

# CoreCon

## Setting Manual -



Revised on 2018. 03. 22

본 제품 설명서에서 제공되는 정보는 코아로봇의 자산입니다.

코아로봇의 서면에 의한 동의 없이 전부 또는 일부를 무단 전재 및 재배포할 수 없으며,  
제 3 자에게 제공되거나 다른 목적에 사용할 수 없습니다.

본 설명서는 성능 및 기능 개선을 위해 사전 예고 없이 변경될 수 있습니다.

Printed in the Republic Of Korea - 2016 년 4 월. 초판

Copyright © 2016 by CoreRobot Inc.

## Corecon Settings Manual Revision History

Revision	Data	Comment	비고
1.01	16.04.05	최초 작성	
1.02	16.04.12	전용 신호선 설정 방법	
1.03	16.06.05	Pallet robot, Home 찾기 기능 설정	
1.04	17.03.17	CoreCon 이미지 및 MMPU 기능 추가	
1.05	18.03.22	신규추가된 option 에 대한 설명 정리	

## 목차

1. Setting 파일 .....	7
2. corecon.conf .....	7
1) ROBOT_CONFIG .....	7
2) DRY_RUN .....	8
3) UI_STYLE .....	8
4) OVERRIDE_SYSTEM_INPUT .....	8
5) CONVEYOR_CONFIG2 .....	8
6) DEBUG_MODE .....	9
7) DEBUG_INFO .....	9
8) ENABLE_CONFIG_INS .....	9
9) SERVO_ON_MODE .....	9
10) PREFETCH_SIGNAL .....	10
11) INSTRUCTION_FILE = myInst.txt .....	11
12) TASK1.SLEEP, TASK2.SLEEP .....	12
13) TASK1.PRIORITY, TASK2.PRIORITY .....	12
14) PLUGIN = libmyscreen.so .....	13
15) LOGO = custom.png .....	13
16) LANGUAGE = ko .....	13
17) EMSTOP_MODE = 1 .....	13
18) EMSTOP_ACC_TIME = 0.2 .....	14
19) STEP_BACK = TRUE .....	14
20) STEP_OVER = TRUE .....	14

21)	TEACH_SINGULAR = TRUE.....	14
22)	TEACH_SINGULAR_SPEED_LIMIT = 0.001 ~ 1.0 .....	14
23)	REPEAT_SINGULAR = TRUE .....	15
24)	REPEAT_SINGULAR_SPEED_LIMIT = 0.001 ~ 1.0 .....	15
25)	MAX_LOG_SIZE = 100 .....	15
26)	REMOVE_LOG_RATIO = 0.1.....	15
27)	RECORD_ON = TRUE.....	15
28)	RECORD_AUTORUN = arg1, arg2.....	15
29)	DOMAIN_COUNT = 1 .....	16
30)	SUB_TASK_COUNT = 4 .....	16
31)	WC_ERROR_DELAY = 2 .....	16
3.	Robot Configuration File .....	17
1)	기본 설정 .....	18
2)	EtherCAT slaves device 설정 .....	19
3)	Robot Type 설정 – Robot 기구 사양 .....	21
4)	Axis 설정 .....	27
5)	Cartesian Motion 설정 .....	30
6)	System I/O 설정 .....	31
7)	User Coordinate 설정 .....	35
8)	Zeroing Position 설정 .....	36
9)	Home search 파라미터 설정 .....	36
10)	Run/Hold/Servo on Sequence.....	38
11)	Jog Label 설정 .....	39
12)	그 외 설정 .....	40

4. 예제 : scara.conf.....	42
1) 로봇 사양 확인하기 .....	42
2) 기본설정 .....	42
3) EtherCAT slaves Device 설정 .....	43
4) Robot Type 설정 .....	43
5) Axis 설정 .....	44
6) Cartesian Motion 설정 .....	44
5. 예제 : Virtual Axis.....	45

## Setting 파일

Corecon 의 설정은 몇 개의 텍스트 파일로 설정할 수 있습니다. 프로그램 실행 시 가장 먼저, corecon.conf 를 읽어 들입니다. Corecon.conf 에는 전역적 설정내용과, 상세 로봇 파일 및 상세 컨베이어 설정 파일을 지정합니다. 따라서, 몇 가지 상세 설정을 정의해 두고, corecon.conf 에서 실제 사용할 설정을 지정함으로써, 설정 파일 관리를 용이하게 할 수 있습니다.

### corecon.conf

Configuration 의 상세 내용에 해당하는 파일을 지시하는 항목을 모아 놓은 Configuration File 입니다.

- Configuration 구성은 parameter\_name = value 형식으로 구성됩니다.
- #이 표시된 라인은 comment 입니다.
- corecon.conf 파일에서는 dry\_run, robot\_config, ui\_style 등을 설정할 수 있습니다.

```
# coreCon Configuration File  
  
DRY_RUN = TRUE  
  
ROBOT_CONFIG = serial.conf  
UI_STYLE = yeji.css
```

Figure 1 corecon.conf 내용

#### 1) ROBOT\_CONFIG

corecon.conf 라는 파일에서 실제 적용될 로봇의 Configuration File 을 정의하면, 미리 정의된 구체적인 configure 파일로 연결됩니다. 몇 가지의 Configuration 파일을 정의함으로써 편리하게 로봇을 구분할 수 있습니다.

- ROBOT\_CONFIG = serial.conf → 상세 로봇 파일은 robots/serial.conf 임
- ROBOT\_CONFIG = wtr.conf → 상세 로봇 파일은 robots/wtr.conf 임

별도의 상세 파일 안에서는 axis\_count, ecat\_device, robot\_type 과 같은 구체적인 내용을 서술하고 있습니다.

## 2) DRY\_RUN

시운전 모드로 설정할 때 사용됩니다. TRUE 로 설정할 시 시운전모드, FALSE 또는 comment 로 설정하여 DRY\_RUN 을 사용하지 않으면, 로봇 운전 모드로 설정됩니다.

- DRY\_RUN = TRUE / FALSE

DRY\_RUN 모드는 로봇 UI 를 개발할 경우 등에 사용이 되며, 사용자의 dryrun 모드와는 별개의 모드입니다. 또한 DRY\_RUN 모드로 설정될 경우, 디지털 입력은 UI 의 디지털 입력의 아이콘을 클릭함으로써, 강제로 입력시킬 수 있습니다.

## 3) UI\_STYLE

프로그램의 디자인을 설정할 때, UI\_STYLE 의 값을 변경하여 설정 가능합니다. 기본값은 default.css 파일로 설정되어있고, 사용자가 원하는 css 파일을 작성하여 변경 가능합니다.

- UI\_STYLE = default.css

## 4) OVERRIDE\_SYSTEM\_INPUT

TP 에 있는 Enable Switch(Dead man switch), Select Key, 그리고 Emergency Stop 는 외부 Digital Input 을 통해 입력되어야 합니다. 만일 Digital Input 을 사용하지 않고, UI 버튼으로 Teach/Repeat 전환, TP Enable Key 전환을 사용하려면 이 값을 TRUE 로 정의하면 됩니다.

기본 값이 TRUE 이기 때문에, Digital Input 을 통해 하드웨어 입력 키를 사용하기 위해선, 반드시 이 값을 FALSE 로 해 주어야 합니다.

OVERRIDE\_SYSTEM\_INPUT = TRUE (기본값) UI Teach/Repeat, Enable Key 사용

OVERRID\_SYSTEM\_INPUT = FALSE 하드웨어 키 사용

하드웨어 입력 키가 어느 입력을 사용할 지 설정은 상세 로봇 설정을 참고하기 바랍니다.

## 5) CONVEYOR\_CONFIG2



컨베이어 트랙킹 기능을 사용할 때, 컨베이어 정의 파일을 지정합니다. 컨베이어 설정을 통해, 컨베이어 설정을 완료한 후, 저장하게 되면, 이 항목으로 설정된 파일에 저장이 되게 됩니다.

- CONVEYOR\_CONFIG2 = rbconv.conf

Rbconv.conf 파일은 corecon.conf 와 동일한 폴더에 존재하게 됩니다.

#### 6) **DEBUG\_MODE**

Corecon 을 이용하여, 적용테스트를 하거나, 전용 UI 를 개발할 경우, debug 모드를 설정하여, 다양한 정보의 On/Off 를 관리할 수 있습니다. 기본은 설정은 FALSE 입니다.

- DEBUG\_MODE = TRUE

#### 7) **DEBUG\_INFO**

DEBUG\_MODE 가 TRUE 로 설정할 경우 각각의 기능에 대해 적용 테스트를 진행할 수 있습니다. DEBUG\_INFO 는 bit 로 관리합니다. CONVEYOR, TOUCH, PLANTEST(bit: 1,2,4)

- DEBUG\_INFO = 3 ;Conveyor 와 touch 두가지 기능에 대해 테스트합니다.

#### 8) **ENABLE\_CONFIG\_INS**

Lefty/Righty 와 같은 로봇 동작 명령어를 사용 할 수 있게 하는 설정 항목입니다. 기본적으로 로봇 동작 중의 Config 변경은 금지되어 있으나, 이 설정 항목을 TRUE 로 두면, Config 설정 관련 명령어를 사용할 수 있습니다. Config 설정은 MoveJ 명령어에서만 사용할 수 있습니다. MoveL 이나 MoveC 를 사용할 시에는 특이점을 지나가게 되어, 지정된 움직임을 실행할 수 없기 때문에, 무시됩니다.

- ENABLE\_CONFIG\_INS = TRUE

#### 9) **SERVO\_ON\_MODE**

Servo On 동작을 설정합니다. 기본 동작은 Motor On 신호는 Controller 의 Power 가 공급되고 있는 State 신호로만 사용됩니다. Servo On Timing 은 Teach mode 와 Repeat mode 의 경우에 따라 달라집니다. Teach mode 의 경우, Motor On 신호 후에, TP 의 Enable switch 가 On 이 되면, 전원부의 MC 가 작동이 되어, Motor Driver 에 Motor Power 를

공급하고, 잠시 후, ServoOn 이 작동됩니다. TP 의 Enable 을 끄면, Servo Off 상태가 됩니다. Repeat 모드일 경우엔, Hold 상태에서 Run 으로 이동 시 Servo On 이 됩니다. Hold 로 변경되면 Servo Off 가 되고, Teach mode 로 변경되면, Servo Off 와 더불어, MC 도 Off 되게 됩니다. (MC Off 는 제어함 interface 구현에 따라 달라질 수 있습니다. )

만일 Servon Timing 을 기본 동작과는 다르게, Motor On 신호시, 바로 Servo On 까지 수행하고자 한다면, SERVO\_ON\_MODE = 1 로 설정합니다.

Motor On 신호를 무시하고, 개별적으로 Servo On 신호를 하고자 한다면, SERVO\_ON\_MODE = 2 로 합니다. 각 축 별로 Servo On 을 별도로 하고자 할 때 이 모드를 사용하면 됩니다.

- SERVO\_ON\_MODE = 0 : Default 동작, 정적 servo on/off - Robot 의 MotorOn 신호로 설정.

OVERRIDE\_SYSTEM\_INPUT = TRUE 인 경우, 외부 시그널을 무시하므로, UI 의 Servo On 버튼에 따라 구동된다.

OVERRIDE\_SYSTEM\_INPUT = FALSE 인 경우, 외부 Motor On signal 의 상태에 따라 servo on/off 가 전환된다.

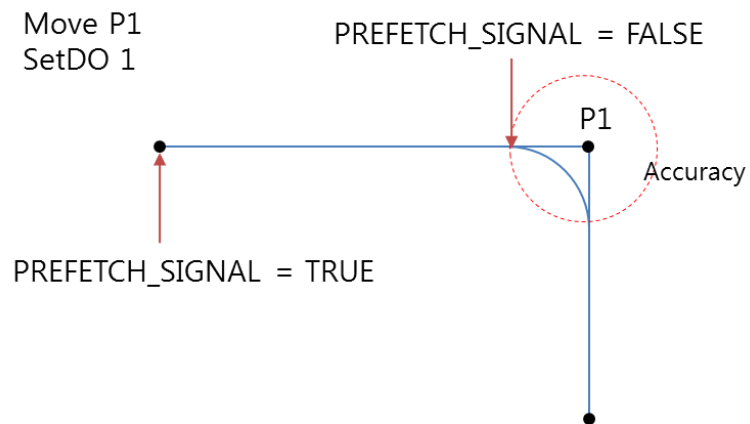
- SERVO\_ON\_MODE = 1 : Motor on 신호 시 즉시, Servo On 시키고, 동적 servo on/off - Robot 의 Servo On sequence 에 따라 구동됩니다.
- SERVO\_ON\_MODE = 2 : Motor on 신호 무시, Robot engine 과 servo on 은 관계가 없다. 즉, servo on 을 위해서, 별도의 command 를 구현해 줘야 합니다.
- SERVO\_ON\_MODE = 3 : Driver 별 servo on/off 를 구현합니다. Robot config 파일에 AXIS\_SERVO\_ON\_MODE = 0, 1, 0, 1 이런 식으로 각각의 축에 mode 0, mode 1 을 별도로 할당합니다. 0 으로 설정된 축은 motor on 상태에 따라 상시 servo on 상태가 되고 1 로 설정된 축은 run/hold 시퀀스에 따라 동적으로 servo on/off 가 설정됩니다.

#### 10) PREFETCH\_SIGNAL

로봇의 모션 명령 후의 Digital output 혹은 input 출력 Timing 을 설정합니다.

PREFETCH\_SIGNAL = TRUE 이면, 로봇 Motion 이 시작할 때, 바로 Output 이 나가게 되며, FALSE 이면, 모션 종료 후 출력이 나가게 됩니다. 기본 설정은 TRUE 입니다.

- PREFETCH\_SIGNAL = FALSE



관련 명령어 :

RunMask – FALSE 일 경우 동작 완료 후 mask 설정

WatiTime – FALSE 일 경우 동작 완료 후 timer start

WaitSig - FALSE 일 경우 동작 완료 후 signal checking

SetDO

Reset

Bits

PulseDO

DelayDO

SetAO

## 11) INSTRUCTION\_FILE = myInst.txt

Robot Instruction 을 rename 할수 있습니다. 예를 들어 MoveJ, MoveL 등 명령을 JMove, LMove 등으로 변경 가능합니다.

INSTRUCTION\_FILE 에 변경하고자 하는 Instruction 파일을 corecon.conf 의 INSTRUCTION\_FILE = myInst.txt 와 같이 설정합니다. 이때 myInst.txt 파일 형식은 old name -> new name 을 pair 로 나열 합니다.

ex) myInst.txt

MoveJ JMove

MoveL LMove

Waittime TimeWait

## 12) TASK1.SLEEP, TASK2.SLEEP

로봇 프로그램을 실행하는 Task 의 명령간 쉬는 시간을 설정합니다. 단위는 msec 이며, -1 이면 쉬없이 연속적으로 프로그램을 실행합니다. Task1 은 Main Task 이고, Task2 는 Sub Task 입니다.

Sub Task 에서 우선 순위를 높게 하여, 프로그램을 실행하고자 할 때, TASK2.SLEEP = -1 로 설정합니다. 사용자가 적절히 프로그램 내에 쉬는 시간을 설정해 주지 않으면, UI 등의 성능에 영향을 미쳐, 성능이 떨어질 수 있습니다. TASK2.SLEEP 의 값을 조정하여, 이 문제를 해결할 수 있습니다.

TASK2.SLEEP = -1

TAKS1.SLEEP = 4           # 4msec sleep

## 13) TASK1.PRIORITY, TASK2.PRIORITY

로봇 프로그램을 실행하는 Task 의 우선 순위 레벨을 설정합니다. 1 ~ 99 까지 설정할 수 있으며, -1 이면 UI 와 동일한 우선 순위로 설정됩니다. -1 로 설정시 과도한 UI 실행에서 로봇 프로그램 실행이 다소 지연될 수 있으나, UI 의 응답성은 좋아 집니다.

TASK1.PRIORITY = -1

TAKS2.PRIORITY = 20

**14) PLUGIN = libmyscreen.so**

coreCon 을 실행 했을 때, 정규 coreCon 프로그램 화면이 아닌, 사용자가 정의한 화면이 로드됩니다. 전용 application 을 위한 화면 구성이나, 운용 화면을 간단히 하고자 할 때, custom ui 용 api 를 사용하여, 화면 프로그램을 제작하여, corecon 에 연결할 수 있습니다. 자세한 설명은 coreCon custom 화면 설계편을 참조해 주십시오.

**15) LOGO = custom.png**

coreCon 을 실행하였을 때, UI 화면에서 나타나는 코아로봇의 로고가 아닌 원하는 로고로 변경하고 싶을 때 설정하면 원하는 로고가 로드됩니다. 예를 들어 cusom.png 파일로 설정하게 되면 크기에 맞는 로고가 로드됩니다. 파일은 .png 파일만 가능하며, 권장 파일 크기는 146x56 입니다. 크기에 맞지 않는 이미지를 사용할 시, 이미지가 원하는 모습으로 나타날 수 있음을 주의하시기 바랍니다.

**16) LANGUAGE = ko**

coreCon 에서 사용하고자 하는 언어를 바꾸고자 할 때 설정하게 되면, 원하는 언어로 로드됩니다. Default 언어는 영어로서 설정을 하지 않는 경우에는 기본적으로 영어가 로드됩니다. 현재 설정가능한 언어는 영어, 한국어, 중국어 입니다. 영어는 en, 한국어는 ko, 중국어는 zh-cn 으로 설정하여야 합니다. 언어는 추가적으로 업데이트 될 예정입니다.

**17) EMSTOP\_MODE = 1**

coreCon 에서 Emergency Stop 을 실행할 때, 정지되는 속도에 따라 설정할 수 있습니다. 설정 값은 0, 1, 2 로 가능합니다.

0 일 경우, 감속시간이 없이 즉시 정지되는 모드입니다.

1 일 경우, default 설정 값으로 macro 명령어 Hold 와 동일한 감속 시간으로 정지되는 모드입니다.

2 일 경우, 설정된 EMSTOP\_ACC\_TIME 의 감속 시간으로 정지되는 모드입니다.

**18) EMSTOP\_ACC\_TIME = 0.2**

EMSTOP\_MODE = 2 를 위한 값을 설정합니다.

**19) STEP\_BACK = TRUE**

coreCon 에서 Step 명령어를 실행할 때, 이전 스텝으로 돌아가서 실행할 수 있는 기능을 추가해줍니다. 기본 설정은 false 로 되어 있습니다.

**20) STEP\_OVER = TRUE**

coreCon 에서 Step 명령어를 실행할 때 Call 명령어를 실행할 경우, Call 실행 프로그램으로 화면이 변경되어 Call 프로그램을 한 스텝씩 실행하게 되는데 STEP\_OVER 값이 TRUE 일 경우, Call 프로그램을 내부에서 실행된 후, 그 다음 스텝을 실행하게 됩니다. 기본 설정은 false 로 되어 있습니다.

**21) TEACH\_SINGULAR = TRUE**

로봇이 Teach 혹은 Repeat 모드로 Singular 근처를 통과할 때, 모터의 속도가 급격히 증가되는 현상이 나타나며 Over Speed Limit Error 가 발생하고 로봇은 정지하게 됩니다. 이런 현상을 회피하는 방법으로 로봇의 Speed 를 줄여 Singular 영역을 통과하게 할 수 있지만 속도의 조절만으로 회피하는 것이기 때문에 경우에 따라 작동이 완벽하게 되지 않을 수도 있습니다. 따라서 Singular 구간을 필연적으로 통과해야 될 경우, MoveJ 와 같은 모션으로 변경하도록 하고, Serial6 인 경우 MoveH 를 이용하도록 한다.

TEACH\_SINGULAR = TRUE 일 경우 Teach 모드에서 Singular 구간 Joging 을 가능하게 합니다.

기본 설정은 FALSE 입니다.

**22) TEACH\_SINGULAR\_SPEED\_LIMIT = 0.001 ~ 1.0**

Teach 모드에서 Singular 구간 회피 시 TEACH\_SINGULAR = TRUE 일 때 TEACH\_SINGULAR\_SPEED\_LIMIT 값이 1.0 일 경우 Teach mode 의 Max speed limit 까지 허용합니다.

**23) REPEAT\_SINGULAR = TRUE**

REPEAT\_SINGULAR = TRUE 일 때 Repeat 모드에서 Singular 구간 통과를 가능하게 합니다.

기본설정은 FALSE 입니다.

**24) REPEAT\_SINGULAR\_SPEED\_LIMIT = 0.001 ~ 1.0**

Repeat 모드에서 Singular 구간 회피시 REPEAT\_SINGULAR = TRUE 일 때

REPEAT\_SINGULAR\_SPEED\_LIMIT = 1.0 일 때 Repeat mode 의 max speed 로 Singular 구간을 통과하게 됩니다(보통 정격속도의 2 배로 설정). 따라서 AXIS Spec 항목인 AXIS\_MAXRPM, AXIS\_RPM 등 설정치를 같이 조절할 필요가 있습니다.

**25) MAX\_LOG\_SIZE = 100**

Corecon.log 의 max log size 를 설정합니다. 기본설정은 1000 개입니다.

**26) REMOVE\_LOG\_RATIO = 0.1**

Corecon.log 가 max log 보다 커지면 10% 만큼의 log 를 삭제합니다. 삭제된 log 는 corecon.log.2 로 옮겨지고 기존의 corecon.log.2 는 삭제됩니다.

MAX\_LOG\_SIZE = 100

REMOVE\_LOG\_RATIO = 0.1

Log 사이즈가 100 개가 넘으면 10%씩 삭제되고 삭제된 로그는 corecon.log.2 로 옮겨집니다.

**27) RECORD\_ON = TRUE**

마우스 클릭 이벤트를 record 하는 기능입니다. RECORD\_ON = TRUE 로 설정하면 Jog 화면에서 Sub Menu 에 Record 기능이 추가 됩니다. 기본 설정은 FALSE 입니다. Record 관련 자세한 사항은 Manual 을 참고하시기 바랍니다.

**28) RECORD\_AUTORUN = arg1, arg2**

RECORD\_ON = TRUE 일 때 프로그램 로딩 후, 자동으로 Recording Macro 프로그램을 실행합니다.

arg1: recording macro program name

arg2: 0(1cycle),1(repeat)

ex) RECORD\_AUTORUN = rec\_0, 0

위 예제는 Recording Macro 프로그램 rec\_0 을 1cycle 실행 시합니다.

#### 29) DOMAIN\_COUNT = 1

사용중인 Servo Driver 가 Ethercat Command 명령어 LRW(Read/Write), LRD(Read), LWR(Write)중 어떤 명령어를 지원하는지에 따라 Domain 을 1 개(TxPDO + RxPDO)로 구성할지 아니면 TxPDO 와 RxPDO 를 분리하여 Domain 을 2 개로 생성할지를 결정합니다.

기본 설정 값은 1 입니다.

#### 30) SUB\_TASK\_COUNT = 4

Run tap 에서의 SunTask 의 개수를 설정합니다. 최대 4 까지 사용 가능합니다. 기본 설정은 4 입니다.

#### 31) WC\_ERROR\_DELAY = 2

Working counter 는 EtherCAT 전송 시 Data frame 이나, slave 의 문제로 인해, 제어기(마스터)와 드라이버(Slave)간의 데이터 교환에 실패 시 Error 발생합니다. Working counter Error 발생시 Error 로 인식하고 로봇의 동작을 멈추는 조건을 설정합니다. Working counter Error 는 1 cycle(2msec) 마다 체크를 합니다.

WC\_ERROR\_DELAY = 2 로 설정하면 2cycle 이상 Working counter Error 발생시 로봇의 동작을 멈춥니다.



## Robot Configuration File

로봇의 구성요소 및 EtherCAT 관련 설정 변수를 정의합니다.

기본 단위는 길이 mm, 각도 degree, 시간 sec, 속도 mm/sec, 각속도 deg/sec 입니다.

예외의 경우는 모터 속도는 RPM 을 CYCLE\_TIME 은 msec 를 사용합니다.

```
ROBOT_NAME = ABB IRB4600
CYCLE_TIME = 2
CONTROLLER_MODEL = DTP7-L

#####
# Motor Maker
# LSMPO, HIGEN, OMRON, SERVOTRONIX
#####
ECAT_DEVICE = LSMPO, LSMPO, LSMPO, LSMPO, LSMPO, LSMPO

#####
# Link parameter : ABB IRB4600
#####
ROBOT_TYPE = SERIAL6

L_V1 = 495
L_V2 = 900
L_V3 = 175
L_H1 = 175
L_H2 = 960
L_H3 = 135

#####
# per axis spec
#####
AXIS_COUNT = 6
AXIS_TYPE = R, R, R, R, R, R
AXIS_PPR = 524287, 524287, 524287, 524287, 524287, 524287
AXIS_GEAR = 105, 100, 100, 80, 80, 50
AXIS_RPM = 3000, 3000, 3000, 3000, 3000, 3000
```

```

AXIS_ACCTIME = 0.5, 0.5, 0.5, 0.5, 0.5, 0.5

AXIS_MSPEED = 30, 40, 40, 60, 60, 60

AXIS_LIMITP = 180, 150, 150, 400, 150, 400
AXIS_LIMITM = -180, -150, -180, -400, -150, -400

# world coordiante
W_SPEED = 2000, 180
W_ACCTIME = 0.8
W_MSPEED = 250, 50

```

Figure 2 serial.conf 파일 내용

### 1) 기본 설정

Robot 의 Name 과 Cycle Time, 사용하는 Controller 의 모델을 설정해 줍니다. Robot 의 Name 은 필수사항은 아닙니다. 설정을 하지 않으면, 빈칸으로 표시됩니다. CYCLE\_TIME 은 2 msec 로 기본값으로 설정되어있습니다.

■ ROBOT\_NAME = ABB\_ROBOT

■ CYCLE\_TIME = 2

coreCon 을 사용할 Controller Model 을 설정해 줍니다. Key 입력이나, I/O 입력등이 Controller 모델에 따라 달라지기 때문에, 사용하는 모델에 따라 설정해 줍니다. 기본값은 DTP7-L 로 되어 있고, 이 값이 맞지 않을 때, Jog Key 가 동작하지 않거나, I/O 가 동작하지 않을 수 있습니다.

■ CONTROLLER\_MODEL = DTP7-L

- DMC : DMC 모델 I/O 서브모듈은 사용하지 않음
- DMC-IO : DMC 모델, I/O 서브 모듈을 사요함. I/O 는 1~16, 1001~1016 에 할당되게 됩니다.

■ SHIFT\_PERCENT = 30

- Ethercat Master 에서 Slave 로 Position data 를 보낼 때 Slave 에서 data 를 다 받을 때 까지 지연시간을 주는 기능입니다. SHIFT\_PERCENT 를 설정할 경우

Master 에서 position data 를 보내자마자 Slave 에서 바로 처리하는 것이 아니라 설정한 Shift percent 만큼 position data 를 계속 Read 후 motor 를 동작하게 됩니다. 기본 설정은 30%로 되어 있습니다.

■ SPEED\_RATE\_TYPE = 0

- Macro 프로그램에서 Speed 를 %로 사용시 가/감속 시간으로 인해 동작시간이 속도의 배수단위로 변하지 않습니다. SPEED\_RATE\_TYPE = 1 로 설정할 경우 Macro 프로그램의 Speed 명령어와 Override Speed(Run tap 의 SPEED)가 같은 동작을 수행합니다. 기본 설정은 0 입니다.

■ TRACK\_OPTION = FALSE

- Track 이 있는지 여부를 설정합니다. 기본값은 FALSE 로 되어있고 TURE 로 변경 시 마지막 축이 track axis 로 설정됩니다.

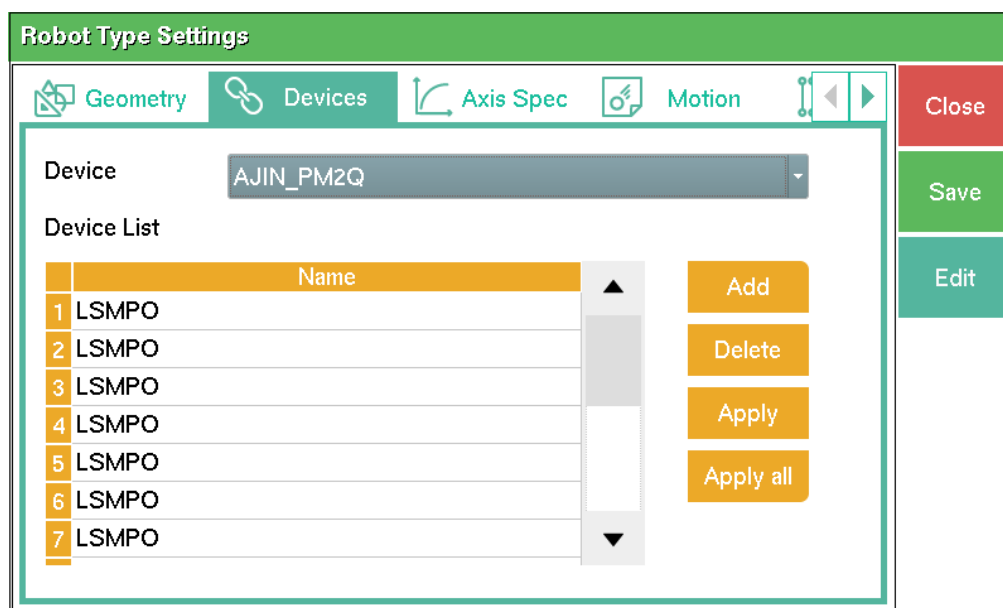
■ INCHING\_ACCTIME = 0.5

- Inching 사용시 acctime 을 설정합니다. 기본 설정은 0.5 입니다.

■ ROBOT\_SOLVER = cusrobot.so

- Custom 로봇의 solver 파일을 설정합니다.

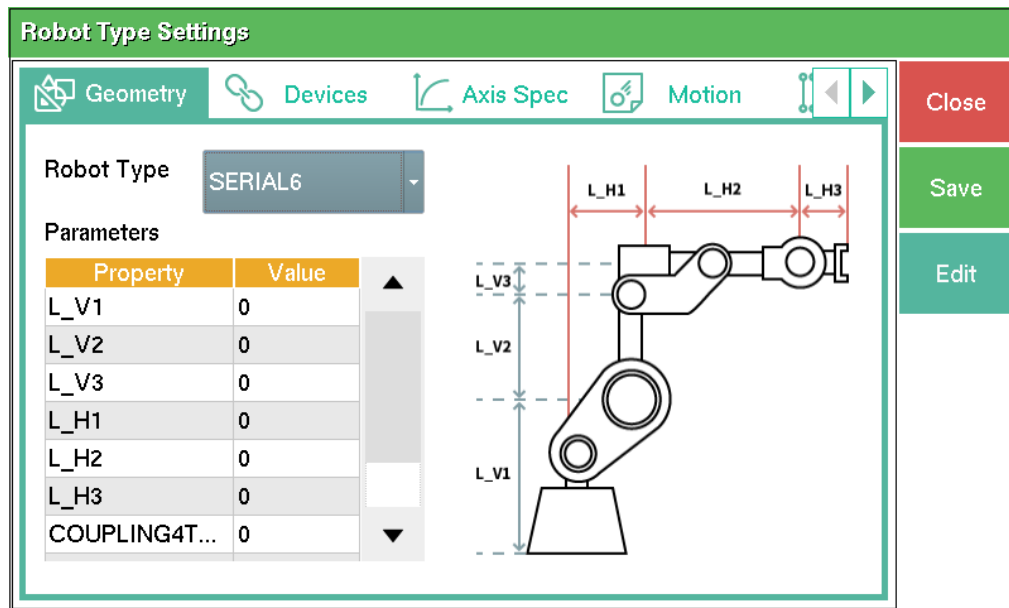
## 2) EtherCAT slaves device 설정



Robot 의 EtherCAT slaves device 에 대한 설정을 해야 합니다. 설정 가능한 EtherCAT device 의 종류는 다음과 같습니다.

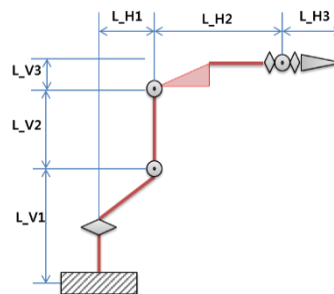
- Servo Driver : LSMPO, OMPON, LSMPO\_PEGA, HIGEN, PANASONIC\_5150A1, PANASONIC\_515070A1, SERVOTRONIX 등 기본으로 저장되어있는 servo driver 이외에 다른 제품(Ethercat type Servo drive)을 사용하고 싶을 경우에는 본사로 문의 바랍니다.
- Servo Driver 에서 PANASONIC 이 Vendor name 이 되고, 5150A1 과 같은 이름은 Product code 가 된다. Panasonic 은 용량 별로 product code 가 다르기 때문에 product code 로 분류하도록 하였다.
- I/O Driver : BECKHOFF\_EK1100, BECKHOFF\_EL1008, BECKHOFF\_EL2008, BECKHOFF\_EL3008, BECKHOFF\_EL4008 등 다양한 IO 모듈 선택이 가능함

### 3) Robot Type 설정 – Robot 기구 사양

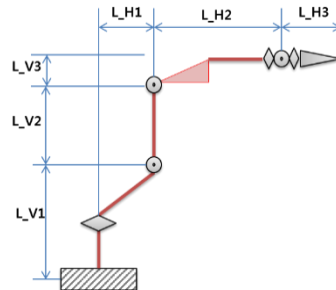


Configuration File 에서 Robot 의 타입은 ROBOT\_TYPE = Serial6 와 같은 형식으로 사용합니다. Robot type 은 SERIAL6, PARALLEL6, UR6, CARTESIAN, SCARA, CUSTOM 등이 있습니다. 각 Type 마다 설정해야 할 Link Parameter 가 있으며, 이를 설정해 주어야 합니다.

- SERIAL6 : "L\_V1", "L\_V2", "L\_V3", "L\_H1", "L\_H2", "L\_H3"

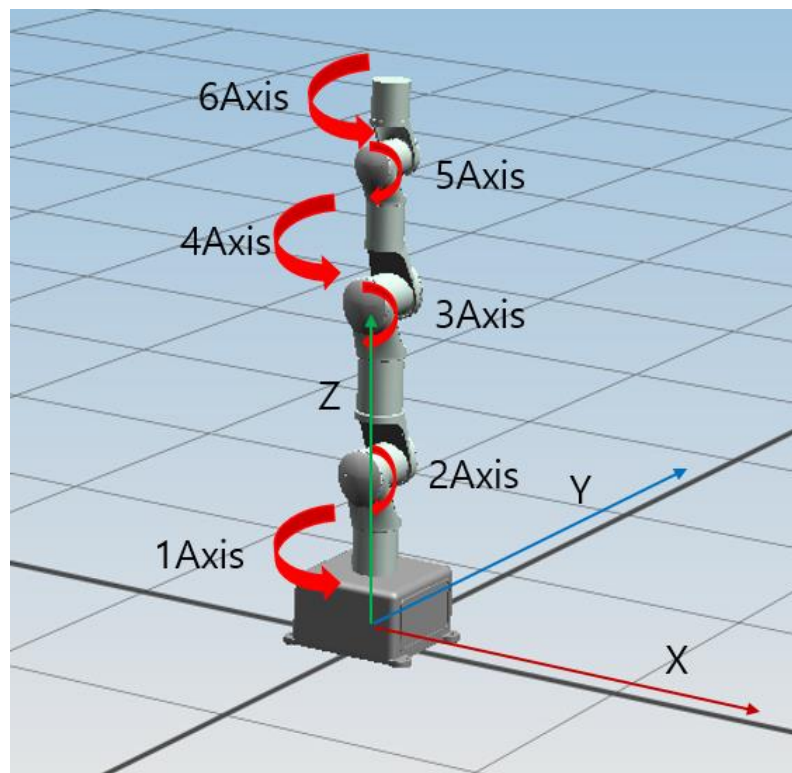


- SERIAL6-2 : "L\_V1", "L\_V2", "L\_V3", "L\_H1", "L\_H2", "L\_H3"



L\_V3 = 0 인 경우 기존의 6 축과 Zero Position 이 변경된다.

전체를 세운 위치가 Zero 위치이며, 각 Joint 의 방향은 그림과 같다.

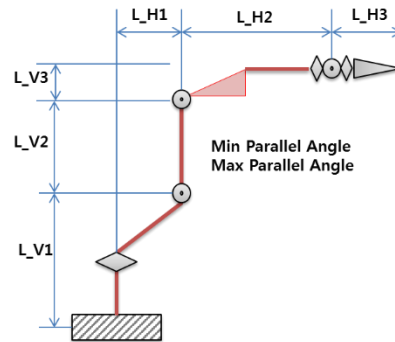


각축의 회전 방향

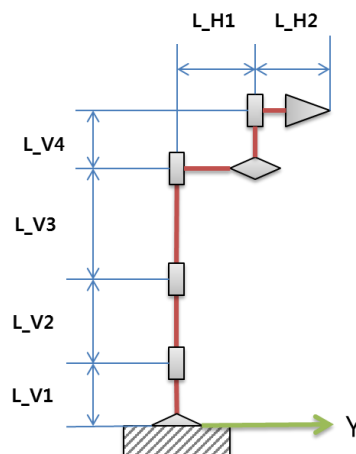
1, 4, 6 축 : Z 방향 회전

2, 3, 5 축 : Y 방향 회전

- PARALLEL6 : "L\_V1", "L\_V2", "L\_V3", "L\_H1", "L\_H2", "L\_H3"

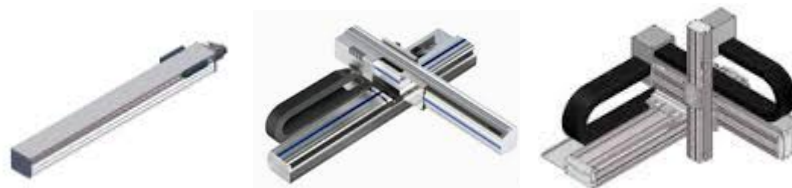


- UR6 : "L\_V1", "L\_V2", "L\_V3", "L\_V4", "L\_H1", "L\_H2"



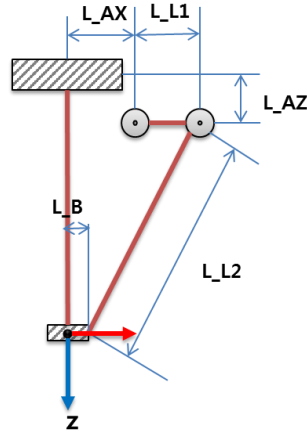
- CARTESIAN : "L\_AXIS"

: 1 이면 단축, 2 이면 X-Y, 3 이면 X-Y-Z



1 축, 2 축, 3 축 구성

- DELTA : "L\_AXIS\_TYPE", "L\_AX", "L\_AZ", "L\_L1", "L\_L2", "L\_B", "L\_ROTATION"

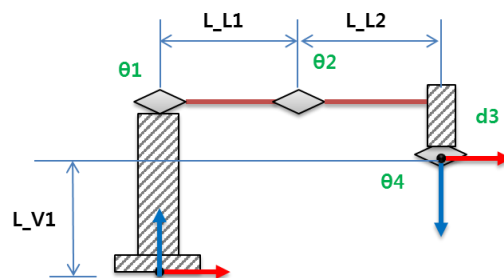


L\_AXIS\_TYPE = 1 이면 회전축이 있는 4 축 delta 가 되고, 0 이면 회전축이 없는 3 축 delta 가 됩니다.

L\_ROTATION = x 축 회전 각도

Delta robot 의 x 축은 첫번째 관절의 방향이 x 축이 됩니다. X 축 방향을 변경하고자 할 때는 여기에 회전 값을 입력하면, 회전된 축이 x 축이 됩니다.

- SCARA : "L\_V", "L\_H1", "L\_H2", L\_COFACTOR



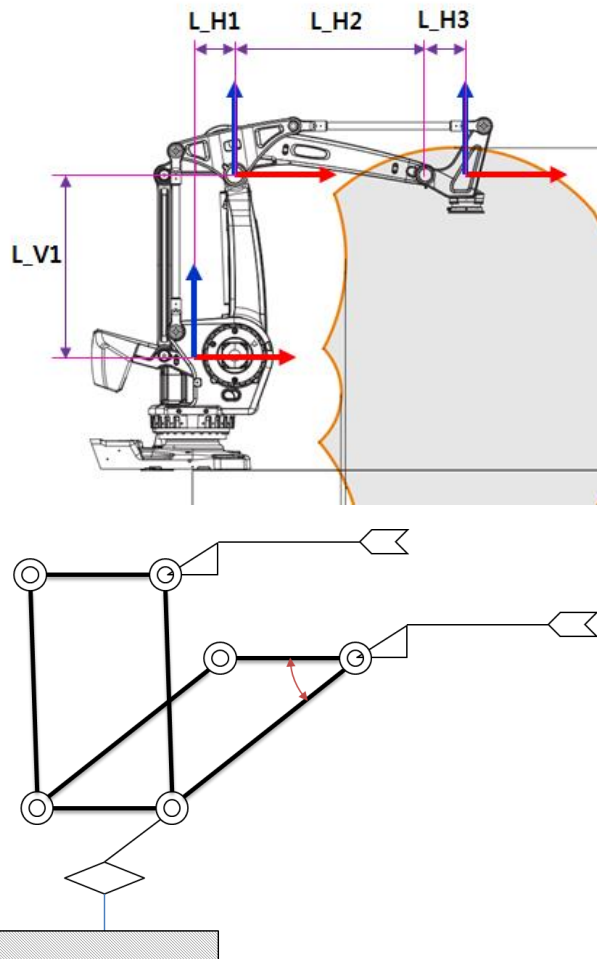
- PALLET : "L\_TYPE" "L\_V1", "L\_H1", "L\_H2", "L\_H3"

PALLET robot 은 4 축 혹은 5 축으로 정의된 로봇입니다. L\_TYPE = 1 이면 아래 그림과 같이 4 축 Palletizing robot 으로, 3 축이 Parallel link 로 구동되는 Type 입니다. 따라서, Parallel link 각의 최소 최대 값도 입력해 주어야 합니다. Parallel link 각의 기본 값은



30 ~ 150 도 입니다.

L\_TYPE = 2 이면, 4 축이면서, Parallel Link 가 없는 Type 이며, L\_TYPE = 3 이면, 5 축으로 Wrist 부의 관절이 Control 가능한 Type 이 됩니다. 현재 L\_TYPE = 1 만 제공되고 있습니다.



Parallel link 사잇각 범위 정의

MIN\_PARALLEL\_ANGLE = 30

MAX\_PARALLEL\_ANGLE = 150

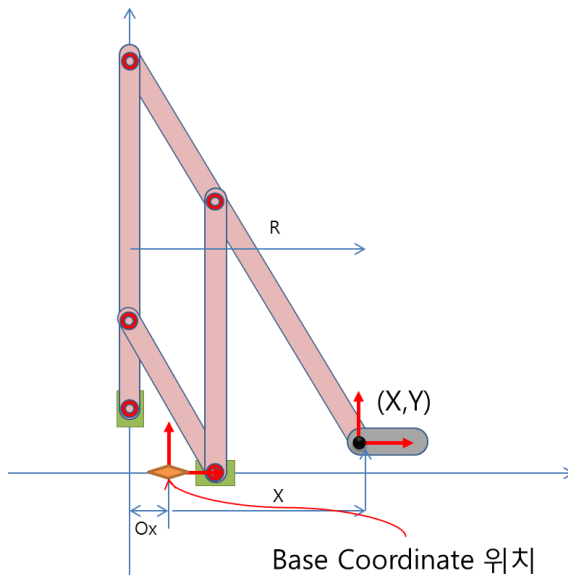
- PALLET\_L : "L\_TYPE" "L\_V1", "L\_H1"

PALLET\_L 은 4 축으로 구성된 Palletizing robot 중에서, 2, 3 축의 구동이 Linear 구동장치에 의해 구동되는 로봇입니다.

이 로봇의 특징은 2,3 축이 회전형 관절로 구성되어 있으나, Parallel link 의 동작으로 인해 End Effector 의 작동은 항상 Linear 가 된다는 것입니다.

L\_V1, L\_H1 은 1 축의 회전축에 Base 좌표계를 두기위한 Offset 값입니다.

Home 위치 예를 아래 그림과 같이 2 축의 angle, 3 축의 angle 위치를 각각 0, 60 도로 하고, 이때의 Joint 좌표값이 (0, 0, 0, 0)으로 설정한다면, 이 상태에서, End Effector 위 X, Z 의 Joint 값은 (0, 0)이 됩니다. 이를 Base 좌표계에 대한 값으로 변환하기 위해선, 아래와 같은 L\_H1, L\_V1 값을 입력합니다. L\_V1 의 값은 적당한 값을 입력해도 되나, L\_H1 은 직선동작을 위해서, 회전축과 일치시켜야 하므로, 이 자세에서 계산된 값을 입력해 주어야 합니다.



$$L_{H1} = X_0 = R - O_x$$

$$L_{V1} = Z_0$$

Joint 축의 Limit 설정은 이 자세를 기준으로, +/- limit 영역을 기구 동작이 가능한 범위로 설정하면 됩니다..

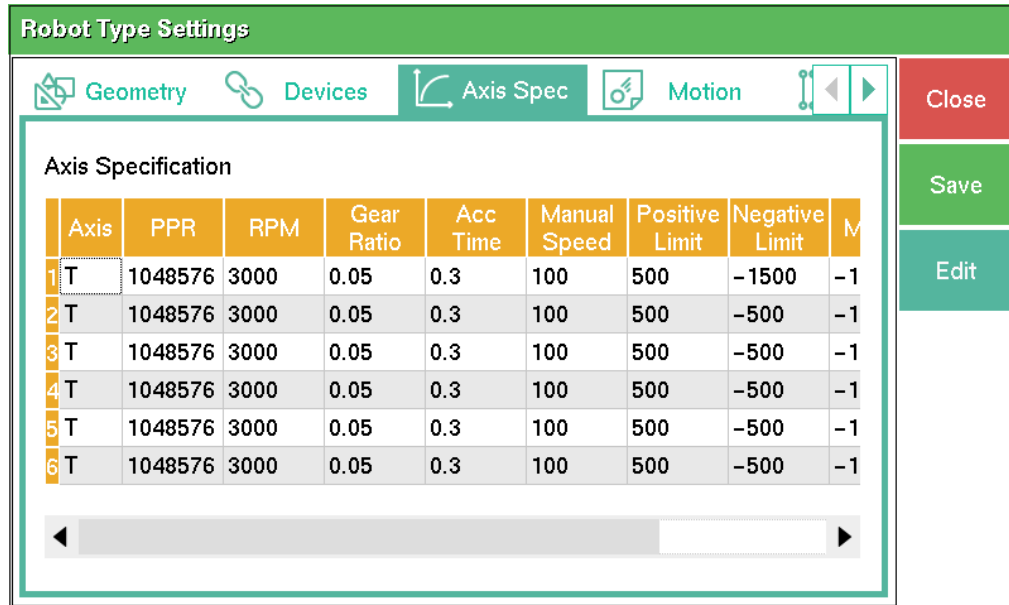
#### ● CUSTOM : "L\_TYPE "

LTYPE : 0 or 1

: 0 이면 User Coordinate 를 사용합니다. 따라서, User Coordinate 도 정의해줘야 합니다.

: 1 이면 Custom IK/FK 를 제공하는 모델이 됩니다. User coordinate 는 사용하지 않고, joint vs 직교좌표의 관계를 별도의 모듈로 제공하면 됩니다.

#### 4) Axis 설정



**Robot Type Settings**

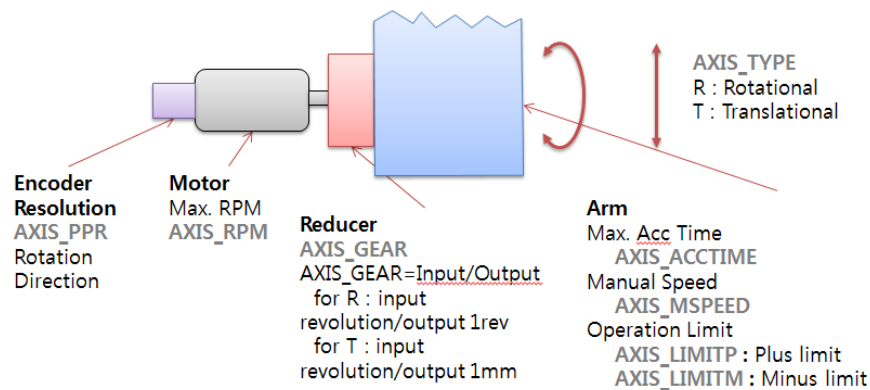
Geometry Devices **Axis Spec** Motion

Close Save Edit

**Axis Specification**

	Axis	PPR	RPM	Gear Ratio	Acc Time	Manual Speed	Positive Limit	Negative Limit	M
1	T	1048576	3000	0.05	0.3	100	500	-1500	-1
2	T	1048576	3000	0.05	0.3	100	500	-500	-1
3	T	1048576	3000	0.05	0.3	100	500	-500	-1
4	T	1048576	3000	0.05	0.3	100	500	-500	-1
5	T	1048576	3000	0.05	0.3	100	500	-500	-1
6	T	1048576	3000	0.05	0.3	100	500	-500	-1

Parameters : Motor – Encoder – Reducer - Arm



- AXIS\_COUNT = 6 : 축의 개수
- AXIS\_TYPE = R, R, R, R, R, R

: 축의 타입으로서 T 와 R 이 있으며, R : Rotational, T : Prismatic 을 의미합니다.

- AXIS\_PPR = 524287, 524287, 524287, 524287, 524287, 524287 :

: PPR 은 Pulse Per Revolution 의 약자로 1 바퀴 회전당 나오는 진동 수를 의미합니다.

Encoder 의 bit 수를 의미하며, 위의 예시는 19bit Encoder 를 뜻하게 됩니다. 회전 방향이 반대로 될 경우, 앞의 숫자에 - 표시를 붙입니다.

- `AXIS_ENCBIT = 19, -19, 19, 19, 19, 19 :`

: PPR 을 bit 수로 입력합니다.

주) 이 값은 실제의 Encoder bit 보다, Motor driver 의 command pulse bit 으로 입력하여야 합니다. 드라이버에 따라, Encoder bit 수와 Driver command bit 수가 다를 수 있습니다.

`AXIS_PPR` 또는 `AXIS_ENCBIT` 는 같은 내용이기 때문에, 둘 중 하나를 사용하면 됩니다.

- `AXIS_GEAR = 105, 100, 100, 80, 80, 50`

: 이 값은 Gear ratio 로서 Translational Axis 의 경우, 1mm 를 가기 위해 필요한 회전 수를 의미하고, Rotational Axis 의 경우, 1 회전을 하기 위해 필요한 회전 수를 의미합니다.

Ex) 회전축이면서, 감속비가 100 일 경우, Axis type 이 R 이고, 100 바퀴를 돌아야 Arm 이 1 회전을 할 수 있으므로 Gear Ratio = 100 이 됩니다.

만약, Axis type 이 T 일 경우, 직선운동으로 변환시켜주는 볼스크류등의 리드 사양을 보고 판단할 수 있습니다. 볼스크류의 리드가 25 라면, 모터 1 회전에 25mm 를 이동하기 때문에, 1mm 를 가기 위해선 1/25 회전이 필요합니다. 즉, Gear Ratio 는 0.04 가 나오게 됩니다.

- `AXIS_RPM = 3000, 3000, 3000, 3000, 3000, 3000`

: RPM 은 Revolution per Minute 의 약자로 1 분당 돌아가는 회전 수를 의미하며 모터의 정격 RPM 값으로 설정합니다. Joint Move 할 때의 Max speed 는 이 값으로 설정됩니다. 위의 예시는 1 분당 3000 바퀴를 회전하는 속도를 의미하며 정격 RPM 값이 3000 이라는 것을 의미합니다.

- `AXIS_MAXRPM = 6000, 6000, 6000, 6000, 6000, 6000`

모터가 순간적으로 작동할 수 있는 최대 RPM 을 설정합니다. 이 값 이상의 입력값이 들어오게 되면, 동작을 중지하게 됩니다. 기본은 6000RPM 으로 설정되어 있습니다.

- `AXIS_ACCTIME = 0.5, 0.5, 0.5, 0.5, 0.5, 0.5`

:ACCTIME 은 각 축의 최소 가속 시간으로 최고 속도까지 올라가는 데 소요되는 시간을

나타냅니다. 이 값은 수동으로 설정해야 하며, 위의 예시는 0.5 초만에 최고 속도에 도달한다는 의미입니다.

- `AXIS_MACCTIME` = 0.5, 0.5, 0.5, 0.5, 0.5, 0.5

:`AXIS_MACCTIME` 은 Joging 시 각 축의 가/감속 시간입니다. 기본 설정은 0.5 입니다.

- `AXIS_MSPEED` = 30, 40, 40, 60, 60, 60

: Jogging 할 시, 최대 Jogging 속도로서 수동으로 직접 입력할 경우, 지정된 값으로 설정됩니다. 기본 설정은 50 입니다.

(Manual Speed 의 경우 국제 표준상 선속도 기준 250mm/s 를 max 로 하고 있습니다. 이를 고려하여 값을 입력하시기 바랍니다.)

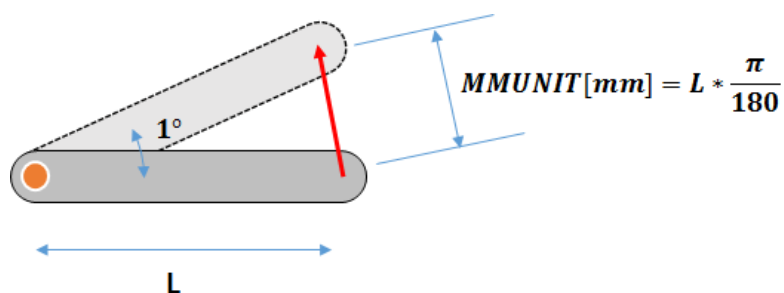
- `AXIS_LIMITP` = 180, 150, 150, 400, 150, 400

- `AXIS_LIMITM` = -180, -150, -180, -400, -150, -400

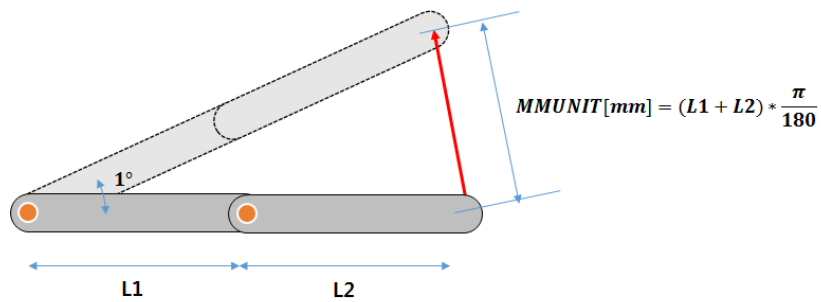
: 동작 Limit 값으로서 최대 이동 가능한 범위를 나타냅니다. P 는 Plus 값을, M 은 Minus 값을 의미합니다.

- `AXIS_MMPU` = 2, 2, 2, 2, 2, 2

: MMPU 는 mm per unit 입니다. 즉, 1 unit 을 움직일 때, 로봇 TCP 의 직선 이동량 mm 단위입니다. 직진 운동 Joint 로 구동되는 Link 의 경우, 대부분 1mm = 1mm 로 `AXIS_MMUNIT` 은 1 이 됩니다. 회전 운동 Joint 로 구동되는 Link 의 경우, Link 길이에 따라, 다음과 같이 계산하여 구하여야 합니다.

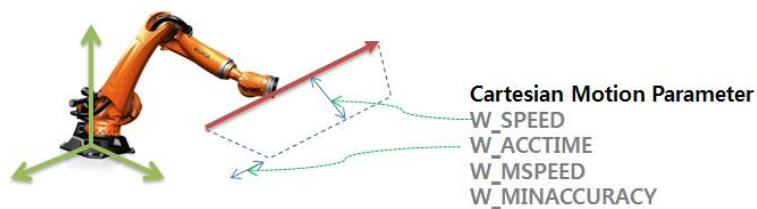
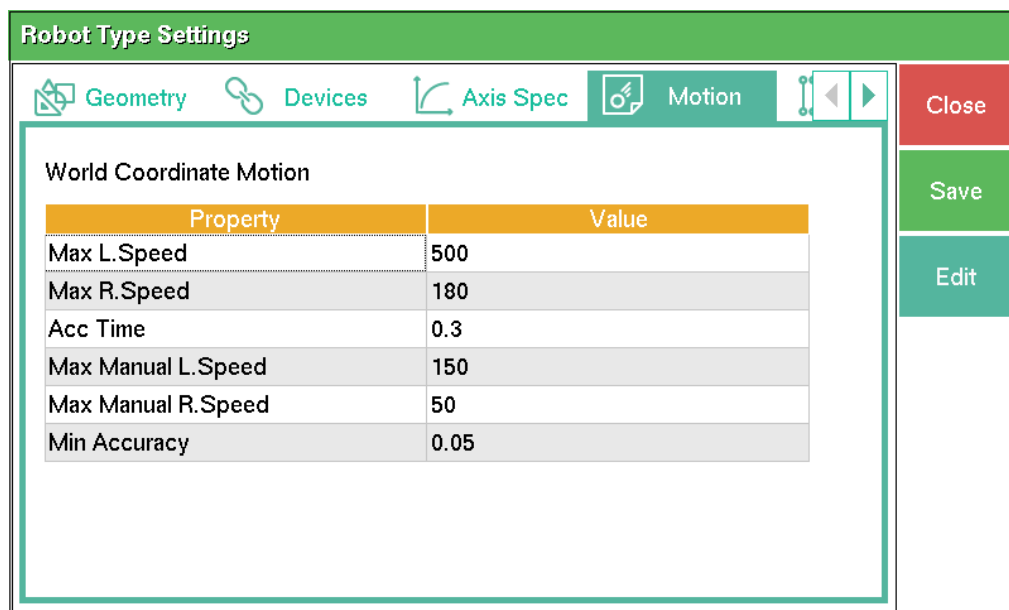


만일 두개의 링크로 연결된 경우엔, 링크 끝단의 END 점의 이동량을 입력합니다.



이 값은 Accuracy 설정과 관계가 있습니다. Accuracy 단위의 의미를 scaling 하고자 할 때는 이 값을 scaling 하면 됩니다. 설정을 하지 않는 경우 Default 값으로 -1 이 설정됩니다.

#### 5) Cartesian Motion 설정



- W\_ACCTIME = 0.8

: World Coordinate 상에서의 가속 시간으로 단위는 sec 입니다. 기본 설정은

0.5 입니다.

- W\_MACCTIME = 0.8

: Joging 시 World Coordinate 상에서의 가/감속 시간입니다. 기본 설정은 0.5 입니다.

- W\_SPEED = 1500, 180

: World Coordinate 상의 최대 선속도와 각속도입니다. 각 단위는 선속도일 경우 mm/sec, 각속도일 경우 deg/sec 을 사용합니다.

(최대 선속도 구하는 방법 : 부하를 가장 많이 받는 축을 기준으로 하여 최대속도 산출

최대속도 :  $(RPM(정격)/60s/Gear Ratio) * 2\pi * \text{최대길이}$

위 공식으로 구한 최대 선속도는 모든 영역을 만족하지는 않습니다.)

- W\_MSPEED = 250, 50

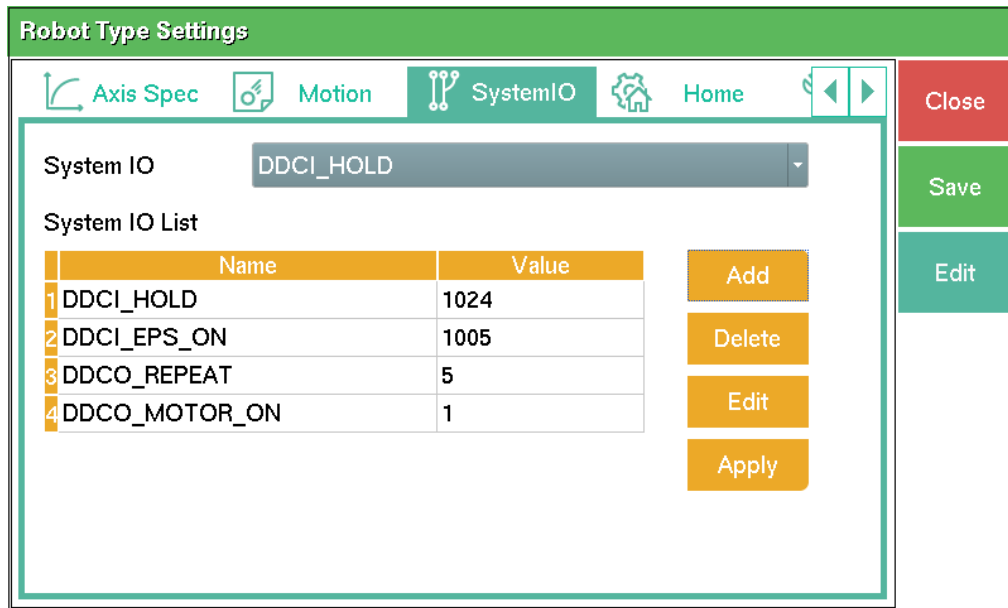
: World Coordinate 상의 jog 최대 속도를 설정하는 값으로, 각 단위는 W\_SPEED 와 같습니다.

(Manual Speed 의 경우 국제 표준상 선속도 기준 250mm/s 를 max 로 하고 있습니다. 이를 고려하여 값을 입력하시기 바랍니다.)

- W\_MINACCURACY = 0.1

: 축일치 등을 체크할 때 사용하는 최소 ACCURACY 값으로 단위는 mm 입니다.

## 6) System I/O 설정



**Robot Type Settings**

Axis Spec Motion **SystemIO** Home

System IO: DDCI\_HOLD

System IO List

	Name	Value
1	DDCI_HOLD	1024
2	DDCI_EPS_ON	1005
3	DDCO_REPEAT	5
4	DDCO_MOTOR_ON	1

Add Delete Edit Apply

Close Save Edit

로봇이 사용하는 전용 신호가 I/O 단자의 어디를 사용할지 설정하는 파라미터입니다. TP의 Teach/Repeat 절환 switch, Enable switch 등을 사용하려면, 반드시 System을 I/O를 설정하고, OVERRIDE\_SYSTEM\_INPUT 값을 FALSE로 해 두어야 합니다.

전용 신호는 출력신호와 입력신호가 있습니다.

신호의 정의는

신호 이름 = IO 번호, 사용 여부 (1 : 사용, 0 : 사용하지 않음)

형식으로 설정하고, 정의 할 수 있는 I/O는 아래와 같습니다.

- 출력 신호

- DDCO\_MOTOR\_ON = 16, 1 : Motor On 상태를 출력합니다.
- DDCO\_ERROR = 17, 1 : Error 상태를 출력합니다.
- DDCO\_TEACH = 18, 1 : Teach mode 일 시 On
- DDCO\_REPEAT = 19, 1 : Repeat mode 일 시 On
- DDCO\_CHECK = 20, 1 : Check mode 일 시 On
- DDCO\_RUN1 = 21, 1 : Main Task 가 실행 중이면 On
- DDCO\_RUN2 = 22, 1 : Sub Task 가 실행 중이면 On
- DDCO\_EPS\_MODE = 23, 1 : External program Selet Mode 를 사용중이면 On



- DDCO\_EPS\_STATUS = 24, 1 : EPS\_ON 신호가 있으면 On
- DDCO\_HOME = 25, 1 : Home 위치에 있으면 On
- DDCO\_HOME2 = 26, 1 : Home2 위치에 있으면 On
- DDCO\_REMOTE = 27, 1 : Remote mode 일 시 On
- DDCO\_HIGH\_SPEED = 28, 1 : 한 축 이상이 RPM 이상의 속도일 때 On
- DDCO\_SYSTEM\_READY = 29, 1 : EtherCAT 이 op 상태일 때 On
- DDCO\_SERVO\_ERROR = 30, 1 : EtherCAT Servo driver 에서 오류가 났을 때 On
- DDCO\_HS\_MODE = 31, 1 : Home Search mode 일 시 On

- 입력 신호

- DDCI\_MOTOR\_POWER\_ON = 1001, 1 : Motor power 를 On 시킵니다. Pulse 입력을 주고, Off→On 일 때 Motor On 을 시키고, Servo Driver 의 상태를 Servo On 으로 하여, 로봇의 동작이 가능한 상태가 되게 합니다.

- DDCI\_MOTOR\_POWER\_STATE = 1002, 1 : Motor power 상태를 입력합니다. Motor Power 상태를 모니터링 할 수 있는 입력 신호가 있다면, 여기에 연결해 줍니다. 만일 이 신호가 Off 가 되면, EMSTOP 과 동일하게, Servo Off 를 시키고, 프로그램은 모두 정지하게 됩니다. 이 신호를 무효화 시키면, 모터 파워 입력 상태는 체크하지 않습니다.

- DDCI\_SELECT1 = 1003, 1

- DDCI\_SELECT2 = 1004, 1

DDCI\_SELECT1 이 On, DDCI\_SELECT2 가 Off 일 때 Teach 모드이고, 그 외 경우 Repeat 모드 입니다.

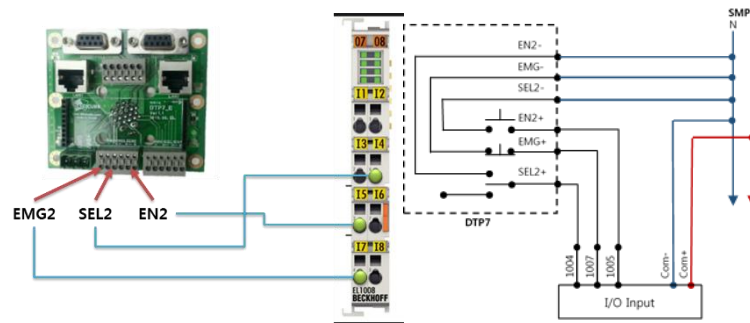
- DDCI\_TPENABLE = 1005, 1 : On 이면 Enable 상태가 됩니다. 이 상태에서 조강이 가능하게 됩니다.

- DDCI\_ERESET = 1006, 1 : Error 를 Reset 합니다.

- DDCI\_EMSTOP = 1007, 1 : EM Stop 이 눌러 졌음을 인지 하게 됩니다. EMSTOP 은 B 접점으로, ON 일 때가 EMSTOP 버튼이 눌러지지 않게 된 경우 입니다.

DTP7-L 장비의 경우, 커넥터 뒷면에, EMG1, EMG2, SEL1, SEL2, EN1, EN2 단자가 있습니다.

위 예제의 경우, 아래와 같이 EMG2, SEL2, EN2 단자를 사용하여, Digital Input 의 첫 번째 slave 에 연결합니다.

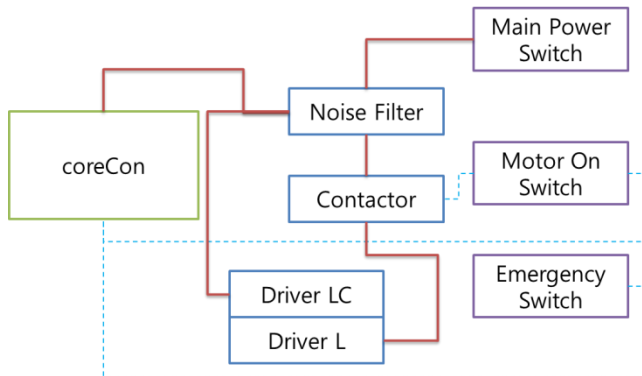


TEACH\_REPEAT : 1004 / TP\_ENABLE : 1005 / EMSTOP : 1007

- DDCI\_SAFETY\_ERROR = 1008, 1 : External error signal to need servo off
- DDCI\_HOLD = 1009, 1 : External Hold/Run Switch
- DDCI\_PROGRAM\_START = 1010, 1 : External Program start
- DDCI\_PROGRAM\_RESET = 1011, 1 : External Program Reset
- DDCI\_EPS\_ON = 1012, 1 : External Program select On
- DDCI\_EPS\_OFF = 1013, 1 : External program select Off
- DDCI\_EPS\_START\_BIT = 1014, 1 : External program select start IO, 1014 부터 시작하여 대응되는 프로그램은 pg1, pg2, ...
- DDCI\_EPS\_END\_BIT = 1015, 1 : External program select end IO
- DDCI\_START\_SUB1 = 1016, 1 : External start sub task
- DDCI\_RESET\_SUB1 = 1017, 1 : External reset sub task
- DDCI\_STOP\_SUB1 = 1018, 1 : External end sub task
- DDCI\_ACFAIL = 1019, 1 : AC FAIL 이 났을 때 입력되는 신호로 정전 등 전원 차단 문제가 발생할 중요 데이터를 backup 합니다. 현재 기능 구현이 되어있지 않음.

- System Input 신호 연결

아래 그림과 같이 노이즈 필터를 거친 전원은 드라이버의 제어 전원, 그리고, coreCon 의 전원으로 인가 되고, 별도의 라인을 통해, 모터의 전원으로 인가되고, 이는 외부 Motor on switch 로 제어합니다.



coreCon 의 중요한 System input 신호선이 이 Motor on switch 와 Emergency switch 입니다. System 신호를 활성화시키면, Software 로 조정하는 Motor On 버튼은 작동이 되지 않고, 이 Motor on switch 의 상태를 읽어 Motor on sequence 를 수행하게 됩니다. 또한 , Emergency switch 신호로 내부 제어 상태를 정지 시킬 수 있으나, 이 Emergency switch 는 별도로 Motor 전원을 off 시키는 회로를 구성해 두어야 합니다.

## 7) User Coordinate 설정

Robot Type Settings

⚙️ Motion   ⚙️ SystemIO   ⚙️ Home   ⚙️ Etc
Close

☐ User Coordinate
Axis Count : 8

	Angular	Speed	Acc Time	Manual Speed
1	0	500	0.3	150
2	0	500	0.3	150
3	0	500	0.3	150
4	0	500	0.3	150
5	0	500	0.3	150

☐ Zeroing Position

1	2	3	4	5	6
0	0	0	0	0	0

Save  
Edit

일반적인 직교 좌표계를 사용하지 않을 경우, 사용자 정의 좌표계를 정의하여 사용할 수 있습니다. 이 경우 Robot 의 Type 은 반드시 Custom 으로 설정되어 있어야 하며, Joint Space 에서 User Space 간 변환하는 기구해는 별도의 library 로 입력되어 있어야 합니다.

예를 들어, XYZ 직교좌표계는 3 개의 직선 component 를 갖는 user 좌표계로 생각할 수 있습니다.

- UC\_SIZE = 4 : Coordinate 의 Component 개수입니다.

- UC\_ANGULAR = 0, 1, 0, 0

: Axis component 중 각도단위를 사용하는 축과 직선축을 설정합니다. 1 일 경우 각도단위, 0 일 경우 직선축이 됩니다.

- UC\_SPEED = 400, 290, 1700, 1700

: 각 축 상의 최대 속도를 설정합니다. 속도의 단위는 직선속도일 경우 mm/sec, 각속도일 경우 deg/sec 가 됩니다.

- UC\_ACCTIME = 0.4, 0.2, 0.2, 0.2

: 각 축 상의 최대 가속 시간을 설정합니다. 즉 최대 속도에 도달하는 시간을 의미합니다.

- UC\_MSPEED = 100, 50, 200, 200

: 각 축 상에서 jogging 할 때, 최대 jogging 속도를 설정합니다. UC\_SPEED 와 동일한 단위를 사용합니다.

## 8) Zeroing Position 설정

Zeroing 을 실시 했을 때의 위치가 (0, 0, 0, 0.... ) 이 아닌 특정 위치가 되도록 설정할 수 있습니다. ( 7) User Coordinate 그림 참고.)

- ZERO\_POSITION = -20, -20, -20, 0

이 예제의 경우, zeroing 을 실행시키면, 설정된 값으로 위치가 설정됩니다.

## 9) Home search 파라미터 설정

Robot Type Settings

⚙️ Motion
🔗 SystemIO
🏠 Home
🔧 Etc

Close
Save
Edit

Home Search

☒ Home Search Enable

	Axis	Method	Speed	Speed Fine
1	False	0	0	0
2	False	0	0	0
3	False	0	0	0
4	False	0	0	0
5	False	0	0	0
6	False	0	0	0
7	False	0	0	0
8	False	0	0	0
9	False	0	0	0

Incremental Encoder 를 장착한 모터를 사용할 시, 로봇 구동전에 Home(또는 Reference)를 찾아야 합니다. 이러한 Home search 기능에 필요한 파라미터를 설정은 아래와 같은 변수를 이용하게 하게 됩니다.

- HS\_EXT\_ENABLE = 1 : 외부 신호를 이용하여 home search 를 필요로 하는 상황에서 사용하게 됩니다. 1 로 설정을 할 시 pthread 를 이용하여 DDCI\_HS\_MODE 를 확인하게 되고, hs mode 가 true 가 될 때, home search 를 실행 시키는데 실행 전 System IO 인 DDCI\_MOTOR\_POWER\_ON 를 확인하여 Motor power 가 들어와 있을시 실행하고, 아닐 때에는 모터에 파워를 인가 후 home search 를 실행합니다. 완료 시 leave 하게 됩니다. 그리고 다시 homing 을 하기 위해서는 DDCI\_HS\_MODE 를 0 으로 변경했다 1 로 다시 올려줘야 합니다.
- HS\_ENABLE = 1 : Home searching 기능을 사용한다는 표시입니다. 0 일 경우, UI 에서 Home searching 을 시도할 시 에러가 발생하게 됩니다.
- HS\_AXIS = 1, 0, 1 : Home searching 을 실시하는 축을 표시합니다. Incremental Encoder 와 Absolute Encoder 를 혼용하여 사용할 경우, 필요한 축만 Home 을 찾게 됩니다.
- HS\_METHOD = 7, 8, 7 : Home 찾는 방법을 각 축마다 다르게 설정할 수 있습니다.
- HS\_START\_ORDER = 0 : Home Searching 우선순위 축입니다. 기본값은 0 입니다.
- HS\_SPEED = 10, 0, 10 : Home sensor 를 찾기 위한 속도입니다. 이 속도가 너무 빠르게 되면, Home sensor 인식이 안될 수 있으므로, 적절한 속도를 설정해 주어야

합니다.

- HS\_SPEED\_FINE = 2, 0, 2 : Home sensor 를 찾고 난 후, 정확한 reference 점으로 이동할 때의 속도입니다. 통상 Encoder 의 z 상 위치까지 움직일 때의 속도가 됩니다
- HS\_ACCEL = 10,10,10 : home 이동 가감속을 설정

#### 10) Run/Hold/Servo on Sequence

Servo On Option 은 0, 1, 2, 3 이 존재하는데 1 일 경우 동적 Servo on/off 로서 Robot 의 Servo On Sequence 에 따라 구동됩니다.

Robot 의 servon on/off 동작은 브레이크 작동을 고려하여, 다음의 순서대로 진행됩니다.

servo on 지시 --> servo on --> brake release --> run OK

servo off 지시 --> motion hold 시 까지 대기 --> brake on --> servo off --> hold OK

이 진행 순서에서 servo on 상태가 되면, driver 에 servo on 지령을 내리게 되는데 수시로 servo on/off 가 일어나게 됩니다. 그리고 이 상태에선 Motor on 신호는 Robot 모터 전원의 마그네틱 contact 를 연결하여 Power 공급을 완료하는 신호로 사용되고, 실제 servo on 상태까지는 진행이 되지 않게 됩니다.

현재 EtherCAT driver 의 경우, brake 작동은 driver 의 자체 sequence 대로 동작하게 되어 있으나, 로봇 엔진의 내부에서도 위의 sequence 로 처리되게 되어 있습니다. 따라서, 로봇 엔진의 sequence 대로 처리하고자 할 때는, brake 신호선을 robot 의 전용 신호선에 연결하여 처리하면 됩니다. driver 자체 시퀀스를 사용할 경우, 위의 brake on/off 동작은 의미가 없습니다.

servo on 지시는 Teach mode 일 경우, Motor on 상태에서, Enable switch 를 풀게 되는 상태이거나, Repeat mode 에서 program start 명령을 내린 상태가 됩니다. 특히, repeat 모드에선 servo on sequence 과 완료하고 난 뒤, run ok 가 되어야 프로그램을 진행하게 됩니다.

Servo On 시퀀스의 각각의 상태로 전이하기 위해선 약간의 time delay 가 필요한데, 이는 Robot config file 에서 RUN\_TIMER, HOLD\_TIMER 옵션으로 다음과 같이 설정할 수 있습니다.

- HOLD\_TIMER = 0.5, 0.1, 0.4
- RUN\_TIMER = 0.1, 0.4, 0.8

기본 값은 RUN\_TIMER = 0.1, 0.4, 0.8 이고, HOLD\_TIMER = 0.5, 0.1, 0.4 입니다. 주의할 점은 HOLD\_TIMER 는 상대 시간 간격으로 , 지시한 순간부터 0.5 초 그 다음 0.1 초 그리고 0.4 초 이다. 절대 시간 간격으로 환산하면, 0.5, 0.6, 1.0 이 됩니다.

반면에 RUN\_TIMER 는 모든 시간이 지시한 순간부터의 시간이므로, 각각의 상태로 전환되기 위한 시간 간격은 각각, 0.1 --> 0.3 --> 0.4 가 됩니다.

RUN\_TIMER 의 각각의 시간의 의미는

run 상태 지시 --> 0.1 sec --> servo on --> 0.3 sec --> brake release --> 0.4 --> run OK

HOLD\_TIMER 의 각각의 시간의 의미는

hold 상태 지시 --> 0.5 sec --> hold motion done --> 0.1 sec --> brake on --> 0.4 sec --> servo off --> hold OK

hold motion 이 종료 대기 시간이 0.5sec 이므로 로봇의 가감속 설정에 따라 긴 hold motion 시간이 필요할 경우, 0.5 sec 을 0.8, 1.0sec 등으로 늘려 설정합니다.

## 11) Jog Label 설정

Jog 화면에서 display 되는 label 또는 실제 데이터 값, euler angle type, control key 를 사용자가 원하는 대로 바꿀 수 있습니다. 예를 들어 scara 같은 경우, x, y 축 회전이 필요하지 않아 거의 쓰이지 않습니다. 이럴 때, 기존 6 개의 x, y, z, a, b, c 의 Cartesian 화면을 x, y, z, c 로 표현할 수 있습니다.

- JOG\_JOINT\_LABEL = J1, J2, J3, J4 : custom joint display label 표시 설정
- JOG\_TRANS\_LABEL = X, Y, Z, C : custom trans display label 표시 설정
- JOG\_TRANS\_ANGLE = DEF : Trans angle 값을 Euler Angle Type 으로 설정할 수 있습니다. Default 값은 DEF 이며, ZYX, XYZ 로 변경할 수 있습니다.
- JOG\_TRANS\_DATA = X, Y, Z, C : custom trans display data 표시 설정, label 을 바꿔도 실제 data 는 이 설정을 통해 따로 설정을 해주어야 합니다. 그렇지 않으면 trans label 은 X, Y, Z, C 이나 data 값은 X, Y, Z, A 를 표시하게 됩니다.
- JOG\_TRANS\_KEY = X, Y, Z, C : custom trans control key 설정, label 과 data 를

바뀌도 jog key 는 이 설정을 통해 따로 설정을 해주어야 합니다. 그렇지 않으면 trans label 과 data 는 X, Y, Z, C 이나 key 값은 X, Y, Z, A 로 움직이게 됩니다.

●

## 12) 그 외 설정

위의 설정 외에도 다음과 같은 설정을 수행할 수 있습니다.

- REPORT\_ILA = TRUE/FALSE : Report Internal limit alarm 의 약자로서 POT/NOT 신호가 잡혔을 때, alarm 을 띄울 것인지 말 것인지에 대한 설정을 합니다. 실제 Driver 의 H/W Limit 이 커지거나, 토크 limit 이 초과하는 경우가 발생할 때, Internal Limit Active (ILA) bit 가 켜지게 됩니다. Default 값은 true 로서 Internal limit alarm 을 표시하게 되고, alarm 을 표시하고 싶지 않을 때, false 로 설정하면 error alarm 을 표시하지 않습니다. 그리고 Teach mode 에선 두 경우 모두 에러로 report 하지 않고, repeat mode 에서만, error 를 report 합니다.

Driver 의 연결 상태가 좋지 않을 경우, 제어기에서 지령치를 계속 내리지만 드라이버가 동작하지 않으면 두 값이 차이가 커지게 됩니다. 이 설정은 이러한 차이 값이 특정 값 이상이면, 에러를 발생시키게 됩니다. 이 때, 에러코드는 -15, 메시지는 Servo driver pos. and command has too much difference 가 출력됩니다.

이 기능 자체를 Off 시키거나, check 설정 값을 크게 하도록 하기 위해 다음과 같은 설정을 할 수 있습니다.

- MONITOR\_CMD\_JUMP = TRUE/FALSE : cmd Pos 와 cur Pos 가 일정 값 이상 차이가 날 경우 에러를 발생시킬지 말지를 설정하는 것입니다. Default 값은 false 로서 기능을 무효화 시킵니다.
- MONITOR\_CMD\_JUMP\_RATE = 20 : 에러를 발생시키는 cmd Pos 와 cur Pos 값의 차의 비로서 비가 클수록 차이 커야 에러가 발생하게 됩니다. error check 값은 MAX\_RPM 으로 계산된 매 cycle 당 encoder diff 의 X 배로 설정되며, 이 X 배는 기본 default 값이 20 입니다. MAX\_RPM 은 AXIS\_MAX\_RPM 을 설정하자 않을 시 AXIS\_RPM 의 2 배가 된다. 즉, 3000RPM 일 경우 6000RPM 이 됩니다.

동작 중 정지하는 Hold 동작은 최대 감속으로 정지하도록 구현되어 있습니다. Hold 동작은



Hold, Teach 로 전환, Brake, Interrupt on 등의 동작 시 작동됩니다. 이러한 Hold 동작을 할 때에도 acc/dcc 적용치를 적용할 때 사용할 수 있습니다.

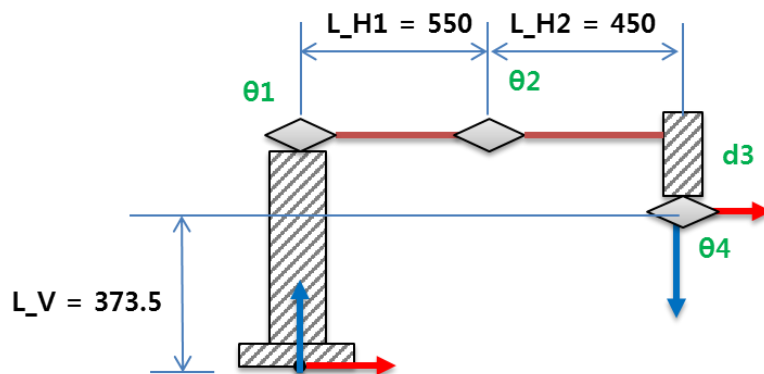
- VARIABLE\_HOLD\_DCC = TRUE/FALSE : hold 감속이 Macro 명령어인 accel/decel 명령에 따라 변경되는지 여부를 설정합니다.
- MAX\_HOLD\_DCC\_TIME = 0.6 : hold 감속 시간이 주어진 시간보다 길어지지 않게 최대치를 설정합니다. Default 값은 0.6sec 입니다.

## 예제 : scara.conf

SCARA Robot 의 구성사양을 보고 Configuration File 을 작성하는 법은 다음과 같습니다.

### 1) 로봇 사양 확인하기

로봇 사양이 다음 그림과 표와 같이 로봇의 크기와 사양이 구성되어 있는지를 확인합니다.



모터	모델명	용량[Kw]	정격속도[rpm]	최대속도[rpm]
JOINT1	MSMD08**	0.75	3000	4500
JOINT2	MSMD08**	0.75	3000	4500
JOINT3	MSMD04**	0.4	3000	5000
JOINT4	MSMD02**	0.2	3000	5000

감속비	감속기	볼스크류리드
JOINT1	80	-
JOINT2	80	
JOINT3		25
JOINT4	15	-

### 2) 기본설정

특별한 경우가 아니라면, CYCLE\_TIME 은 2msec 로 설정해 주시고, Controller 를 원하는 모델로 작성합니다.

```
CYCLE_TIME = 2
CONTROLLER_MODEL = DTP7-L
```

### 3) EtherCAT slaves Device 설정

모터의 모델명으로 관련된 EtherCAT Driver 모델을 설정합니다. 제시된 모델은 PANASONIC 의 드라이버를 사용하고, PRODUCT CODE 가 각각 535200A1, 525100A1, 515070A1, 임을 확인할 수 있습니다. 이러한 경우, 아래와 같이 Panasonic 명으로 설정하면 됩니다.

```
ECAT_DEVICE = PANASONIC_535200A1,W
              PANASONIC_535200A1,W
              PANASONIC_525100A1,W
              PANASONIC_525070A1,W
```

### 4) Robot Type 설정

로봇의 타입은 SCARA 임으로 SCARA 로 작성합니다. SCARA 로봇의 경우 1.2 의 3)을 확인하면, "L\_V", "L\_H1", "L\_H2" 값이 필요함을 알 수 있습니다. 예제에 제시된 그림을 보고 각 위치에 맞는 값을 입력하면 됩니다.

만일 3, 4 축의 기계 요소를 Ball Screw Spline 을 사용하게 되면, 4 축 회전시 3 축이 움직이게 되므로 이를 보정하기 위한 값을 입력해야 합니다. 이 값이 L\_COFACTOR 입니다. 보정량은 radian 으로 변환한 값을 이용해야 하기 때문에, Ball Screw 리드 / (2\*pi) 값을 넣어 줍니다.

```
ROBOT_TYPE = SCARA
L_V = 373.5
L_H1 = 550
L_H2 = 450
L_COFACTOR = 3.978873577297      -리드가 25 일 때
```

주) L\_V 의 값은 초기 위치에서의 로봇의 Z 축 값을 정하는 것이기 때문에, 도면상의 값인 373.5 외의 다른 값을 입력해도 됩니다. 만일 L\_V = 400 을 입력하면, 조인트 값이 (0, 0, 0, 0)일 때, 로봇의 위치는 (0 1000, 373.5) 에서 (0, 1000, 400)으로 됩니다.

## 5) Axis 설정

SCARA 로봇의 축의 개수는 4 개이고, 각각의 축은 로봇의 Base 를 기준으로 R, R, T, R 로 구성되어 있습니다.

또한 PPR 은 이 로봇의 Encoder bit 수가 17bit 임으로 1048567 로 설정합니다. Gear Ratio 는 위 표의 감속비로 확인할 수 있는데 축의 타입이 R 일 경우 감속비 값을 그대로 작성해주고, T 일 경우 1 을 볼스크류리드의 값으로 나눠줍니다. 따라서 T 타입의 Gear Ratio 는 1/25 을 해서 0.04 가 됩니다.

그 다음 RPM 값은 위 표에서 정격속도의 값으로 작성해 주고, ACCTIME 은 임의의 값을 설정해 줍니다. 로봇의 사양과 모터 사양으로 계산할 수 있으나, 간단히 0.3 정도를 기본 값으로 입력한 후, 문제가 없을 경우, 더욱 작은 값을 입력하여 시험합니다. 이 값은 작을 수록 로봇의 성능이 향상되므로, 최대한 작은 값을 갖도록 합니다.

MSPEED 값은 Jogging 할 때의 최고 속도이므로 이 값 또한, 사용자가 원하는 값으로 입력해 줍니다. LIMITP 는 +방향으로 이동할 수 있는 최대 값을 말하므로, 임의의 +값으로, LIMITM 은 -방향으로 이동할 수 있는 최대 값을 말하므로, 임의의 -값으로 입력하여 줍니다.

```

AXIS_COUNT = 4
AXIS_TYPE = R, R, T, R
AXIS_PPR = 1048567, 1048567, 1048567, 1048567
AXIS_GEAR = 80, 80, 0.04, 15
AXIS_RPM = 3000, 3000, 3000, 3000
AXIS_ACCTIME = 0.3, 0.3, 0.3, 0.3
AXIS_MSPEED = 150, 40, 40, 60
AXIS_LIMITP = 170 170, 145, 360
AXIS_LIMITM = -170, -170, -145, -360

```

## 6) Cartesian Motion 설정

다음은 마지막으로 World Coordinate 상에서의 값을 설정하는 것입니다. 이 또한 사용자가 원하는 값으로 설정하면 됩니다.

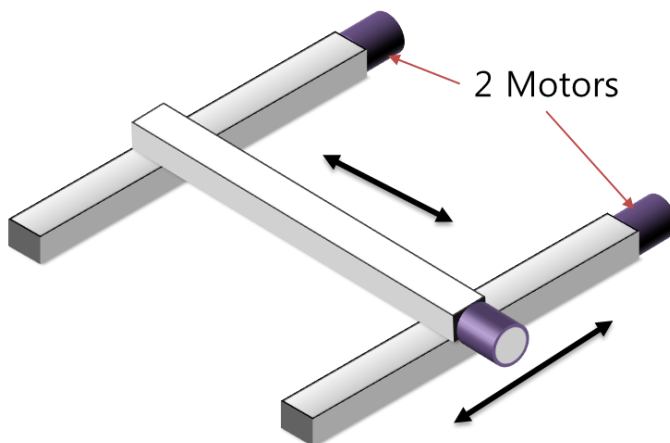
```

W_SPEED = 1000, 180
W_ACCTIME = 0.3
W_MSPEED = 150, 50

```

## 예제 : Virtual Axis

Cartesian Robot 을 사용하나, 특정 축이 2 개의 모터로 구동이 될 경우, 두 축을 그룹 설정할 경우가 있습니다. 아래의 예는 x 축이 2 개의 모터로 구동시키고, y 축은 한 개의 모터로 구동하는 예입니다.



직교 로봇이기 때문에, Robot Type 은 CARTESIAN 이 됩니다.

모션에 관련하는 축은 X, Y 축 2 개이나, 실제 Actuator 축은 3 축이 되기 때문에, 가상축 개념을 적용시킵니다. 아래의 예는 3 축의 모터를 2 축의 XY 축으로 변환시키는 예입니다.

VA_COUNT = 2	: Virtual Axis 축의 개수
VA_ENABLE = 1	: Virtual Axis 설정 여부
VA_TYPE = T, T	: Virtual Axis 축의 타입
VA_SPEED = 1500, 1500	: Virtual Axis Speed
VA_ACCTIME = 0.3, 0.3	: Virtual Axis Acctime
VA_MSPEED = 250, 250	: Virtual Axis Manual Speed
VA_LIMITP = 500, 500	: Virtual Axis LIM
VA_LIMITM = -500, -500	: Virtual Axis LIMITM
VA_SOLVER = libcart2.so.	: Virtual Axis SOLVER

VA\_SOLVER 에 다양한 변환식을 넣어서 사용할 수 있습니다. 이 예의 경우는, VA[0] 의 값을 실제 joint[0], [1]의 값으로 설정하는 간단한 식으로 만 구성하면 됩니다.

```
int cx_ptol(float* pv, float* lv)
{
    // check pv[0] == pv[1]

    lv[0] = pv[0];
    lv[1] = pv[2];
    return 0;
}
```

```
int cx_ltop(float* lv, float* pv)
{
    pv[0] = lv[0];
    pv[1] = lv[0];
    pv[2] = lv[1];
    return 0;
}
```



**CoreCon Quick Operation Manual**

Copyright © 2016–2017 CoreRobot

All rights reserved.

Printed in the Republic Of Korea

#603, IT MIRAE Tower, 33, Digital-ro 9-gil,  
Geumcheon-gu, Seoul, Republic Of Korea

CoreRobot Inc.

Web: [www.core-robot.com](http://www.core-robot.com)

Tel: 82-2-2027-6565

Fax: 82-2-2027-6565