Program Guide

PRESTO
프레스토솔루션

Preface

본 문서는 로봇 제어기(PRC 시리즈)의 프로그램 가이드로, 로봇 명령어 상세 설명과 예제에 대한 내용을 포함합니다.

PRC 시리즈 로봇 제어기를 최고의 성능으로 유지하기 위해서 본 매뉴얼을 사전에 충분히 숙지한 후 설치, 운전, 점검 및 유지 보수 작업을 정확한 방법으로 실행하기 바랍니다.

㈜프레스토솔루션은 고객의 의견을 수렴해 제품 개선에 반영하고자 항상 노력하고 있습니다. 정기/비정기적 제품 개선, 사양 변경 또는 사용자의 편의를 위해 별도의 고지 없이 본 매뉴얼의 내용을 변경할 수 있습니다.

본 매뉴얼 내용 중 의문 사항이 있거나 개선에 관한 의견이 있으면, 언제든지 ㈜프레스토솔루션 본사로 문의해 주시기 바랍니다.

Table of Contents

Chapter 1. Introduction

본 문서는 PRC(Presto Robot Controller) 로봇 프로그래밍 언어를 구성하는 모든 요소를 자세히 설명합니다.

PRC 로봇 언어 RobotGul(로봇글)은 C, Python 와 유사한 문법 구조로 사용자가 쉽게 이해할 수 있는 고수준(High Level) 언어입니다. RobotGul 은 로봇을 실시간 제어할 수 있도록 빠른 해독과 실행이 가능하게 설계되어 있으며, 완벽한 로봇 제어를 위한 다양한 명령어를 제공합니다.

1.1 Software Architecture

PRC 언어는 변수, 식, 함수 이름, 키워드 등 모든 요소에 대소문자를 구분합니다.

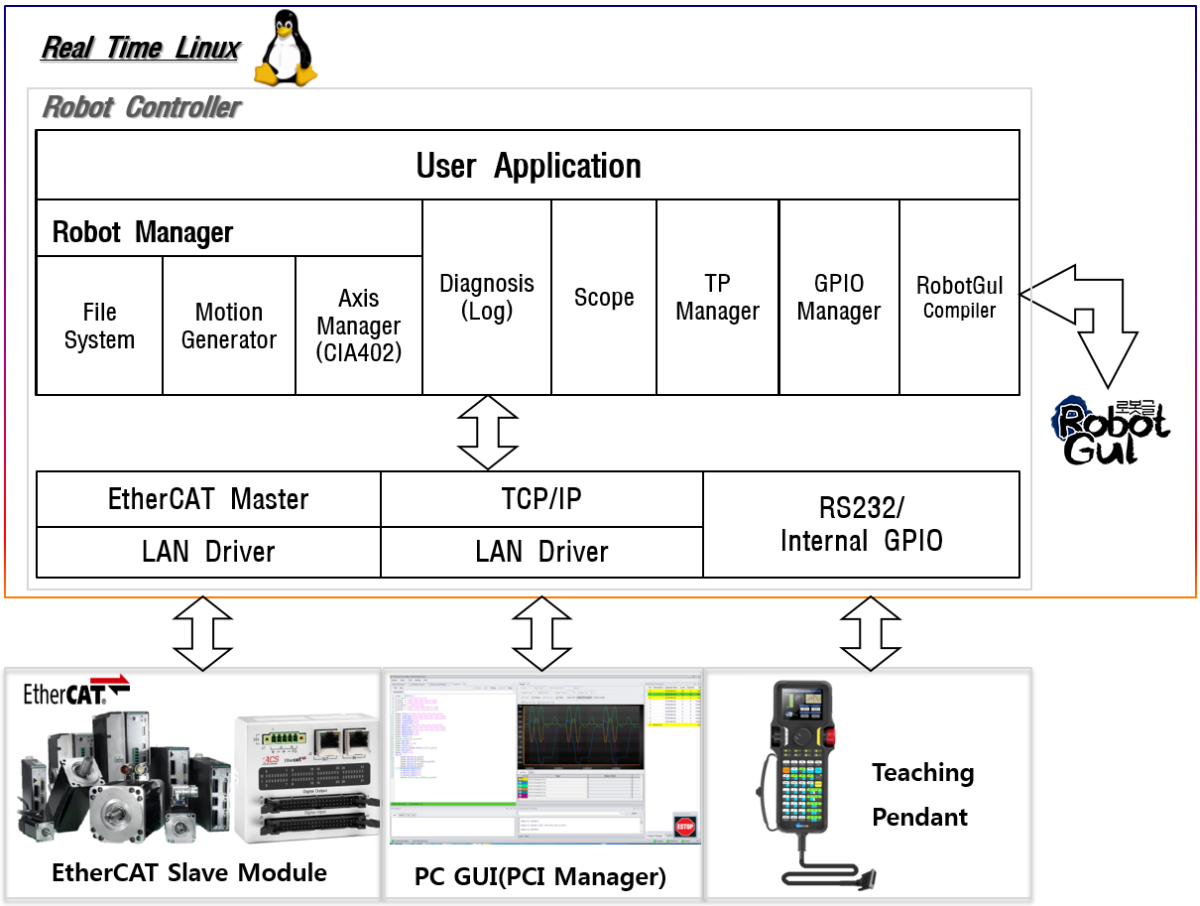


Figure 1 Software Architecture

1.2 Program Architecture

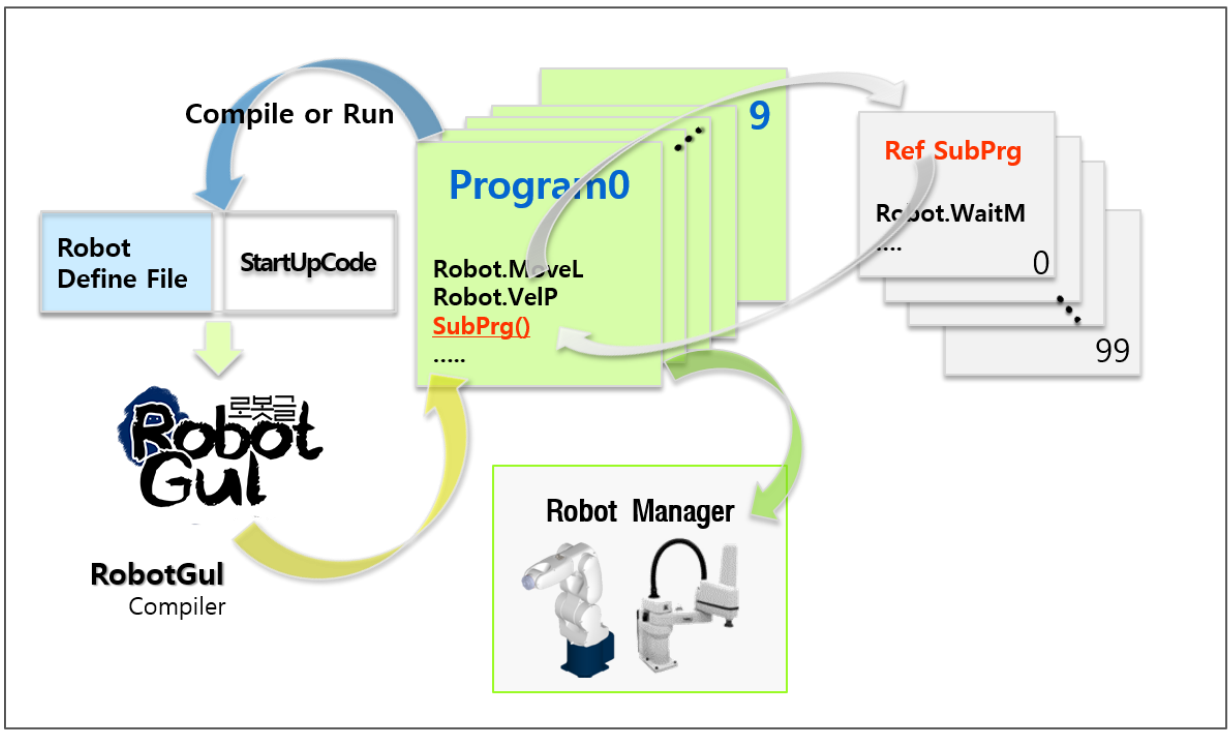


Figure 2 Program Architecture

○ 10 개의 독립적인 Program Task 지원

○ Program 에서 함수 호출처럼 사용할 수 있는 Sub Program 이 최대 100 개까지 지원

Chapter 2. Robot Motion

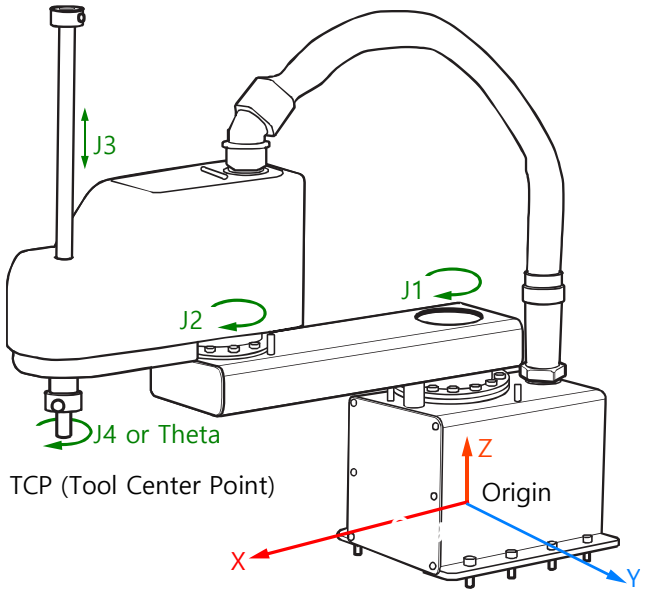
2.1 Motion Type & Mode

메뉴 및 구동 모드에 따라 Manual, Program(Auto), Homing Mode 로 구분 되고, 이동 축 방향에 따라 Joint, Work, Ready Motion 으로 구분됩니다.

1) Manual, Program(Auto), Homing Mode

- Manual: PRC Manager(GUI)의 Manual Motion Menu 를 통해 구동합니다.
- Program(Auto): PRC Manager 의 Program 에서 모션 명령어를 수행해서 구동합니다.
- Homing: PRC Manager 의 Homing Menu 를 통해 원점 복귀 모션을 수행합니다.

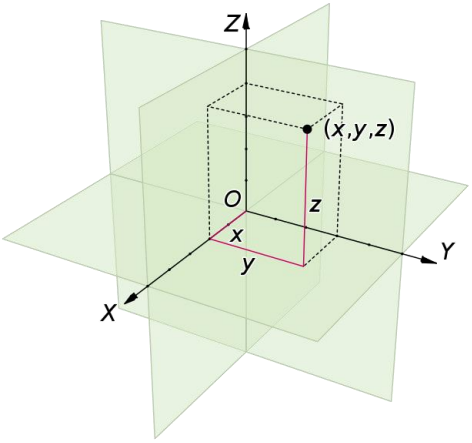
2) Joint, Work, Ready Motion



※ 위 그림은 3 차원 직교 좌표계를 World 좌표계로 사용하는 스카라 로봇입니다.

- **Joint(단축):** 축(Axis)별로 제어, 위 그림의 J1~J4 로 표시된 축 방향으로 이동하는 모션으로, 해당 축의 연결된 링크의 이동각도 또는 이동거리로 이동하는 모션을 말합니다.
- **Work:** World 좌표(직교좌표의 경우 X, Y, Z) 기준으로 로봇 제어, World 좌표로 TCP 위치를 이동하는 모션을 말합니다.
- **Ready:** System Configuration or Motion Parameter 에서 설정되어 있는 Ready 위치로 이동합니다.

2.2 Coordinate System



1) 직교 좌표(XYZ)

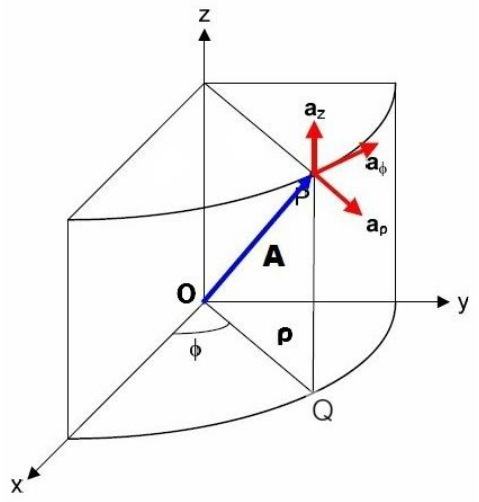
Coordinate	Cartesian	
	- (Negative)	+ (Positive)
X	-500.000	500.000
Y	-500.000	500.000
Z	-0.100	200.100

2) 원통 좌표

· ρ: x-y 평면에서 원점으로부터 떨어진 거리

· Φ: x 축 양의 방향으로부터 OQ 가 이루는 각도

· z: x-y 평면으로부터 높이



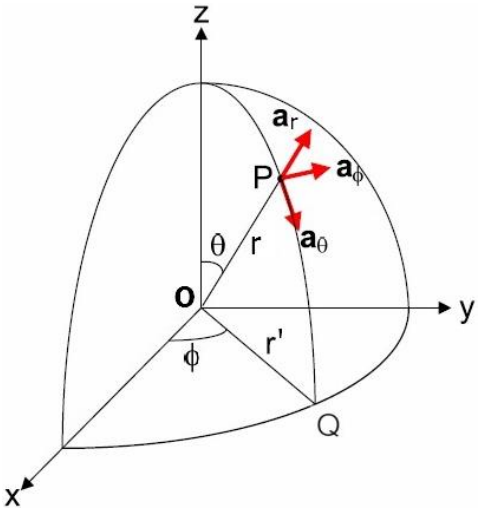
Coordinate	Cylindrical	
	- (Negative)	+ (Positive)
ρ	-500.000	500.000
ϕ	-500.000	500.000
z	-0.100	200.100

3) 구 좌표

· r: 원점에서 점 P 까지의 거리

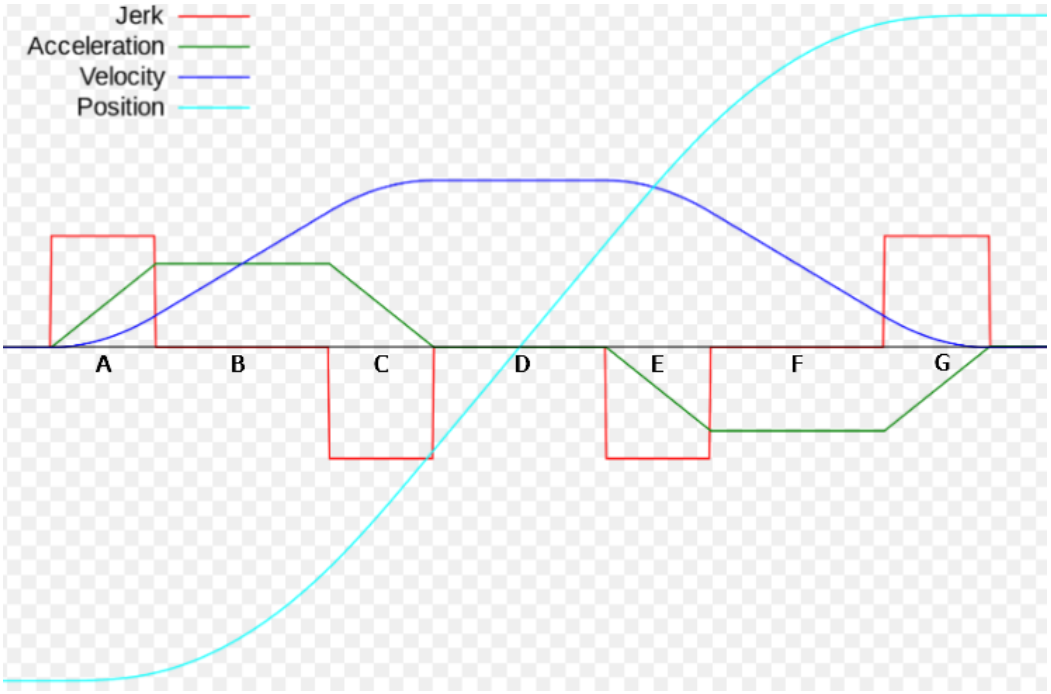
· θ: z 축 양의 방향으로부터 OP 가 이루는 각도

· Φ: x 축 양의 방향으로부터 OQ 가 이루는 각도



Coordinate	Spherical	
	- (Negative)	+ (Positive)
r	-500.000	500.000
θ	-500.000	500.000
ϕ	-0.100	200.100

2.3 Motion Profile

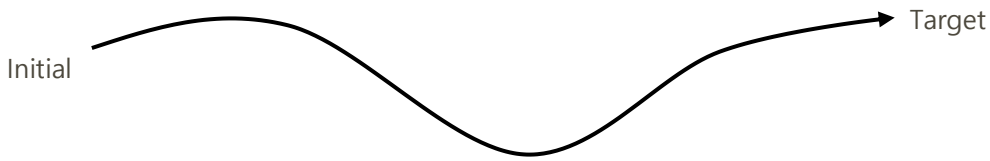


- **Velocity:** D 구간의 속도
- **Acceleration Time:** A+B+C 구간의 총 시간
- **Deceleration Time:** E+F+G 구간의 총 시간
- **Acceleration Jerk:** Acceleration Time 중 A+C 구간 시간의 백분율
- **Deceleration Jerk:** Deceleration Time 중 E+G 구간 시간의 백분율
- **Kill Time:** Motion 을 급속 정지할 때 정지까지 걸리는 시간

2.4 Motion 궤적

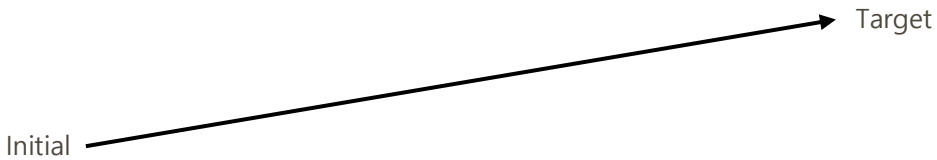
Move, MoveJ

현재 위치에서 Target Point 까지 Joint Motion 으로 움직입니다.
Move 는 각 축의 Motion Parameter 기준으로 구동되기 때문에 종료 시점이 다르고, MoveJ 는 가장 오랜 모션 시간을 가지는 축을 기준으로 보간 모션으로 구동됩니다.



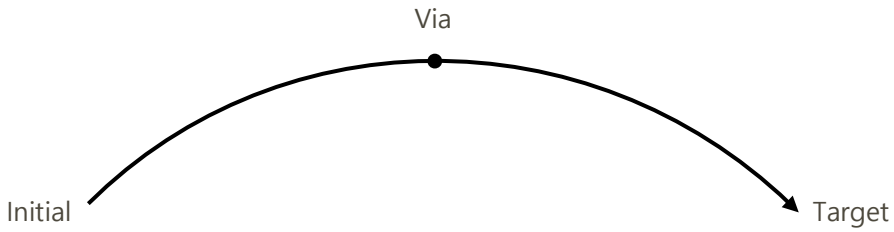
MoveL

현재위치에서 Target Point 로 직선 보간 이동합니다.



MoveC(원호보간)

Initial Point, Via Point, Target Point 는 원 위의 세 점이 되어 Motion 궤적을 만듭니다.



Chapter 3. Standard Information

3.1 Variable type

Item	Command	Description
Global variable	global	Program, Sub-program, Terminal 간에 공유되는 변수 타입
Local variable	local	프로그램 내의 한 Scope 에서만 사용할 수 있는 변수 타입

3.1.1 Global variable

Define

Program, Sub-program, Terminal 간에 공유되는 변수를 만들고 싶을 때 global 키워드를 사용합니다. Program 간에 공유되지만, 사용하기에 앞서 사용될 모든 Program 에서 global 키워드를 사용하여 동일한 변수를 선언해주어야 합니다.

Form

```
global VARIABLE_NAME;
```

Parameter

None

Return

None

Example

```
global intVal = 1;


Description
global 변수 intVal 을 선언함과 동시에 1 로 초기화
```

3.1.2 Local variable

Define

프로그램 내의 한 Scope 에서만 사용할 수 있는 변수 타입입니다. 다른 Scope 동일한 이름을 가진 변수일지라도 의미가 다를 수 있습니다.

Form

```
local VARIABLE_NAME;
```

Parameter

None

Return

None

Example

```
local doubleVal = 4.11;
local stringVal="Hello world";

Description
local 변수 doubleVal 을 선언함과 동시에 4.11 로 초기화
local 변수 stringVal 을 선언함과 동시에 "Hello world"로 초기화
```

3.2 Data type

Type	Domain	Example
Boolean	true, false	local a = true;
Integer	64-bit signed integer	local a = 123;
Double	C/C++ standard 8 bytes (64-bit)	local a = 123.456;
String	-	local a = "Hello world";

3.2.1 Array

Define

변수의 집합(배열)을 만듭니다.

Form

```
local VARIABLE_NAME = [arr1, arr2, arr3...arrN];
```

Parameter

None

Return

None

Example

```
local a = [1, 2, "string", true, 5.1]; // 배열 a 선언 및 초기화
a[1] = "123"; // 기존에 a[1]에 있던 정수 2 에 문자열 "123"을 대입
a[3] = 123; // 기존에 a[3]에 있던 boolean 형 TRUE 에 정수 123 을 대입
local b = a[3]; // 변수 b 에 정수 123 대입
```

Description

동일한 Data Type 을 가진 같은 크기의 배열이라면 연산이 가능합니다. (+, -, !=, ==)

3.2.2 2D Array

Define

2 차원 배열을 만듭니다.

Form

```
local VARIABLE_NAME = [[arr1, arr2], [arr3, arr4], [arr5,...arrN]];
```

Parameter

None

Return

None

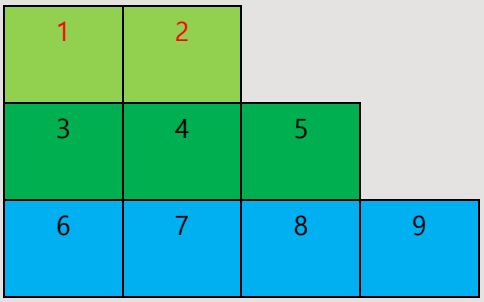
Example

```
local a = [[1, 2], [3,4], [5, 6]]; //Array 3 row x 2 col
local b = a[0];                    //b = [1, 2]
Print(b)

local a = [[1, 2], [3,4,5], [6,7,8,9]]; //Array with 3 row
local b0 = a[0];                    //b0 = [1, 2] size 2
local b1 = a[1];                    //b1 = [3, 4, 5] size 3
local b2 = a[2];                    //b1 = [6,7,8,9]] size 4
Print(b0);
Print(b1);
Print(b2);

local a = [[1, 2], [3,4,5], [6,7,8,9]];
a[1][2] = 20;            //Return: [[1, 2], [3, 4, 20], [6, 7, 8, 9]]
Print(a);

local a = [[1, 2], [3,4,5], [6,7,8,9]];
a[1] = [20, 30, 40, 50, 60];   //Return: [[1, 2], [20, 30, 40, 50, 60], [6, 7, 8, 9]]
```



3.2.3 CreateArray

Define

배열을 만듭니다.

Form

```
local VARIABLE_NAME = CreateArray(size1) // or CreateArray(size1, size2);
```

Parameter

Array Size

Return

Array Variable

Example

```
local a = CreateArray(3);
local b = CreateArray(3, 4);
a[1] = 10;
b[1][1] = 20;
Print(a);
Print(b);
```

Output:

```
 [0, 10, 0]  // a
 [[0, 0, 0, 0], [0, 20, 0, 0], [0, 0, 0, 0]] // b
```

3.3 Operator

Category	Precedence	Operator	Description	Example
Math arithmetic operator	1.	*^	Power	2*^3 = 8
	2.	*	Multiplication	2*3=6
		/	Division	4/3=1
		%	Remainder	5%3=2
	3.	+	Addition	5+3=8
		-	Subtraction	5-3=2
Bitwise shift operator	4.	<<	Bitwise left shift	1<<3 = 8
		>>	Bitwise right shift	8 >> 1 = 4
Relational Operator	5.	<	Less	1 < 2 (true)
		<=	Less equal	1 <= 2 (true)
		>	Greater	1 > 2 (false)
		>=	Greater equal	1 >= 2(false)
	6.	==	Equal	1 == 2 (false)
		!=	Not equal	1 != 2 (true)
Bitwise logical operator	7.	&	Bitwise AND	1 & 0 = 0
	8.	^	Bitwise XOR	1 ^ 0 = 1
	9.	|	Bitwise OR	1 | 0 = 1
Logical operator	10.	&&	Logical AND	true&&false = false
	11.	||	Logical OR	true||false = true

3.4 Comment

코드 안에 설명을 추가하기 위해 사용하며, 실제 코드 실행에는 아무런 영향을 주지 않습니다.

Item	Comment	Description
Line Comment	//	//로 시작한 한 줄이 주석으로 처리됩니다.
Block Comment	start with /* end with */	/* 부터 */ 까지의 모든 내용이 블록 주석으로 처리됩니다.

Example1

```
global a; //This is comment with global variable
local b; //This is comment with local variable
```

Example2

```
/*
What inside is block comment
*/
```

3.5 Sub-program

Define

사용자 함수를 만듭니다. 사용자 함수는 정상 컴파일 후 어떤 프로그램에서도 호출할 수 있습니다.

Form

```
sub subProgramName(arguments)
{
    return expression;
}
```

Parameter

arguments: Sub program 에서 사용할 인자 입력

Return

원하는 값을 반환 할 수 있습니다.

Example

```
-Example 1: sub program without return value
sub printSum(a, b)
{
    print(a + b);
}

-Example 2: sub program with return value
sub Sum(a, b)
{
    return a + b;
}
```

Description

각 서브 프로그램 파일에는 모든 프로그램의 무결성을 보장하기 위해 하나의 서브 프로그램 이름만 포함할 수 있습니다.

서브 프로그램은 다른 서브 프로그램에서 호출할 수 없습니다.

Chapter 4. Pre-defined Commands

○ Pre-defined Command 는 System, Robot Command 로 구분됩니다.

○ PRC Series 의 경우 최대 10 대 로봇이 지원되기 때문에 Robot Command 는 Robot Class 의 멤버 함수(Method)로 정의되어 있습니다.

○ 멤버 함수의 사용

Robot[0].MoveL(WkPos[0]) // 0 번째 Robot 이 WkPos[0] 위치로 직선 보간(MoveL) 명령 수행

4.1 System Commands

Item	Command	Description
System Control	Date	현재 시스템 날짜 읽기
	Time	현재 시스템 시간 읽기
	SetDateTime	현재 시스템 시간 설정
	TimerRead	타이머 시간 읽기
	TimerStart	타이머 기능 시작
	TimerStop	타이머 기능 정지
	Print	입력한 내용을 User Log Message 창에 출력
	Wait	지정한 시간(msec)만큼 대기
	ToInt	정수형 외의 타입을 정수형으로 변환
	ToString	문자열 외의 타입을 문자열로 변환
	SubString	문자열의 일부분을 출력
	batch	여러 명령어를 동시(1 Cycle time)에 실행
	HexStringToDec	16 진 문자열을 10 진수로 변경
	HexStringToDec	10 진수를 16 진 문자열으로 변경
	IsNull	변수가 초기화 되었는지 체크
Program	PrgPause	프로그램 일시 정지
	PrgResume	프로그램 재 시작
	PrgStart	프로그램 시작

	PrgStop	프로그램 정지
	PrgState	프로그램의 현재 상태
System IO	DIn	Digital Input
	DOut	Digital Output
	AIn	Analog Input
	AOut	Analog Output

4.1.1 System Control

4.1.1.1 Date()

Define

현재 시스템의 날짜를 가져올 때 사용됩니다.

Form

```
Date();
```

Parameter

None

Return

현재 시스템의 날짜 String 리턴합니다.

Example

```
local strDate = Date();
Print(strDate);
```

Description

시스템의 현재 날짜를 User Log Message 창에 출력합니다.

2019-2-21

4.1.1.2 Time()

Define

현재 시스템의 시간을 가져올 때 사용됩니다.

Form

```
Time();
```

Parameter

None

Return

현재 시스템의 시간 String 리턴합니다.

Example

```
strTime = Time();
Print(strTime);

Description
시스템의 현재 시간을 User Log Message 창에 출력합니다.
3:33:45
```

4.1.1.3 SetDateTime("timestring")

Define

현재 시스템의 시간을 설정합니다.

Form

SetDateTime("timestring");

Parameter

Time string

Return

None

Example

SetDateTime("2020-04-07 14:08:11")

Description

시스템의 현재 시간을 2020 년 04 월 07 일 14 시 08 분 11 초로 설정합니다.

4.1.1.4 TimerRead(timerID)

Timer 함수는 명령수행 시간을 측정할 때 유용하게 사용할 수 있습니다.

Define

지정한 타이머의 시간을 읽어올 때 사용됩니다.

Form

```
TimerRead(id);
```

Parameter

id (default = 0)

Return

Timer Start 부터 Stop 까지 걸린 시간을 msec 로 반환합니다.

Example

```
TimerStart(); // 0 번 Timer 시작
Robot[0].MoveL(Robot[0].WkPos[0]);
TimerStop(); // 0 번 Timer 정지
strTime = TimerRead();
Print(strTime); // 0 번 Timer 시간 읽기
```

Description

id 를 지정하지 않으면 기본 0 번입니다.

4.1.1.5 TimerStart(timerID)

Define

타이머 기능을 시작하기 위한 명령어입니다.

Form

```
TimerStart(id);
```

Parameter

id (default = 0)

Return

None

Example

```
TimerRead(id)의 예시와 동일
```

Description

4.1.1.6 TimerStop(timerID)

Define

타이머 기능을 멈추는 명령어입니다.

Form

```
TimerStop(id);
```

Parameter

id (default = 0)

Return

None

Example

TimerRead(id)의 예시와 동일

Description

4.1.1.7 Print(arg1, arg2..., argN)

Define

입력한 내용을 User Log Message 창에 출력

Form

```
Print(arg1, arg2..., argN);
```

Parameter

All data type

Return

None

Example

```
local a = 1;
Print("Test :", a+1);
```

Description

Test : 2

comma(,)로 나누어진 인자 사이에는 자동으로 공백이 추가됩니다.

4.1.1.8 Wait(time)

Define

프로그램을 지정한 시간(msec)만큼 대기시킵니다.

Form

```
Wait(time);
```

Parameter

time(msec): 상수 또는 변수

Return

None

Example

```
TimerStart(1);
Wait(1000);
TimerStop(1);
Print(TimerRead(1));

Description
User Log Message 창에 출력 결과: 1000.632065 msec ....
// 1 초가 Wait 명령어에 의한 지연이고, 나머지는 명령어가 수행될 때 걸리는 시간입니다
```

4.1.1.9 ToInt(argument)

Define

정수형이 아닌 Data type 을 정수형로 변환하여 반환합니다.

Form

```
ToInt(argument);
```

Parameter

실수, 문자, 문자열, 변수

Return

Integer 형으로 반환합니다.

Example

```
local ver = 1.048;
Print(2+ToInt(1.048));
```

Description

결과: 3

Double type 인 ver 변수를 ToInt 함수의 인자로 넣어 정수형으로 변환하면 숫자 연산을 할 수 있습니다.

4.1.1.10 ToString(argument)

Define

문자열이 아닌 Data type 을 문자열로 변환하여 반환합니다.

Form

```
ToString(argument);
```

Parameter

정수, 실수, 문자, 변수

Return

String 형으로 반환합니다.

Example

```
local ver = 1.048;
Print("PRC GUI Version : " + ToString(ver));
```

Description

Double type 인 ver 변수를 ToString 함수의 인자로 넣어 문자열로 변환하면 문자열 연산(+)을 할 수 있습니다.

4.1.1.11 SubString(string, index, size)

Define

문자열의 일부분을 출력할 때 사용합니다.

Form

```
SubString(string, index, size);
```

Parameter

string: 문자열

index: 문자열 중 원하는 일부분의 시작 지점

size: index 를 시작으로 원하는 일부분의 끝 지점까지의 크기

Return

index 부터 size 만큼의 String 이 반환됩니다.

Example

```
local a = "PRESTO SOLUTION";
Print(SubString(a, 7, 8));
```

Description

결과: SOLUTION

문자열의 맨 앞이 index 0 이며, 공백도 index 를 차지합니다.

4.1.1.12 batch

Define

여러 명령어를 동시(1 Cycle time)에 실행할 때 사용됩니다.

Form

```
batch{
    ……
}
```

Parameter

None

Return

None

Example

```
batch{
    TimerStart();
    Robot[0].On();
    Robot[1].On();
    TimerStop();
}
TimerRead();
```

Description

batch 내의 명령어 실행 시간이 설정한 Cycle time 을 초과하면 에러가 발생합니다.

Robot 또는 IO 를 동시에 제어할 때 사용할 수 있습니다.

4.1.1.13 HexStringToDec(“stringInHexa”)

Define

16 진 문자열을 10 진수로 변경할 때 사용됩니다.

Form

HexStringToDec(“stringInHexa”)

Parameter

None

Return

Integer Type

Example

```
local a = HexStringToDec("0x55")   // Return 85
```

4.1.1.14 DecToHexString(number)

Define

10 진수를 16 진 문자열으로 변경할 때 사용됩니다.

Form

```
DecToHexString(number);
```

Parameter

None

Return

Integer Type

Example

```
local a = DecToHexString(85); // Return "0x55"
```

4.1.1.15 IsNull(ariable)

Define

변수가 초기화 되었는지 체크할 때 사용됩니다. 여러 프로그램에서 쓰이는 전역 변수를 체크할 때 유용합니다.

Form

```
IsNull(Variable);
```

Parameter

Variable

Return

Boolean Type

Example

program 1 source code	Program 1 source code

global g; //Not initialized
 	
while (IsNull(g)) {Wait(100);} //Wait another program init this variable.
 	
while (true) {
 	
g = g + 1; //write g
 	
Wait(1000);
 	
}
	global g;
	
	g = 1; //Init g
	
	while (true){
	
	Print(g); //Read g
	
	Wait(1000);
	
	}

4.1.2 Program

4.1.2.1 PrgPause(PrgNum)

Define

현재 실행되고 있는 해당 프로그램을 일시 정지시킬 때 사용합니다.

Form

```
PrgPause(PrgNum);
```

Parameter

PrgNum: Program Number(0~9)

Return

ERR_NONE(0): 에러가 없을 경우

ERR_NUM(Error Code): 에러가 있을 경우

Example

```
PrgPause(1);
```

Description

프로그램 1 번을 일시 정지시킵니다.

4.1.2.2 PrgResume(PrgNum)

Define

PrgPause 에 의해 일시 정지된 해당 프로그램일 재 시작할 때 사용합니다.

Form

```
PrgResume(PrgNum);
```

Parameter

PrgNum: Program Number(0~9)

Return

ERR_NONE(0): 에러가 없을 경우

ERR_NUM(Error Code): 에러가 있을 경우

Example

```
PrgResume(1);
```

Description

프로그램 1 번을 재 시작합니다.

4.1.2.3 PrgStart(PrgNum)

Define

해당 프로그램을 시작할 때 사용합니다.

Form

```
PrgStart(PrgNum);
```

Parameter

PrgNum: Program Number(0~9)

Return

ERR_NONE(0): 에러가 없을 경우

ERR_NUM(Error Code): 에러가 있을 경우

Example

```
PrgStart(4);
```

Description

프로그램 4 번을 시작합니다.

4.1.2.4 PrgStop(PrgNum)

Define

실행하고 있는 해당 프로그램을 정지할 때 사용합니다.

Form

```
PrgStop(PrgNum);
```

Parameter

PrgNum: Program Number(0~9)

Return

ERR_NONE(0): 에러가 없을 경우

ERR_NUM(Error Code): 에러가 있을 경우

Example

```
a = 3;
PrgStop(a);
```

Description

프로그램 3 번을 정지시킵니다.

4.1.2.4 PrgState(PrgNum)

Define

해당 프로그램의 현 상태를 나타냅니다.

Form

```
PrgState(PrgNo);
```

Parameter

PrgNo: Program Number(0~9)

Return

프로그램의 상태를 Integer 값으로 반환합니다.

Example

```
PrgStart(5);
Print(PrgState(5));
```

Description

결과 : 2 (Program Running 상태)

프로그램 5 번의 상태를 나타냅니다.

```
NONE = 0
START = 1
RUN = 2
STOP = 3
PAUSE = 4
END = 5
COMPILED = 6
COMPILE ERROR = 7
RUNTIME ERROR = 8
```

4.1.3 System IO

4.1.3.1 DIn[portIndex][bitIndex]

Define

Digital Input 값을 가져올 때 사용합니다. (read only)

Form

```
DIn[portIndex];
DIn[portIndex][bitIndex];
```

Parameter

portIndex: System 에 연결된 Digital Input Module Number, EtherCAT Slave Scan 시 연결된 순서대로 0 부터 자동으로 값이 주어집니다.

bitIndex: 해당 Module 의 Channel Number 입니다. (0~31)

Return

현재 Digital Input 값을 반환합니다.

Example

```
local dIn = DIn[0];   // 0 번째 Digital Input Module 의 값을 dIn 변수에 저장

local dInBit = DIn[0][1];  // 0 번째 Digital Input Module 의 1 번째 Channel(bit)의 값을 dInBit 에 저장.
```

Description

4.1.3.2 DOut[portIndex][bitIndex]

Define

Digital Output 값을 설정하거나 현재 값을 가져올 때 사용합니다.

Form

```
DOut[portIndex];
DOut[portIndex][bitIndex];
```

Parameter

portIndex: System 에 연결된 Digital Output Module Number, EtherCAT Slave Scan 시 연결된 순서대로 0 부터 자동으로 값이 주어집니다.

bitIndex: 해당 Module 의 Channel Number 입니다. (0~31)

Return

현재 Digital Output 값을 반환합니다.

Example

```
dOut = DOut[0];  // 0 번째 Digital Output Module 의 값을 dOut 변수에 저장

dOutBit = DOut[0][1]; // 0 번째 Digital Output Module 의 1 번째 Channel(bit)의 값을 dOutBit 에 저장

DOut[1] = 255;  // 1 번째 Digital Output Module 의 값을 255 으로 설정
DOut[2][3] = 1;   // 2 번째 Digital Output Module 의 3 번째 Channel(bit) 값을 1 로 설정
```

Description

4.1.3.3 AIn[index]

Define

Analog Input 값을 가져올 때 사용합니다. (read only)

Form

```
AIn[index];
```

Parameter

index: System 에 연결된 Analog Input Module Channel 들을 EtherCAT Slave Scan 시 순서대로 0 부터 자동으로 값이 주어집니다.

예를 들어 IOMps-EA0400(4 개 Channel 이 있음) 2 개의 Slave 를 사용할 경우, 1 번째 제품의 Channel 이 0~3 까지 주어지고, 두 번째 연결된 제품의 Channel 이 4~7 로 주어집니다.

Return

현재 Analog Input 값을 반환합니다.

Example

```
local aIn = AIn[0];   // 0 번째 Analog Input Channel 의 값을 aIn 변수에 저장
```

Description

4.1.3.4 AOut[index]

Define

Analog Output 값을 설정하거나 현재 값을 가져올 때 사용합니다.

Form

```
AOut[index];
```

Parameter

index: System 에 연결된 Analog Input Module Channel 들을 EtherCAT Slave Scan 시 순서대로 0 부터 자동으로 값이 주어집니다.
예를 들어 IOMps-EA0004(4 개 Channel 이 있음) 2 개의 Slave 를 사용할 경우, 1 번째 제품의 Channel 이 0~3 까지 주어지고, 두 번째 연결된 제품의 Channel 이 4~7 로 주어집니다.

Return

현재 Analog Output 값을 반환합니다.

Example

```
local aOut = AOut[0];   // 0 번째 Analog Output Channel 의 값을 aOut 변수에 저장.

local AOut[1] = 8196;  // 1 번째 Analog Output Channel 을 8196 값으로 설정
```

Description

4.2 Robot Commands

Item	Command	Description
Robot Class	Robot	Robot 객체에 접근할 때 사용 (모든 Robot Command 는 Robot 객체의 메소드)
Robot Status	Status	Robot 의 동작 상태를 나타냅니다.
	IsOn	Robot 의 전원이 켜져 있는지 체크
	IsMoving	Robot 이 움직이고 있는 상태인지 체크
	IsAlarm	Robot 에 Alarm 이 있는지 체크
Power (Servo ON/OFF)	On	Robot 에 연결된 모든 Motor Drive On.
	Off	Robot 에 연결된 모든 Motor Drive Off.
Motion	Move	현 위치에서 목표위치로 직선 이동. * 목표 위치는 Joint Position 으로 인식하고 Joint Motion 으로 이동
	MoveW	Move 동작에 WaitM 이 결합된 형태
	MoveJ	현 위치에서 목표위치로 PTP 이동 * 목표 위치는 Joint Position 으로 인식하고 Joint Motion 으로 이동
	MoveJW	MoveJ 동작에 WaitM 이 결합된 형태
	MoveL	현 위치에서 목표위치로 직선 보간 * 목표 위치는 Work Position 으로 인식하고 Work Motion 으로 이동
	MoveLW	MoveL 동작에 WaitM 이 결합된 형태
	MoveC	현 위치에서 경유위치를 지나 목표위치로 원호 보간 경유 및 목표위치는 Work Position 으로 주어지고, Work Motion 으로 이동
	MoveCW	MoveC 동작에 WaitM 이 결합된 형태

	MoveR	현 위치에서 Ready 위치로 이동. * 목표 위치는 Joint Position 으로 인식하고 Ready Motion 으로 이동
	MoveRW	MoveR 동작에 WaitM 이 결합된 형태
	MoveRel	현 위치에서 입력된 위치만큼 상대 위치로 직선 이동 * 목표 위치는 Joint Position 으로 인식하고 Joint Motion 으로 이동
	MoveRelW	MoveRel 동작에 WaitM 이 결합된 형태
	MoveJRel	현 위치에서 입력된 위치만큼 상대 위치로 PTP 이동 * 목표 위치는 Joint Position 으로 인식하고 Joint Motion 으로 이동
	MoveJRelW	MoveJRel 동작에 WaitM 이 결합된 형태
	MoveLRel	현 위치에서 입력된 위치만큼 상대 위치로 직선 보간 이동 * 목표 위치는 Work Position 으로 인식하고 Work Motion 으로 이동.
	MoveLRelW	MoveLRel 동작에 WaitM 이 결합된 형태
	MoveTX	현 위치에서 입력된 위치만큼 Tool X 축 상대 위치로 이동 * 목표 위치는 Work X 축 Position 으로 인식하고 Work Motion 으로 이동.
	MoveTXW	MoveTX 동작에 WaitM 이 결합된 형태
	MoveTY	현 위치에서 입력된 위치만큼 Tool Y 축 상대 위치로 이동 * 목표 위치는 Work Y 축 Position 으로 인식하고 Work Motion 으로 이동.
	MoveTYW	MoveTY 동작에 WaitM 이 결합된 형태
	MoveTZ	현 위치에서 입력된 위치만큼 Tool Z 축 상대 위치로 이동 * 목표 위치는 Work Z 축 Position 으로 인식하고 Work Motion 으로 이동.

	MoveTZW	MoveTZ 동작에 WaitM 이 결합된 형태
	MoveX	현 위치에서 입력된 위치만큼 X 축 상대 위치로 이동 * 목표 위치는 Work X 축 Position 으로 인식하고 Work Motion 으로 이동.
	MoveXW	MoveX 동작에 WaitM 이 결합된 형태
	MoveY	현 위치에서 입력된 위치만큼 Y 축 상대 위치로 이동 * 목표 위치는 Work Y 축 Position 으로 인식하고 Work Motion 으로 이동.
	MoveYW	MoveY 동작에 WaitM 이 결합된 형태
	MoveZ	현 위치에서 입력된 위치만큼 Z 축 상대 위치로 이동 * 목표 위치는 Work Z 축 Position 으로 인식하고 Work Motion 으로 이동.
	MoveZW	MoveZ 동작에 WaitM 이 결합된 형태
	StartBlend EndBlend	여러 모션 명령을 중첩(Blending)하여 이동하는 명령어 StartBlend: Blending 모션의 시작. EndBlend: Blending 모션의 끝.
	MoveStop	수행 중인 로봇 모션을 정지시킵니다.
	MoveEStop	모션중인 로봇을 비상 정지 시킵니다. # 비상정지 감속시간은 JtKTime or WkKTime 을 참조합니다.
	WaitM	진행중인 모션이 끝난 후 지정한 시간(msec)만큼 대기
	JtJogP	원하는 축을 Joint 좌표계 기준으로 Positive 조그 모션
	JtJogN	원하는 축을 Joint 좌표계 기준으로 Negative 조그 모션
	WkJogP	원하는 축을 Work 좌표계 기준으로 Positive 조그 모션
	WkJogN	원하는 축을 Work 좌표계 기준으로 Negative 조그 모션
	MoveH	모든 축 Homing 모션

Motion Parameter	Joint	JtAJerkP	가속구간의 Jerk(가가속) 범위 (Unit: %)
		JtATime	가속시간 (Unit: msec)
		JtVel	속도(Unit: mm/sec or °/s)
		JtDTime	감속시간 (Unit: msec)
		JtDJerkP	감속구간의 Jerk(가가속) 범위 (Unit: %)
		JtKTime	급속정지(MoveEStop)시 감속시간 (Unit: msec)
		JtCPos	현재 지령 위치 값
		JtFPos	현재 실제 모터 위치(Feedback Encoder)값
	Work	WkAJerkP	가속구간의 Jerk(가가속) 범위 (Unit: %)
		WkATime	가속시간 (Unit: msec)
		WkVel	속도 (Unit: mm/sec or °/s)
		WkDTime	감속시간 (Unit: msec)
		WkDJerkP	감속구간의 Jerk(가가속) 범위 (Unit: %)
		WkKTime	급속정지(MoveEStop)시 감속시간 (Unit: msec)
		WkCPos	현재 지령 위치 값
		WkFPos	현재 실제 모터 위치 (Feedback Encoder)값
	Ready	RdAJerkP	가속구간의 Jerk(가가속) 범위 (Unit: %)
		RdATime	가속시간 (Unit: msec)
		RdVel	속도 (Unit: mm/sec or °/s)
		RdDTime	감속시간 (Unit: msec)
		RdDJerkP	감속구간의 Jerk(가가속) 범위 (Unit: %)
	Velocity	VelP	구동 속도의 Percent (Unit: %) # 실제 로봇 구동 속도는 Vel(JtVel, WkVel, RdVel) × VelP 로 구동
Non-Volatile Variable (Store in Flash)		JtPos	해당 로봇의 Joint Position 저장하는 비 휘발성 변수
		WkPos	해당 로봇의 Work Position 저장하는 비 휘발성 변수

	IVar	Integer(정수)값을 저장할 수 있는 비 휘발성 변수
	DVar	Double(실수)값을 저장할 수 있는 비 휘발성 변수
	JogVelP	Jog 시, 구동 속도의 Percent(%)
	JtJogVel	Jog 시, Joint 좌표계의 구동 속도(mm/s)
	WkJogVel	Jog 시, Work 좌표계의 구동 속도(mm/s)
	JtJogATime	Jog 시, Joint 좌표계의 가속 시간(msec)
	WkJogATime	Jog 시, Work 좌표계의 가속 시간(msec)
	JtJogDTime	Jog 시, Joint 좌표계의 감속 시간(msec)
	WkJogDTime	Jog 시, Work 좌표계의 감속 시간(msec)

4.2.1 Robot Class

4.2.1.1 Robot[index].Command(argument)

Define

Robot Class 에 접근하여 다양한 메소드를 이용할 수 있습니다.

Form

```
Robot[index].Command(argument);
```

Parameter

Index: 로봇 Index 로 0 부터 9 까지 총 10 개의 값이 주어집니다.

Return

None

Example

```
Robot[0].On(); // 0 번 로봇의 모든 Motor Drive On


Description
index 를 입력하지 않으면 0 번 로봇을 칭합니다.
ex)
   Robot.On(); // 0 번 로봇의 모든 Motor Drive On
```

4.2.2 Robot Status

4.2.2.1 Status()

Define

로봇의 상태를 알 수 있습니다.

Status 배열엔 4 개의 요소가 아래와 같은 순서로 저장됩니다.

[Robot On , Robot Run , Robot Alarm , Robot Homing]

① Robot On : 서보 드라이브 On(1)/Off(0)
② Robot Run : 로봇 움직임(1)/멈춤(0)
③ Robot Alarm : 서보 드라이브 에러 있음(1)/없음(0)
④ Robot Homing :

로봇의 상태가 업데이트 되기까지에는 최소 1 Cycle Time 이 필요합니다.

Form

```
Status();
```

Parameter

None

Return

4 개의 요소를 가지고 있는 Integer 배열을 반환합니다.

Example

```
local status; // 로봇의 상태를 저장하기 위한 변수 선언
Robot[0].MoveJ(100); // 단축 로봇 움직임
Wait(1); // CycleTime 을 1ms 로 설정했을 때, 로봇의 상태를 업데이트 하기 위해 필요한 딜레이 1ms
status = Robot[0].Status(); // 0 번 로봇의 상태를 가져옵니다. ex) 로봇 작동 중 [1,1,0,0]

Print(status); // [1,1,0,0] 출력
```

```
if(status[0] == 1) // status 배열의 첫번째 요소(Robot On)의 상태를 확인합니다.
{
    Print("Robot 0 is On"); // 0 번 로봇이 On 상태이면 이 부분을 출력합니다.
}
else
{
    Print("Robot 0 is Off"); // 0 번 로봇이 Off 상태이면 이 부분을 출력합니다.
}
```

Description

4.2.2.2 IsOn()

Define

Robot 의 전원이 켜져 있는지 체크합니다.

Form

```
IsOn();
```

Parameter

None

Return

Boolean Type

Example

```
Robot[0].On();      //Command

//Wait until Robot Really Power On
while (!Robot[0].IsOn()) { Wait(100); }
```

4.2.2.3 IsMoving()

Define

Robot 이 움직이는 상태인지 체크합니다.

Form

```
IsMoving();
```

Parameter

None

Return

Boolean Type

Example

```
Robot[0].MoveJ([10, 20]);

while (Robot[0].IsMoving))
{
        Wait(100);
        Print("Robot is moving");
}
```

4.2.2.4 IsAlarm()

Define

Robot 에 Alarm 이 있는지 체크합니다.

Form

```
IsAlarm();
```

Parameter

None

Return

Boolean Type

Example

```
local isAlarm = Robot[0].IsAlarm();

if(isAlarm == true)
{
    Print("The Robot has an alarm.");
}
```

4.2.3 Power

4.2.3.1 On()

Define

Robot 에 연결된 모든 Motor Drive 를 Servo On 합니다.

Form

```
On();
```

Parameter

None

Return

ERR_NONE(0): 에러가 없을 경우

ERR_NUM(Error Code): 에러가 있을 경우

Example

```
Robot[1].On() // 1 번째 로봇에 연결된 모든 Motor Drive On
```

Description

로봇에 연결된 모든 Drive 가 On 된 상태여야 Motion 구동이 가능합니다.

4.2.3.2 Off()

Define

Robot 에 연결된 모든 Motor Drive 를 Servo Off 합니다.

Form

```
Off();
```

Parameter

None

Return

ERR_NONE(0): 에러가 없을 경우

ERR_NUM(Error Code): 에러가 있을 경우

Example

```
Robot[0].Off() //0 번째 로봇에 연결된 모든 Motor Drive Off
```

Description

4.2.4 Motion

4.2.4.1 Move(JointPos)

Define

이 명령은 로봇의 TCP(Tool Center Point)를 입력한 목표 위치로 PTP 이동을 시킬 때 사용합니다.
Joint Motion 으로 Joint Motion Parameter 값을 참조하여 모션 경로를 생성합니다.

MoveJ 와 차이점

Move 는 각 축 마다 설정된 Variables 에 의해 구동되며, 모션 종료 시점이 축 마다 다를 수 있습니다.
MoveJ 는 모든 축이 동기화 되어 모션이 동시에 시작하고 동시에 종료됩니다.

Form

```
Move(JointPos);
```

Parameter

JointPos: 각 축마다 Target Point 의 Joint 절대 위치 List or Array 형의 Variable

Return

ERR_NONE(0): 에러가 없을 경우
ERR_NUM(Error Code): 에러가 있을 경우

Example

```
local JointPos = [A1, A2, .. An];
Robot[0].Move(JointPos); // Robot[0].Move([A1, A2, .. An]);
```

Description

4.2.4.2 MoveJ(JointPos)

Define

이 명령은 로봇의 TCP(Tool Center Point)를 입력한 목표 위치로 PTP 이동을 시킬 때 사용합니다. Joint Motion 으로 Joint Motion Parameter 값을 참조하여 모션 경로를 생성되며, 모든 축이 동기화 되어 모션이 동시에 시작하고 동시에 종료됩니다.

Form

```
MoveJ(JointPos);
```

Parameter

JointPos: 각 축마다 Target Point 의 Joint 절대 위치 List or Array 형의 Variable

Return

ERR_NONE(0): 에러가 없을 경우
ERR_NUM(Error Code): 에러가 있을 경우

Example

```
JointPos = [A1, A2, .. An];
Robot[0].MoveJ([A1, A2, .. An]);
Robot[0].MoveJ{JointPos};
```

Description

4.2.4.3 MoveL(WorkPos)

Define

이 명령은 로봇의 TCP(Tool Center Point)를 입력한 목표 위치로 직선 보간 이동을 시킬 때 사용합니다.
Work Motion 으로 Work Motion Parameter 값을 참조하여 모션 경로를 생성됩니다.

Form

```
MoveL(WorkPos);
```

Parameter

WorkPos: 각 축마다 Target Point 의 Work 절대 위치 List or Array 형의 Variable

Return

ERR_NONE(0): 에러가 없을 경우
ERR_NUM(Error Code): 에러가 있을 경우

Example

```
local WorkPos = [X, Y, .. Tz];
Robot[0].MoveL(WorkPos);
```

Description

4.2.4.4 MoveC(ViaWorkPos, TargetWorkPos)

Define

이 명령은 로봇의 TCP(Tool Center Point)를 입력한 경유 위치를 지나 목표 위치로 원호 보간 이동을 시킬 때 사용합니다. Work Motion 으로 Work Motion Parameter 값을 참조하여 모션 경로를 생성됩니다.

Form

```
MoveC(ViaWorkPos, TargetWorkPos);
```

Parameter

ViaWorkPos: 각 축마다 Target Point 의 Work 절대 위치 List, Array 형 Variable

TargetWorkPos: 각 축마다 Via Point 의 Work 절대 위치 List, Array 형 Variable

Return

ERR_NONE(0): 에러가 없을 경우

ERR_NUM(Error Code): 에러가 있을 경우

Example

```
local ViaWorkPos = [X1, Y1, .. Tz1];
local TargetWorkPos = [X2, Y2, .. Tz2];
Robot[0].MoveC(ViaWorkPos, TargetWorkPos);
```

Description

4.2.4.5 MoveR()

Define

이 명령은 로봇의 TCP(Tool Center Point)를 설정된 준비위치(=RdPos)로 PTP 이동 시킬 때 사용합니다. Ready Motion 으로 Ready Motion Parameter 값을 참조하여 모션 경로를 생성됩니다.

Form

```
MoveR();
```

Parameter

None

Return

ERR_NONE(0): 에러가 없을 경우
ERR_NUM(Error Code): 에러가 있을 경우

Example

```
Robot[0].MoveR();
```

Description

4.2.4.6 MoveRel(dJointPos)

Define

이 명령은 Move 의 응용 모션이며, 로봇의 TCP(Tool Center Point)를 입력한 목표 위치로 PTP 이동을 시킬 때 사용합니다. Joint Motion 으로 Joint Motion Parameter 값을 참조하여 모션 경로를 생성합니다.

Rel(Relative Position): 상대 위치를 의미합니다.

Form

```
Move(dJointPos);
```

Parameter

dJointPos: 각 축마다 Target Point 의 Joint 상대 위치 List or Array 형의 Variable

Return

ERR_NONE(0): 에러가 없을 경우

ERR_NUM(Error Code): 에러가 있을 경우

Example

```
local dJointPos = [10, 0, 0, 0];
Robot[0].MoveRel(dJointPos);
```

Description

만약에 현재 로봇 위치가 [300, 200, 10, 50] 이였다면, Robot[0].MoveRel(dJointPos) 명령어를 수행할 경우 로봇은 [310, 200, 10, 50] 위치가 됩니다.

4.2.4.7 MoveJRel(JointPos)

Define

이 명령은 MoveJ 의 응용 모션이며, 로봇의 TCP(Tool Center Point)를 입력한 목표 위치로 PTP 이동을 시킬 때 사용합니다. Joint Motion 으로 Joint Motion Parameter 값을 참조하여 모션 경로를 생성되며, 모든 축이 동기화 되어 모션이 동시에 시작하고 동시에 종료됩니다.

Form

```
MoveJRel(JointPos);
```

Parameter

JointPos: 각 축마다 Target Point 의 Joint 상대 위치 List or Array 형의 Variable

Return

ERR_NONE(0): 에러가 없을 경우
ERR_NUM(Error Code): 에러가 있을 경우

Example

```
local dJointPos = [A1, A2, .. An];
Robot[0].MoveJRel(dJointPos);
```

Description

4.2.4.8 MoveLRel(dWorkPos)

Define

이 명령은 MoveL 의 응용 모션이며, 로봇의 TCP(Tool Center Point)를 입력한 목표 위치로 직선 보간 이동을 시킬 때 사용합니다. Work Motion 으로 Work Motion Parameter 값을 참조하여 모션 경로를 생성됩니다.

Form

```
MoveLRel(dWorkPos);
```

Parameter

dWorkPos: 각 축마다 Target Point 의 Work 상대 위치 List or Array 형의 Variable

Return

ERR_NONE(0): 에러가 없을 경우
ERR_NUM(Error Code): 에러가 있을 경우

Example

```
local dWorkPos = [X, Y, .. Tz];
Robot[0].MoveL(dWorkPos);
```

Description

4.2.4.9 MoveTX(dTX), MoveTY(dTY), MoveTZ(dTZ)

Define

이 명령은 MoveL 의 응용 모션이며, 입력변수로 World 좌표계의 TX, TY, TZ 축의 상대 위치를 사용합니다. Work Motion 으로 Work Motion Parameter 값을 참조하여 모션 경로를 생성됩니다.
(TX, TY, TZ): 6 축 다관절 로봇에서 Tool 이 바라보는 직선상 임의의 한 점.

Form

```
MoveTX(dTX);
MoveTY(dTY);
MoveTZ(dTZ);
```

Parameter

dTX(Y, Z): World 좌표계에서 X(Y, Z)축 Target Point 의 Work 상대 위치 값, 실수형 Variable

Return

ERR_NONE(0): 에러가 없을 경우
ERR_NUM(Error Code): 에러가 있을 경우

Example

```
local dTX = 100;
Robot[0].MoveTX(dTX);
```

Description

4.2.4.10 MoveX(dX), MoveY(dY), MoveZ(dZ)

Define

이 명령은 MoveL 의 응용 모션이며, 입력변수로 World 좌표계의 X, Y, Z 축의 상대 위치를 사용합니다. Work Motion 으로 Work Motion Parameter 값을 참조하여 모션 경로를 생성됩니다.

Form

```
MoveX(dX);
MoveY(dY);
MoveZ(dZ);
```

Parameter

dX(Y, Z): World 좌표계에서 X(Y, Z)축 Target Point 의 Work 상대 위치 값, 실수형 Variable

Return

ERR_NONE(0): 에러가 없을 경우
ERR_NUM(Error Code): 에러가 있을 경우

Example

```
local dX = 100;
Robot[0].MoveX(dX);
```

Description

4.2.4.11 StartBlend(), EndBlend()

Define

이 명령은 여러 모션 명령을 중첩하여 이동 시킬 때 사용합니다. StartBlend 명령과 EndBlend 명령 사이에 있는 모션명령들이 순서대로 실행되며, 이전 모션이 감속하는 시점에 다음 모션이 중첩되어 실행됩니다.

Form

```
StartBlend();
EndBlend();
```

Parameter

None

Return

ERR_NONE(0): 에러가 없을 경우
ERR_NUM(Error Code): 에러가 있을 경우

Example

```
Robot[0].StartBlend()
Robot[0].MoveL(Pos1);
Robot[0].MoveC(vPos, tPos);
Robot[0].MoveL(Pos2);
Robot[0].EndBlend();
```

Description

4.2.4.12 MoveStop()

Define

이 명령은 모션 중인 로봇을 정지 시킬 때 사용합니다. 진행 중인 모션은 감속도(Dec)값을 참조하여 정지합니다.

Form

```
MoveStop();
```

Parameter

None

Return

ERR_NONE(0): 에러가 없을 경우

ERR_NUM(Error Code): 에러가 있을 경우

Example

```
local WorkPos = [A1, A2, .. An];
Robot[0].MoveL(WorkPos);
Robot[0].MoveStop();
```

Description

4.2.4.13 MoveEStop()

Define

이 명령은 모션 중인 로봇을 비상 정지 시킬 때 사용합니다. 진행 중인 모션은 설정된 KDec(or KTime) 값을 참조하여 정지합니다.

Form

```
MoveEStop();
```

Parameter

None

Return

ERR_NONE(0): 에러가 없을 경우
ERR_NUM(Error Code): 에러가 있을 경우

Example

```
local WorkPos = [A1, A2, .. An];
Robot[0].MoveL(WorkPos);
Robot[0].MoveEStop();
```

Description

4.2.4.14 WaitM(time)

Define

이 명령은 해당 로봇의 모션이 종료 한 후 설정된 대기 시간만큼 기다려 줍니다.

Form

```
WaitM(time);
```

Parameter

time(msec): 상수 또는 변수

Return

ERR_NONE(0): 에러가 없을 경우

ERR_NUM(Error Code): 에러가 있을 경우

Example

```
local WorkPos1 = [A1, A2, .. An];
local WorkPos2 = [B1, B2, .. Bn];
Robot[0].MoveL(WorkPos1);
Robot[0].WaitM(1000);
Robot[0].MoveL(WorkPos2);
```

Description

WorkPos1 지점으로 직선 보간으로 이동한 후 1 초 대기 후 WorkPos2 지점으로 직선 보간 이동합니다.

4.2.4.15 JtJogP, JtJogN, WkJogP, WkJogN

Define

원하는 축을 Joint 좌표계 혹은 Work 좌표계 기준으로 조그 모션합니다. (P: Positive, N: Negative)

Form

```
JtJogP(AxisNo)
JtJogN(AxisNo)
WkJogP(AxisNo)
WkJogN(AxisNo)
```

Parameter

AxisNo : 조그 모션할 축 번호

Return

None

Example

```
Robot[0].JtJogP(0);
Wait(1000);
Robot[0].MoveStop();
```

Description

Jog axis 0 of robot 0 to positive direction within 1 second.

4.2.4.16 MoveH()

Define

해당 로봇의 모든 축을 Homing 할 때 사용됩니다. (Homing Parameters 를 따릅니다.)

Form

```
MoveH();
```

Parameter

None

Return

None

Example

```
Robot[0].MoveH();
WaitM(1000);
Print(Robot[0].JtFPos());
```

4.2.5 Motion Parameter

Jt: Joint Motion, Wk: Work Motion, Rd: Ready Motion 의 Parameter 를 의미합니다.

4.2.5.1 JtAJerkP/JtDJerkP(WkAJerkP/WkDJerkP, RdAJerkP/RdDJerkP)

Define

이 명령은 로봇 모션의 가가속(Jerk) 범위(%)값을 설정하거나 설정된 값을 가져올 때 사용합니다.

JtAJerkP(WkAJerkP, RdAJerkP): 가속시간의 가가속 범위(0~100%)

JtDJerkP(WkDJerkP, RdDJerkP): 감속시간의 가가속 범위(0~100%)

Form

```
JtAJerkP(JerkArray);
JtAJerkP(JerkP);
local JerkArray = JtAJerkP();

JtDJerkP(JerkArray);
JtDJerkP(JerkP);
local JerkArray = JtDJerkP();
```

Parameter

JerkArray: 모션 가가속 범위 값의 배열

JerkP: 모션 가가속 범위 값, Double or Integer 변수 또는 상수

Return

설정되어 있는 현재 모션의 가가속 범위(0 ~ 100%)를 반환합니다.

Example

```
local JerkArray = [10, 20, 30...];
local JerkP = 100;
Robot[0].JtAJerkP(JerkArray);  // 축의 JerkP 값을 개별 설정.
Robot[0].JtAJerkP(JerkP);  // 로봇의 모든 축의 값을 100%로 설정.
local JerkArray2 = Robot[0].JtAJerkP(); // 설정된 로봇의 Jerk 값을 가지고 옵니다.
```

Description

4.2.5.2 JtATime/JtDTime(WkATime/WkDTime, RdATime/RdDTime)

Define

이 명령은 로봇의 모션의 가속 또는 감속 시간을 설정하거나 설정된 값을 가져올 때 사용합니다.

JtATime(WkATime, RdATime): 가속시간(msec)

JtDTime(WkDTime, RdDTime): 감속시간(msec)

Form

```
JtATime(ATimeList);
JtATime(ATime);
local ATimeList = JtATime();

JtDTime(DTimeList);
JtDTime(DTime);
local DTimeList = JtDTime();
```

Parameter

ATimeList(DTimeList): 모션 가속 또는 감속 시간 리스트, Array 형변수

ATime(DTime): 모션 가속 시간, Double or Integer 변수 또는 상수

Return

설정되어 있는 현재 모션의 가속 시간

Example

```
local ATimeList = [1000, 1500, 500...];
local ATime = 1000;
Robot[0].JtATime(ATimeList);  // 축의 가속시간을 개별 설정.
Robot[0].JtATime(ATime);  // 로봇의 모든 축의 가속 시간을 1000msec 로 설정.
local ATimeList2 = Robot[0].JtATime(); // 설정된 로봇의 ATime 값을 가지고 옵니다.
```

Description

4.2.4.3 JtVel(WkVel, RdVel)

Define

이 명령은 로봇 모션의 속도 값을 설정하거나 가져올 때 사용합니다.

Form

```
JtVel(VelList);
JtVel(Vel);
local VelList = JtVel();
```

Parameter

VelList: 모션 속도 리스트, Array 형 변수

Vel: 모션의 속도, Double 변수 또는 상수

(Unit: mm/sec or °/sec)

Return

설정되어 있는 현재 모션의 속도를 반환합니다.

Example

```
local VelList = [1000, 1500, 500...];
local Vel = 1000;  // 1000mm/sec
Robot[0].JtVel(VelList);  // 축의 속도를 개별 설정.
Robot[0].JtVel(Vel);  // 로봇의 모든 축 속도를  1000mm/s 로 설정.
local VelList2 = Robot[0].JtVel(); // 설정된 로봇의 Vel 값을 가지고 옵니다.
```

Description

4.2.4.4 JtKTime(WkKTime)

Define

이 명령은 로봇 모션의 급속 정지시(MoveEStop()) 감속 시간(Kill Time)을 설정하거나 설정된 값을 가져올 때 사용합니다.

Form

```
JtKTime(KTimeList);
JtKTime(KTime);
KTimeList = JtKTime();
```

Parameter

KTimeList: 모션 급·감속 시간 리스트, Array 형변수
KTime: 모션의 급·감속 시간 속도, Double or Integer 변수 또는 상수
(Unit: msec)

Return

설정되어 있는 현재 모션의 급·감속 시간을 반환합니다.

Example

```
local KTimeList = [100, 150, 50...];
local KTime = 100;
Robot[0].JtKTime(KTimeList);  // 축의 급·감속 시간을 개별 설정
Robot[0].JtKTime(KTime);  // 로봇의 모든 축의 급·감속 시간을 100msec 로 설정
local KTimeList2 = Robot[0].JtKTime(); // 설정된 로봇의 KTime 값을 변수에 대입
```

Description

4.2.5.5 JtCPos(WkCPos)

Define

이 명령은 로봇의 현재 지령(Command) 기준의 Joint or Work(World 좌표) 위치 값을 읽어올 때 사용합니다.

Form

```
JtCPos();
```

Parameter

None

Return

현재 Joint or Work(World 좌표) Command 위치 값을 반환합니다.

Example

```
local JointList = Robot[0].JtCPos();
Print(JointList); // 0 번 로봇의 모든 축의 위치 값을 출력합니다.
Print(JointList[0]); // 0 번 로봇의 처음 축의 위치 값을 출력합니다. (단축)
```

Description

4.2.5.6 JtFPos(WkFPos)

Define

이 명령은 로봇의 현재 엔코더(Feedback) 기준의 Joint or Work(World 좌표) 위치 값을 읽어올 때 사용합니다.

Form

```
JtFPos();
```

Parameter

None

Return

현재 Joint or Work(World 좌표) Feedback 위치 값을 반환합니다.

Example

```
local JointList = Robot[0].JtFPos();
Print(JointList); // 0 번 로봇의 모든 축의 위치 값을 출력합니다.
Print(JointList[0]); // 0 번 로봇의 처음 축의 위치 값을 출력합니다. (단축)
```

Description

4.2.5.7 VelP(Percent)

Define

이 명령은 로봇의 구동 속도 범위 값을 설정하거나 가져올 때 사용합니다.

실제 로봇 구동 속도는 Vel(JtVel, WkVel, RdVel) × VelP 로 구동됩니다.

Form

```
VelP(Percent);
```

Parameter

Percent: 로봇 속도 백분율 값, Integer 또는 Double 변수

Return

설정되어 있는 로봇 속도의 백분율 값을 반환합니다.

Example

```
Robot[0].JtVel(300) // 300mm/sec
Robot[0].VelP(30) // 30%로 설정
Robot[0].MoveJ(JointArray)  // JointArray 위치로 300*0.3 = 90mm/sec 로 속도로 이동
```

Description

4.2.6 Non-Volatile Variable

4.2.6.1 JtPos/WkPos[Index]

Define

해당 변수는 로봇의 위치를 저장하기 위한 비 휘발성 변수입니다.
변수의 저장은 PRC Manager 또는 TP 로만 가능합니다.

Form

```
JtPos[Index];
WkPos[Index];
```

Parameter

Index: 변수의 Index.

Return

해당 Index 에 저장되어 있는 변수 값을 반환합니다.

Example

```
Robot[0].MoveJ(Robot[0].JtPos[0]);
Robot[0].MoveL(Robot[0].WkPos[1]);
Print(Robot[0].JtPos[0]);
Print(Robot[0].WkPos[1]);
```

Description

4.2.6.2 IVar/DVar

Define

해당 변수는 Integer or Double Type 을 비 휘발성 변수로 사용할 때 사용됩니다.

Form

```
IVar[Index];
DVar[Index];
```

Parameter

Index: 변수의 Index.

Return

해당 Index 에 저장되어 있는 변수 값을 반환합니다.

Example

```
Wait(IVar[0]);


Robot[0].WkATime(IVar[0]);
Robot[0].WkVel(DVar[1]);


Description
```

4.2.6.3 JogVelP, JtJogVel, WkJogVel, JtJogATime, WkJogATime, JtJogDTime, WkJogDTime

Define

조그 모션 파라미터를 설정할 때 사용합니다.

Form

```
JogVelP(Value)
JtJogVel(Value)
WkJogVel(Value)
JtJogATime(Value)
WkJogATime(Value)
JtJogDTime(Value)
WkJogDTime(Value)
```

Parameter

Value: 해당 Parameter 에 설정할 값

Return

None

Example

```
//Set Jog Velicty
Robot[0].JtJogVel(200);
//Get Jog Velocity
local a = Robot[0].JtJogVel(200);

Axis Keywors with DIn and DOut
We can access Digital Input/Output on motor drive by using
local a= Axis[0].DIn; //Port Access 32bit value
local b= Axis[0].DIn[12]; //Bit Access value (0 or 1)
Axis[0].DOut[10] = 1; //Turn on Output bit 10 on drive 0
```

4.3 Mathematics Commands

Item	Command	Description
π	Pi()	원주율(PI)을 사용합니다.
sine	Sin(deg)	입력된 degree 의 sine 을 계산합니다.
	Asin(deg)	입력된 degree 의 arc sine 을 계산합니다.
cosine	Cos(deg)	입력된 degree 의 cosine 을 계산합니다.
	Acos(deg)	입력된 degree 의 arc cosine 을 계산합니다.
tangent	Tan(deg)	입력된 degree 의 tangent 를 계산합니다.
	Atan(deg)	입력된 degree 의 arc tangent 를 계산합니다.
absolute value	Abs(value)	입력된 value 의 절댓값을 계산합니다.
square root	Sqrt(value)	입력된 value 의 제곱근을 계산합니다.
exponential	Exp(value)	자연상수 e 의 지수제곱을 계산합니다.
Logarithm	Log(value)	자연 로그를 계산합니다.
	Log2(value)	이진 로그를 계산합니다.
	Log10(value)	상용 로그를 계산합니다.
Convert	DegToRad(value)	Degree 값을 Radian 값으로 변환하여 반환합니다.
	RadToDeg(value)	Radian 값을 Degree 값으로 변환하여 반환합니다.

4.3.1 Pi()

Define

원주율(π)을 반환하는 함수입니다.

Form

```
Pi();
```

Parameter

None

Return

원주율 3.141593 이 Double type 으로 반환합니다.

Example

```
local pi = Pi();
Print(pi);
```

Description

결과: 3.141593

4.3.2 Sin(deg)

Define

Sine 값을 계산하는 함수입니다.

Form

```
Sin(deg);
```

Parameter

deg: degree 값 입력

Return

계산된 Sin 값이 Double type 으로 반환합니다.

Example

```
Print(Sin(90));
```

Description

결과: 1.000000

4.3.3 Asin(deg)

Define

Arc sine 값을 계산하는 함수입니다.

Form

```
Asin(deg);
```

Parameter

deg: degree 값 입력

Return

계산된 Asin 값이 Double type 으로 반환합니다.

Example

```
Print(Asin(0.5));
```

Description

결과: 30.000000

4.3.4 Cos(deg)

Define

Cosine 값을 계산하는 함수입니다.

Form

```
Cos(deg);
```

Parameter

deg: degree 값 입력

Return

계산된 Cos 값이 Double type 으로 반환합니다.

Example

```
Print(Cos(90));
```

Description

결과: 0.000000

4.3.5 Acos(deg)

Define

Arc cosine 값을 계산하는 함수입니다.

Form

```
Acos(deg);
```

Parameter

deg: degree 값 입력

Return

계산된 Acos 값이 Double type 으로 반환합니다.

Example

```
Print(Acos(0.5));
```

Description

결과: 60.000000

4.3.6 Tan(deg)

Define

Tangent 값을 계산하는 함수입니다.

Form

```
Tan(deg);
```

Parameter

deg: degree 값 입력

Return

계산된 Tan 값이 Double type 으로 반환합니다.

Example

```
Print(Tan(-80));
```

Description

결과: -5.671282

4.3.7 Atan(deg)

Define

Arc tangent 값을 계산하는 함수입니다.

Form

```
Atan(deg);
```

Parameter

deg: degree 값 입력

Return

계산된 Atan 값이 Double type 으로 반환합니다.

Example

```
Print(Atan(5.67));
```

Description

결과: 79.997785

4.3.8 Abs(value)

Define

절댓값을 계산하는 함수입니다.

Form

```
Abs(value);
```

Parameter

value: 절댓값으로 반환할 값 입력

Return

value 에 Integer 입력이 들어오면 Integer type 을 반환,
Double 입력이 들어오면 Double type 을 반환합니다.

Example

```
Print(Abs(-80));
Print(Abs(-90.0));

Description
결과: 80
      90.000000
```

4.3.9 Sqrt(value)

Define

제곱근을 구하는 함수입니다.

Form

```
Sqrt(value);
```

Parameter

value: 제곱근을 계산할 값 입력

Return

value 의 제곱근을 계산하여 Double type 을 반환합니다.

Example

```
Print(Sqrt(81));
Print(Sqrt(9.0));

Description
결과: 9.000000
      3.000000
```

4.3.10 Exp(value)

Define

자연상수 e 의 지수 제곱을 계산하는 함수입니다.

Form

```
Exp(value);
```

Parameter

value: 지수 제곱을 계산할 값 입력

Return

value 의 지수 제곱을 계산하여 Double type 을 반환합니다.

Example

```
Print(Exp(1));
Print(Exp(-1));

Description
결과: 2.718282
      0.367879
```

4.3.11 Log(value)

Define

자연 로그를 계산하는 함수입니다.

Form

```
Log(value);
```

Parameter

value: 자연 로그를 계산할 값 입력

Return

value 의 자연 로그를 계산하여 Double type 을 반환합니다.

Example

```
local a = Exp(2);
Print(Log(a));
Print(Log(a*Exp(3)));

Description
결과: 2.000000
      5.000000
```

4.3.12 Log2(value)

Define

이진 로그를 계산하는 함수입니다.

Form

```
Log2(value);
```

Parameter

value: 이진 로그를 계산할 값 입력

Return

value 의 이진 로그를 계산하여 Double type 을 반환합니다.

Example

```
Print(Log2(8));
Print(Log2(1024));

Description
결과: 3.000000
      10.000000
```

4.3.13 Log10(value)

Define

상용 로그를 계산하는 함수입니다.

Form

```
Log10(value);
```

Parameter

value: 상용 로그를 계산할 값 입력

Return

value 의 상용 로그를 계산하여 Double type 을 반환합니다.

Example

```
Print(Log10(100));
Print(Log10(10*^10));
Print(125*^(1.0/3.0));
// 뒤에 소수점을 안 붙이고 그냥 (1/3)으로 하면 소수가 아닌 정수 계산이 되어 0 이 되니 주의

Description
결과: 2.000000
      10.000000
       5.000000
```

4.3.14 DegToRad(value)

Define

Degree 값을 Radian 값으로 변환하는 함수

Form

```
DegToRad(value);
```

Parameter

value: Radian 으로 변환할 Degree 값

Return

변환된 Radian 값을 Double type 으로 반환합니다.

Example

```
Print(DegToRad(90));
```

Description

결과: 1.570796

4.3.15 RadToDeg(value)

Define

Radian 값을 Degree 값으로 변환하는 함수

Form

```
RadToDeg(value);
```

Parameter

value: Degree 로 변환할 Radian 값 입력

Return

변환된 Degree 값을 Double type 으로 반환

Example

```
Print(RadToDeg(2*pi));
```

Description

결과: 360.000000

4.4 EtherCAT

Item	Command	Description
SDO IN/OUT	EcSdoIn	EtherCAT Slave 모듈의 비주기 데이터를 읽어올 때 사용
	EcSdoOut	EtherCAT Slave 모듈의 비주기 데이터를 설정할 때 사용
PDO IN/OUT	EcPdoIn	EtherCAT Slave 모듈의 주기 데이터를 읽어올 때 사용
	EcPdoOut	EtherCAT Slave 모듈의 주기 데이터를 설정할 때 사용

4.4.1 EcSdoIn(MapSlaveIdx, Index, SubIndex, Size)

Define

EtherCAT Slave 모듈의 비주기 데이터를 읽어올 때 사용됩니다.

Form

EcSdoIn(MapSlaveIdx, ObjAddress, SubIdx, Size);

Parameter

MapSlaveIdx : Master 로부터 연결된 Slave 순서로 첫 번째 Slave 가 0 이 됩니다.

Index : 해당 Slave 에서 읽고자 하는 Object Address

SubIndex : Object Sub Index

Size : 데이터 사이즈(Bytes)

Return

해당 Object 의 값을 읽어온다.

Example

```
local OpMode;
OpMode = EcSdoIn(0, HexStringToDec("0x6061"), 0, 1); //현재 실행되고 있는 동작모드를 읽어올 때
Print("Op Mode = ", OpMode);
```

Description

ObjAddress 의 값을 십진수로 바로 입력해도 되지만, 매뉴얼에서 일반적으로 헥사값으로 표기되어 있기 때문에 HexStringToDec()함수를 사용하면 편리합니다.

* 제공되는 Object List 는 각 Slave 제품 매뉴얼을 참고하시기 바랍니다.

4.4.2 EcSdoOut(MapSlaveIdx, Index, SubIndex, Size)

Define

EtherCAT Slave 모듈의 비주기 데이터를 설정할 때 사용됩니다.

Form

EcSdoIn(MapSlaveIdx, ObjAddress, SubIdx, Data, Size);

Parameter

MapSlaveIdx : Master 로부터 연결된 Slave 순서로 첫 번째 Slave 가 0 이 됩니다.

Index : 해당 Slave 에서 설정하고자 하는 Object Address

SubIndex : Object Sub Index

Data : 설정하고자 하는 데이터

Size : 데이터 사이즈(Bytes)

Return

Error Code (정상적으로 Write 할 경우 ERR_NONE(0) 값을 리턴)

Example

```
local OpMode;
OpMode = EcSdoIn(0, HexStringToDec("0x6061"), 0, 1); //현재 실행되고 있는 동작모드를 읽어올 때
Print("Op Mode = ", OpMode);
```

Description

ObjAddress 의 값을 십진수로 바로 입력해도 되지만, 매뉴얼에서 일반적으로 헥사값으로 표기되어 있기 때문에 HexStringToDec()함수를 사용하면 편리합니다.

* 제공되는 Object List 는 각 Slave 제품 매뉴얼을 참고하시기 바랍니다.

4.4.3 EcPdoIn(OffsetBit, Size)

Define

EtherCAT Slave 모듈의 주기 데이터를 읽어올 때 사용됩니다.

Form

```
EcPdoIn(OffsetBit, Size);
```

Parameter

OffsetBit: bitOffset of PDO data in EtherCat Network

Size : 데이터 사이즈(Bytes)

Return

해당 Object 값을 읽어온다.

Example

```
local off = [74*8, 75*8, 76*8, 77*8]; // byte to bit

for(local i = 0; i < 4; i = I + 1) {
    local val = EcPdoIn(off[i], 1);
    Print(val);
}
```

4.4.4 EcPdoIn(OffsetBit, Size)

Define

EtherCAT Slave 모듈의 주기 데이터를 설정할 때 사용됩니다.

Form

```
EcPdoIn(OffsetBit, Size);
EcPdoIn(OffsetBit, Size, Val);
```

Parameter

OffsetBit: bitOffset of PDO data in EtherCat Network

Size : 데이터 사이즈(Bytes)

Val : 설정할 값(Value)

Return

Error Code (정상적으로 Write 할 경우 ERR_NONE(0) 값을 리턴)

Example

```
local off = [74*8, 75*8, 76*8, 77*8]; // byte to bit
local val = [1, 2, 3, 4];

for(local i = 0; i < 4; i = I + 1) {
    EcPdoOut(off[i], 1, val[i]);
}
```

4.5 Axis Class

Item	Command	Description
Axis IO	DIn	드라이브에 탑재된 Digital Input 값을 가져올 때 사용
	DOut	드라이브에 탑재된 Digital Output 값을 설정하거나 현재 값을 가져올 때 사용
Error Code	ErrCode	드라이브(Axis)의 Error Code 를 읽어올 때 사용

4.5.1 Axis IO

4.5.1.1 DIn[bitIndex]

Define

드라이브에 탑재된 Digital Input 값을 가져올 때 사용합니다. (read only)

Form

```
Axis[AxisNo].DIn; // DIn Port 전체 값을 가져올 때 사용
Axis[AxisNo].DIn[BitIndex]; // DIn 의 해당 Bit 값(0 또는 1)을 가져올 때 사용
```

Parameter

AxisNo: System Configuration 에서 맵핑된 Axis Index

BitIndex: 해당 Axis 의 Digital Input 의 Channel Number 입니다. (0~31)

* 일반적으로 Bit0(Negative Limit), Bit1(Positive Limit), Bit2(Home Switch), Bit16~31(User Input)

Return

현재 Digital Input 값을 반환합니다.

Example

```
local AxisDIn = Axis[0].DIn;  // DIn Port 전체 값을 읽어 올 때
Print(DecToHexString(AxisDIn));

local AxisDInBit = Axis[0].DIn[16]; // DIn 의 16 번째 Bit 값을 읽어 올 때
Print(AxisDInBit);r
```

Description

User Input 이 16bit 부터 시작됨으로 Port 전체를 읽어올 때 DecToHexString()사용하면 보기가 용이

* 0x60FD Object 가 PDO 할당되게 ENI 파일을 만들어야 합니다.

(대부분의 드라이브의 ESI XML 파일에서 Default 로 할당되게 처리되어 있으나 ENI 파일을 만들 때 주의 하시기 바랍니다.)

4.5.1.2 DOut[bitIndex]

Define

드라이브에 탑재된 Digital Output 값을 설정하거나 현재 값을 가져올 때 사용합니다.

Form

```
Axis[AxisNo].DOut; // DOut Port 전체 값을 설정하거나 현재 값을 가져올 때 사용
Axis[AxisNo].DOut[BitIndex]; // DOut 의 해당 Bit 값(0 또는 1)을 설정하거나 현재 값을 가져올 때 사용
```

Parameter

AxisNo: System Configuration 에서 맵핑된 Axis Index

BitIndex: 해당 Axis 의 Digital Output 의 Channel Number 입니다. (0~31)

* 일반적으로 Bit0(Brake), Bit16~31(User Output)

Return

현재 Digital Output 값을 반환합니다.

Example

```
Axis[0].DOut = HexStringToDec("0x0F0000");  // DOut Port 전체 값을 설정할 때
local AxisDOut = Axis[0].DOut;
Print(DecToHexString(AxisDOut));

Axis[0].DOut[16] = 1; // DOut 의 16 번째 Bit 값을 1 로 Set
local AxisDOutBit = Axis[0].DOut[16];
Print(AxisDOutBit);
```

Description

User Output 이 16bit 부터 시작됨으로 Port 전체를 설정할 때는 HexStringToDec()를 사용하고, 읽어올 때 DecToHexString()사용하면 편리합니다.

* 0x60FE Object 가 PDO 할당되게 ENI 파일을 만들어야 합니다.

(대부분의 드라이브의 ESI XML 파일에서 Default 로 할당되게 처리되어 있으나 ENI 파일을 만들 때 주의 하시기 바랍니다.)

`

4.5.2 Error Code

4.5.2.1 ErrCode

Define

드라이브(Axis)의 Error Code 를 읽어올 때 사용합니다.

Form

```
Axis[AxisNo].ErrCode(); // 해당 Axis 의 Error Code 를 읽어옵니다.
```

Parameter

AxisNo: System Configuration 에서 맵핑된 Axis Index

Return

Error Code 반환합니다.

* 상세 코드 정보는 각 드라이브사의 매뉴얼을 참조하십시오.

Example

```
Axis[0].ErrCode();
```

Description

드라이브 알람이 발생할 경우 Command 창을 통해 해당 명령어를 입력하면 Error Code 를 받아 올 수 있습니다.

Chapter 5. Basic Flow Commands

조건문과 반복문의 중괄호 속 내용이 단 한 줄이라면 중괄호를 생략할 수 있습니다.

Item	Command	Description
조건문	if, else if, else	조건을 판별하여 조건에 맞는 프로그램을 실행합니다.
반복문	while	특정 조건이 참일 경우 계속해서 블록의 명령문들을 반복하여 실행할 수 있도록 합니다.
	for	객체의 열거형을 따라서 반복하여 실행할 때 사용합니다.
	loop	일정 횟수만큼 반복합니다.
제어문	break	루프 문을 강제로 빠져나올 때, 즉 아직 루프 조건이 'False'가 되지 않았거나 열거형의 끝까지 루프가 도달하지 않았을 경우에 루프 문의 실행을 강제로 정지시키고 싶을 때 사용됩니다.
	pause	프로그램을 일시 정지합니다.
	stop	프로그램을 정지합니다.

5.1 조건문

5.1.1 if, else if, else

Define

조건을 판별할 때 사용됩니다. if(만약) 조건이 참이라면, _if 블록_의 명령문을 실행하며 else(아니면) _else 블록_의 명령문을 실행합니다. 이 때 else if 와 else 조건절은 생략 가능합니다.

Form

```
if(condition)
{
    ……..;
}
else if(condition)
{
    ……..;
}
else
{
    ……..;
}
```

Parameter

condition: 참 거짓을 판단할 수 있는 조건

Return

None

Example

```
local a = Robot[0].JtFPos();
if(a[2] > 0.0)
{
    Robot[0].MoveL(Robot[0].WPos[1]);  // if 조건이 참이면 이 명령을 실행합니다.
}
else if(a[2] < -5.0)
{
    Robot[0].MoveStop();               // if 조건은 거짓, else if 조건은 참이면 이 명령을 실행합니다.
}
else
{
    Robot[0].MoveStop();               // if 조건, else if 조건 모두 거짓이면 이 명령을 실행합니다.
}
```

Description

5.2 반복문

5.2.1 while

Define

특정 조건이 참일 경우 계속해서 블록의 명령문들을 반복하여 실행합니다.

Form

```
while(condition)
{
    ……;
}
```

Parameter

condition: 참 거짓을 판단할 수 있는 조건

Return

None

Example

```
local a = Robot[0].WkFPos();
while(a[1] > 50)
    Wait(10);           // while 조건이 참이면 이 명령을 실행합니다.
```

Description

5.2.2 for

Define

객체의 열거형을 따라서 반복하여 실행할 때 사용되는 또 하나의 반복문으로, 조건에 맞을 때까지 증감한다.

Form

```
for(초기식; 조건식; 증감식)
{
    ……
}
```

Parameter

초기식: 초기값을 설정합니다.
조건식: 조건을 설정합니다.
증감식: 증감에 이용될 식를 설정합니다.

Return

None

Example

```
for(local i = 1; I <= 10; i = i+1)
{
    Robot[0].MoveLRel([i, i+1, i+2, i+3]);
    WaitM(10);
}
```

Description

5.2.3 loop

Define

단순하게 지정한 횟수만큼 반복합니다.

Form

```
loop(num)
{
    ……
}
```

Parameter

num: 반복할 횟수를 입력합니다.

Return

None

Example

```
local i = 0;
loop(10)
{
    Robot[0].MoveJW([i, i*2, 0, 0]);
i = i+1;
}
```

Description

5.3 제어문

5.3.1 break

Define

루프 문을 강제로 빠져나올 때, 즉 아직 루프 조건이 'False'가 되지 않았거나 열거형의 끝까지 루프가 도달하지 않았을 경우에 루프 문의 실행을 강제로 정지시키고 싶을 때 사용됩니다.

Form

```
break;
```

Parameter

None

Return

None

Example

```
local cnt = 0
while(true) // 조건을 항상 true 로 설정했으므로 무한 루프
{
    Print(cnt);
    if(cnt > 50)
        break;
    cnt = cnt + 1;
}
```

Description

5.3.2 pause

Define

현재 실행중인 프로그램을 일시 정지할 때 사용합니다.

Form

```
pause;
```

Parameter

None

Return

None

Example

```
local a = Robot[0].WkFPos();
while(a[1] > 50)
{
    Robot[0].MoveLRel([0,-10,0,0]);
    Robot[0].WaitM(10);
    if(a[0] > 50)
        pause;
    else
        break;
}
```

Description

5.3.3 stop

Define

현재 실행중인 프로그램을 완전히 정지할 때 사용합니다.

Form

```
stop;
```

Parameter

None

Return

None

Example

```
local a = 0
for(a = 0; a < 50; a = a+1)
{
    if(a>=30)
        stop;
}
```

Description

Chapter 6. Communication Interface

6.1 Melsec Ethernet Protocol

Mitsubishi MELSEC PLC 의 Ethernet Card 와 통신 하기 위한 함수들을 지원합니다.

Item	Command	Description
Connect /Disconnect	PLC_Connect	생성된 소켓을 통해 Melsec PLC 서버에 접속을 요청
	PLC_Disconnect	소켓을 소멸시키고 데이터 통신을 종료
Data Transmission	PLC_SetBit	해당 Device 의 Memory Bit 를 Set(1)
	PLC_ClearBit	해당 Device 의 Memory Bit 를 Clear(0)
	PLC_ReadBit	해당 Device 의 Memory Bit 읽기 (0 or 1)
	PLC_WriteW	해당 Device 의 Memory 의 주어진 크기만큼 데이터 쓰기
	PLC_ReadW	해당 Device 의 Memory 의 주어진 크기만큼 데이터 읽기

Device(PLC 의 특정 데이터를 저장할 수 있는 내부 메모리 영역) Definition

Definition	Description
MC_DEVICE_DATA_X	입력(X)
MC_DEVICE_DATA_Y	출력(Y) # Y area using HEX
MC_DEVICE_DATA_M	내부 릴레이(M)
MC_DEVICE_DATA_D	데이터 레지스터(D)
MC_DEVICE_DATA_T	타이머(T)

6.1.1 Connect / Disconnect

6.1.1.1 PLC_Connect(IP, PortNum)

Define

생성된 소켓을 통해 Melsec PLC 서버에 접속을 요청합니다.

Form

```
PLC_Connect(IP, PortNum);
```

Parameter

IP: Melsec PLC 서버의 IP Address (String)

PortNum: Port Number

Return

None

Example

```
PLC_Connect("192.168.100.220", 4096);
```

Description

6.1.1.2 PLC_Disconnect()

Define

소켓을 소멸시키고 데이터 통신을 종료합니다.

Form

```
PLC_Disconnect();
```

Parameter

None

Return

None

Example

```
PLC_Connect("192.168.100.220", 4096);
PLC_Disconnect();
```

Description

Connect 이후 사용해야 하며, 소켓을 소멸시킨 이후 다시 재 접속(Connect)시 수십 msec 의 Delay 가 발생할 수 있습니다.

6.1.2 Data Transmission

6.1.2.1 PLC_SetBit(Device, Address, BitOffset)

Define

해당 Device 의 Memory Bit 를 Set(1) 합니다.

Form

```
PLC_SetBit(Device, Address, BitOffset);
```

Parameter

Device: MC_DEVICE_DATA_X, MC_DEVICE_DATA_Y 등의 PLC Device

Address: Device 기준 Word Start Address

BitOffset: Word Address 기준의 BitOffset (PLC 의 Word 는 2Bytes 임으로 0~15 의 값을 입력할 수 있습니다.)

Return

None

Example

```
PLC_Connect("192.168.100.220", 4096);
PLC_SetBit(MC_DEVICE_DATA_Y, 0x30, 1)  // 출력(Y)의 0x31(bit Address) 을 Set(1) 합니다.
```

Description

6.1.2.2 PLC_ClearBit(Device, Address, BitOffset);

Define

해당 Device 의 Memory Bit 를 Clear(0) 합니다.

Form

```
PLC_ClearBit(Device, Address, BitOffset);
```

Parameter

Device: MC_DEVICE_DATA_X, MC_DEVICE_DATA_Y 등의 PLC Device

Address: Device 기준 Word Start Address

BitOffset: Word Address 기준의 BitOffset (PLC 의 Word 는 2Bytes 임으로 0~15 의 값을 입력할 수 있습니다.)

Return

None

Example

```
PLC_Connect("192.168.100.220", 4096);
PLC_ClearBit(MC_DEVICE_DATA_Y, 0x30, 1);  // 출력(Y)의 0x31(bit Address) 을 Clear(0) 합니다.
```

Description

6.1.2.3 PLC_ReadBit(Device, Address, BitOffset)

Define

해당 Device 의 Memory Bit 를 읽어옵니다.

Form

PLC_ReadBit(Device, Address, BitOffset)

Parameter

Device: MC_DEVICE_DATA_X, MC_DEVICE_DATA_Y 등의 PLC Device

Address: Device 기준 Word Start Address

BitOffset: Word Address 기준의 BitOffset (PLC 의 Word 는 2Bytes 임으로 0~15 의 값을 입력할 수 있습니다.)

Return

해당 Bit 의 데이터 반환 (0 or 1)을 반환합니다.

Example

```
PLC_Connect("192.168.100.220", 4096);
local ret = PLC_ReadBit(MC_DEVICE_DATA_Y, 0x30, 1);  // 출력(Y)의 0x31(bit Address) 값을 읽어옵니다.
```

Description

6.1.2.4 PLC_WriteW(Device, Address, Size, Word)

Define

해당 Device 의 Memory 의 주어진 크기만큼 데이터 쓰기를 합니다.

Form

```
PLC_WriteW(Device, Address, Size, Word);
PLC_WriteW(Device, Address, Word);
```

Parameter

Device: MC_DEVICE_DATA_X, MC_DEVICE_DATA_Y 등의 PLC Device

Address: Device 기준 Word Start Address

Size: Word Size

Word: Write 하고자 하는 데이터 버퍼(List or Array)

Return

None

Example

```
PLC_Connect("192.168.100.220", 4096);
local words = [1, 2, 3];
PLC_WriteW (MC_DEVICE_DATA_D, 1000, 3, words);
```

Description

결과

Device	Value
D2000	1
D2001	2
D2002	3

6.1.2.5 PLC_ReadW(Device, Address, Size)

Define

해당 Device 의 Memory 의 주어진 크기만큼 데이터 읽어옵니다.

Form

```
PLC_ReadW(Device, Address, Size);
PLC_ReadW(Device, Address);
```

Parameter

Device: MC_DEVICE_DATA_X, MC_DEVICE_DATA_Y 등의 PLC Device

Address: Device 기준 Word Start Address

Size: Word Size

Return

해당 위치 데이터 Size 만큼 반환, List or Array 변수에 저장해서 사용합니다.

Example

```
PLC_Connect("192.168.100.220", 4096)
local ret = PLC_ReadWord(MC_DEVICE_DATA_D, 1000, 3)
Print(ret);
```

Description

PRESTO

프 레 스 토 솔 루 션

www.prestosolution.co.kr

101-1404, Digital Empire Building 2nd, 88 Sinwon RD, Yeongtong-gu, Suwon-si, Gyeonggi-do, KOREA

T. +82-70-7167-8606 | F. +82-70-7159-2628