SEL = $1
        P_B00_A_CMD_RUN =1
        While(P_B00_A_CMD_RUN!=0)
        ENDWHILE
```

```
P_B00_ A_CMD_CODE = $360  // HOME SEARCH
P_B00_ A_CMD_DATA0 = $02      //HOME Condition
P_B00_ A_CMD_DATA1 = $0       //Home Flag Select
P_B00_ A_CMD_DATA2 = 1000  //Home 구동 속도 및 방향
P_B00_ A_CMD_DATA3 = 0        // Home Offset
P_B00_A_MOTOR_SEL = $1        // 해당 축만 1 로 설정
P_B00_A_CMD_RUN =1
While(P_B00_A_CMD_RUN!=0)
ENDWHILE
WHILE(P_B00_A_X_STS & $1 =0 )
ENDWHILE
WHILE(P_B00_A_X_STS & $40 =0 or P_B00_A_X_STS & $1 =0 )
        //HOME 진행 시 Error detect 및 Error clear하는 루틴
        IF( P_B00_A_Z_STS & $4 !=0   or   P_B00_A_Z_STS & $8 !=0 )
                                        // Error or 알람 detect되면
            P_B00_A_CMD_CODE = $241     // Error READ
            P_B00_A_CMD_DATA0 = 0
            P_B00_A_CMD_DATA1 = 0
            P_B00_A_CMD_DATA2 = 0
            P_B00_A_CMD_DATA3 = 0
            P_B00_A_MOTOR_SEL = $1
            P_B00_A_CMD_RUN    = 1
            While(P_B00_A_CMD_RUN!=0)
            ENDWHILE
                P101 =P_B00_A_RCMD_DATA0
                        // Error 상태를 읽어온 값을 TEMP variable에 저장
//////HOME 시 ERROR 발생했음을 알려 주는 FLAG 설정 바랍니다.
                IF( P101&$FF0000 =$F20000)
                        // $F2xxxx 의 Error는 HOME 도중에 Error 발생
                P100=1
                ENDIF
                IF( P101&$FF0000=$F40000)
                        //$F4xxxx는 HOME OFFSET 적용 도중에 Error 발생
                P100=2
                ENDIF
ENDWHILE // INP 및 PULSE 종료
```

구동 시 주의 사항

속도 프로파일 관련

#1) 이동 거리가 짧은 경우 가감속 시간을 반드시 동일 하게 설정 하여야 합니다.
가감속이 다른 경우 가속 시간보다 이동량이 적게 되면 최저 속도로 움직입니다.

모터 동작 관련

#1) ABS Move는 현재의 위치를 기반으로 움직여야 할 거리를 자동으로 생성 합니다.
만약 움직이는 도중에 ABS Move를 사용하면 의도하지 않은 위치로 움직이게 됩니다.

#2) In position이 없는 경우 HOME 속도를 낮게 설정하여야 HOME 과정이 안정되게 완료 됩니다.

1) 상대치 혹은 절대치로 움직일 때 이동 거리가 0 일 경우(PULSE 출력이 안 나가는 경우) PLC 프로그램에서 WAIT 항목 적용 시 조건에 맞지 않아 PLC가 HOLDING 되게 됩니다. 프로그램 시 이러한 경우가 없도록 주의 하여야 합니다.

AGENT PROGRAM 사용 관련

#1) AGENT 창을 되도록 한 개씩 띄워서 사용 하여야 합니다. 속도저하의 원인이 될 수 있습니다.
#2) 개별 Motion을 클릭 했을 때 Window 창이 뜨지 않는 경우 Master Scan을 다시 실행해 주시기 바랍니다.