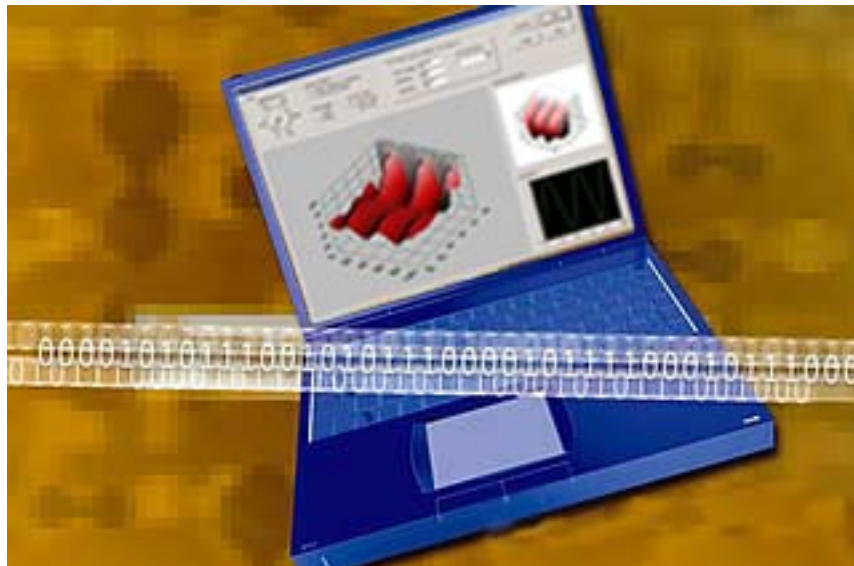

COMIZOA DAQ System (LX 시리즈) Visual Basic 매뉴얼





*COMputer Innovation
is Zoomed by Our Affection!*



저작권자 : ㈜커미조아

Copyright (c) by COMIZOA CO.,LTD. All right reserved.

2001 년 11 월 20 일 중 판 인쇄

이 사용자 설명서는 저작권법에 의해 보호되고 있습니다.

㈜커미조아의 사전 서면 동의 없이 사용자설명서의 일부 또는 전체를 어떤 형태로든 복사, 전재할 수 없습니다.

Hardware Support : Hardware@comizoa.co.kr

Software Support : Software@comizoa.co.kr



㈜커미조아

www.comizoa.co.kr

www.comizoa.com

Tel) 042 - 861 - 3301~3

Fax) 042 - 861 - 3304





CHAPTER 1 라이브러리 설치 및 배포	2
1.1 라이브러리 설치	2
1.2 프로그램의 배포 방법	4
CHAPTER 2 라이브러리 사용과 지원 method	8
2.1 디바이스 시작/종료	9
2.2 아날로그 입력(Analog Input)	12
2.2.1 아날로그 입력(General)	13
2.2.2 COMI-LX10x 시리즈 전용 A/D 스캔	18
2.2.3 COMI-LX20x 시리즈 전용 A/D 스캔	29
2.3 아날로그 출력(Analog Output)	50
2.3.1. 일반적인 Analog Output 메소드	51
2.3.2. Waveform Generation	52
2.4 디지털 입출력(Digital Input/Output)	59
2.4.1 일반적인 디지털 입출력	60
2.4.2 시리얼 통신을 이용한 디지털 입출력	67
2.5 카운터	80
2.6 모션 제어(Motion Control)	83
2.6.1 모션 초기화 및 환경설정 메소드	84
2.6.2 Single Axis 모션 제어 메소드	103
2.6.3 Multi-Axis 동시제어 메소드	130
2.6.4 Coordinated Motion 메소드	144
2.6.5 속도 및 위치 오버라이딩(Overriding) 메소드	194
2.6.6 원점 복귀(Home Return) 메소드	201
2.6.7 Manual Pulser 모드 모션 제어 메소드	211
2.6.8 External Switch Operation 모드 모션 제어 함수	221
2.6.9 리스트 모션(Listed Motion) 메소드	225
2.6.10 상태 감시 및 제어 메소드	234
2.6.11 I/O(입출력) 환경설정 메소드	258
2.6.12 인터럽트 관련 메소드	294
Appendix A 라이브러리 메소드 리스트	303
A.1 기능별 메소드 색인	303

A.2 메쏘드별 지원 가능 디바이스 리스트	311
-------------------------------	-----

CHAPTER 1

라이브러리 설치 및 배포

본 장에서는 COMI-LX 디바이스 시리즈를 Visual Basic 에서 제어하기 위한 라이브러리 설치 및 배포 방법을 설명합니다. 사용자는 이 라이브러리를 설치하셔야 Visual Basic 에서 COMIDAS 디바이스를 제어하실 수 있습니다. 물론, COMI-LX 디바이스 시리즈를 제어하는 Visual Basic 으로 작성된 프로그램을 제작, 배포하실 때에도, 라이브러리를 함께 배포하셔야 합니다.

CHAPTER 1 라이브러리 설치 및 배포

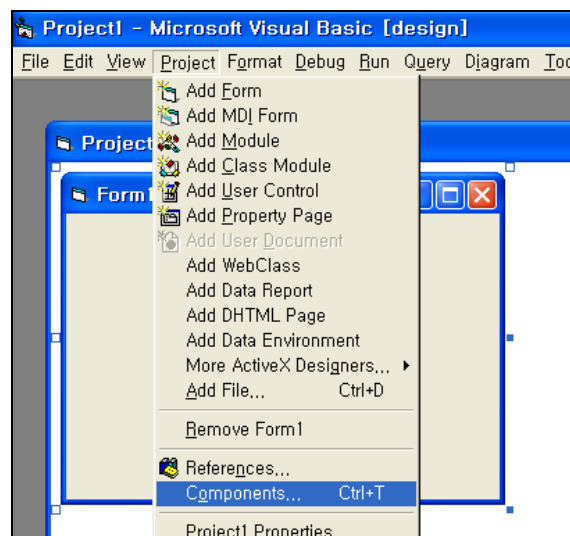
본 장에서는 사용자가 COMI-LX 시리즈 디바이스를 제어하는 Visual Basic 프로그램을 구현할 때 유용하게 사용될 수 있는 라이브러리의 설치와 배포방법을 설명합니다. (주)커미조아에서 제공하는 COMI-LX 시리즈용 Visual Basic 라이브러리는 모든 종류의 COMI-LX 시리즈 디바이스에 적용 가능한 통합 라이브러리입니다.

COMI-LX 시리즈 디바이스의 Visual Basic 용 라이브러리는 ComiLxAx.ocx 라는 ActiveX 컴포넌트 형태로 제공됩니다. ComiLxAx.ocx 라는 컴포넌트는 ComidasLX.dll 과 사용자 프로그램과의 통신을 담당해주는 ActiveX 로서, 이를 통해 드라이버를 Access 할 수 있습니다.

즉, Visual Basic 에서 COMI-LX 시리즈 디바이스를 제어하려면, 디바이스 드라이버와 ComidasLX.dll 파일 그리고, ComiLxAx.ocx 파일이 시스템에 설치되어야 합니다.

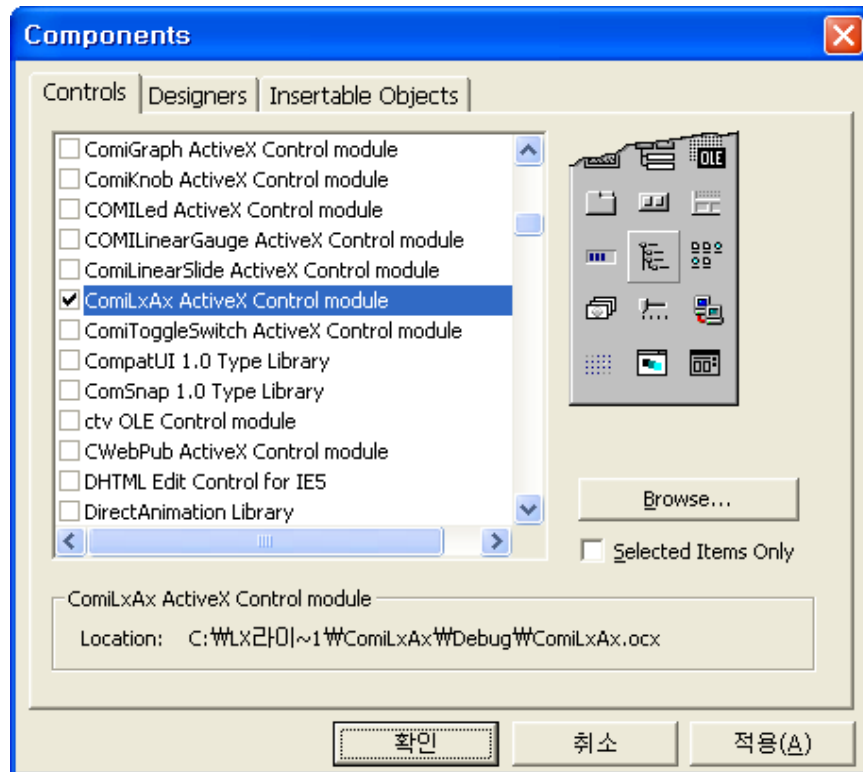
1.1 라이브러리 설치

‘LX 소프트웨어 매뉴얼’ 을 참조하여, 디바이스의 드라이버와 응용프로그램을 설치하시면, C:\WProgram Files\WCOMIZOAW\Comidas-LX\Visual Basic 폴더내에, Visual Basic 용 라이브러리, 예제, 매뉴얼이 복사되어집니다. 라이브러리인 ComiLxAx.ocx 는, 설치마법사가 자동으로 윈도우에 등록해주므로 사용자는 Visual Basic 에서 컴포넌트를 추가하여 주기만 하면 됩니다.



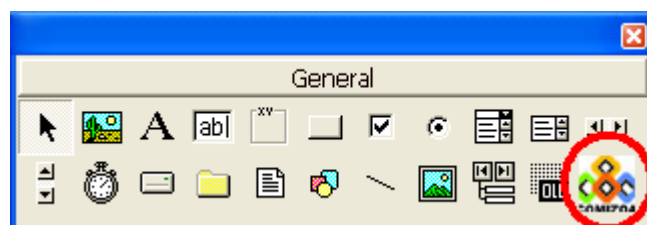
[그림 1.1] Components 창 열기

메뉴에서 'Project' 를 선택하신 후, 'Components' 를 클릭하시면, Components 창이 열립니다. (단축키 : ctrl + T)



[그림 1.2] Components 창에서의 ComiLxActiveX.ocx 콘트롤 선택

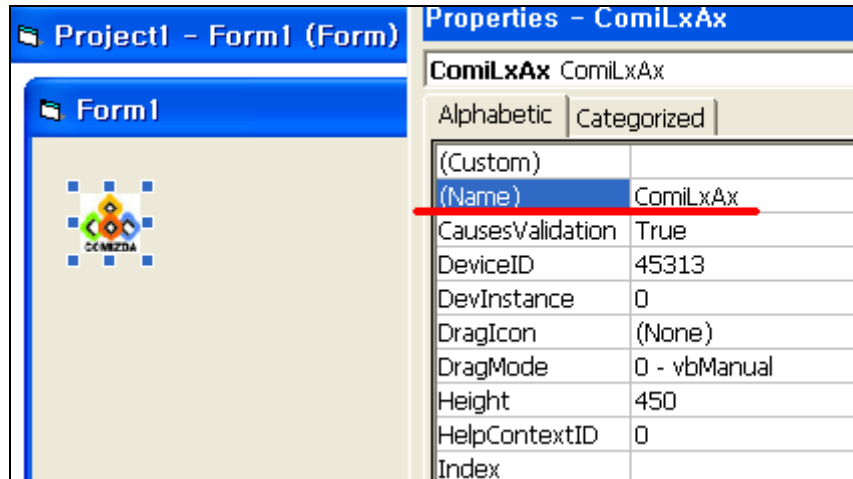
'ComiLxActiveX Control module' 을 선택하시어, 체크표시를 하시고 '확인' 을 클릭합니다. (윈도우에 등록되어 있지 않은 ocx 인 경우에는 'Browse(찾아보기)' 를 클릭하여, 그 위치를 설정해 주어야 하지만, ComiLxActiveX.ocx 는 드라이버 및 응용프로그램 설치시에 자동으로 윈도우의 시스템 폴더에 복사되어진 후, 레지스트리에 자동 등록됩니다.)



[그림 1.3] toolbox 에 등록된 ComiLxActiveX.ocx

[그림 1.3]과 같이 toolbox 에 ComiLxActiveX 콘트롤이 등록되면, 이제 다른 Visual Basic

컨트롤들을 사용하듯이 폼에 삽입할 수 있습니다. 하지만, 다른 컨트롤들과는 달리 Timer 컨트롤처럼, Design time 에서는 폼에서 그 위치나 존재가 보이지만, Runtime 에서 최종사용자에게는 컨트롤이 보여지지 않습니다. 즉, ComiLxAx 컨트롤은 내부적으로 COMI-LX 디바이스 시리즈를 제어하고 디바이스로부터 데이터를 전달받을 때만 사용되어집니다.



[그림 1.4] 폼에 삽입된 ComiLxAx 컨트롤의 인스턴스와 인스턴스명 설정

하나의 폼에는 여러 ComiLxAx 컨트롤의 인스턴스가 삽입될 수 있습니다. 이는, 하나의 프로그램이 여러 디바이스를 동시에 제어, 계측할 수 있음을 의미합니다. 그 방법들은 다음 장에 설명됩니다. 다른 Visual Basic 컨트롤과 마찬가지로, 컨트롤의 인스턴스에 대한 접근은 인스턴스명을 이용합니다. 즉, 인스턴스의 Property(속성)인 경우, '인스턴스명.속성이름' 이 되며, method(메소드)인 경우, '인스턴스명.메소드()' 가 됩니다. 본 매뉴얼에서는 인스턴스명을 'ComiLxAx1' 로 설정하고 프로그래밍을 하는것으로 간주하고 설명하겠습니다.

1.2 프로그램의 배포 방법

프로그램 배포시에도 디바이스 드라이버와 ComidasLX.dll, ComiLxAx.ocx 파일을 같이 배포하셔야 합니다.

'LX 소프트웨어 매뉴얼' 을 참조하여, 디바이스의 드라이버와 응용프로그램을 설치하시면, C:\WProgram Files\COMIZOA\Comidas-LX\Driver 폴더내에 디바이스 드라이버가 담겨있습니다. 배포시에는 이 드라이버 파일을 같이 배포하셔서 최종사용자로 하여금 디바이스 설치시 드라이버를 설치하도록 하셔야 합니다.

또한, C:\Program Files\COMIZOAWComidas-LXWC_PPPWLibrary 폴더에 ComidasLX.dll 파일이 담겨있습니다. 배포시에는 이 파일이 대상 시스템의 시스템 폴더에 복사되도록 하셔야 합니다. (시스템 폴더란 Windows 9x 와 ME 인 경우 C:\WINDOWS\system이며, NT 와 XP 인 경우 C:\WINDOWS\system32 입니다.)

마지막으로, C:\Program Files\COMIZOAWComidas-LXWVisual BasicWLibrary 폴더에 ComiLxAx.ocx 파일이 담겨있습니다. 배포시에는 이 파일이 대상 시스템의 OS 에 등록되어야 합니다. 즉, 이 파일의 위치는 상관없이 OS 의 레지스트리에 등록이 되어야하는데, 등록방법으로는 DOS-command 창에서 'regsvr32 ComiLxAx.ocx' 라는 명령을 실행시켜주는 방법과 인스톨 실투드와 같은 셋업 프로그램을 생성해주는 유틸리티를 이용하여, 최종사용자가 배포된 프로그램을 셋업이 자동으로 동시에 ocx 파일이 OS 에 등록되도록 하는 방법이 있습니다

CHAPTER 2

라이브러리 사용과 지원 method

본 장에서는 COMI-LX 디바이스 시리즈의 Visual Basic 라이브러리인 ComiLxAx 컨트롤의 속성과 이 컨트롤이 제공하는 메소드에 대해 설명합니다. 지원되는 디바이스별로 정리되어 있는 각각의 메소드의 특성과 기능을 숙지한다면 보다 쉽게 강력하고 안정적인 프로그램 제작이 가능할 것입니다.

2.1 라이브러리 및 디바이스 시작/종료	9
2.2 아날로그 입력(Analog Input)	12
2.3 아날로그 출력(Analog Output)	50
2.4 디지털 입출력(Digital Input/Output)	59
2.5 카운터(Counter)	80
2.6 모션 제어(Motion Control)	83

CHAPTER 2 라이브러리 사용과 지원 method

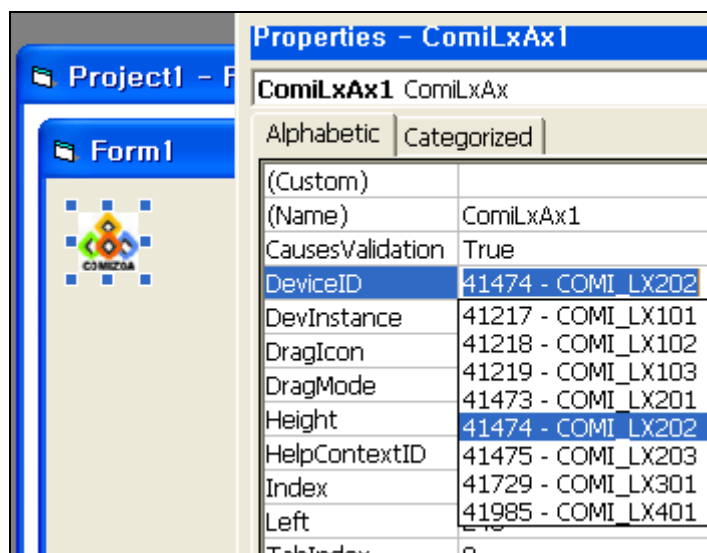
하나의 폼에는 다수의 ComiLxAx 컨트롤의 인스턴스가 삽입될 수 있고, 이들은 각각 자신만의 디바이스 ID (DeviceID Property)와 디바이스 Instance (DeviceInstance Property)로 시스템에 연결된 디바이스를 가르킵니다. 또한, ComiLxAx 컨트롤이 제공하는 메소드들을 이용하여 계측하고 또한 제어하게 됩니다. Visual Basic 의 다른 컴포넌트와 마찬가지로, COMI-LX 디바이스 시리즈를 이용한 프로그램을 잘 구현하려면, ComiLxAx 가 가지고 있는 Property (프라퍼티 : 속성)와 Method (메소드 : 함수의 개념)를 잘 숙지하여야 합니다.

ComiLxAx 는 DeviceID 와 DeviceInstance 라는 두 개의 Property 를 가지고 있습니다. DeviceID 는 'COMI-LX102' , 'COMI-LX301' 등의 디바이스 명을 가르킵니다. 즉, 여러 종류의 디바이스는 DeviceID 속성으로 구분되어집니다.

이러한 DeviceID 속성은, Design Time 에서 [그림 2.1]과 같이 선택하여 줄 수도 있고, 실제 소스코드에서

```
ComiLxAx1.DeviceID = COMI-LX201
```

과 같이 설정하여 줄 수도 있습니다.



[그림 2.1] 폼에 삽입된 ComiLxAx 컨트롤의 DeviceID 설정

DeviceInstance 속성은, 같은 종류의 대바이스가 여러 개 설치되었을 때, 이를 구분하는 속성입니다. 예를 들어, 각각 하나씩의 COMI-LX201 과 COMI-LX301 이 있다면 이는

```
ComiLxAx1.DeviceID = COMI-LX201
```



```
ComiLxAx2.DeviceID = COMI-LX301
```

라고 설정하여 주면 되지만 같은 COMI-LX201 디바이스가 두 개 이상 설치되어 있다면,

```
ComiLxAx1.DeviceInstance = 0
```

```
ComiLxAx2.DeviceInstance = 1
```

라고 설정해주어야 합니다.

각 메소드들은 그 기능에 따라 분류되어 수록되었습니다. 사용자는 각 메소드 그룹의 설명 및 부록에서 제공되는 각 메소드의 지원가능한 디바이스 리스트를 참조하여 각 디바이스에 맞는 메소드들을 사용하시기 바랍니다.

2.1 디바이스 시작/종료

이 단원에서는 COMIDAS 라이브러리와 각 디바이스를 로드(Load)/언로드(Unload)하는 메소드들을 소개합니다. 이 메소드들은 COMIDAS 라이브러리를 사용하기 위해 필수적으로 적용되어야 할 메소드들입니다. 이에 관련된 메소드들의 리스트는 다음과 같습니다.

메소드명	각 보드별 지원 여부								
	LX101	LX102	LX103	LX201	LX202	LX203	LX301	LX401	LX50x
LoadDevice	V	V	V	V	V	V	V	V	V
UnloadDevice	V	V	V	V	V	V	V	V	V

[표 2-1] 디바이스 시작/종료 메소드 리스트 및 각 보드별 지원 여부

■ LoadDevice

메소드 원형

Function **LoadDevice** As Boolean

메소드 설명

이 메소드는 하나의 COMIDAS 디바이스를 로드(load)합니다. 각 디바이스를 제어하기 위해서는 먼저 이 메소드를 이용하여 해당 디바이스를 준비시켜야 합니다.

Return 값

메소드 수행의 성공 여부

Value	Meaning
0	메소드 수행 실패 (디바이스 로드 실패)
1	메소드 수행 성공 (디바이스 로드 성공)

참 고

이 메소드는 제어하고자 하는 디바이스 수만큼 수행되어야 합니다.

예 제

2 대의 COMI-SD201 디바이스를 이용하여 프로그램 할 때의 코드 예

```
ComiLxAx1.DeviceID = COMI_SD201
ComiLxAx2.DeviceID = COMI_SD201

ComiLxAx1.DevInstance = 0
ComiLxAx2.DevInstance = 1

Success = ComiLxAx1.LoadDevice

If (Success = True) Then
    '
Else
    Call MsgBox("Can't load first COMI-SD201 device!",vbOKOnly)
End If

Success = ComiLxAx2.LoadDevice

If (Success = True) Then
    '
Else
    Call MsgBox("Can't load second COMI-SD201 device!",vbOKOnly)
End If

Call ComiLxAx1.UnloadDevice
Call ComiLxAx2.UnloadDevice
```

■ UnloadDevice

메소드 원형

```
sub UnloadDevice
```

메소드 설명

이 메소드는 하나의 COMIDAS 디바이스를 언로드(unload)합니다.

예제

```
ComiLxAx1.DeviceID = COMI_SD201
Success = ComiLxAx1.LoadDevice

If (Success = True) Then
    '
Else
    Call MsgBox("Can't load COMI-SD201 device!", vbOKOnly)
End If

Call ComiLxAx1.UnloadDevice
```

2.2 아날로그 입력(Analog Input)

이 장에서는 A/D 에 관련된 메소드들을 소개합니다. A/D 는 아날로그(Analog) 신호를 입력 받아 디지털(Digital)값으로 변환해주는 기능입니다.

COMIDAS 에서 지원하는 A/D 방식에는 두 가지가 있습니다. 첫 번째는 Single PointA/D 방식이며 두 번째는 A/D Scan 방식입니다.

Single PointA/D 는 사용자가 원하는 시점에서 소프트웨어적으로 A/D trigger 를 하여 A/D 데이터를 획득하는 방식입니다. 단 LX20-시리즈 디바이스는 Single PointA/D 방식은 지원하지 않습니다.

A/D Scan 방식은 사용자가 직접 A/D trigger 를 하지 않고, 디바이스에 내장된 타이머가 일정 주기로 A/D trigger 를 하고 변환된 A/D 데이터를 특정 버퍼에 저장하는 방식입니다. 이 방식은 Single PointA/D 방식에 비해 속도가 빠르고 정확한 샘플링 주기를 보장할 수 있습니다.

LX10-시리즈 보드와 LX20-시리즈 보드는 A/D 스캔 방식에 있어서 많은 차이점이 있습니다. 따라서 A/D 스캔 메소드는 LX10-시리즈 보드에 사용되는 메소드와 LX20-시리즈 보드에 사용되는 메소드가 구분되어 있습니다. Us1XXXXX 의 형태로된 메소드는 LX10-시리즈 보드에 적용되는 A/D 스캔 메소드며, Us2XXXXX 의 형태로된 메소드는 LX20-시리즈 보드에 적용되는 A/D 스캔 메소드입니다.

본 매뉴얼의 2.2.2 단원에서 LX10-시리즈 보드용 A/D 스캔 메소드를 수록하였으며, 2.2.3 단원에서는 LX20-시리즈 보드용 A/D 스캔 메소드를 따로 수록하였습니다.

2.2.1 아날로그 입력(General)

이 단원에서는 Single Point A/D 메소드를 포함한 일반적으로 사용되는 A/D 관련 메소드들을 소개합니다. 이에 관련된 메소드들의 리스트는 다음과 같습니다.

메소드명	각 보드별 적용 여부								
	LX101	LX102	LX103	LX201	LX202	LX203	LX301	LX401	LX50x
AdSet InputType	V	V	V						
AdSetRange	V	V	V	V	V	V			
AdGetDigit	V	V	V						
AdGetVolt	V	V	V						

[표 2-2] Analog Input 일반 메소드 리스트 및 각 보드별 지원 여부

■ AdSetInputType

메소드 원형

Sub **AdSetInputType** (ByVal InputMode As Long)

메소드 설명

이 메소드는 아날로그 입력 신호의 연결 형식을 소프트웨어적으로 설정합니다. 아날로그 입력 신호의 연결 형식에는 Single ended 방식과 Differential 방식의 두 가지가 있습니다(하드웨어 매뉴얼 참조).

매개 변수

▶ **InputMode** : 아날로그 입력 신호의 연결 형식을 설정합니다. 이 값은 다음 중 하나의 값이어야 합니다. 컴퓨터 부팅시에 연결 형식의 기본값은 AI_SINGLE 로 설정됩니다.

Value	Meaning
0 또는 AI_DIFF	아날로그 입력 신호의 연결 형식을 <u>Differential</u> 로 설정합니다.
1 또는 AI_SINGLE	아날로그 입력 신호의 연결 형식을 <u>Single ended</u> 로 설정합니다.

참 고

LX10-시리즈만이 소프트웨어적으로 연결 형식을 설정할 수 있습니다. LX20-시리즈는 하드웨어의 점퍼 셋팅을 통하여 연결 형식을 설정하여야 합니다.

■ AdSetRange

메소드 원형

Function **AdSetRange** (ByVal ch As Long, ByVal vmin As Single, ByVal vmax As Single)
As Boolean

메소드 설명

이 메소드는 각 A/D 채널의 입력 범위를 정해줍니다.

매개 변수

- ▶ **ch** : A/D 범위를 정해줄 채널 번호를 지정합니다. 채널 번호는 0 부터 시작합니다.
- ▶ **vmin** : A/D 범위의 최소값을 지정합니다. 유효한 vmin 값은 보드 종류에 따라 다음과 같습니다.
 - COMI-LX10x 디바이스 ~ 0, -1, -2, -5, -10
 - COMI-LX20x 디바이스 ~ -1, -2, -5, -10
- ▶ **vmax** : A/D 범위의 최대값을 지정합니다. 유효한 vmax 값은 1, 2, 5, 10 입니다.

Return 값

메소드 수행의 성공 여부

Value	Meaning
0	메소드 수행 실패
1	메소드 수행 성공

참 고

디바이스별 설정 가능한 A/D 범위는 다음과 같습니다.

디바이스	설정 가능한 A/D 범위
COMI-LX10x	±1, ±2, ±5, ±10, 0~1, 0~2, 0~5, 0~10
COMI-LX20x	±1, ±2, ±5, ±10

예 제

A/D CH0 은 -10 ~ 10 의 입력 범위를, 그리고 A/D CH1 은 -5 ~ 5 의 입력 범위를 가지도록 설정하는 예

```
Call ComiLxAx1.LoadDevice
Call ComiLxAx1.AdSetRange(0, -10, 10)
Call ComiLxAx1.AdSetRange(1, -5, 5)
```

■ AdGetDigit

메소드 원형

Function **AdGetDigit** (ByVal ch As Long) As Long

메소드 설명

이 메소드는 주어진 채널에 대하여 A/D 변환을 수행하고 그 값을 정수값으로 반환합니다.

매개 변수

▶ **ch** : A/D 를 수행할 채널 번호. 채널 번호는 0 부터 시작합니다.

Return 값

□ 정수형의 A/D 결과값. 이 값의 범위는 A/D 분해능에 따라 다르며 디바이스별로 보면 다음과 같습니다.

디바이스	분해능	반환되는 정수값의 범위
COMI-LX101	12 Bit	0 ~ 4095
COMI-LX102	14 Bit	0 ~ 16383
COMI-LX103	16 Bit	-32768 ~ 32767

□ 이 메소드에서 반환하는 정수값을 Voltage 값으로 변환하려면 다음과 같은 식이 적용되어야 합니다.

$$V = \frac{V_{\max} - V_{\min}}{D_{\max} - D_{\min}} (D - D_{\min}) + V_{\min}$$

여기서

V : 정수값으로부터 환산되는 Voltage 값

D : 환산하고자 하는 대상 정수값

V_{\max} : A/D 범위의 최대값 (AdSetRange 메소드 참조)

V_{\min} : A/D 범위의 최소값 (AdSetRange 메소드 참조)

D_{\max} : 정수값의 최대 범위값(A/D 분해능에 따라 다르며 앞의 표 참조)

D_{\min} : 정수값의 최소 범위값(A/D 분해능에 따라 다르며 앞의 표 참조)

예제

A/D CH0 의 A/D 값을 정수값으로 받는 예

```
Call ComiLxAx1.LoadDevice
Call ComiLxAx1.AdSetInputType(AI_SINGLE)
Call ComiLxAx1.AdSetRange( 0 , VMIN , VMAX)
Result = ComiLxAx1.AdGetDigit(0)
'스캔된 값 처리부..
Call ComiLxAx1.UnloadDevice
}
```


■ AdGetVolt

메소드 원형

```
Function AdGetVolt (ByVal ch As Long) As Single
```

메소드 설명

이 메소드는 주어진 채널에 대하여 A/D 변환을 수행하고 그 값을 voltage 값으로 반환합니다.

매개 변수

▶ *ch* : A/D 를 수행할 채널 번호. 채널 번호는 0 부터 시작합니다.

Return 값

A/D 결과값을 Voltage 값으로 반환합니다.

예제

A/D CH0 의 A/D 값을 Voltage 값으로 받는 예

```
Call ComiLxAx1.LoadDevice

Call ComiLxAx1.AdSetInputType(AI_SINGLE)
Call ComiLxAx1.AdSetRange( 0 , VMIN , VMAX)

Result = ComiLxAx1.AdGetVolt(0)
'스캔된 값 처리부..

Call ComiLxAx1.UnloadDevice
}
```

2.2.2 COMI-LX10x 시리즈 전용 A/D 스캔

이 단원에서는 LX10-시리즈 디바이스 전용으로 사용되는 A/D 스캔 메소드들을 소개합니다. LX10-시리즈 전용 A/D 스캔에 관련된 메소드들은 메소드명이 **Us1xxxx** 의 형식으로 구성됩니다.

Unlimited A/D Scan 방식은 사용자가 직접 A/D trigger 를 하지 않고, 디바이스에 내장된 타이머가 일정 주기로 A/D trigger 를 해주고 변환된 A/D 데이터를 특정 버퍼에 저장하는 방식입니다. 이 때 Scan 데이터를 저장하는 버퍼는 환형 버퍼 형식으로 운용되며 사용자는 필요시에 이 버퍼로부터 데이터를 취하게 됩니다. 이 방식은 Single PointA/D 방식에 비해 속도가 빠르고 정확한 샘플링 주기를 보장할 수 있습니다. 따라서 이 방식은 고속 A/D 를 할 때 아주 유용하게 사용될 수 있습니다.

LX10-시리즈 디바이스 전용 A/D 스캔 메소드의 리스트는 다음과 같습니다.

메소드명	각 보드별 적용 여부								
	LX101	LX102	LX103	LX201	LX202	LX203	LX301	LX401	LX50x
Us1Start	V	V	V						
Us1Stop	V	V	V						
Us1CurCount	V	V	V						
Us1GetBuffer	V	V	V						
Us1SBPos	V	V	V						
Us1RetrvOne	V	V	V						
Us1RetrvChannel	V	V	V						
Us1RetrvBlock	V	V	V						
Us1ReleaseBuf	V	V	V						

[표 2-3] LX10-시리즈 디바이스 전용 A/D 스캔 메소드 리스트 및 각 보드별 지원 여부

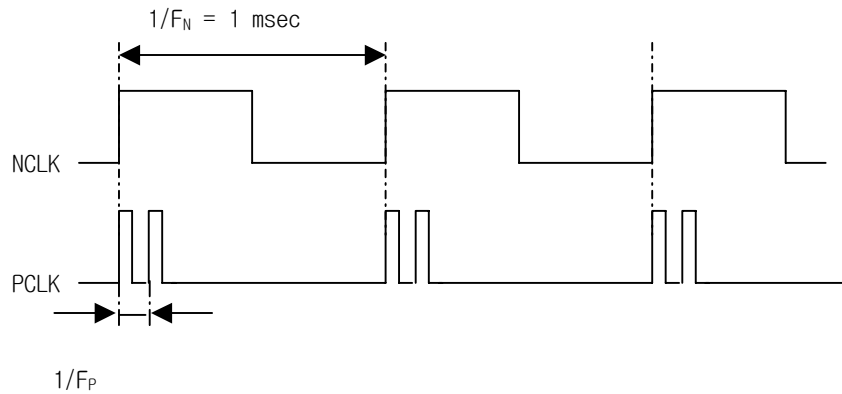
우선 Unlimited A/D Scan 메소드들을 소개하기 앞서 몇 가지 미리 숙지해야할 사항들을 수록합니다.

□ SCAN 이란 ?

A/D SCAN 이라함은 지정된 여러 채널을 순차적으로 A/D 변환한다는 뜻입니다. 따라서 1 회의 SCAN 은 사용자가 지정한 모든 채널에 대하여 1 회씩 A/D 변환이 완료되었을 때를 1 회의 SCAN 으로 정의합니다. 따라서 이후에 사용되는 Scan rate(또는 Scan frequency)라는 용어의 의미는 스캔채널로 지정된 각각의 채널에 대하여 1 초당

변환되는 A/D 횟수를 말하게 됩니다. 이는 동일 스캔내에서의 각 채널간 A/D 변환 주기를 결정해주는 Sampling rate(또는 Sampling frequency)와는 구분이 되어야 합니다. External trigger 를 사용하는 경우를 제외하곤 Scan rate 와 Sampling rate 는 디바이스 내부의 타이머에 의해 제어됩니다.

0 번 채널과 1 번 채널을 스캔 채널로 지정하고 Scan rate 를 1 KHz 로 지정한 경우 각 디바이스에 따른 Scan rate 와 Sampling rate 를 그림으로 표시하면 다음과 같습니다.



NCLK : 스캔 타이머 신호

PCLK : 샘플링 타이머 신호, 이 신호가 실제 A/D Trigger 를 한다.

F_N : Scan frequency

F_P : Sampling frequency (이 값은 디바이스의 A/D 칩이 지원하는 최대 주파수로 설정되며 디바이스에 따라 달라진다.)

□ 환형 버퍼

Unlimited A/D Scan 에서 A/D 데이터가 저장되는 버퍼는 환형 버퍼 형식으로 운용됩니다. 환형 버퍼는 한정된 버퍼에 무한히 데이터를 기록하기 위해 사용되는 것으로써, 데이터가 버퍼의 마지막 위치까지 다 채워지면 버퍼의 처음 위치부터 다시 채워 나가는 방식을 말합니다.

■ Us1Start

메소드 원형

Function **Us1Start** (ByVal NumChannel As Long, ByRef ChanList As Long, ByVal ScanFreq As Long, ByVal BufSize As Long, ByVal TrsMethod As Long) As Long

메소드 설명

이 메소드는 Unlimited scan 기능을 시작합니다.

매개 변수

- ▶ **NumChannel** : A/D scan 할 A/D 채널의 수.
- ▶ **ChanList** : A/D scan 을 수행할 채널 리스트를 담고 있는 배열
- ▶ **ScanFreq** : A/D scan frequency 를 Hz 단위로 지정합니다. 이 값은 SCAN 과 SCAN 사이의 시간차를 결정합니다.
- ▶ **BufSize** : 스캔 데이터를 저장할 환형버퍼의 크기를 결정하는 값으로써 각 채널의 데이터가 환형버퍼에 오버랩(Overlap)되지 않고 담길 수 있는 최대 데이터 수를 의미합니다. 환형 버퍼는 디바이스 드라이버에서 자동으로 할당하며 실제 크기는

$$\text{환형 버퍼의 실제 크기(bytes)} = \text{NumChannel As Long} * \text{BufSize As Long} * \text{sizeof(short)}$$
 가 됩니다.
- ▶ **TrsMethod** : A/D 디바이스에서 스캔버퍼로 데이터를 전송하는 방식을 지정합니다. 이 값은 다음의 두 값 중의 하나이어야 합니다.

Value	Meaning
1 또는 TRS_SINGLE	1 회의 SCAN 이 완료될 때마다 인터럽트를 발생시켜 데이터를 전송합니다. 인터럽트의 한계에 따라 Scan frequency 가 30 KHz 이상이 되면 이 방식이 적절히 작동하지 않을 수 있습니다.
2 또는 TRS_BLOCK	이 방식을 선택하면, A/D 디바이스는 A/D Conversion 데이터를 디바이스에 내장되어 있는 FIFO 메모리에 일단 저장하고, 1024 개의 데이터가 쌓이면 인터럽트를 발생시켜 데이터를 1024 개 단위로 사용자 버퍼에 전송합니다. 이 방식은 고속 A/D scan 시에 적합합니다.

Return 값

실제로 설정되는 스캔 주파수를 Hz 단위로 반환합니다. 사용자가 지정한 스캔 주파수와

실제로 설정되는 스캔 주파수는 약간의 차이가 있을 수 있습니다. 이 값이 0 보다 작으면 수행도중 에러가 발생하였음을 의미합니다.

예제

A/D CH0 의 A/D 값을 Voltage 값으로 받는 예

□ 0 번 채널 한 채널을 1KHz 로 SCAN 할 때의 Start 메소드 예제

```
Call ComiLxAx1.LoadDevice
```

```
Dim ChanList(0)
ChanList(0) = 0
```

```
ActFreq = ComiLxAx1.Us1Start(1, ChanList(0), 1000, 10240, TRS_SINGLE)
Call ComiLxAx1.UnloadDevice
```

□ 32 채널 모두에 대하여 채널당 10KHz 로 SCAN 할 때의 Start 메소드 예제 (TRS_BLOCK 모드 사용)

```
Call ComiLxAx1.LoadDevice
Dim ChanList(31)
```

```
Call ComiLxAx1.AdSetInputType(AI_SINGLE)
    ' 32 채널을 사용하기 위해서는 연결 형식이 SINGLE-ENDED 로 설정되어야함
```

```
For i=0 To 31
    ChanList(i) = i
Next i
```

```
Call ComiLxAx1.Us1Start(32, ChanList(0), 10000, 8192, TRS_BLOCK)
```

```
Call ComiLxAx1.UnloadDevice
```

or

■ Us1Stop

메소드 원형

```
Sub Us1Stop ( ByVal ReleaseBuf As Integer)
```

메소드 설명

이 메소드는 Unlimited scan 을 종료합니다.

매개 변수

▶ **ReleaseBuf** : Us1Start()가 수행될 때 할당되었던 스캔 버퍼를 메모리 해제시킬것인지를 결정합니다. 만일 이 값을 FALSE 로 지정하면 후에 반드시 Us1ReleaseBuf() 사용하여 버퍼를 해제하여야 합니다. 이 값을 TRUE 로 지정하면 Us1ReleaseBuf() 메소드를 수행할 필요가 없습니다.

■ Us1RetrvOne

메소드 원형

Function **Us1RetrvOne** (ByVal ChOrder As Long, ByVal scanCount As Long) As Integer

메소드 설명

이 메소드는 원하는 위치의 A/D Scan 데이터를 Voltage 값으로 반환합니다.

매개 변수

▶ *chOrder* : 데이터를 취하기 원하는 채널의 채널 리스트 상의 순서(0 based)입니다. 이 값은 채널 번호가 아님을 주의하여야 합니다.

▶ *scanCount* : 원하는 데이터의 Scan count.

Return 값

chOrder 와 scanCount 에 의해 지정된 데이터를 Voltage 형식으로 반환합니다.

or

■ Us1RetrvChannel

메소드 원형

```
Function Us1RetrvChannel (ByVal chOrder As Long, ByVal startCount, ByVal maxNumData,
ByRef DestBuf As Double ) As Long
```

메소드 설명

이 메소드는 A/D Scan 채널 중에서 하나의 채널에 대한 데이터 블록을 Voltage 값으로 환산하여 전달합니다. 데이터 블록은 사용자가 지정한 startCount 에서부터 maxNumData 에서 지정한 수만큼이 됩니다.

매개 변수

- ▶ **chOrder** : 데이터를 취하기 원하는 채널의 채널 리스트 상의 순서(0 based)입니다. 이 값은 채널 번호가 아님을 주의하여야 합니다.
- ▶ **startCount** : 전달 받고자 하는 데이터 블록의 시작 Scan count.
- ▶ **maxNumData** : 전달 받고자 하는 데이터 블록의 크기(데이터 수)를 지정 합니다. 이 값이 양수이면 startCount 부터 이후에 스캔된 데이터 중 maxNumData 에서 지정한 수만큼 데이터를 전달합니다. 이 값이 음수이면 startCount 부터 이전에 스캔된 데이터 중 maxNumData 에서 지정한 수만큼 데이터를 전달합니다.
- ▶ **DestBuf** : 데이터를 전달 받을 배열을 지정합니다. 이 버퍼의 크기는 maxNumData 에서 지정한 값보다 크거나 같아야 한다.

Return 값

실제 전달된 데이터 수. 만일 startCount 이후에 현재까지 스캔된 데이터 수가 maxNumData 에서 지정한 수 보다 작으면, 현재 스캔된 데이터까지만 전달하게됩니다.

예제

□ 예제 1.

이프로그램은 A/D CH0 와 CH2 의 두채널을 Unlimited scan 을 이용하여 A/D 변환을 수행하고 그 결과를 파일로 저장하는 예제입니다.

```
Const NUM_CH = 2          ` Number of channels
Const S_FREQ = 1000      ` Scan freq. -> 1000 Hz
Const MSB = 10240
Dim DestBuf(S_FREQ * 2, 1) As Double
Dim PreCount
Dim ch_list(1) As Long
ch_list(0) = 0
ch_list(1) = 2          ` Scan channel list : 0 번 과 2 번 채널

Call ComiLxAx1.LoadDevice

Call ComiLxAx1.US1_Start(NUM_CH,ch_list(0),S_FREQ,MSB,TRS_SINGLE)
Set ResultFile = CreateObject ("Scripting.FileSystemObject").
```



```
CreateTextFile("result.txt", True)

Private Sub Timer_Timer()

    NumOfData=ComiLxAx1.Us1RetrvChannel(0,PreCount,2048,DestBuf(0,0))
    Call ComiLxAx1.Us1RetrvChannel(2,PreCount,NumOfData,DestBuf(0,1))

    For i = 0 To NumOfData - 1
        For j = 0 To 1
            ResultFile.write Format(DestBuf(i, j), "0.000 ")
        Next j
        ResultFile.writeline
    Next i
    PreCount = PreCount + NumOfData

End Sub

Call ComiLxAx1.Us1Stop(1)
ResultFile.Close
Call ComiLxAx1.UnloadDevice
```

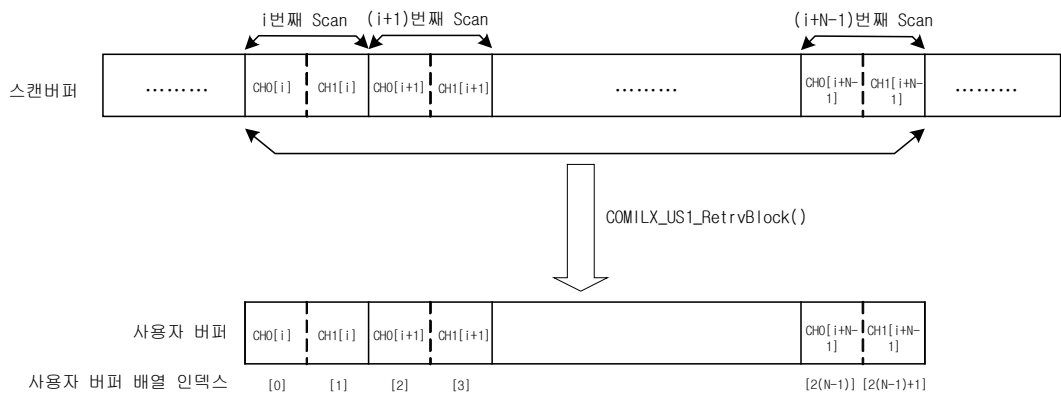
Us1RetrvBlock

메소드 원형

Function **Us1RetrvBlock** (ByVal startCount As Long, ByVal maxNumScan As Long, ByRef DestBuf As Double) As Long

메소드 설명

이 메소드는 A/D Scan 전 채널에 대한 데이터를 사용자가 지정하는 버퍼에 전달합니다. 전달되는 데이터 블록의 시작 위치는 startCount 에 의해 결정되며, 그 크기는 maxNumScan 에 의하여 결정됩니다. 데이터 블록의 실제 크기는 maxNumScan * 채널수가 됩니다. 예를 들어 CH0 와 CH1 의 두 채널에 대하여 A/D 스캔을 수행할 때 startCount 를 1, maxNumScan 을 N 으로 하였다면 스캔 버퍼에서 사용자 버퍼로 데이터가 전달되는 것은 다음 그림과 같습니다.



매개 변수

- ▶ **chOrder** : 데이터를 취하기 원하는 채널의 채널 리스트 상의 순서(0 based)입니다. 이 값은 채널 번호가 아님을 주의하여야 합니다.
- ▶ **startCount** : 전달 받고자 하는 데이터 블록의 시작 Scan count.
- ▶ **maxNumScan** : 전달 받고자 하는 데이터 블록의 크기(스캔 횟수)를 지정합니다. 이 값이 양수이면 startCount 부터 이후에 스캔된 데이터 중 maxNumScan 에서 지정한 수만큼 데이터를 전달합니다. 이 값이 음수이면 startCount 부터 이전에 스캔된 데이터 중 maxNumScan 에서 지정한 수만큼 데이터를 전달합니다.
- ▶ **DestBuf** : 데이터를 전달 받을 배열을 지정합니다. 이 버퍼의 크기는 maxNumScan*채널 수 보다 크거나 같아야 합니다.

Return 값

실제 전달된 데이터 수. 만일 startCount 이후에 현재까지 스캔된 데이터 수가 maxNumData

에서 지정한 수 보다 작으면, 현재 스캔된 데이터까지만 전달하게됩니다.

참 고

□ 사용자 버퍼의 배열 크기는 반드시 (채널수 * maxNumData) 보다 크거나 같아야 합니다.

예 제

A/D CH0 와 CH1 의 두채널을 Unlimited scan 을 이용하여 A/D 변환을 수행하고 그 결과를 파일로 저장하는 예제입니다.

```
Const NUM_CH = 2          ` Number of channels
Const S_FREQ = 1000      ` Scan freq. -> 1000 Hz
Const MSB = 10240
Dim DestBuf(S_FREQ * 2) As Double
Dim PreCount
Dim ch_list(1) As Long
ch_list(0) = 0
ch_list(1) = 2 ` Scan channel list : 0 번 과 2 번 채널

Call ComiLxAx1.LoadDevice

Call ComiLxAx1.Us1_Start(NUM_CH,ch_list(0),S_FREQ,MSB,TRS_SINGLE)
Set ResultFile = CreateObject ("Scripting.FileSystemObject").
    CreateTextFile("result.txt", True)

Private Sub Timer_Timer()

    NumOfData=ComiLxAx1.Us1RetrvBlock(PreCount,2048,DestBuf(0))

    For i = 0 To NumOfData - 1
        ResultFile.write Format(DestBuf(i*NUM_CH), "0.000 ")
        ResultFile.write Format(DestBuf(i*NUM_CH+1), "0.000 ")
    Next i
    PreCount = PreCount + NumOfData

End Sub

Call ComiLxAx1.Us1Stop(1)
ResultFile.Close
Call ComiLxAx1.UnloadDevice
```

or

■ Us1ReleaseBuf

메소드 원형

Function **Us1ReleaseBuf** As Boolean

메소드 설명

Us1Start()가 수행될 때 할당되었던 스캔 버퍼를 메모리 해제 시킵니다. 이 메소드는 Us1Stop() 메소드가 호출되기 전에 수행되어서는 안되며, Us1Stop() 메소드를 호출할 때 두 번째 파라미터를 TRUE 로 지정했을 때는 사용하지 않아도 됩니다.

Return 값

Value	Meaning
False	메소드 수행 실패
True	메소드 수행 성공

2.2.3 COMI-LX20x 시리즈 전용 A/D 스캔

이 단원에서는 LX20-시리즈 디바이스 전용으로 사용되는 A/D 스캔 메소드들을 소개합니다. LX20-시리즈 전용 A/D 스캔에 관련된 메소드들은 메소드명이 **Us2xxxx** 의 형식으로 구성됩니다. LX20-시리즈 디바이스는 고속 샘플링을 구현하기 위하여 LX10-시리즈 디바이스들과 여러 가지 차이점을 가지게 되어 그 메소드를 구분하여 구현하였습니다.

메소드명	각 보드별 적용 여부								
	LX101	LX102	LX103	LX201	LX202	LX203	LX301	LX401	LX50x
Us2SetTriggerEvent				V	V	V			
Us2Start				V	V	V			
Us2Resume				V	V	V			
Us2IsBufFull				V	V	V			
Us2ChangeScanFreq				V	V	V			
Us2DmaCount				V	V	V			
Us2GetBuffer				V	V	V			
Us2RetrvChannel				V	V	V			
Us2Stop				V	V	V			
Us2ReleaseBuf				V	V	V			

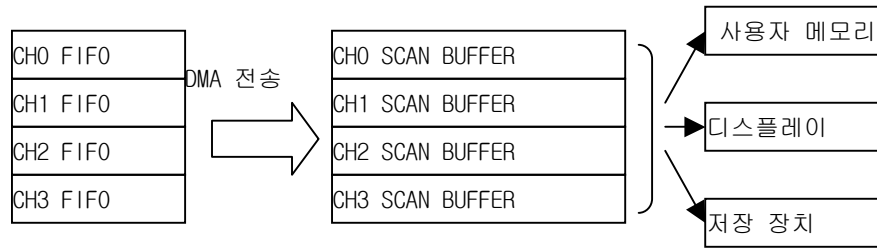
[표 2-4] LX20-시리즈 디바이스 전용 A/D 스캔 메소드 리스트 및 각 보드별 지원 여부

우선 LX20-시리즈 전용 A/D 스캔 메소드들을 설명하기 전에 자주 사용되는 용어에 대하여 먼저 설명합니다. 이 용어들을 먼저 숙지하고 각 메소드들을 파악한다면 좀더 쉽게 이해하실 수 있을 것입니다.

□ 스캔 버퍼 (Scan Buffer)

이 단원에서 사용되는 스캔 버퍼란 용어는 드라이버가 자동 할당하여 A/D 된 데이터들을 임시 저장하는 PC 메모리 공간을 말합니다. 스캔 버퍼는 드라이버에서 자동적으로 할당합니다. 단, 그 크기는 사용자가 지정하게 되는데 이 것은 Us2Start() 메소드의 BufSize 매개 변수가 스캔 버퍼의 크기를 결정하게 됩니다.

LX20-시리즈 디바이스를 사용할 때 A/D 데이터의 흐름을 그림으로 표현하면 다음과 같습니다.



[LX20x 디바이스 내장] [PC 메모리 공간] [사용자의 데이터 처리]
 [그림 2-1] COMI-LX20 시리즈 디바이스를 이용한 A/D 스캔 시에 데이터의 흐름도

디바이스에 내장된 FIFO 는 각 채널당 8192 개의 데이터를 담을 수 있는 크기이며 4096 개씩으로 나뉘어 더블 버퍼링(Double Buffering)방식으로 운용됩니다. DMA 전송은 각 채널당 4096 개의 데이터 블록 단위로 전송됩니다. 따라서 1 회의 DMA 전송이 이루어졌으면 각 채널당 4096 개의 데이터가 전송되었음을 의미합니다.

□ 환형 버퍼

환형 버퍼는 한정된 버퍼에 무한히 데이터를 기록하기 위해 사용되는 것으로써, 데이터가 버퍼의 마지막 위치까지 다 채워지면 버퍼의 처음 위치부터 다시 채워 나가는 방식을 말합니다.

Us2Start()메소드의 매개 변수중에서 PauseBufFull 값을 FALSE 로 설정하면 드라이버는 스캔버퍼를 환형버퍼 형식으로 운용하여 데이터를 채워나갑니다.

※ 버퍼크기가 N 인 경우 스캔 버퍼에 데이터가 저장되는 예 (여기서 BufSize Gain 값을 n 이라 하면 $N = n * 4096$ 이 된다)

- 현재 N개의 데이터가 스캔된 경우

Buf[0]	Buf[1]	Buf[2]		Buf[N-1]
DATA (1)	DATA (2)	DATA (3)	DATA (N)

- 현재 N+2개의 데이터가 스캔된 경우

Buf[0]	Buf[1]	Buf[2]		Buf[N-1]
DATA (N+1)	DATA (N+2)	DATA (3)	DATA (N)

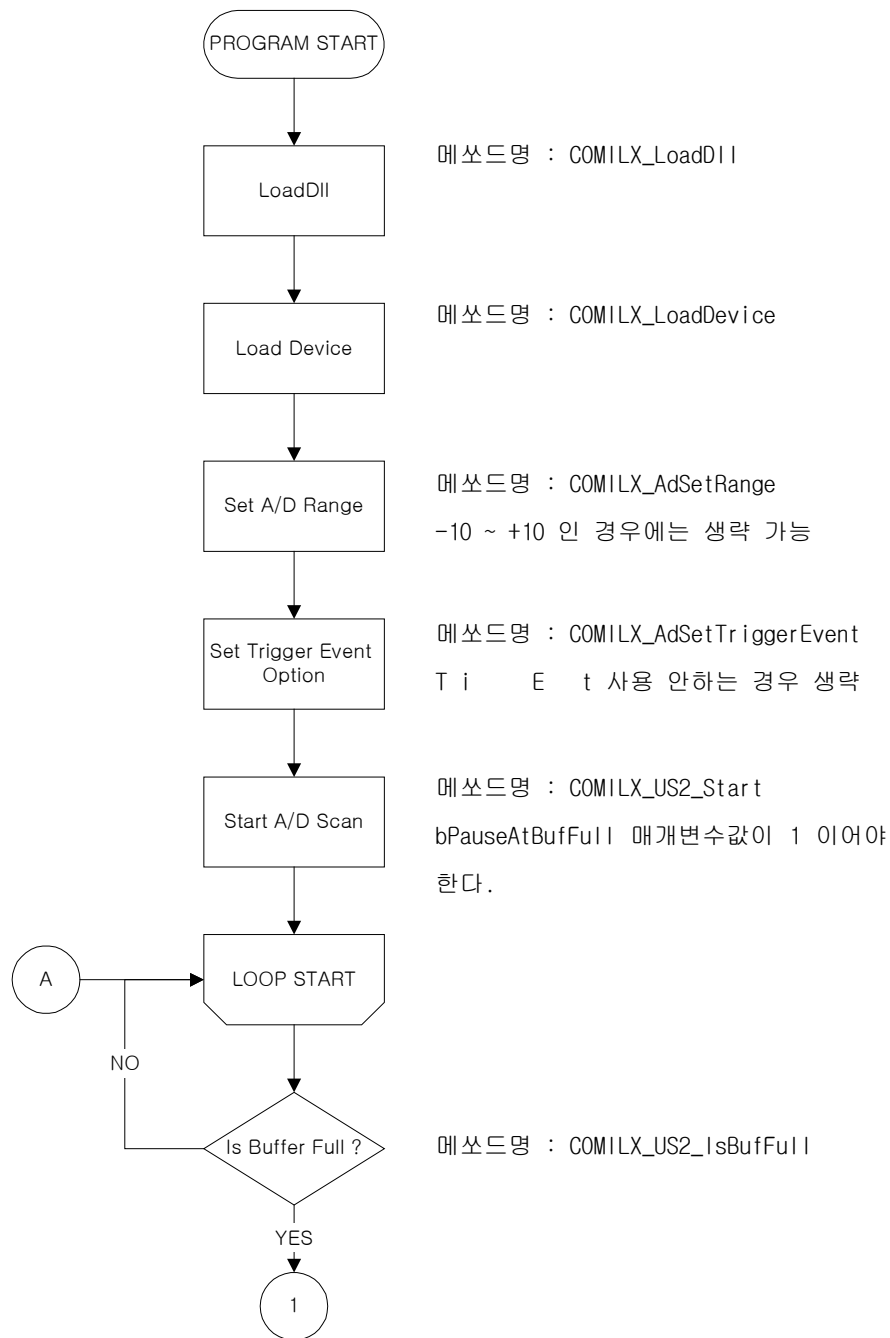
□ Frame Scan 과 Continuous Scan

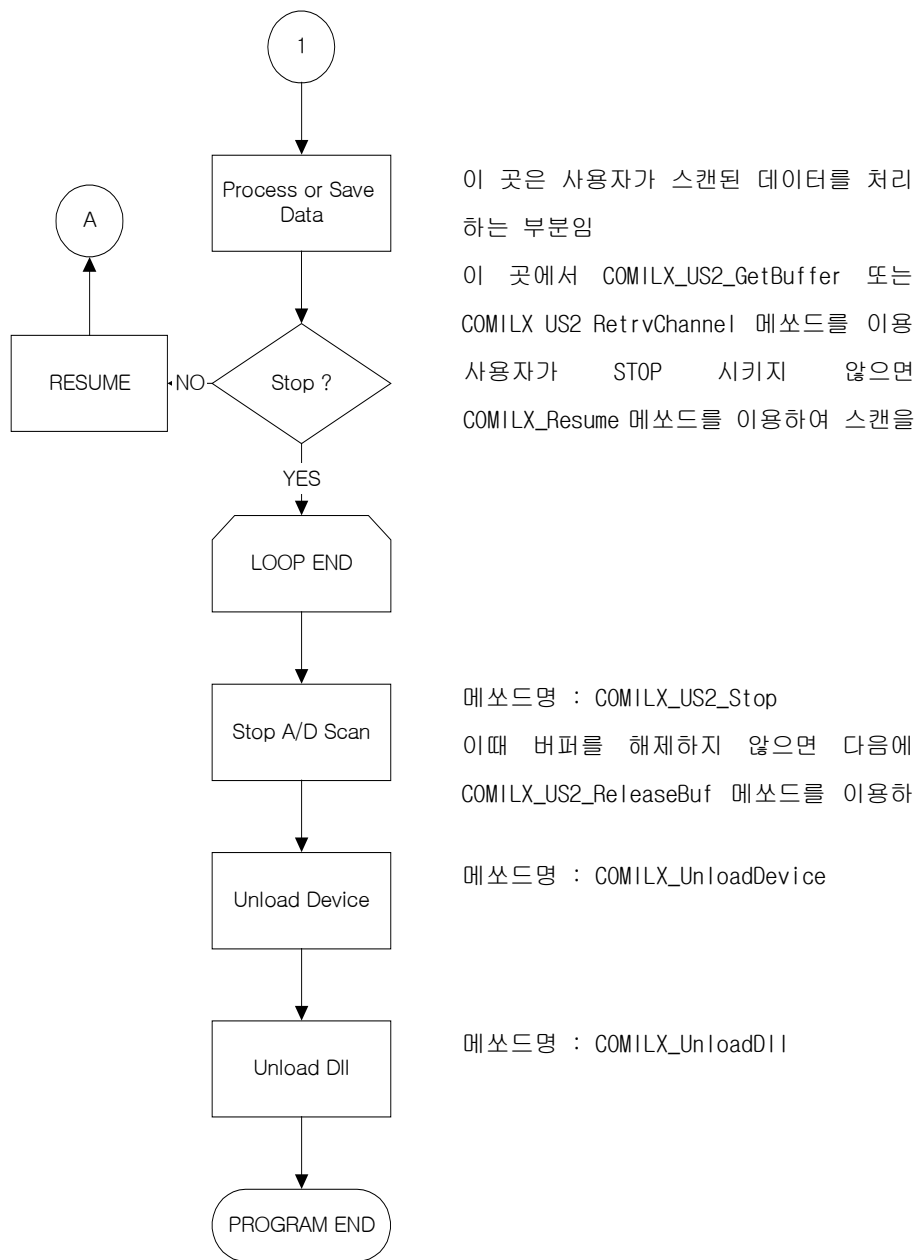
본 라이브러리를 이용하여 고속으로 A/D 스캔을 수행할 때에 Us2Start()메소드의 PauseAtBufFull 매개 변수의 값에 따라 다음의 두 가지 방식으로 수행됩니다.

Frame Scan : 일정 크기의 데이터 블록이 스캔되면 사용자가 Us2Resume()메소드를 이용하여 재개하기 전까지 자동으로 스캔을 일시 중지합니다. 이 것은 고속으로 A/D 스캔하는 경우 데이터의 Over lap 을 방지하기 위한 방식입니다.

Continuous Scan : 스캔 버퍼를 환형 버퍼 형식으로 운용하여 사용자가 중지할 때까지 계속하여 A/D 스캔을 수행하는 방식입니다. 이 방식을 사용하면 데이터를 끊임 없이 계속할 수 있지만 데이터의 처리 속도에 따라 데이터가 Over lap 되는 경우가 발생할 수 있습니다.

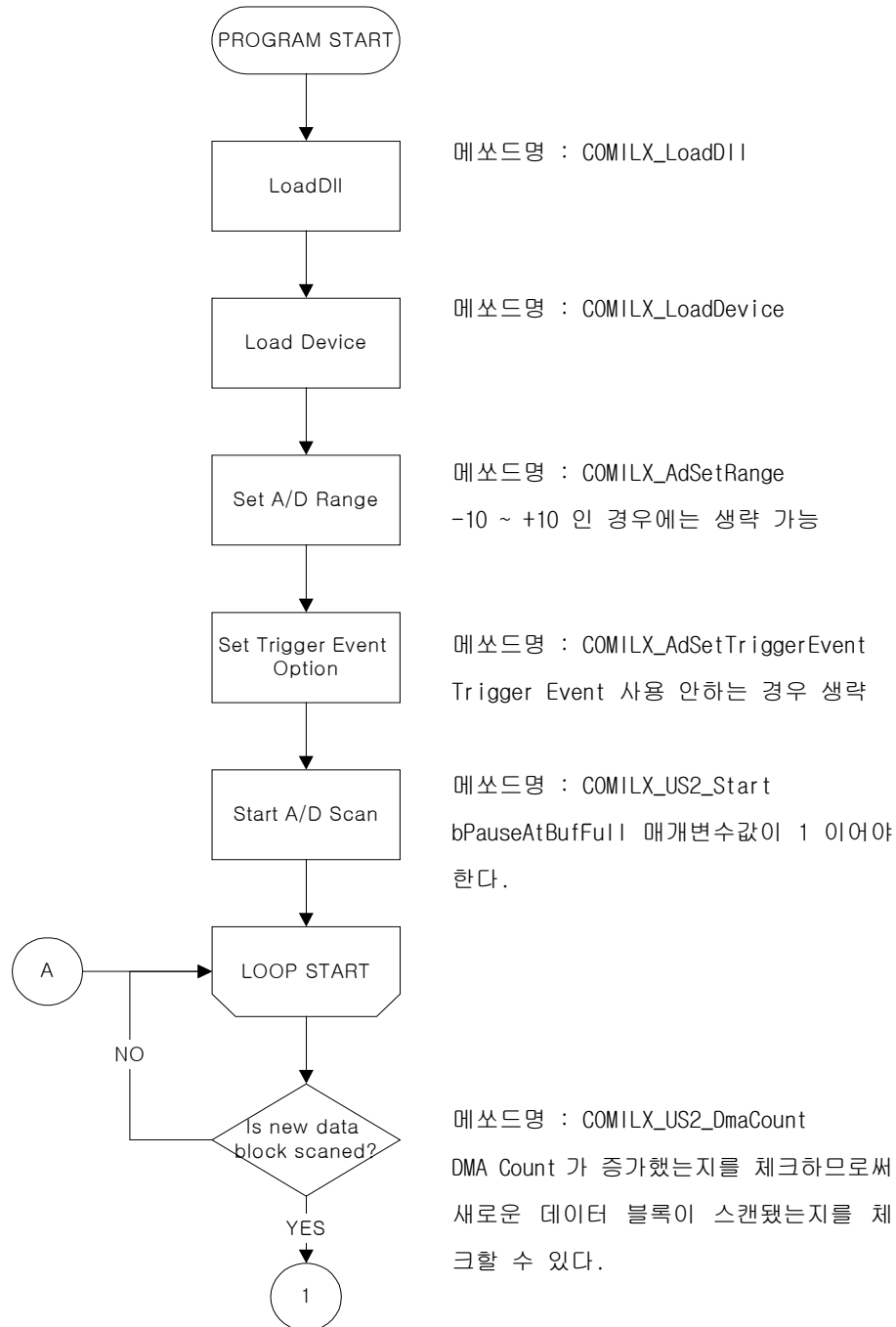
□ Frame Scan 방식을 이용할 때의 일반적인 순서도

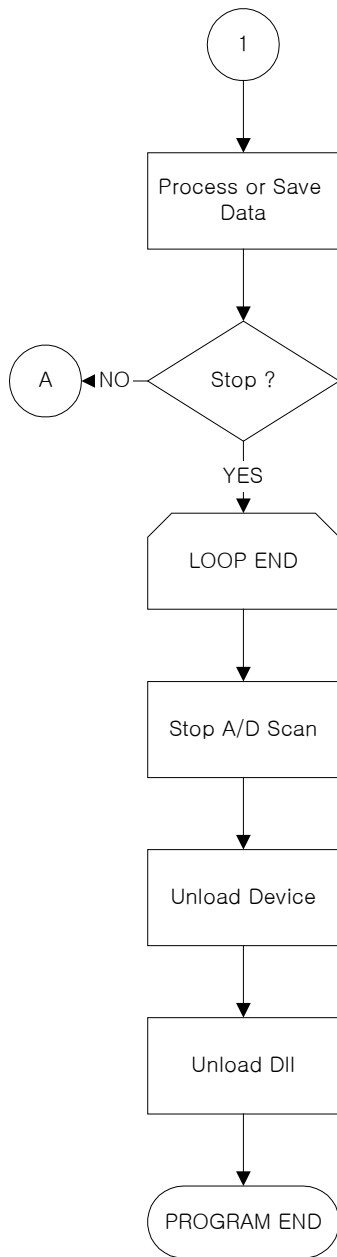




□ Continuous Scan 방식을 이용할 때의 순서도

UI





이 곳은 사용자가 스캔된 데이터를 처리하는 부분임

이 곳에서 COMILX_US2_GetBuffer 또는 COMILX_US2_RetrChannel 메소드를 이용

메소드명 : COMILX_US2_Stop

이때 버퍼를 해제하지 않으면 다음에 COMILX_US2_ReleaseBuf 메소드를 이용하

메소드명 : COMILX_UnloadDevice

메소드명 : COMILX_UnloadDll

■ Us2SetTriggerEvent

메소드 원형

Sub **Us2SetTriggerEvent** (ByVal InputSource As Long, ByVal EdgeType As Long, ByVal TrgMode As Long, ByVal AiRef As Single, ByVal AiRefBand As Single)

메소드 설명

이 메소드는 Trigger Event 를 사용할 것인지를 결정하고, Trigger Event 신호의 종류와 운용 방법을 설정합니다.

Trigger Event 는 A/D 스캔을 시작하는 신호를 의미합니다. Trigger Event 를 사용하지 않는 경우에는 A/D 스캔 시작 메소드를 호출함과 동시에 A/D 스캔이 시작됩니다. 그러나 Trigger Event 를 사용하면 A/D 스캔 시작 메소드를 호출하여도 외부에서 Trigger Event 신호가 입력되기 전까지 스캔 데이터를 버퍼에 저장하지 않습니다.

매개 변수

▶ **InputSource** : Trigger Event 신호로 사용되는 신호원(Signal Source)을 설정합니다. 이 값은 다음 중 하나의 값이어야 합니다.

Value	Meaning
0 또는 TS_NONE	이 값을 지정하면 Trigger Event 를 사용하지 않음을 의미합니다. 즉, A/D Scan 시작 메소드가 호출됨과 동시에 스캔 데이터는 버퍼에 저장됩니다.
1 또는 TS_ANALOG	이 값을 지정하면 A/D CH0 의 신호를 Trigger Event 의 신호원으로 사용합니다.
2 또는 TS_DIGITAL	이 값을 지정하면 Digital 신호(ON/OFF)가 Trigger Event 의 신호원으로 사용됩니다. COMI-LX20x 디바이스는 Trigger Event 전용 Digital Input 단자를 제공합니다. 자세한 사항은 H/W 매뉴얼을 참조하십시오.

▶ **EdgeType** : Trigger Event 신호원에서 발생하는 신호가 Trigger Event 로서 동작하기 위한 신호의 상태를 설정합니다. 이 값은 다음 중 하나의 값이어야 하며 자세한 사항은 아래의 참고 항목을 참조하십시오.

Value	Meaning
0 또는 TE_POSITIVE	Positive Edge
1 또는 TE_NEGATIVE	Negative Edge

▶ **TrgMode** : 이 값은 나중을 위하여 예약된 것이며 현재는 0 으로 지정하여야 합니다.

▶ **AiRef** : 이 값은 nInputSource 가 TS_ANALOG 로 지정되었을 때만 의미를 갖는 값으로써, 이 때에는 A/D CH0 의 값이 레퍼런스값으로 사용됩니다.

▶ **AiRefBand** : 이 값은 nInputSource 가 TS_ANALOG 로 지정되었을 때만 의미를 갖는 값으로써, 유효한 Trigger 신호를 검출하기 위하여 A/I Reference 의 Band 를 설정하기 위한 값입니다 (참고 항목 참조). 이 값의 단위는 A/D 입력 범위에 대한 백분율(%)입니다. 예를 들어, 만일 Input Range 가 -10 ~ +10 이고, fAiRefBand 의 값이 1 인 경우에는 Range 크기가 20 이므로 Band 크기는 $20 * 0.01 = 0.2 \text{ Volt}$ 가 됩니다.

Return 값

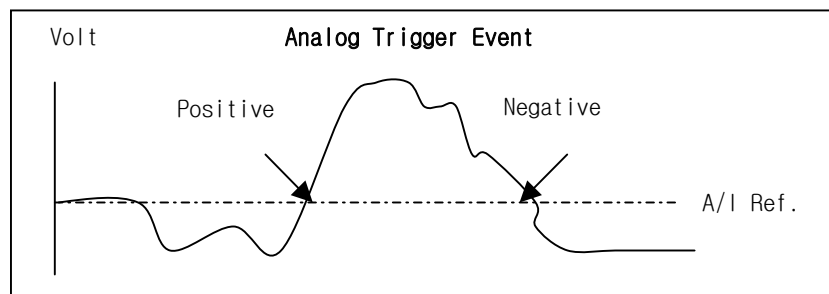
메소드 수행의 성공 여부

Value	Meaning
0	메소드 수행 실패
1	메소드 수행 성공

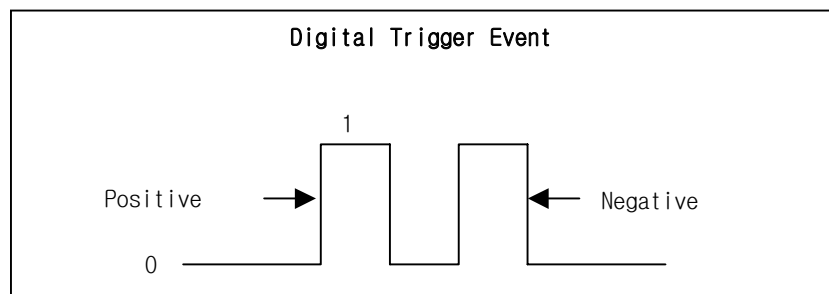
참 고

□ 컴퓨터가 처음 부팅될 때에 COMI-LX20x 디바이스는 기본적으로 Trigger Event 를 사용하지 않는 것으로 설정됩니다.

□ **Edge Type** 을 그림과 함께 설명하면 다음과 같습니다.



[그림 2-2] Analog Trigger Event 시에 Edge Type 의 종류

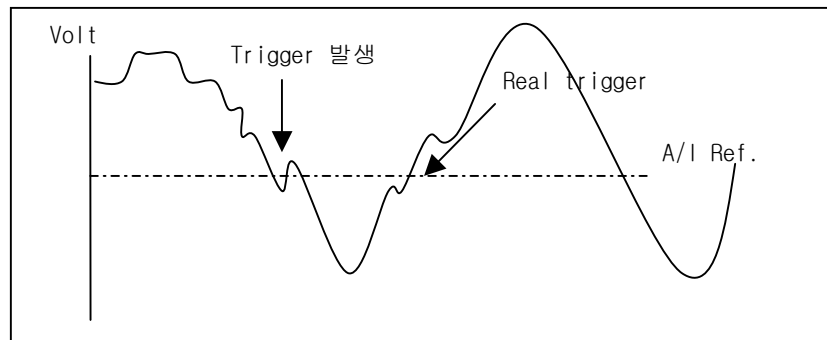


[그림 2-3] Digital Trigger Event 시에 Edge Type 의 종류

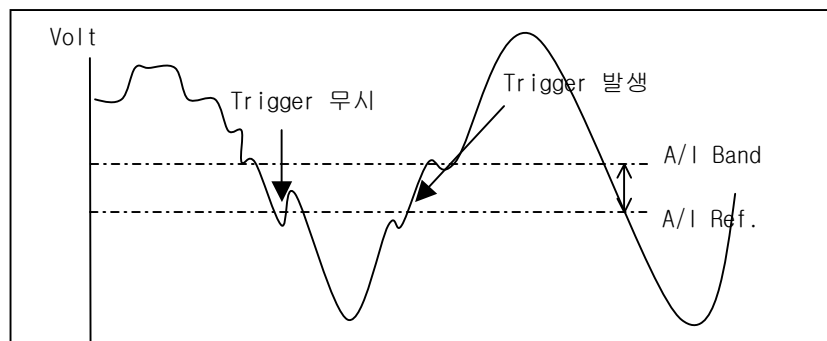
□ Analog Input Reference Band 의 역할

Analog Input Reference Band 는 Noise 등에 의하여 발생할 수 있는 잘못된 Trigger 를 방지하기 위한 것입니다. 예를 들어 [그림 2-4]와 같이 Positive Edge Trigger 모드에서 Negative Edge 순간임에도 불구하고 Noise 의 영향에 의하여 순간적으로 Positive Edge 가 발생할 수 있습니다. 이 때에는 원하지 않는 Trigger 가 발생하게 됩니다.

그러나 [그림 2-5]와 같이 Band 를 지정하면 A/I Reference 에서뿐만 아니라 Band limit 값에서도 Positive Edge 가 발생 해야만 진정한 Trigger Event 로 간주되므로 잘못된 Trigger 를 방지할 수 있습니다.



[그림 2-4] Positive Edge 방식에서 Analog Input Reference Band 를 사용하지 않아 잘못된 Trigger 가 발생하는 예



[그림 2-5] Positive Edge 방식에서 Analog Input Reference Band 를 사용하므로써 잘못된 Trigger 의 발생을 방지하는 예

예제

1) Digital 신호가 LOW 에서 HIGH 로 되는 순간부터 스캔이 시작되도록 하는 경우.

```
Call ComiLxAx1.LoadDevice
Call ComiLxAx1.us2SetTriggerEvent(TS_DIGITAL, TE_POSITIVE, 0, 0, 0)
Call ComiLxAx1.us2Start (...)
```

2) Analog Input(CH0) 신호가 5 Volt 이상이 되는 순간부터 스캔이 시작되도록 하되, 0.1 volt 의 Reference Band 크기를 갖도록 하는 경우.

```
Call ComiLxAx1.LoadDevice
Call ComiLxAx1.AdSetRange(0, -10, 10)
Call ComiLxAx1.Us2SetTriggerEvent(TS_ANALOG, TE_POSITIVE, 0, 5, 0.5)
Call ComiLxAx1.Us2Start(...)
.....
```

3) Trigger Event 를 사용하지 않는 경우

```
Call ComiLxAx1.LoadDevice
Call ComiLxAx1.Us2SetTriggerEvent(TS_NONE, 0, 0, 0, 0)
Call ComiLxAx1.Us2Start(...)
.....
```

■ Us2Start

메소드 원형

```
Function Us2Start(ByVal NumChannel As Long,ByRef ChanList As Long,ByVal ScanFreq As Long, ByVal BufSize As Integer , ByVal PauseAtBufFull As Integer) As Double
```

메소드 설명

이 메소드는 Unlimited scan 기능을 시작합니다.

매개 변수

- ▶ **NumChannel** : 스캔에 사용되는 A/D 채널 수를 지정합니다.
- ▶ **ChanList** : A/D scan 을 수행할 채널 리스트를 담고 있는 배열
- ▶ **ScanFreq** : A/D scan 주파수를 설정합니다. 단위는 Hz 입니다.
- ▶ **BufSize** : 이 값은 스캔 버퍼의 크기를 결정합니다. 그러나 이 값이 스캔 버퍼의 직접적인 크기를 지정하지는 않습니다. 실제 버퍼의 크기는 $BufSize * 4096$ 개의 데이터를 저장할 수 있는 크기로 할당됩니다(여기서 4096 값은 한번의 DMA 가 수행될 때 전송되는 데이터 개수입니다). 예를 들어 채널 당 4096 개의 데이터를 저장할 수 있는 크기로 버퍼를 할당하려면 이 값을 1 로 하여야 하며, 40960 개의 데이터를 저장할 수 있도록 하려면 이 값을 10 으로 하여야 합니다.
- ▶ **PauseAtBufFull** : 이 값은 스캔 버퍼에 데이터가 꽉 찬 경우에 스캔을 일시 중지할 것인지 결정합니다. 이 값을 1 로 하면 Frame Scan, 0 으로 하면 Continuous Scan 방식으로 운용됩니다. 자세한 사항은 단원 앞 부분의 “Frame Scan 과 Continuous Scan” 설명을 참조하십시오.

Return 값

실제 스캔 주파수를 Hz 단위로 반환합니다. 스캔 주파수를 결정하는 내장 타이머가 정수값을 분주하여 주파수를 결정하므로 사용자가 지정한 스캔 주파수와 실제 스캔 주파수는 약간의 차이가 있을 수 있습니다.

참고

□ BufSize 및 PauseBufFull 값을 설정할 때 주의 사항

- ① COMI-LX20x 디바이스는 각 채널당 8192 개의 데이터를 저장할 수 있는 FIFO 메모리를 가지고 있습니다. 따라서 BufSize 값을 2 로 하고 PauseBufFull 을 TRUE 로 하면 속도와 상관없이 8 K 의 연속 데이터를 얻을 수 있습니다.
- ② (채널수 * 스캔 주파수) 가 5 MHz 보다 큰 경우에는 BufSize 을 2 로하고

PauseBufFull 을 TRUE 로 하는 것이 좋습니다. 이는 DMA 속도가 스캔속도보다 느리게 되어 연속적으로 데이터를 전송할 수 없기때문입니다.

③ CH0 와 CH3 만을 사용하는 경우에 BufSize 을 4 로 하고 PauseBufFull 을 TRUE 로 하면 자동적으로 2 개의 FIFO 를 각 채널에 할당하는 CASCADE 방식으로 운용하여 스캔 속도와 관계없이 16384 (16K)개의 연속 데이터를 얻을 수 있습니다.

④ Us2SetTrgEvent()메소드를 수행하여 트리거 이벤트를 TM_MIDDLE 로 설정한 경우에는 BufSize 을 2 로 하고 PauseBufFull 을 TRUE 로 하여야 합니다.

■ Us2Resume

메소드 원형

Sub Us2Resume

메소드 설명

이 메소드는 일시 중지된 A/D 스캔을 재개하여 줍니다. Us2Start()메소드에서 PauseAtBufFull 값을 TRUE 로 지정한 경우에는 스캔 버퍼에 데이터가 다 차게 되면 스캔이 일시 중지됩니다. 이 때 사용자는 필요에 따라 스캔 데이터를 처리하고 Us2Resume () 메소드를 이용하여 스캔을 재개할 수 있습니다.

예제

```
Const NUM_CH =4 ` 채널 수 = 4 채널
Const SCAN_FREQ = 100000 ` 100 KHz sampling
Const BUF_SIZE_GAIN = 2 ` 채널당 버퍼 크기 = 2 * 4096 = 8192 개의 데이터를
담을 수 있는 공간
.....
Call ComiLxAx1.LoadDevice
Dim ChanList(3)
For i=0 To 3
    ChanList(i) = i
Next i

Call ComiLxAx1.Us2Start(NUM_CH,ChanList(0),SCAN_FREQ, BUF_SIZE_GAIN ,
TRUE)

While ( Not (IsStop())) ` IsStop()은 강제 종료에 해당하는 가상의 메소드임.
    CurCount = ComiLxAx1.Us2DmaCount ` 버퍼가 차면 데이터를 처리하고, 다시 스
캔을 재개한다.
    If ( CurCount = BUF_SIZE_GAIN) Then
        ` ProcessData()는 스캔 데이터를 취하여 처리하는 가상의 메소드
        Call ProcessData(...)
        .....
        Call ComiLxAx1.Us2Resume
    Wend

Call ComiLxAx1.Us2Stop(TRUE)
Call ComiLxAx1.UnloadDevice
```

■ Us2IsBufFull

메소드 원형

Function **Us2IsBufFull** As Boolean

메소드 설명

이 메소드는 지정한 크기(개수)의 스캔 버퍼에 데이터가 다 찼는지를 알려주는 메소드입니다. 이 것은 Us2Start(...)메소드에서 PauseAtBufFull 매개 변수 값을 TRUE 또는 1로 하였을 경우에만 해당하는 것입니다.

예제

```
Const NUM_CH =4 ` 채널 수 = 4 채널
Const SCAN_FREQ = 100000 ` 100 KHz sampling
Const BUF_SIZE_GAIN = 2 ` 채널당 버퍼 크기 = 2 * 4096 = 8192 개의 데이터를
담을 수 있는 공간
.....
Call ComiLxAx1.LoadDevice
Dim ChanList(3)
For i=0 To 3
    ChanList(i) = i
Next i

Call ComiLxAx1.Us2Start(NUM_CH,ChanList(0),SCAN_FREQ, BUF_SIZE_GAIN ,
TRUE)

While ( Not (IsStop())) ` IsStop()은 강제 종료에 해당하는 가상의 메소드임.
    ` 버퍼가 차면 데이터를 처리하고, 다시 스캔을 재개한다.
    If ( ComiLxAx1.Us2IsBufFull) Then
        ` ProcessData()는 스캔 데이터를 취하여 처리하는 가상의 메소드
        Call ProcessData(...)
        .....
        Call ComiLxAx1.Us2Resume
    Wend

Call ComiLxAx1.Us2Stop(TRUE)
Call ComiLxAx1.UnloadDevice
```

■ Us2ChangeScanFreq As Long

메소드 원형

Function **Us2ChangeScanFreq** (ByVal ScanFreq As Long) As Double

메소드 설명

이 메소드는 스캔이 진행되는 중에 스캔 주파수를 변경합니다.

매개 변수

▶ **ScanFreq** : A/D scan 주파수를 설정합니다. 단위는 Hz 입니다.

Return 값

실제 스캔 주파수를 Hz 단위로 반환합니다. 스캔 주파수를 결정하는 내장 타이머가 정수값을 분주하여 주파수를 결정하므로 사용자가 지정한 스캔 주파수와 실제 스캔 주파수는 약간의 차이가 있을 수 있습니다.

■ Us2DmaCount

메소드 원형

Function **Us2DmaCount** As Long

메소드 설명

이 메소드는 스캔이 시작된 후에 현재까지 몇회의 DMA 데이터블록이 전송됐는지를 알려주는 메소드입니다.

Return 값

현재까지 진행된 DMA 전송 횟수를 반환합니다. 1 회의 DMA 전송에 의해 각 채널당 4096 개의 데이터가 전송되므로 실제로 획득된 데이터수 수는

$N = 4096 * C$ 가 됩니다.

여기서

N : 각 채널당 획득된 데이터 수

C : DMA 전송 횟수

■ Us2RetrvChannel

메소드 원형

Function **Us2RetrvChannel** (ByVal ChanOrder As Long, ByVal StartCount As Long, ByVal MaxNumData As Long, ByRef DestBuf As Double) As Long

메소드 설명

이 메소드는 A/D Scan 채널 중에서 하나의 채널에 대한 데이터 블록을 Voltage 값으로 환산하여 전달합니다. 데이터 블록은 사용자가 지정한 StartCount 에서부터 MaxNumData 에서 지정한 수 만큼이 됩니다.

매개 변수

- ▶ **ChanOrder** : 데이터를 취하기 원하는 채널의 채널 리스트 상의 순서(0 based)입니다. 이 값은 채널 번호가 아님을 주의하여야 한다.
- ▶ **StartCount** : 전달 받고자 하는 데이터 블록의 시작 Scan count.
- ▶ **MaxNumData** : 전달 받고자 하는 데이터 블록의 크기(데이터 수)를 지정 합니다. 이 값이 양수이면 StartCount 부터 이후에 스캔된 데이터 중 MaxNumData 에서 지정한 수만큼 데이터를 전달합니다. 이 값이 음수이면 StartCount 부터 이전에 스캔된 데이터 중 MaxNumData 에서 지정한 수만큼 데이터를 전달합니다.
- ▶ **DestBuf** : 데이터를 전달 받을 배열을 지정합니다. 이 버퍼의 크기는 MaxNumData 에서 지정한 값보다 크거나 같아야 한다.

Return 값

실제 전달된 데이터 수를 반환합니다. 만일 StartCount 이후에 현재까지 스캔된 데이터 수가 MaxNumData 에서 지정한 수 보다 작으면, 현재 스캔된 데이터까지만 전달하게됩니다.

예제

□ **예제 1** : Frame Scan 방식을 사용할 때 Us2RetrvChannel 메소드를 이용하여 데이터를 취하는 예

```
Const NUM_CH =4 ' 채널 수 = 4 채널
Const SCAN_FREQ = 100000 ' 100 KHz sampling
Const BUF_SIZE_GAIN = 2 ' 채널당 버퍼 크기 = 2 * 4096 = 8192 개의 데이터를 담을 수 있는 공간
Dim DestBuf(8192) As Double

Call ComiLxAx1.LoadDevice
Dim ChanList(3)
For i=0 To 3
    ChanList(i) = i
Next i

Call ComiLxAx1.Us2Start(NUM_CH,ChanList(0),SCAN_FREQ, BUF_SIZE_GAIN ,
```

TRUE)

While (Not (IsStop())) ' IsStop()은 강제 종료에 해당하는 가상의 메소드임.

 ' 버퍼가 차면 데이터를 처리하고, 다시 스캔을 재개한다.

 If (ComiLxAx1.Us2IsBufFull) Then

 For i=0 To NUM_CH - 1

 count = ComiLxAx1.Us2RetrvChannel (i, 1, BUF_SIZE_GAIN *4096, DestBuf(0))

 PlotData(DestBuf(0), count)

 ' PlotData()메소드는 가상의 메소드임.

 Next i

 End If

 Call ComiLxAx1.Us2Resume

Wend

Call ComiLxAx1.Us2Stop(TRUE)

Call ComiLxAx1.UnloadDevice

□ 예제 2 : Continuous Scan 방식을 사용할 때 Us2RetrvChannel 메소드를 사용하여 데이터를 취하는 예

Const NUM_CH =4 ' 채널 수 = 4 채널

Const SCAN_FREQ = 100000 ' 100 KHz sampling

Const BUF_SIZE_GAIN = 2 ' 채널당 버퍼 크기 = 2 * 4096 = 8192 개의 데이터를 담을 수 있는 공간

Dim DestBuf(8192) As Double

Call ComiLxAx1.LoadDevice

Dim ChanList(3)

For i=0 To 3

 ChanList(i) = i

Next i

Call ComiLxAx1.Us2Start(NUM_CH, ChanList, SCAN_FREQ, BUF_SIZE_GAIN, FALSE)

While(Not (IsStop())) ' IsStop()은 가상의 메소드임.

 CurCount = ComiLxAx1.Us2DmaCount

 ' 새로운 스캔 데이터가 있는지 체크하고 있으면 데이터를 처리한다.

 If (CurCount > PrvCount) Then

 NumData = (CurCount - PrvCount) * 4096

 For i=0 To NUM_CH -1

 count = ComiLxAx1.Us2RetrvChannel (i, PrvCount*4096+1, USER_BUF_SIZE, DestBuf(0))

 PlotData(DestBuf(0), count) ' PlotData()메소드는 가상의 메소드임.

 End If

 PrvCount = CurCount

 Wend

Call ComiLxAx1.Us2Stop(TRUE)

Call ComiLxAx1.UnloadDevice

or

■ Us2Stop

메소드 원형

Sub **Us2Stop** (ByVal ReleaseBuf As Integer)

메소드 설명

이 메소드는 A/D 스캔을 종료합니다.

매개 변수

▶ **ReleaseBuf** : Us2Start()가 수행될 때 할당되었던 스캔 버퍼를 메모리 해제시킬것인지를 결정합니다. 만일 이 값을 FALSE 로 지정하면 후에 반드시 Us2ReleaseBuf()를 사용하여 버퍼를 해제하여야 합니다. 이 값을 TRUE 로 지정하면 Us2ReleaseBuf() 메소드를 수행할 필요가 없습니다.

■ Us2ReleaseBuf

메소드 원형

Function **Us2ReleaseBuf** As Boolean

메소드 설명

Us2Start()가 수행될 때 할당되었던 스캔 버퍼를 메모리 해제 시킵니다. 이 메소드는 Us2Stop() 메소드가 호출되기 전에 수행되어서는 안되며, Us2Stop() 메소드를 호출할 때 두 번째 파라미터를 TRUE 로 지정했을 때는 사용하지 않아도 됩니다.

Return 값

Value	Meaning
False	메소드 수행 실패
True	메소드 수행 성공

2.3 아날로그 출력(Analog Output)

이 단원에서는 Analog Output 에 관한 메소드를 소개합니다. COMIDAS 에서는 두 가지 형태의 Analog Output 기능이 있습니다.

첫 번째는 일반적인 Analog Output 기능으로써 사용자가 지정한 전압을 출력하는 기능입니다.

두 번째는 Waveform Generation 기능입니다. Waveform Generation 기능은 Sine Wave 또는 Square Wave 등과 같이 사용자가 지정하는 주기성을 가지는 신호를 자동으로 생성해주는 기능입니다.

Analog Output 과 관련된 메소드들의 리스트는 다음과 같습니다.

메소드명	각 보드별 지원 여부								
	LX101	LX102	LX103	LX201	LX202	LX203	LX301	LX401	LX50x
DaOut	V	V	V				V		
WfmStart	V	V	V				V		
WfmReload	V	V	V				V		
WfmRateChange	V	V	V				V		
WfmGetCurPos	V	V	V				V		
WfmGetCurLoops	V	V	V				V		
WfmStop	V	V	V				V		

[표 2-5] Analog Output 관련 메소드 리스트 및 각 보드별 지원 여부

2.3.1. 일반적인 Analog Output 메소드

■ DaOut

메소드 원형

Function **DaOut** (ByVal ch As Long, ByVal volt As Single) As Integer

메소드 설명

이 메소드는 지정한 Analog Output 채널에 지정한 Voltage 를 출력합니다.

매개 변수

- ▶ **ch** : Analog Output 채널번호. 채널 번호는 0 부터 시작합니다.
- ▶ **volt** : Analog Output 출력 Voltage.

Return 값

Value	Meaning
0	메소드 수행 실패
1	메소드 수행 성공

2.3.2. Waveform Generation

■ WfmStart

메소드 원형

```
long WfmStart(ByVal ch As Long, ByRef DataBuffer As Single, ByVal NumData As Long,
    ByVal PPS As Long, ByVal MaxLoops As Long) As Long
```

메소드 설명

Waveform Generation 을 시작합니다.

매개 변수

- ▶ **ch** : Analog Output 채널번호. 채널 번호는 0 부터 시작합니다.
- ▶ **DataBuffer** : Waveform 데이터를 담은 버퍼 배열
- ▶ **NumData** : 버퍼에 담겨진 데이터의 수
- ▶ **PPS** : Waveform Generation 의 주기를 결정합니다. 이 값은 Points/Second 입니다. 예를 들어 100 개의 데이터로 한 주기를 구성하였다면 10Hz 의 신호를 만들기 위해서는 PPS 는 1000 이 되어야 합니다.
- ▶ **MaxLoops** : 이 값이 0 보다 크면 생성되는 Wave 신호의 수를 제한합니다. 이 값이 0 이면 WfmStop()메소드가 수행되기 전까지 계속하여 Wave 신호를 생성합니다.

Return 값

실제로 설정되는 Points/Second 를 반환합니다. 사용자가 지정한 PPS 와 실제로 설정되는 PPS 는 약간의 차이가 있을 수 있습니다. 이 값이 0 보다 작으면 수행도중 에러가 발생하였음을 의미합니다.

예제

□ 10 KHz, -5 ~ 5 Volt, Square Wave 신호를 계속 발생시킬 때의 예.

```
Dim DataBuffer(1)
DataBuffer(0) = -5
DataBuffer(1) = 5
count = ComiLxAx1.WfmStart(0, DataBuffer(0), 2, 10000 * 2, 0)
Call MsgBox ("Actual Freq(Hz)" & count, vbOKOnly)
```

□ 10 KHz, -5 ~ 5 Volt, Square Wave 신호를 1000 개 발생시킬 때의 예.

```
Dim DataBuffer(1)
DataBuffer(0) = -5
DataBuffer(1) = 5
Call ComiLxAx.WfmStart(0, DataBuffer(0), 2, 10000 * 2, 1000)
```

□ 1 KHz, -5 ~ 5 Volt, Sine Wave 신호를 계속 발생시킬 때의 예.

```
Const NUM_DATA = 50
Dim DataBuffer(NUM_DATA-1)
rad = 2*3.141592/ NUM_DATA
For i=0 To NUM_DATA-1
    DataBuffer(i) = 5 * sin(i*rad)
Next i
Call ComiLxAx1.WfmStart(0, DataBuffer(0), NUM_DATA,1000*NUM_DATA,0)
```

■ WfmReload

메소드 원형

```
Function WfmReload (ByVal ch As Long, ByRef DataBuffer As Single, ByVal NumData As Long) As Integer
```

메소드 설명

Waveform Generation 이 진행되고 있는 중에 주기(Wave 신호) 데이터를 변경합니다.

매개 변수

- ▶ *ch* : Analog Output 채널번호. 채널 번호는 0 부터 시작합니다.
- ▶ *DataBuffer* : Waveform 데이터를 담은 버퍼 배열
- ▶ *NumData* : 버퍼에 담겨진 데이터의 수

■ WfmRateChange

메소드 원형

Function **WfmRateChange** (ByVal ch As Long, ByVal PPS As Long) As Long

메소드 설명

Waveform Generation 이 진행되고 있는 중에 주파수(PPS)를 변경합니다.

매개 변수

- ▶ *ch* : Analog Output 채널번호. 채널 번호는 0 부터 시작합니다.
- ▶ *PPS* : Points/Second. WfmStart 메소드 참조.

Return 값

실제로 설정되는 Points/Second 를 반환합니다. 사용자가 지정한 PPS 와 실제로 설정되는 약간의 차이가 있을 수 있습니다. 이 값이 0 보다 작으면 수행도중 에러가 발생하였음을 의미합니다.

■ WfmGetCurPos

메소드 원형

Function **WfmGetCurPos** (ByVal ch As Long) As Long

메소드 설명

현재 출력되고 있는 주기 데이터의 위치를 반환합니다. 즉, 현재 출력되고 있는 데이터 포인트가 주기 데이터의 몇 번째 데이터인지를 알려줍니다.

매개 변수

▶ *ch* : Analog Output 채널번호. 채널 번호는 0 부터 시작합니다.

Return 값

현재 출력되고 있는 주기 데이터의 인덱스(Index).

■ WfmGetCurLoops

메소드 원형

Function **WfmGetCurLoops** (ByVal ch As Long) As Long

메소드 설명

현재 남아있는 Wave 신호의 주기 수를 반환합니다.

매개 변수

▶ *ch* : Analog Output 채널번호. 채널 번호는 0 부터 시작합니다.

Return 값

이 메소드는 현재 남아있는 Wave 신호의 주기 수를 반환합니다. 예를 들어 MaxLoops 를 1000 으로 하였을 때 이 메소드가 100 을 반환한다면 현재까지 900 회의 Wave 신호가 발생하였으며, 100 회의 Wave 신호가 남았음을 의미합니다.

or

■ WfmStop

메소드 원형

Sub **WfmStop** (ByVal ch As Long)

메소드 설명

Waveform Generation 을 종료합니다.

매개 변수

▶ *ch* : Analog Output 채널번호. 채널 번호는 0 부터 시작합니다.

2.4 디지털 입출력(Digital Input/Output)

이 단원에서는 Digital Input 과 Output 에 관한 메소드를 소개합니다. 일반적으로 Digital Input 은 스위치(Switch)의 상태를 읽어들이는데 사용되고, Digital Output 은 스위치의 상태를 제어하는데 사용됩니다.

(주)커미조아에서 제공하는 COMI-LX 시리즈 디바이스는 크게 두 가지로 구분되는 디지털 입출력 방식을 제공합니다.

하나는 일반적인 디지털 입출력 기능입니다. 일반적인 디지털 입출력은 PC 에 장착된 PCI 보드에서 디지털 입출력을 직접제어하는 것을 의미하며 이와 관련된 메소드들은 [2.4.1 일반적인 디지털 입출력] 단원에서 설명됩니다. COMI-LX402 Serial DIO Master Board 를 제외한 대부분의 COMI-LX 시리즈 디바이스들이 일반적인 디지털 입출력 기능을 지원하며 각 디바이스별 일반적인 디지털 입출력 기능 지원 여부는 [표 2-6]을 참조하십시오.

또 하나의 디지털 입출력 방식은 시리얼 통신(RS-422)을 이용하여 디지털 입출력을 제어하는 방식입니다. 이 방식은 PC 에 장착되어 시리얼 통신을 관장하는 마스터보드(COMI-LX402 보드)와 외부에 설치되어 실제 디지털 입출력을 제어하는 터미널 모듈(COMI-LXTM4A)이 RS-422 시리얼 통신으로 연결되어 제어되는 방식입니다. 이 방식은 사용자가 프로그램 상에서 디지털 입출력 명령을 수행하면 마스터보드가 해당 터미널 모듈에 시리얼통신을 이용하여 명령을 전달하고 해당 터미널 모듈이 디지털 입출력 명령을 수행하는 메카니즘을 사용합니다. 이 방식은 하나의 마스터보드에 최대 16 개까지 터미널모듈을 확장하여 사용할 수 있다는 큰 장점이 있습니다. 이와 관련된 메소드들은 [2.4.2 시리얼 통신을 이용한 디지털 입출력] 단원에서 설명됩니다.

2.4.1 일반적인 디지털 입출력

이 단원에서는 일반적인 디지털 입출력 기능에 관련된 메소드들을 소개합니다. 일반적인 디지털 입출력은 PC 에 장착된 PCI 보드에서 디지털 입출력을 직접 제어하는 것을 의미합니다. COMI-LX402 Serial DIO Master Board 를 제외한 대부분의 COMI-LX 시리즈 디바이스들이 일반적인 디지털 입출력 기능을 지원하며 각 디바이스별 일반적인 디지털 입출력 기능 지원 여부는 [표 2-6]을 참조하십시오.

메소드명	각 보드별 적용 여부								
	LX101	LX102	LX103	LX201	LX202	LX203	LX301	LX401	LX50x
DioSetUsage	V	V	V	V	V	V	V	V	V
DiGetOne	V	V	V	V	V	V	V	V	V
DiGetAll	V	V	V	V	V	V	V	V	V
DoPutOne	V	V	V	V	V	V	V	V	V
DoPutAll	V	V	V	V	V	V	V	V	V

[표 2-6] Digital Input/Output 에 관련된 메소드 리스트 및 각 보드별 지원 여부

■ DioSetUsage

메소드 원형

Sub **DioSetUsage** (ByVal usage As TCmDiDoUsage)

메소드 설명

이 메소드는 Digital Input/Output 단자의 용도를 설정합니다. COMILX-시리즈 디바이스는 Digital Input/Output 단자가 따로 구분되어 있지 않고 필요에 따라 용도를 설정할 수 있도록 되어 있습니다. 예를 들어 COMI-LX201 보드에는 8 개의 Digital Input / Output 단자가 있는데, 8 개의 단자를 모두 Input 전용으로 사용하거나 Output 전용으로 사용할 수 있습니다.

매개 변수

► **usage** : DIO 단자의 용도를 설정합니다. 이 값은 다음의 세 값 중 하나이어야 합니다.

Value	Meaning
0 또는 DI_ONLY	전 채널을 Digital Input 채널로 사용합니다. 이 때 채널 수는 디바이스에 따라 다르므로 H/W 매뉴얼을 참조하십시오.
1 또는 DI_DO	전 채널의 반은 Digital Input 으로, 나머지 반은 Digital Output 으로 사용합니다. 단, COMI-LX20 시리즈에서는 이 모드를 지원하지 않습니다.
2 또는 DO_DI	전 채널의 반은 Digital Output 으로, 나머지 반은 Digital Input 으로 사용합니다. 단, COMI-LX20 시리즈에서는 이 모드를 지원하지 않습니다.
3 또는 DO_ONLY	전 채널을 Digital Output 채널로 사용합니다. 이 때 채널 수는 디바이스에 따라 다르므로 H/W 매뉴얼을 참조하십시오.

참 고

□ D/I 와 D/O 채널 번호에 관하여

DIO 단자의 용도 설정에 따라 디지털 입력과 디지털 출력 채널 번호가 어떻게 되는지에 대해 혼동될 수 있습니다. 디지털 입력과 출력의 채널번호는 언제나 각각 0 번부터 시작합니다. 예를 들어 COMI-LX101 보드는 16 채널의 DIO 단자를 제공하는데 이를 DI_DO 로 설정하였다면 DI00 ~ DI07 의 단자는 디지털 입력 CH0 ~ CH7 로 사용되고 DI08 ~ DI015 의 단자는 디지털 출력 CH0 ~ CH7 로 사용됩니다. 반대로 DO_DI 모드로 설정하였다면 DI00 ~ DI07 의 단자는 디지털 출력 CH0 ~ CH7 로 사용되고 DI08 ~ DI015 의 단자는 디지털 입력 CH0 ~ CH7 로 사용됩니다.

예 제

이 프로그램은 DoPutOne(..)과 DiGetOne(..) 메소드를 사용하여 D/O CH0 을 통하여

ur

STATUS 를 반복적으로 반전하면서 출력을 내보내고 D/I CH0 의 STATUS 를 반복적으로 체크하는 것입니다. D/O CH0 와 D/I CH0 를 서로 연결한다면 D/O CH0 의 출력을 D/I CH0 를 통하여 반복적으로 체크할 수 있습니다.

```
Dim do_state
```

```
Call ComiLxAx1.LoadDevice
```

```
Call ComiLxAx1.DioSetUsage(DI_DO)
```

```
Private Sub Timer_Timer()
```

```
do_state = ( do_state + 1 ) mod 2 ` state 반전
```

```
Call ComiLxAx1.DoPutOne(0, do_state) ` Put D/O
```

```
`Get D/I and print on screen
```

```
Label1.caption = ComiLxAx1.DiGetOne(0)
```

```
End Sub
```

```
Call ComiLxAx1.UnloadDevice
```

■ DiGetOne

메소드 원형

Function **DiGetOne** (ByVal ch As Long) As Long

메소드 설명

이 메소드는 지정한 Digital Input 채널의 Status 를 반환합니다.

매개 변수

▶ **ch** : Digital Input 채널번호. 채널번호는 0 부터 시작합니다.

Return

Digital Input 채널의 Status.

Value	Meaning
0	OFF
1	ON

예 제

이 프로그램은 DoPutOne(..)과 DiGetOne(..) 메소드를 사용하여 D/O CH0 을 통하여 STATUS 를 반복적으로 반전하면서 출력을 내보내고 D/I CH0 의 STATUS 를 반복적으로 체크하는 것입니다. D/O CH0 와 D/I CH0 를 서로 연결한다면 D/O CH0 의 출력을 D/I CH0 를 통하여 반복적으로 체크할 수 있습니다.

```
Dim do_state

Call ComiLxAx1.LoadDevice
Call ComiLxAx1.DioSetUsage(DI_DO)

Private Sub Timer_Timer()

    do_state = ( do_state + 1 ) mod 2 ` state 반전
    Call ComiLxAx1.DoPutOne(0, do_state) ` Put D/O

    `Get D/I and print on screen
    Label1.caption = ComiLxAx1.DiGetOne(0)

End Sub

Call ComiLxAx1.UnloadDevice
```

or

■ DiGetAll

메소드 원형

Function **DiGetAll** As Long

메소드 설명

이 메소드는 해당 디바이스의 모든 Digital Input 채널의 Status 를 반환합니다.

Return 값

32 개의 채널에 대한 Input Status 를 32 비트 값으로 반환합니다. 각비트는 비트 순서와 일치하여 각 채널의 ON/OFF 상태를 나타냅니다. 단, 디바이스에 따라 32 채널 미만의 Digital 채널을 지원하는 경우에는 BIT0 부터 해당 채널 수 만큼의 비트만 사용하시면 됩니다.

예 제

이 프로그램은 DiGetAll(..)과 DoPutAll(..) 메소드를 사용하여 D/I 와 D/O 의 8 채널을 동시에 컨트롤하는 예제입니다.

```
Dim do_state

Call ComiLxAx1.LoadDevice
Call ComiLxAx1.DioSetUsage(DI_DO)

Private Sub Timer_Timer()
    do_state = ( do_state + 1 ) mod 2 ` state 반전
    Call ComiLxAx1.DoPutAll( do_state ) ` Put D/O

    `Get D/I and print on screen
    di_state = ComiLxAx1.DiGetAll
    For i=0 To 7
        Label1(i).caption = readBitFromByte(di_state, i)
    Next i
End Sub

`readBitFromByte () : 바이트 데이터의 특정 비트의 값을 반환하는 메소드
Private Function readBitFromByte(Status, index) As Boolean
    readBitFromByte = (Status And (2 ^ index)) / (2 ^ index)
End Function

Call ComiLxAx1.UnloadDevice
```


■ DoPutOne

메소드 원형

```
Sub DoPutOne (ByVal ch As Long, ByVal status As Long)
```

메소드 설명

이 메소드는 지정한 Digital Output 채널에 지정한 Status 로 출력을 내보냅니다.

매개 변수

- ▶ **ch** : Digital Output 채널번호. 채널 번호는 0 부터 시작한다.
- ▶ **status** : 출력 Status. 0 - OFF, 1 - ON.

예 제

이 프로그램은 DoPutOne(..)과 DiGetOne(..) 메소드를 사용하여 D/O CH0 을 통하여 STATUS 를 반복적으로 반전하면서 출력을 내보내고 D/I CH0 의 STATUS 를 반복적으로 체크하는 것입니다. D/O CH0 와 D/I CH0 를 서로 연결한다면 D/O CH0 의 출력을 D/I CH0 를 통하여 반복적으로 체크할 수 있습니다.

```
Dim do_state

Call ComiLxAx1.LoadDevice
Call ComiLxAx1.DioSetUsage(DI_DO)

Private Sub Timer_Timer()

    do_state = ( do_state + 1 ) mod 2 ` state 반전
    Call ComiLxAx1.DoPutOne(0, do_state) ` Put D/O

    'Get D/I and print on screen
    Label1.caption = ComiLxAx1.DiGetOne(0)

End Sub

Call ComiLxAx1.UnloadDevice
```

or

■ DoPutAll

메소드 원형

```
Sub DoPutAll (ByVal Statuses As Long)
```

메소드 설명

이 메소드는 해당 디바이스의 모든 Digital Output 채널에 출력을 내보낸다.

매개 변수

▶ **Statuses** : 모든 Digital Output 채널의 출력 Status 를 나타내는 32 bit 값. 이 값의 각 비트의 값이 각 채널의 Status 를 나타냅니다.

예 제

이 프로그램은 DiGetAll(...)과 DoPutAll(...) 메소드를 사용하여 D/I 와 D/O 의 8 채널을 동시에 컨트롤하는 예제입니다.

```
Dim do_state

Call ComiLxAx1.LoadDevice
Call ComiLxAx1.DioSetUsage(DI_DO)

Private Sub Timer_Timer()
    do_state = ( do_state + 1 ) mod 2 ` state 반전
    Call ComiLxAx1.DoPutAll( do_state ) ` Put D/O

    `Get D/I and print on screen
    di_state = ComiLxAx1.DiGetAll
    For i=0 To 7
        Labell(i).caption = readBitFromByte(di_state, i)
    Next i
End Sub

`readBitFromByte () : 바이트 데이터의 특정 비트의 값을 반환하는 메소드
Private Function readBitFromByte(Status, index) As Boolean
    readBitFromByte = (Status And (2 ^ index)) / (2 ^ index)
End Function

Call ComiLxAx1.UnloadDevice
```

2.4.2 시리얼 통신을 이용한 디지털 입출력

이 단원에서는 시리얼 통신을 이용한 디지털 입출력 기능에 관련된 메소드들을 소개합니다. 시리얼 통신을 이용한 디지털 입출력 방식은 시리얼 통신(RS-422)을 이용하여 디지털 입출력을 제어하는 방식입니다. 이 방식은 PC 에 장착되어 시리얼 통신을 관장하는 마스터보드(COMI-LX402 보드)와 외부에 설치되어 실제 디지털 입출력을 제어하는 터미널 모듈(COMI-SDM4A)이 RS-422 시리얼 통신으로 연결되어 제어되는 방식입니다. 이 방식은 사용자가 프로그램 상에서 디지털 입출력 명령을 수행하면 마스터보드가 해당 터미널 모듈에 시리얼 통신을 이용하여 명령을 전달하고 해당 터미널 모듈이 디지털 입출력 명령을 수행하는 메카니즘을 사용합니다. 이 방식은 하나의 마스터보드에 최대 16 개까지 터미널모듈을 확장하여 사용할 수 있다는 큰 장점이 있습니다.

각 COMI-SDM4A 터미널 모듈은 16 채널의 디지털 입출력 채널을 제공하며, 하나의 COMI-SD401 마스터 보드에 16 개의 COMI-SDM4A 터미널 모듈이 확장되어 연결될 수 있어서 하나의 마스터 보드가 총 256 개의 디지털 입출력 채널을 제어할 수 있습니다.

메소드명	각 보드별 적용 여부					
	LX10x	LX20x	LX301	LX401	LX402	LX50x
SdioInitComm					V	
SdioCheckModule					V	
SdioCheckModule					V	
SdioSetDioUsage					V	
SdioReadLowByte					V	
SdioReadHighByte					V	
SdioWriteLowByte					V	
SdioWriteHighByte					V	

[표 2-7] Digital Input/Output 에 관련된 메소드 리스트 및 각 보드별 지원 여부

■ SdioInitComm

메소드 원형 :

Function **SdioInitComm** As Boolean

메소드 설명 :

이 메소드는 COMI-LX402 마스터 보드의 통신 포트를 초기화합니다. 일반적으로 컴퓨터가 부팅되면서 통신 초기화는 자동으로 이루어집니다. 따라서 사용자는 통신초기화를 별도로 하지 않아도 상관은 없으나 프로그램 시작부분에서 통신 초기화를 해주는 것이 좋습니다.

Return

메소드 수행의 성공 여부

Value	Meaning
False	실패
True	성공

예 제

```
Call ComiLxAx1.LoadDevice
Call ComiLxAx1.SdioInitComm
```

、 여기에서 필요한 SDIO 메소드들을 수행한다.

```
Call ComiLxAx1.UnloadDevice
```

■ SdioCheckModule

메소드 원형 :

Function **SdioCheckModule** (ByVal ModuleNo As Long) As Boolean

메소드 설명 :

이 메소드는 지정한 주소값(모듈 번호)을 가지는 COMI-SDM4A 디지털 입출력 터미널 모듈이 COMI-LX402 마스터 보드에 현재 연결되어 있는지를 체크하는 메소드입니다.

매개 변수

▶ **ModuleNo** : 검색하고자 하는 COMI-SDM4A 디지털 입출력 터미널 모듈 번호를 지정합니다. 모듈 번호는 0 ~ 15 까지 지정할 수 있으며 해당 COMI-SDM4A 터미널 모듈에서 점퍼로 설정되는 모듈번호와 일치하여야 합니다.

Return

지정한 COMI-SDM4A 터미널 모듈이 연결되었는지를 나타내는 값

Value	Meaning
False	연결되어 있지 않음
True	연결되어 있음

예 제

다음의 예제는 0 번 모듈부터 15 번 모듈까지 모두 검색하여 각 모듈의 연결상태를 화면에 표시해주는 예제입니다.

```

Call ComiLxAx1.LoadDevice
Call ComiLxAx1.SdioInitComm

For i=0 To 15
    If ( ComiLxAx1.SdioCheckModule(i) ) Then
        Label(i).caption = "연결됨"
    Else
        Label(i).caption = "연결되지 않음"
    End IF
Next i

Call ComiLxAx1.UnloadDevice
    
```

■ SdioSetDioUsage

메소드 원형 :

Function **SdioSetDioUsage** (ByVal ModuleNo As Long, ByVal Usage As Long) As Boolean

메소드 설명 :

이 메소드는 지정한 COM1-SDM4A 터미널 모듈의 Digital Input/Output 단자의 용도를 설정합니다. 각 COM1-SDM4A 터미널 모듈은 16 개의 디지털 입출력 채널을 제공하며 16 채널의 입출력 모드를 4 가지 방법으로 설정할 수 있습니다.

매개 변수

▶ **ModuleNo** : 설정하고자 하는 COM1-SDM4A 디지털 입출력 터미널 모듈 번호를 지정합니다. 모듈 번호는 0 ~ 15 까지 지정할 수 있으며 해당 COM1-SDM4A 터미널 모듈에서 점퍼로 설정되는 모듈번호와 일치하여야 합니다.

▶ **Usage** : DIO 단자의 용도를 설정합니다. 이 값은 다음의 값 중 하나이어야 합니다.

Value	Meaning
0 또는 DI_ONLY	전 채널을 디지털 입력 채널로 사용합니다.
1 또는 DI_D0	CH0~CH7 : 디지털 입력, CH8~CH15 : 디지털 출력
2 또는 D0_DI	CH0~CH7 : 디지털 출력, CH8~CH15 : 디지털 입력
3 또는 D0_ONLY	전 채널을 디지털 출력 채널로 사용합니다.

Return

메소드 수행의 성공 여부

Value	Meaning
False	실패
True	성공

예 제

다음의 예제는 0 번 모듈의 디지털 입출력 모드를 DI_D0 로 설정하고, CH0~CH7 로부터 디지털 입력 상태를 읽어들이 각 채널의 상태를 화면에 표시하고 이를 다시 디지털 출력 (CH8~CH15)으로 내보내는 예이다.

Call ComiLxAx1.**LoadDevice** '그리고, 초기화, 모듈체크가 되어 있다고 가정

Const MOD0 = 0

```
'readBitFromByte () : 바이트 데이터의 특정 비트의 값을 반환하는 메소드
Private Function readBitFromByte(Status, index) As Boolean
    readBitFromByte = (Status And (2 ^ index)) / (2 ^ index)
End Function
```

```
'writeBitToByte () : 바이트 데이터의 특정 비트의 값을 변경하는 메소드
```

```

Private Function writeBitToByte(Status, index, bit) As Byte
    If (bit = 1) Then
        writeBitToByte = Status Or (2 ^ index)
    Else
        writeBitToByte = Status And Not (2 ^ index)
    End If
End Function

Call ComiLxAx1.SdioSetDioUsage(MOD0, DI_DO)

di_byte = ComiLxAx1.SdioReadLowByte(MOD0)

For i=0 to 7
    di_each(i) = readBitFromByte (di_byte, i)
Next i

Call MsgBox("D/I States(CH0~CH7)"& di_each(0)_
    & di_each(1)& di_each(2)& di_each(3) & di_each(4)_
    & di_each(5)& di_each(6)& di_each(7), vbOkOnly )
' 다음의 구문은 읽어들이 CH0~CH7의 D/I 상태를 CH8~CH15를
' 통하여 D/O 출력으로 내보내는 것이다. 사실 do_byte = di_byte
' 를 대입하면 되지만 디지털 출력 각 채널을 개별적으로 설정하는 것
' 을 예로 보이기 위해서 실제로는 불필요한 아래의 for 루프를
' 사용한 것이다.
do_byte = 0
For i=0 to 7
    do_byte = writeBitToByte(do_byte, i, di_each(i))
Next i
Call ComiLxAx1.SdioWriteHighByte( MOD0, do_byte)

Call ComiLxAx1.UnloadDevice

```

 or

■ SdioReadLowByte

메소드 원형 :

Function **SdioReadLowByte** (ByVal ModuleNo As Long) As Integer

메소드 설명 :

이 메소드는 지정한 COM1-SDM4A 터미널 모듈의 0 번 채널부터 7 번 채널까지의 현재 입력 또는 출력 상태를 읽어들이니다. 이 메소드는 CH0 ~ CH7 의 채널 그룹이 디지털 입력용으로 설정되었을때뿐 아니라 디지털 출력용으로 설정된 경우에도 사용할 수 있습니다. 만일 디지털 출력용으로 설정된 경우에 이 메소드를 사용하시면 현재 CH0 ~ CH7 의 출력 상태를 반환받을 수 있습니다.

매개 변수

▶ **ModuleNo** : 설정하고자 하는 COM1-SDM4A 디지털 입출력 터미널 모듈 번호를 지정합니다. 모듈 번호는 0 ~ 15 까지 지정할 수 있으며 해당 COM1-SDM4A 터미널 모듈에서 점퍼로 설정되는 모듈번호와 일치하여야 합니다.

Return

지정한 COM1-SDM4A 터미널 모듈의 0 번 채널부터 7 번 채널까지의 현재 입력 또는 출력 상태. 이 값은 8 비트값으로써 각 비트의 상태가 각 채널의 상태를 나타냅니다.

예 제

다음의 예제는 0 번 모듈의 디지털 입출력 모드를 DI_DO 로 설정하고, CH0~CH7 로부터 디지털 입력 상태를 읽어들이 각 채널의 상태를 화면에 표시하고 이를 다시 디지털 출력 (CH8~CH15)으로 내보내는 예이다.

Call ComiLxAx1.**LoadDevice** '그리고, 초기화, 모듈체크가 되어 있다고 가정

Const MOD0 = 0

```
'readBitFromByte () : 바이트 데이터의 특정 비트의 값을 반환하는 메소드
Private Function readBitFromByte(Status, index) As Boolean
    readBitFromByte = (Status And (2 ^ index)) / (2 ^ index)
End Function
```

```
'writeBitToByte () : 바이트 데이터의 특정 비트의 값을 변경하는 메소드
Private Function writeBitToByte(Status, index, bit) As Byte
    If (bit = 1) Then
        writeBitToByte = Status Or (2 ^ index)
    Else
        writeBitToByte = Status And Not (2 ^ index)
    End If
End Function
```

Call ComiLxAx1.**SdioSetDioUsage**(MOD0, DI_DO)

di_byte = ComiLxAx1.**SdioReadLowByte**(MOD0)

```
For i=0 to 7
    di_each(i) = readBitFromByte (di_byte, i)
```



```

Next i

Call MsgBox("D/I States(CH0~CH7)"& di_each(0)_
    & di_each(1)& di_each(2)& di_each(3) & di_each(4)_
    & di_each(5)& di_each(6)& di_each(7), vbOkOnly )
' 다음의 구문은 읽어들이 CH0~CH7 의 D/I 상태를 CH8~CH15 를
' 통하여 D/O 출력으로 내보내는 것이다. 사실 do_byte = di_byte
' 를 대입하면 되지만 디지털 출력 각 채널을 개별적으로 설정하는 것
' 을 예로 보이기 위해서 실제로는 불필요한 아래의 for 루프를
' 사용한 것이다.
do_byte = 0
For i=0 to 7
    do_byte = writeBitToByte(do_byte, i, di_each(i))
Next i
Call ComiLxAx1.SdioWriteHighByte( MOD0, do_byte)

Call ComiLxAx1.UnloadDevice
    
```

■ SdioReadHighByte

메소드 원형 :

Function **SdioReadHighByte** (ByVal ModuleNo As Long) As Integer

메소드 설명 :

이 메소드는 지정한 COM1-SDM4A 터미널 모듈의 8 번 채널부터 15 번 채널까지의 현재 입력 또는 출력 상태를 읽어들이습니다. 이 메소드는 CH8 ~ CH15 의 채널 그룹이 디지털 입력용으로 설정되었을 때뿐 아니라 디지털 출력용으로 설정된 경우에도 사용할 수 있습니다. 만일 디지털 출력용으로 설정된 경우에 이 메소드를 사용하시면 현재 CH8 ~ CH15 의 출력 상태를 반환받을 수 있습니다.

매개 변수

▶ **ModuleNo** : 설정하고자 하는 COM1-SDM4A 디지털 입출력 터미널 모듈 번호를 지정합니다. 모듈 번호는 0 ~ 15 까지 지정할 수 있으며 해당 COM1-SDM4A 터미널 모듈에서 점퍼로 설정되는 모듈번호와 일치하여야 합니다.

Return

지정한 COM1-SDM4A 터미널 모듈의 8 번 채널부터 15 번 채널까지의 현재 입력 또는 출력 상태. 이 값은 8 비트값으로써 각 비트의 상태가 각 채널의 ON/OFF 상태를 나타냅니다.

예 제

다음의 예제는 0 번 모듈의 디지털 입출력 모드를 DO_DI 로 설정하고, CH8~CH15 로부터 디지털 입력 상태를 읽어들이 각 채널의 상태를 화면에 표시하고 이를 다시 디지털 출력 (CH0~CH7)으로 내보내는 예입니다.

Call ComiLxAx1.**LoadDevice** '그리고, 초기화, 모듈체크가 되어 있다고 가정

Const MOD0 = 0

```
'readBitFromByte () : 바이트 데이터의 특정 비트의 값을 반환하는 메소드
Private Function readBitFromByte(Status, index) As Boolean
    readBitFromByte = (Status And (2 ^ index)) / (2 ^ index)
End Function
```

```
'writeBitToByte () : 바이트 데이터의 특정 비트의 값을 변경하는 메소드
Private Function writeBitToByte(Status, index, bit) As Byte
    If (bit = 1) Then
        writeBitToByte = Status Or (2 ^ index)
    Else
        writeBitToByte = Status And Not (2 ^ index)
    End If
End Function
```

Call ComiLxAx1.**SdioSetDioUsage**(MOD0, DO_DI)

di_byte = ComiLxAx1.**SdioReadHighByte**(MOD0)

```
For i=0 to 7
    di_each(i) = readBitFromByte (di_byte, i)
```

```

Next i

Call MsgBox("D/I States(CH8~CH15)"& di_each(0)_
    & di_each(1)& di_each(2)& di_each(3) & di_each(4)_
    & di_each(5)& di_each(6)& di_each(7), vbOkOnly )
' 다음의 구문은 읽어들이 CH~CH15 의 D/I 상태를 CH0~CH7 를
' 통하여 D/O 출력으로 내보내는 것이다. 사실 do_byte = di_byte
' 를 대입하면 되지만 디지털 출력 각 채널을 개별적으로 설정하는 것
' 을 예로 보이기 위해서 실제로는 불필요한 아래의 for 루프를
' 사용한 것이다.
do_byte = 0
For i=0 to 7
    do_byte = writeBitToByte(do_byte, i, di_each(i))
Next i
Call ComiLxAx1.SdioWriteLowByte( MOD0, do_byte)

Call ComiLxAx1.UnloadDevice
    
```

■ SdioWriteLowByte

메소드 원형 :

Function **SdioWriteLowByte** (ByVal ModuleNo As Long, ByVal LowByte As Integer) As Boolean

메소드 설명 :

이 메소드는 지정한 COM1-SDM4A 터미널 모듈의 CH0 ~ CH7 에 디지털 출력을 내보냅니다. 이 메소드를 사용하기 전에 Call ComiLxAx1.SdioSetDioUsage()메소드를 사용하여 CH0 ~ CH7 을 디지털 출력 채널로 설정하여야 합니다.

매개 변수

- ▶ **ModuleNo** : 설정하고자 하는 COM1-SDM4A 디지털 입출력 터미널 모듈 번호를 지정합니다. 모듈 번호는 0 ~ 15 까지 지정할 수 있으며 해당 COM1-SDM4A 터미널 모듈에서 점퍼로 설정되는 모듈번호와 일치하여야 합니다.
- ▶ **Value** : CH0 ~ CH7 의 디지털 출력 상태를 지정합니다. 이 값은 8 비트 값으로써 각 비트의 상태가 각채널의 ON/OFF 상태를 의미합니다.

Return

메소드 수행의 성공 여부

Value	Meaning
False	실패
True	성공

참고

특정 채널의 상태만 변경하고자 하는 경우 Call ComiLxAx1.SdioReadLowByte() 메소드를 통하여 CH0 ~ CH7 의 현재 출력 상태를 읽어들이고 후 원하는 채널에 해당하는 비트만 변경하여 출력하면 됩니다. CH0 ~ CH7 의 채널 중에 특정 채널만 출력을 변경하는 메소드를 다음과 같이 구성할 수 있습니다.

```
Private Sub writeLow(moduleNo, ch, bit)
    state = ComiLxAx1.SdioReadLowByte(moduleNo)

    If (bit = 1) Then
        State = state Or (2 ^ ch)
    Else
        State = state And Not (2 ^ ch)
    End If

    Call ComiLxAx1.SdioWriteLowByte( moduleNo, state)
End Sub
```

예 제

다음의 예제는 0 번 모듈의 디지털 입출력 모드를 DO_DI 로 설정하고, CH8~CH15 로부터 디지털 입력 상태를 읽어들이 각 채널의 상태를 화면에 표시하고 이를 다시 디지털 출력 (CH0~CH7)으로 내보내는 예입니다.

```
Call ComiLxAx1.LoadDevice '그리고, 초기화, 모듈체크가 되어 있다고 가정

Const MOD0 = 0

'readBitFromByte () : 바이트 데이터의 특정 비트의 값을 반환하는 메소드
Private Function readBitFromByte(Status, index) As Boolean
    readBitFromByte = (Status And (2 ^ index)) / (2 ^ index)
End Function

'writeBitToByte () : 바이트 데이터의 특정 비트의 값을 변경하는 메소드
Private Function writeBitToByte(Status, index, bit) As Byte
    If (bit = 1) Then
        writeBitToByte = Status Or (2 ^ index)
    Else
        writeBitToByte = Status And Not (2 ^ index)
    End If
End Function

Call ComiLxAx1.SdioSetDioUsage(MOD0, DO_DI)

di_byte = ComiLxAx1.SdioReadHighByte(MOD0)

For i=0 to 7
    di_each(i) = readBitFromByte (di_byte, i)
Next i

Call MsgBox("D/I States(CH8~CH15)"& di_each(0)_
    & di_each(1)& di_each(2)& di_each(3) & di_each(4)_
    & di_each(5)& di_each(6)& di_each(7), vbOkOnly )
' 다음의 구문은 읽어들이 CH8~CH15 의 D/I 상태를 CH0~CH7 를
' 통하여 D/O 출력으로 내보내는 것이다. 사실 do_byte = di_byte
' 를 대입하면 되지만 디지털 출력 각 채널을 개별적으로 설정하는 것
' 을 예로 보이기 위해서 실제로는 불필요한 아래의 for 루프를
' 사용한 것이다.
do_byte = 0
For i=0 to 7
    do_byte = writeBitToByte(do_byte, i, di_each(i))
Next i
Call ComiLxAx1.SdioWriteLowByte( MOD0, do_byte)

Call ComiLxAx1.UnloadDevice
```

■ SdioWriteHighByte

메소드 원형 :

Function **SdioWriteHighByte** (ByVal ModuleNo As Long, ByVal HighByte As Integer) As Boolean

메소드 설명 :

이 메소드는 지정한 COMI-SDM4A 터미널 모듈의 CH8 ~ CH15 에 디지털 출력을 내보냅니다.
이 메소드를 사용하기 전에 Call ComiLxAx1.SdioSetDioUsage()메소드를 사용하여 CH8 ~ CH15 를 디지털 출력 채널로 설정하여야 합니다.

매개 변수

- ▶ **ModuleNo** : 설정하고자 하는 COMI-SDM4A 디지털 입출력 터미널 모듈 번호를 지정합니다. 모듈 번호는 0 ~ 15 까지 지정할 수 있으며 해당 COMI-SDM4A 터미널 모듈에서 점퍼로 설정되는 모듈번호와 일치하여야 합니다.
- ▶ **Value** : CH8 ~ CH15 의 디지털 출력 상태를 지정합니다. 이 값은 8 비트 값으로써 각 비트의 상태가 각채널의 ON/OFF 상태를 의미합니다.

Return

메소드 수행의 성공 여부

Value	Meaning
0	실패
1	성공

참고

특정 채널의 상태만 변경하고자 하는 경우 Call ComiLxAx1.SdioReadHighByte() 메소드를 통하여 CH8 ~ CH15 의 현재 출력 상태를 읽어들이고 후 원하는 채널에 해당하는 비트만 변경하여 출력하면 됩니다. CH8 ~ CH15 의 채널 중에 특정 채널만 출력을 변경하는 메소드를 다음과 같이 구성할 수 있습니다.

```
Private Sub writeHigh(moduleNo, ch, bit)
    state = ComiLxAx1.SdioReadHighByte(moduleNo)

    If (bit = 1) Then
        state = state Or (2 ^ (ch-8))
    Else
        state = state And Not (2 ^ (ch-8))
    End If
    Call ComiLxAx1.SdioWriteHighByte( moduleNo, state)
End Sub
```

예 제

다음의 예제는 0 번 모듈의 디지털 입출력 모드를 DI_DO 로 설정하고, CH0~CH7 로부터 디지털 입력 상태를 읽어들이 각 채널의 상태를 화면에 표시하고 이를 다시 디지털 출력 (CH8~CH15)으로 내보내는 예이다.

```
Call ComiLxAx1.LoadDevice '그리고, 초기화, 모듈체크가 되어 있다고 가정

Const MOD0 = 0

'readBitFromByte () : 바이트 데이터의 특정 비트의 값을 반환하는 메소드
Private Function readBitFromByte(Status, index) As Boolean
    readBitFromByte = (Status And (2 ^ index)) / (2 ^ index)
End Function

'writeBitToByte () : 바이트 데이터의 특정 비트의 값을 변경하는 메소드
Private Function writeBitToByte(Status, index, bit) As Byte
    If (bit = 1) Then
        writeBitToByte = Status Or (2 ^ index)
    Else
        writeBitToByte = Status And Not (2 ^ index)
    End If
End Function

Call ComiLxAx1.SdioSetDioUsage(MOD0, DI_DO)

di_byte = ComiLxAx1.SdioReadLowByte(MOD0)

For i=0 to 7
    di_each(i) = readBitFromByte (di_byte, i)
Next i

Call MsgBox("D/I States(CH0~CH7)"& di_each(0)_
    & di_each(1)& di_each(2)& di_each(3) & di_each(4)_
    & di_each(5)& di_each(6)& di_each(7), vbOkOnly )
' 다음의 구문은 읽어들이 CH0~CH7 의 D/I 상태를 CH8~CH15 를
' 통하여 D/O 출력으로 내보내는 것이다. 사실 do_byte = di_byte
' 를 대입하면 되지만 디지털 출력 각 채널을 개별적으로 설정하는 것
' 을 예로 보이기 위해서 실제로는 불필요한 아래의 for 루프를
' 사용한 것이다.
do_byte = 0
For i=0 to 7
    do_byte = writeBitToByte(do_byte, i, di_each(i))
Next i
Call ComiLxAx1.SdioWriteHighByte( MOD0, do_byte)

Call ComiLxAx1.UnloadDevice
```

2.5 카운터

이 단원에서는 카운터와 관련된 메소드를 소개합니다. 카운터는 펄스신호를 카운트하는데 사용되는 메소드입니다. 카운터에 관련된 메소드는 다음과 같습니다.

메소드명	각 보드별 적용 여부								
	LX101	LX102	LX103	LX201	LX202	LX203	LX301	LX401	LX50x
ReadCounter32	V	V	V				V	V	
ClearCounter32	V	V	V				V	V	

[표 2-8] Digital Input/Output 에 관련된 메소드 리스트 및 각 보드별 지원 여부

■ ReadCounter32

메소드 원형

Function **ReadCounter32** (ByVal ch As Long) As Long

메소드 설명

이 메소드는 지정한 카운터 채널의 카운트 값을 읽어옵니다.

매개 변수

▶ *ch* : Counter 채널번호. 채널 번호는 0 부터 시작합니다.

Return 값

32 비트 카운트 값

or

■ ClearCounter32

메소드 원형

Function **ClearCounter32** (ByVal ch As Long) As Long

메소드 설명

이 메소드는 지정한 카운터 채널의 카운트 값을 0으로 리셋(reset)하여 줍니다.

매개 변수

▶ *ch* : Counter 채널번호. 채널 번호는 0부터 시작합니다.

2.6 모션 제어(Motion Control)

이 단원에서 설명하는 모션제어 기능은 ㈜커미조아에서 개발한 COMI-LX50x 모션제어(Motion Control) 전용 보드에서 제공하는 기능입니다. 따라서 이 단원에서 소개되는 모든 메소드는 COMI-LX50x 보드에만 적용가능합니다.

COMI-LX50x 모션제어 보드는 펄스 구동 방식에 의하여 스텝모터나 서보 모터의 정밀 위치제어 및 속도제어를 하기 위한 기능을 제공합니다. 스텝모터나 서보 모터의 위치 제어는 디지털 출력등의 일반적인 방법으로 펄스를 만들어서 제어할 수도 있습니다. 그러나 이러한 방식으로는 정확한 속도 제어, 가/감속 제어, 두 축 이상의 보간(Interpolation)등은 거의 불가능합니다. 모션제어 보드는 단일축의 위치제어는 물론이고 정밀 속도 제어, 가/감속 제어, 두 축 이상의 보간 등을 모두 자동화하여 사용자들이 이러한 기능을 아주 쉽게 구현할 수 있도록 해줍니다.

2.6.1 모션 초기화 및 환경설정 메소드

이 단원에서는 모션을 초기화하고 모션을 수행하기 이전에 환경을 설정하는 메소드들을 소개합니다. 이와 관련된 메소드들은 다음과 같습니다.

메소드 / 설명	페이지
Sub McReset 모션 제어 보드의 하드웨어와 소프트웨어적인 모든 상태를 리셋합니다.	86
sub McServoOn (ByVal Channel As Long, ByVal Enable As Short) 지정한 채널(축)의 SERVO-ON 신호를 제어합니다.	87
function McGetServoOn (ByVal Channel As Long) As Long 지정한 채널(축)의 SERVO-ON 신호의 상태를 반환합니다.	88
Sub McSetBlockingMode (ByVal Blocking As Integer) 모션이 완료될때까지 루프를 돌 때 윈도우 또는 시스템 이벤트를 처리할 수 있도록할지를 결정하는 메소드	89
Sub McSetOutputMode (ByVal Channel As Long, ByVal OutputMode As Long) 현재 설정된 Command 펄스의 출력 모드를 얻어옵니다.	90
Function McGetOutputMode (ByVal Channel As Long) As Long 현재 설정된 Command 펄스의 출력 모드를 반환합니다.	91
Sub McSetInputMode (ByVal Channel As Long, ByVal InputMode As Long, ByVal PulseLogic As Long) Feedback 펄스의 입력 모드를 설정합니다.	92
Sub McGetInputMode (ByVal Channel As Long, ByRef InputMode As Long, ByVal PulseLogic As Long) 현재 설정된 Feedback 펄스의 입력 모드를 얻어옵니다.	93
Sub McSetSpeedRange(ByVal Channel, ByVal MaxSpeed) 모션에 적용할 수 있는 최저/최고 속도를 제한합니다.	94
Sub McSetUnitDistance (ByVal Channel As Long, ByVal UnitDist As Double) 논리적 단위 거리에 대한 펄스 수를 설정합니다.	96
Function McGetUnitDistance (ByVal Channel As Long) As Double 현재 설정된 논리적 단위 거리에 대한 펄스 수를 반환합니다.	98
Sub McSetUnitSpeed (ByVal Channel As Long, ByVal UnitSpeed As Long) 논리적 단위 속도에 대한 펄스 출력 속도(PPS)를 설정합니다.	99
Function McGetUnitSpeed (ByVal Channel) As Double 현재 설정된 논리적 단위 속도에 대한 실제 펄스 출력 속도(PPS)를 반환합니다.	101
sub McSetInOutRatio (ByVal Channel As Long, ByVal Ratio As Double)	102

Feedback 펄스와 Command 펄스의 분해능 비율(Resolution ratio)을 설정합니다.

or

■ McReset

메소드 원형

Sub **McReset**

메소드 설명

이 메소드는 모션 제어 보드의 하드웨어와 소프트웨어적인 모든 상태를 리셋합니다.
프로그램 초기와 종료 부분에서는 이 메소드를 사용하여 모션을 리셋시켜주는 것이
좋습니다.

예 제

```
Call ComiLxAx1.LoadDivice  
  
Call ComiLxAx1.McReset  
  
Call ComiLxAx1.UnloadDevice
```

■ McServoOn

함수 원형

sub **McServoOn** (ByVal Channel As Long, ByVal Enable As Short)

함수 설명

지정한 채널(축)의 SERVO-ON 신호를 제어합니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호(0 부터 시작합니다).
- ▶ **Enable** : SERVO-ON 제어값. 0=>OFF, 1=>ON

참 고

서보 드라이버를 사용하실 때는 외부에서 스위치를 이용하여 서보드라이버의 ON/OFF 를 제어할 수 있도록 하는데, 이를 SERVO-ON 신호라 합니다. 따라서 서보드라이버를 사용하여 모터를 구동하는 경우에는 반드시 이 함수를 이용하여 SERVO-ON 신호를 Enable 시켜야 합니다.

단, 이 함수는 COMI-LX501 보드에서는 지원하지 않습니다. COMI-LX501 에서는 디지털 출력 채널을 이용하여 SERVO-ON 제어를 하셔야합니다.

■ McGetServoOn

함수 원형

```
function McGetServoOn (ByVal Channel As Long) As Long
```

함수 설명

지정한 채널(축)의 SERVO-ON 신호의 상태를 반환합니다.

매개 변수

▶ *Channel* : 채널(축) 번호(0 부터 시작합니다).

반환값

현재 출력되고 있는 SERVO-ON 신호의 상태를 반환합니다.

■ McSetBlockingMode

메소드 원형

Sub **McSetBlockingMode** (ByVal Blocking As Integer)

메소드 설명

이 메소드는 Blocking 모드를 결정합니다. McMove()와 같은 메소드들은 Motion 이 완료될 때까지 내부적으로 루프(Loop)를 돌면서 메소드에서 Return 되지 않습니다. Blocking 모드를 FALSE 로 하면 이러한 경우에도 키보드, 마우스 이벤트 등과 같은 윈도우 이벤트나 메시지를 처리할 수 있습니다.

매개 변수

▶ **Blocking** : Blocking 모드를 결정합니다.

참고

다음과 같은 메소드들은 Motion 이 완료될 때까지 메소드에서 Return 되지 않습니다.

McMove, McMoveTo, McLine, McLineTo, McArc, McArcTo

예 제

다음의 예제는 소스의 간결성을 위하여 McSetBlockingMode() 메소드의 사용법만 예로 들기 위해서 만들어진 것입니다. 실제로는 McSetBlockingMode()는 프로그램이 쓰레드 기반일 때 그 효과를 발휘할 수 있습니다.

```
Call ComiLxAx1.LoadDevice

Call ComiLxAx1.McSetBlockingMode ( 0 )
`Set constant speed mode
Call ComiLxAx1.McSetSpeedMode( 0, 0)
`Set speed as 5000 PPS
Call ComiLxAx1.McSetSpeed(0, 0, 1000)
` Blocking 이 않도록 설정되었으므로 Move() 메소드가 내부적으로
` Loop 를 돌면서 모션이 완료되기를 기다릴 때에도 키보드나 마우스
` 등의 시스템 및 윈도우 이벤트를 처리할 수 있다.
Call ComiLxAx1.McMove(0, 5000)

Call ComiLxAx1.UnloadDevice
```

■ McSetOutputMode

메소드 원형

Sub **McSetOutputMode** (ByVal Channel As Long, ByVal OutputMode As Long)

메소드 설명

Command 펄스의 출력 모드를 설정합니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **OutputMode** : Command 펄스의 출력 모드를 설정합니다. 출력 모드는 다음과 같이 6 가지로 설정할 수 있습니다.

Value	출력 형태			
	(+) 방향 운전 시		(-) 방향 운전 시	
	CW pin	CCW pin	CW pin	CCW pin
0		(High)		(Low)
1		(High)		(Low)
2		(Low)		(High)
3		(Low)		(High)
4		(High)	(High)	
5		(Low)	(Low)	

■ McGetOutputMode

메소드 원형

Function **McGetOutputMode**(ByVal Channel As Long) As Long

메소드 설명

현재 설정된 Command 펄스의 출력 모드를 반환합니다.

매개 변수

▶ *Channel* : 채널(축) 번호, 0 ~ 3

Return 값

현재 설정된 Command 펄스의 출력 모드를 반환합니다. 반환되는 값은 0~5 의 정수이며 각 값의 의미는 McSetOutputMode() 메소드를 참조하십시오.

■ McSetInputMode

메소드 원형

```
Sub McSetInputMode(ByVal Channel As Long, ByVal InputMode As Long, ByVal PulseLogic As Long)
```

메소드 설명

Feedback 펄스의 입력 모드를 설정합니다. 사용자는 4 가지 형태의 Feedback 펄스의 입력모드를 설정할 수 있습니다. 또한 이 메소드는 입력 펄스의 입력 로직(Logic)을 설정합니다.

매개 변수

▶ **Channel** : 채널(축) 번호, 0 ~ 3

▶ **InputMode** : Feedback 펄스의 입력 모드를 설정합니다. 입력 모드는 다음과 같이 4 가지로 설정할 수 있습니다.

Value	Meaning
0	1X A/B (1 채널 엔코더 입력 모드)
1	2X A/B (2 채널 엔코더 입력 모드)
2	4X A/B (4 채널 엔코더 입력 모드)
3	CW/CCW (A 펄스 - 카운트 증가, B 펄스 - 카운트 감소)

▶ **PulseLogic** : 입력 펄스의 로직(Logic)을 설정합니다.

Value	Meaning
0	Normal low
1	Normal high

■ McGetInputMode

메소드 원형

```
Sub McGetInputMode(ByVal Channel As Long, ByRef InputMode As Long, ByRef PulseLogic As Long)
```

메소드 설명

현재 설정된 Feedback 펄스의 입력 모드를 반환합니다.

매개 변수

- ▶ *Channel* : 채널(축) 번호, 0 ~ 3
- ▶ *InputMode* : Feedback 펄스의 입력 모드를 반환 받을 변수(ByRef). 반환되는 값은 0~3 의 정수이며 각 값의 의미는 McSetInputMode() 메소드를 참조하십시오. 이 값이 NULL 이면 입력 모드를 반환하지 않습니다.
- ▶ *PulseLogic* : Feedback 펄스의 입력 로직(Logic)을 반환 받을 변수(ByRef). 반환되는 값은 0~1 의 정수이며 각 값의 의미는 McSetInputMode() 메소드를 참조하십시오. 이 값이 NULL 이면 입력 로직을 반환하지 않습니다.

■ McSetSpeedRange

메소드 원형

Sub **McSetSpeedRange**(ByVal Channel As Long, ByVal MaxSpeed As Long)

메소드 설명

모션에 적용할 수 있는 최저/최고 속도를 제한합니다. 이 메소드는 실제적으로는 출력 펄스의 주파수 범위를 설정하는 역할을 합니다. 출력 펄스의 주파수는 최대 6.5MHz 까지 설정가능하며 기본적으로 설정되는 주파수 범위는 10Hz ~ 655,350Hz 입니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **MaxSpeed** : 모션의 최고 속도를 설정합니다. 이 값에 따라 최저 속도는 자동으로 설정됩니다. 이 값의 단위는 McSetUnitSpeed()메소드에 의하여 설정된 단위가 됩니다. 만일 McSetUnitSpeed()메소드를 사용하지 않았다면 PPS 단위가 됩니다. MaxSpeed 값에 따라 설정되는 출력 펄스의 주파수 범위를 몇 가지 예로 들면 아래의 표와 같습니다.

MaxSpeed 값(Hz)	출력 펄스의 주파수 범위(Hz)
65,535	0.1 to 6,553.5
13,107	0.2 to 13,107
32,767.5	0.5 to 32,767.5
65,535	1 to 65,535
131,070	2 to 131070
327,650	5 to 327,650
655,350	10 to 655,350
1,310,700	20 to 1,310,700
3,276,750	50 to 3,276,750
6,553,500	100 to 6,553,500

참 고

- ▶ 최저 속도는 최대 속도 설정에 따라 다음과 같은 식에 의하여 자동으로 결정됩니다.

$$V_{\min} = \frac{V_{\max}}{65535}$$

예를 들어 V_{\max} (MaxSpeed) 값을 1000000(Hz)로 지정하였다면 V_{\min} 값은 $1000000/65535 = 15.26(\text{Hz})$ 으로 자동 설정됩니다. 따라서 사용자는 15.26 ~ 1000000 (Hz)의 범위에서

속도를 설정할 수 있습니다. 만일 McSetSpeed()등의 메소드를 이용하여 속도를 설정할 때
최저 속도보다 작은 값으로 설정하면 자동으로 최저 속도로 조정되어 적용됩니다.

■ McSetUnitDistance

메소드 원형

Sub **McSetUnitDistance**(ByVal Channel As Long, ByVal UnitDist As Double)

메소드 설명

논리적 단위 거리에 대한 펄스 수를 설정합니다. 여기서 논리적 단위 거리라 함은 Move 메소드에서 사용하는 거리 또는 위치에 대한 단위량을 의미합니다. 이 메소드를 사용하여 특별히 지정하지 않는 경우에는 논리적 단위 거리에 대한 펄스 수는 1 로 사용됩니다.

매개 변수

- ▶ *Channel* : 채널(축) 번호, 0 ~ 3
- ▶ *UnitDist* : 논리적 단위 거리에 대한 펄스 수를 지정합니다.

참고

모터나 모터 드라이버의 특성에 따라 단위 거리 이동에 필요한 펄스의 수는 다르게 됩니다. 또는 사용자의 특성에 따라 이동량에 대한 단위가 다를 수 있습니다. 즉, 어떤 사용자는 이동량의 단위를 각도로 표현하는 것이 용이할 수 있고 어떤 사용자는 mm 또는 cm 등으로 표현하는 것이 용이할 수 있습니다. **McSetUnitDistance** 메소드는 사용자가 이동량의 단위를 결정하도록 하는 메소드입니다. 이 메소드를 다음의 예를 참고하여 사용하십시오.

Ex 1) 1 회전에 필요한 펄스 수가 3600 펄스인 경우에 이동량의 단위를 1° 로 하고자 한다면 fUnitDist 값을 10 으로 하면 됩니다.

Ex 2) 1mm 이송에 펄스 수가 20 펄스인 경우에 이동량의 단위를 1mm 로 하고자 한다면 fUnitDist 값을 20 으로 하면 됩니다.

예제

1mm 이동하는데 필요한 펄스수가 100 펄스이고 1 회전에 필요한 펄스수가 10000 펄스일때 거리의 단위를 mm 로, 속도의 단위를 rpm 으로 설정하는 예제입니다.

Call ComiLxAx1.**LoadDevice**

\ Set 100 pulses for unit distance
 \ 이 예제에서는 1mm 이동에 필요한 펄스수를 100 펄스로
 \ 가정하고 단위 거리를 1mm로 설정한 것이다.

Call ComiLxAx1.**McSetUnitDistance**(0, 100)

\ Set 10000/60(=166.7) PPS for unit speed
 \ 이 예제에서는 1 회전에 필요한 펄스수를 10000
 \ 펄스로 가정하고 단위 속도를 1rpm로 설정한 것이다.

Call ComiLxAx1.**McSetUnitSpeed**(0, 10000./60)


```
Call ComiLxAx1.McSetSpeed(0, 0, 60) ` Set speed as 60 rpm  
Call ComiLxAx1.McSetSpeedMode(0, 0) ` Set contant speed mode  
  
Call ComiLxAx1.McMove(0, 100) `60 rpm 의 속도로 100mm 이동  
  
Call ComiLxAx1.UnloadDevice
```

■ McGetUnitDistance

메소드 원형

Function **McGetUnitDistance**(ByVal Channel As Long) As Double

메소드 설명

현재 설정된 논리적 단위 거리에 대한 펄스 수를 반환합니다. 여기서 논리적 단위 거리라는 것은 Move 메소드에서 사용하는 거리 또는 위치에 대한 단위량을 의미합니다.

매개 변수

▶ *Channel* : 채널(축) 번호, 0 ~ 3

Return 값

현재 설정된 단위 거리에 대한 펄스 수를 반환합니다.

■ McSetUnitSpeed

메소드 원형

Sub **McSetUnitSpeed**(ByVal Channel As Long, ByVal UnitSpeed As Double)

메소드 설명

논리적 단위 속도에 대한 실제 펄스 출력 속도(PPS)를 설정합니다. 여기서 논리적 단위 속도라 함은 속도 지정메소드에서 사용하는 속도 또는 가속도에 대한 단위량을 의미합니다. 이 메소드를 사용하여 특별히 지정하지 않는 경우에는 단위 속도에 대한 펄스 출력 속도는 1PPS 로 사용됩니다. 이 것은 McSetSpeed, McSetAccel, McSetScurve, McSetSpeedMx, McSetAccelMx, McSetScurveMx 등의 메소드에 영향을 미칩니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **UnitSpeed** : 단위 속도에 대한 펄스 출력 속도(PPS)를 지정합니다.

참고

사용자의 특성에 따라 속도에 대한 단위가 다를 수 있습니다. 즉, 어떤 사용자는 속도 단위를 RPM 으로 표현하는 것이 용이할 수 있고 어떤 사용자는 m/sec 로 표현하는 것이 용이할 수 있습니다. **McSetUnitSpeed** 메소드는 사용자가 속도의 단위를 결정하도록 하는 메소드입니다. 이 메소드를 다음의 예를 참고하여 사용하십시오.

Ex 1) 1 회전에 필요한 펄스 수가 3600 펄스인 경우에 속도의 단위를 RPM 으로 하고자 한다면 fUnitDist 값을 3600/60, 즉 60 PPS 로 설정합니다(여기서 60 으로 나누는 것은 RPM 은 분당 회전수이므로 초당 3600/60 펄스를 출력해야 1 분에 3600 펄스가 나가기 때문입니다).

Ex 2) 1cm 이송에 필요한 펄스 수가 1000 펄스인 경우에 이동량의 단위를 cm/sec 로 하고자 한다면 fUnitDist 값을 1000 PPS 로 설정합니다.

관련 메소드

McSetSpeed, McSetAccel, McSetScurve, McSetSpeedMx, McSetAccelMx, McSetScurveMx ,
McGetActualSpeed, McGetCurSpeed

예제

■ 예제 1

1 회전에 필요한 펄스수가 3600 펄스일 때 거리의 단위를 각도(1°)로, 속도의 단위를 rpm 으로 설정하는 예제입니다.

ur

```
Call ComiLxAx1.LoadDivice

` Set 10 pulses for unit distance
` 이 예제에서는 1 회전에 필요한 펄스 수를 3600 펄스로
` 가정하고 단위 거리를 1°로 설정한 것이다.
Call ComiLxAx1.McSetUnitDistance(0, 10)
` Set 3600/60(=60) PPS for unit speed
` 이 예제에서는 1 회전에 필요한 펄스수를 3600 펄스로
` 가정하고 단위 속도를 1rpm으로 설정한 것이다.
Call ComiLxAx1.McSetUnitSpeed(0, 3600./60)
` Set trapezoidal speed mode
Call ComiLxAx1.McSetSpeedMode(0, 1)
` Set speed as 100 rpm
Call ComiLxAx1.McSetSpeed(0, 0, 100)
` 가속도와 감속도를 각각 200rpm/s로 설정한다. 이렇게 하면 작업속도가
` 100rpm이므로 가속 및 감속 시간은 각각 0.5 초 걸린다.
Call ComiLxAx1.McSetAccel(0, 200, 200)
` 모터를 720° 회전한다. 실제로는 720*10 펄스가 출력된다.
Call ComiLxAx1.McMove(0, 720)

Call ComiLxAx1.UnloadDevice
```

■ 예제 2

1cm 이동하는데 필요한 펄스수가 1000 펄스일 때 거리의 단위를 cm로, 속도의 단위를 cm/sec로 설정하는 예제입니다.

```
Call ComiLxAx1.LoadDivice

` Set 1000 pulses for unit distance
` 이 예제에서는 1cm 이동에 필요한 펄스수를 1000 펄스로
` 가정하고 단위 거리를 1cm로 설정한 것이다.
Call ComiLxAx1.McSetUnitDistance(0, 1000)
` Set 1000 PPS for unit speed
` 이 예제에서는 1cm 이동에 필요한 펄스수를 1000 펄스로
` 펄스로 가정하고 단위 속도를 1rpm로 설정한 것이다.
Call ComiLxAx1.McSetUnitSpeed(0, 1000)

Call ComiLxAx1.McSetSpeed(0, 0, 50) ` Set speed as 50 cm/sec/
Call ComiLxAx1.McSetSpeedMode(0, 0) ` Set constant speed mode

Call ComiLxAx1.McMove(0, 10) ` 50 cm/sec의 속도로 10cm 이동 . 실제로는
10*1000=10000 펄스가 출력된다.

Call ComiLxAx1.UnloadDevice
```

■ McGetUnitSpeed

메소드 원형

Function **McGetUnitSpeed** (ByVal Channel As Long) As Double

메소드 설명

현재 설정된 논리적 단위 속도에 대한 실제 펄스 출력 속도(PPS)를 반환합니다. 여기서 논리적 단위 속도라 함은 속도 지정메소드에서 사용하는 속도 또는 가속도에 대한 단위량을 의미합니다.

매개 변수

▶ *Channel* : 채널(축) 번호, 0 ~ 3

Return 값

현재 설정된 단위 거리에 대한 펄스 수를 반환합니다.

■ McSetInOutRatio

함수 원형

```
sub McSetInOutRatio (ByVal Channel As Long, ByVal Ratio As Double)
```

함수 설명

Feedback 펄스와 Command 펄스의 분해능 비율(Resolution ratio)을 설정합니다. 여기서 Feedback 펄스의 분해능이란 엔코더의 1 회전시에 발생하는 펄스 수를 의미합니다. 그리고 Command 펄스의 분해능이란 모터를 1 회전시키기 위해 필요한 Command 펄스수를 의미합니다.

매개 변수

▶ **Channel** : 채널(축) 번호, 0 ~ 3

▶ **Ratio** : Feedback 펄스와 Command 펄스의 분해능 비를 지정합니다. 이 값은

$\text{Ratio} = (\text{Feedback 펄스 분해능}) / (\text{Command 펄스의 분해능})$ 으로 표현됩니다.

참 고

□ In/Out Ratio 는 Actual(Feedback) position 또는 Actual speed 를 논리 단위로 읽을 때 적용됩니다. 논리적 단위 거리나 단위 속도는 Command 펄스기준으로 설정되므로 Command 펄스와 Feedback 펄스의 분해능이 서로 다르다면 Actual position 이나 Actual speed 의 논리 값 계산이 잘못되게 됩니다.

□ In/Out Ratio 는 **McGetPositionA** 함수와 **McGetActualSpeed** 함수에만 영향을 미칩니다. In/Out Ratio 가 적용되는 로직은 이 두 함수의 설명을 참조하십시오.

2.6.2 Single Axis 모션 제어 메소드

이 단원에서는 Single Axis 모션 제어에 관련된 메소드들을 소개합니다. Single Axis 모션은 한 축만을 독립적으로 제어하는 작업을 의미합니다. Single Axis 모션은 먼저 속도설정 메소드들을 이용하여 속도를 설정하고 이동 메소드를 사용하여 이동 작업을 수행합니다. 그리고 필요에 따라 정지 메소드를 사용하여 모션을 정지합니다.

메소드 / 설명	페이지
Sub McSetSpeedMode(ByVal Channel As Long, ByVal ModeIndex As Long) Motion의 속도 모드를 설정합니다.	105
Sub McSetSpeed (ByVal Channel, ByVal IniSpeed As Double, ByVal WorkSpeed As Double) Motion의 속도를 설정합니다.	109
Sub McSetAccel(ByVal Channel, ByVal Accel As Double, ByVal Decel As Double) Motion의 가/감속도를 설정합니다.	112
Sub McSetScurve (ByVal Channel, ByVal Svacc As Double, ByVal Svdec Double) 속도모드를 S-curve 속도 패턴으로 설정한 경우에 S-curve Section의 범위를 속도단위로 설정합니다.	115
Sub McStartVMove (ByVal Channel As Long, ByVal Direction As Long) 작업속도까지 가속한 후에 작업속도를 유지하며 정지메소드가 호출될 때까지 지정한 방향으로의 모션을 계속 수행합니다.	118
Sub McStartMove (ByVal Channel As Long, ByVal Distance As Long) 현재의 위치에서 지정한 거리만큼 이동을 수행합니다. 이 메소드는 모션을 시작시킨 후 바로 Return 합니다.	119
Sub McMove (ByVal Channel As Long, ByVal Distance As Double) 현재의 위치에서 지정한 거리만큼 이동을 수행합니다. 이 메소드는 모션이 완료될때까지 Return 되지 않습니다.	121
Sub McStartMoveTo (ByVal Channel As Long, ByVal Position As Double) 지정한 절대좌표로의 이동을 수행합니다. 이 메소드는 모션(Motion)을 시작 시킨 후에 바로 Return 합니다.	123
Sub McMoveTo (ByVal Channel As Long, ByVal Position As Double) 지정한 절대좌표로의 이동을 수행합니다. 이 메소드는 모션이 완료될때까지 Return 되지 않습니다.	125
Sub McStop (ByVal Channel As Long) 지정한 축에 대한 모션을 감속 후 정지합니다.	127
Sub McEngStop (ByVal Channel As Long)	128



UI

지정한 축에 대한 모션을 감속없이 즉시 정지합니다.	
Function McDone (ByVal Channel As Long) As Boolean 하나의 축에 대하여 모션이 완료됐는지를 체크합니다.	129

■ McSetSpeedMode

메소드 원형

Sub **McSetSpeedMode**(ByVal Channel As Long, ByVal ModeIndex As Long)

메소드 설명

Motion 의 속도 모드를 설정합니다. 단, 이 메소드는 Motion 에 바로 영향을 주는 것이 아니고 Move, MoveTo 등의 이송 메소드가 수행될 때 설정된 내용이 적용됩니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **ModeIndex** : 속도 모드를 지정합니다. 속도 모드는 다음과 같이 3 가지로 설정할 수 있습니다.

Value	Meaning
0	Constant speed mode
1	Trapezoidal speed mode
2	S-curve speed mode

관련 메소드

McSetSpeed, McSetAccel, McSetScurve

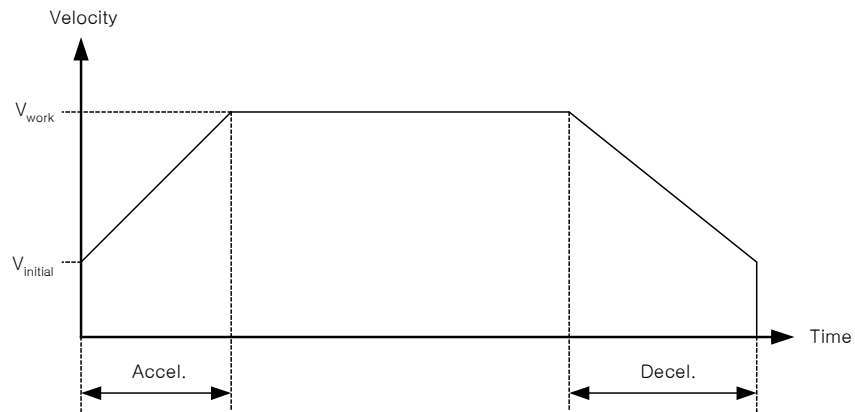
참 고

☐ Constant speed mode

Constant speed mode 에서는 Motion 을 수행할 때 가속/감속을 적용하지 않고 일정속도로 Motion 을 수행합니다. 여기서 적용되는 일정 속도는 **McSetSpeed** 메소드에서 주어지는 EndSpeed 에서 주어진 값이 적용됩니다.

☐ Trapezoidal speed mode

Trapezoidal speed mode 에서는 Motion 을 수행하는데 있어서 속도의 패턴을 [그림 #]과 같이 Linear acceleration -> Working speed(constant) -> Linear deceleration 의 형태로 운용하는 모드입니다.



[그림 #] Trapezoidal speed pattern

여기서 Acceleration time 은

$$T_{acc} = (V_{work} - V_{initial})/a$$

Where,

T_{acc} : Acceleration time

$V_{initial}$: Initial speed

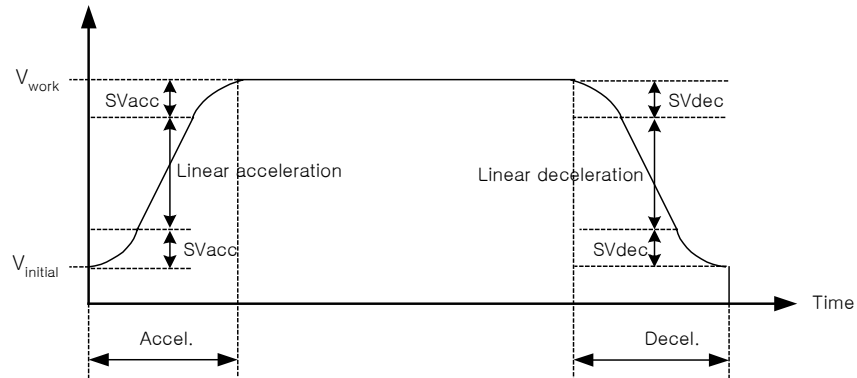
V_{work} : Working speed

a : Acceleration setting value

과 같으며 Deceleration time 또한 위와 같은 계산식이 적용됩니다.

☐ S-curve speed mode

S-curve speed mode에서는 Motion을 수행할 때 S자형 형태로 가속과 감속을 수행합니다. S-curve speed mode에서 가(감)속 구간은 [그림 #]과 같이 S-curve section과 Linear acceleration section으로 구성됩니다.



[그림 #] S-curve speed pattern

※ **S-curve section** : S-curve 형식의 가/감속이 이루어지는 구간. 이 구간은 **McSetSCurve** 메소드의 fSVacc 와 fSVdec 파라미터에 의해 설정됩니다. fSVacc 값이 0 이거나 속도 범위(Working speed - Initial speed)의 50%로 설정되면 가속구간은 Linear acceleration section 이 없이 모두 S-curve section 으로 구성됩니다. fSVdec 값이 0 이거나 속도 범위(Working speed - Initial speed)의 50%로 설정되면 감속구간은 Linear deceleration section 이 없이 모두 S-curve section 으로 구성됩니다.

※ S-curve speed mode 에서 **McSetAccel** 메소드를 통하여 설정한 가(감)속 값은 S-curve section 을 포함한 전체 가(감)속 시간을 결정하는 파라미터로 사용되며 실제 가(감)속도 또는 Jerk 는 자동으로 계산됩니다. 전체 가속 시간 Tacc 는

$$T_{acc} = (V_{work} - V_{initial})/a$$

여기서,

Tacc : Acceleration time

Vinitial : Initial speed

Vwork : Working speed

a : Acceleration setting value

과 같으며 Deceleration time 또한 위와 같은 계산식이 적용됩니다.

예 제

▣ 예제 1

다음의 예제는 Trapezoidal 속도 모드를 설정하는 예제입니다.

```
Call ComiLxAx1.LoadDivice
```

or

```

` Set trapezoidal speed mode
Call ComiLxAx1.McSetSpeedMode(X_AXIS, 1)
` Set speed as 1000
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, 1000)
` 가속도와 감속도를 각각 2000 으로 설정한다. 이렇게 하면 작업속도가
` 1000 이므로 가속 및 감속 시간은 각각 0.5 초 걸린다.
Call ComiLxAx1.McSetAccel(X_AXIS, 2000, 2000)
` 현재의 위치로부터 5000 만큼 이동
Call ComiLxAx1.McMove(X_AXIS, 5000)

Call ComiLxAx1.UnloadDevice
    
```

■ 예제 2

다음의 예제는 S-Curve 속도 모드를 설정하는 예제입니다.

```

Call ComiLxAx1.LoadDivice

` Set S-Curve speed mode
Call ComiLxAx1.McSetSpeedMode(X_AXIS, 2)
` Set speed as 1000
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, 1000)
` 가속도와 감속도를 각각 2000 으로 설정한다. 이렇게 하면 작업속도가
` 1000 이므로 가속 및 감속 시간은 각각 0.5 초 걸린다.
Call ComiLxAx1.McSetAccel(X_AXIS, 2000, 2000)
` Set S-Curve section range : 본 예제는 Linear section 이
` 없는 완 전한 S-curve 가/감속 모드가 되도록 SVacc, SVdec 값을
` 모두 0 으로 설정함
Call ComiLxAx1.McSetScurve(X_AXIS, 0, 0)
` 현재의 위치로부터 5000 만큼 이동
Call ComiLxAx1.McMove(X_AXIS, 5000)

Call ComiLxAx1.UnloadDevice
    
```

■ McSetSpeed

메소드 원형

Sub **McSetSpeed** (ByVal Channel, ByVal IniSpeed As Double, ByVal WorkSpeed As Double)

메소드 설명

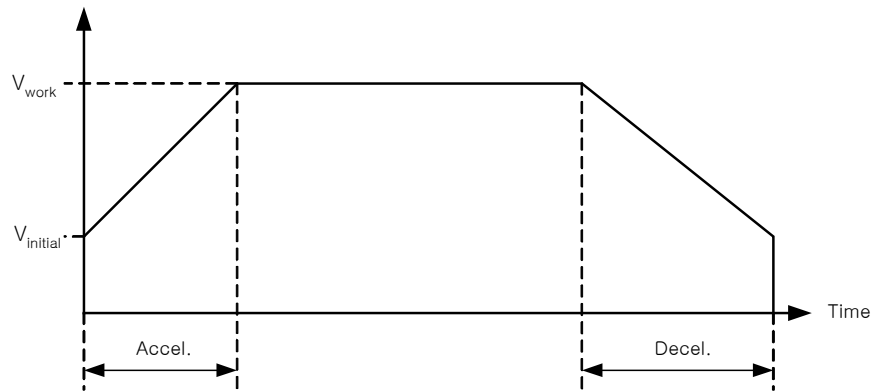
Motion 의 속도를 설정합니다. 단, 이 메소드는 Motion 에 바로 영향을 주는 것이 아니고 Move, MoveTo 등의 이송 메소드가 수행될 때 설정된 내용이 적용됩니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **IniSpeed** : 초기 속도를 설정합니다. 단, Constant 속도 모드에서는 이 값이 무시됩니다.
- ▶ **WorkSpeed** : 작업 속도를 설정합니다.

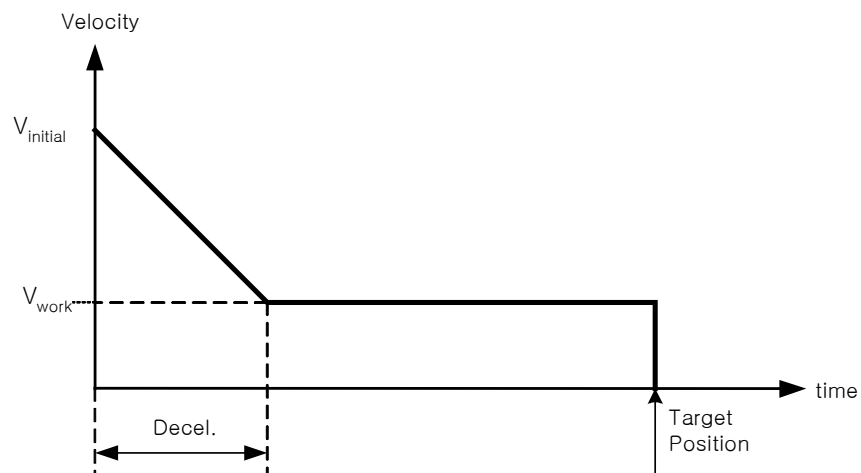
참 고

- 속도의 단위는 **McSetUnitSpeed** 메소드에 의하여 결정되며 기본적으로는 Pulses/sec 입니다.
- 속도의 단위는 **McSetUnitSpeed** 메소드에 의하여 결정되며 기본적으로는 Pulses/sec 입니다.
- 초기속도(fIniSpeed)가 작업속도(fWorkSpeed)가 설정 가능한 속도 범위보다 작거나 크면 자동으로 속도 범위의 최소값 또는 최대값으로 설정됩니다. 속도 범위는 **McSetSpeedRange** 메소드에 의해 결정됩니다.
- Trapezodial 또는 S-curve speed mode 에서 In-Position 명령을 수행할 때 작업속도(fWorkSpeed)가 초기속도(fIniSpeed)보다 크면 [그림 #]과 같이 초기속도⇒가속⇒작업속도⇒감속⇒정지의 동작을 수행합니다.



[그림 #] 작업속도가 초기속도보다 크게 설정된 경우의 속도 구성

□ Trapezoidal 또는 S-curve speed mode 에서 In-Position 명령을 수행할 때 작업속도($f_{workSpeed}$)가 초기속도($f_{iniSpeed}$)보다 작으면 [그림 #]과 같이 초기속도로부터 출발하여 작업속도까지 감속 후에 작업속도를 유지하고 목표 위치까지 이동한 후에는 감속 없이 바로 정지하게 됩니다.



[그림 #] 작업속도가 초기속도보다 작게 설정된 경우의 속도 구성

예 제

■ 예제 1

```
Call ComiLxAx1.LoadDivice
Call ComiLxAx1.McSetSpeedMode(X_AXIS, 0)
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, 1000)
Call ComiLxAx1.McSetStartVMove(X_AXIS, 1)

Call ComiLxAx1.McEmgStop(X_AXIS)

Call ComiLxAx1.UnloadDevice
```

■ 예제 2

1 회전에 필요한 펄스수가 3600 펄스일 때 거리의 단위를 각도(1°)로, 속도의 단위를 rpm으로 설정하는 예제입니다.

```
Call ComiLxAx1.LoadDevice

` Set 10 pulses for unit distance
` 이 예제에서는 1 회전에 필요한 펄스 수를 3600 펄스로
` 가정하고 단위 거리를  $1^{\circ}$ 로 설정한 것이다.
Call ComiLxAx1.McSetUnitDistance(0, 10)
` Set 3600/60(=60) PPS for unit speed
` 이 예제에서는 1 회전에 필요한 펄스수를 3600 펄스로
` 가정하고 단위 속도를 1rpm으로 설정한 것이다.
Call ComiLxAx1.McSetUnitSpeed(0, 3600./60)
` Set trapezoidal speed mode
Call ComiLxAx1.McSetSpeedMode(0, 1)
` Set speed as 100 rpm
Call ComiLxAx1.McSetSpeed(0, 0, 100)
` 가속도와 감속도를 각각 200rpm/s로 설정한다. 이렇게 하면 작업속도가
` 100rpm이므로 가속 및 감속 시간은 각각 0.5초 걸린다.
Call ComiLxAx1.McSetAccel(0, 200, 200)
` 모터를  $720^{\circ}$  회전한다. 실제로는  $720 \times 10$  펄스가 출력된다.
Call ComiLxAx1.McMove(0, 720)

Call ComiLxAx1.UnloadDevice
```

■ McSetAccel

메소드 원형

Sub **McSetAccel**(ByVal Channel, ByVal Accel As Double, ByVal Decel As Double)

메소드 설명

Motion 의 가/감속도를 설정합니다. 단, 이 메소드는 Motion 에 바로 영향을 주는 것이 아니고 Move, MoveTo 등의 이송 메소드가 수행될 때 설정된 내용이 적용됩니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **Accel** : 가속도를 설정합니다. 가속도의 단위는 **McSetUnitSpeed** 메소드에 의하여 결정되며 기본적으로는 1 PPS/SEC 입니다. 이 값이 0 이면 가속은 생략됩니다.
- ▶ **Decel** : 감속도를 설정합니다. 감속도의 단위는 **McSetUnitSpeed** 메소드에 의하여 결정되며 기본적으로는 1 PPS/SEC 입니다. 이 값이 0 이면 감속은 생략됩니다.

참 고

□ 가/감속은 McSetSpeedMode 메소드에서 속도 패턴(Speed Pattern)을 Trapezoidal 또는 S-Curve 로 지정한 경우에 적용됩니다.

□ Trapezoidal 속도 패턴에서는 가/감속 구간의 가/감속도와 일치하게 됩니다. 그러나 S-curve 속도 패턴에서는 이 메소드에서 지정한 가/감속 값은 전체 가/감속 구간(S-curve section 포함)의 시간을 결정하는 파라미터로 사용되며 가속도 값은 S-curve 의 range 에 따라 자동으로 결정됩니다. Trapezoidal 속도 패턴에서는 가/감속 구간의 가/감속도와 일치하게 됩니다. 그러나 S-curve 속도 패턴에서는 이 메소드에서 지정한 가/감속 값은 전체 가/감속 구간(S-curve section 포함)의 시간을 결정하는 파라미터로 사용되며 가속도 값은 S-curve 의 range 에 따라 자동으로 결정됩니다.

전체 가속 시간 Tacc 는

$$Tacc = (Vwork - Vinitial)/a$$

여기서,

Tacc : Acceleration time

Vinitial : Initial speed

Vwork : Working speed

a : Acceleration setting value

와 같으며 Deceleration time 또한 위와 같은 계산식이 적용됩니다.

예 제

■ 예제 1

```

Call ComiLxAx1.LoadDivice

Call ComiLxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, 1000)
Call ComiLxAx1.McSetAccel(X_AXIS, 2000, 2000)
  ` V=1000, Acc=2000, Dec=2000 의 속도 패턴으로
  ` 4000 만큼 이동
Call ComiLxAx1.McMove(X_AXIS, 4000)

Call ComiLxAx1.UnloadDevice

```

■ 예제 2

1 회전에 필요한 펄스수가 3600 펄스일 때 거리의 단위를 각도(1°)로, 속도의 단위를 rpm으로 설정하는 예제입니다.

```

Call ComiLxAx1.LoadDivice

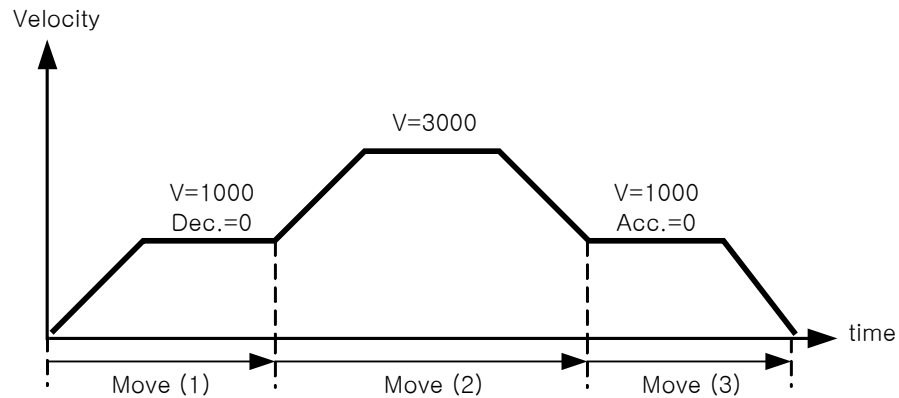
  ` Set 10 pulses for unit distance
  ` 이 예제에서는 1 회전에 필요한 펄스 수를 3600 펄스로
  ` 가정하고 단위 거리를  $1^{\circ}$ 로 설정한 것이다.
Call ComiLxAx1.McSetUnitDistance(0, 10)
  ` Set 3600/60(=60) PPS for unit speed
  ` 이 예제에서는 1 회전에 필요한 펄스수를 3600 펄스로
  ` 가정하고 단위 속도를 1rpm으로 설정한 것이다.
Call ComiLxAx1.McSetUnitSpeed(0, 3600./60)
  ` Set trapezoidal speed mode
Call ComiLxAx1.McSetSpeedMode(0, 1)
  ` Set speed as 100 rpm
Call ComiLxAx1.McSetSpeed(0, 0, 100)
  ` 가속도와 감속도를 각각 200rpm/s로 설정한다. 이렇게 하면 작업속도가
  ` 100rpm이므로 가속 및 감속 시간은 각각 0.5초 걸린다.
Call ComiLxAx1.McSetAccel(0, 200, 200)
  ` 모터를  $720^{\circ}$  회전한다. 실제로는  $720 \times 10$  펄스가 출력된다.
Call ComiLxAx1.McMove(0, 720)

Call ComiLxAx1.UnloadDevice

```

■ 예제 3

아래 그림과 같이 속도의 연속성을 가지는 3 단계의 In-Position 작업을 리스트 모션(Listed Motion) 기능을 이용하여 구현하는 예입니다. 리스트 모션은 하나의 작업과 다음 작업간의 지연시간을 없애고 연속적으로 수행될 수 있도록 일괄처리하는 기능입니다. 이 때 초기 속도, 작업 속도, 가속도, 감속도를 적절히 설정하면 여러 단계의 In-Position 작업을 속도의 연속성을 가지고 연속적으로 수행할 수 있습니다.

ur


작업	초기속도	작업속도	Acceleration	Deceleration
Move (1)	0	1000	2000	0
Move (2)	1000	3000	2000	2000
Move (3)	0	1000	0	2000

[그림 #] 속도의 연속성을 가지는 연속적인 In-Position 모션

```

Call ComiLxAx1.LoadDivice

Call ComiLxAx1.McBeginList '모션 리스트 등록 시작
' Set Trapezoidal Speed Mode
Call ComiLxAx1.McBeginList McSetSpeedMode(0, 1)
' Move (1)
Call ComiLxAx1.McSetSpeed(0, 0, 1000)
Call ComiLxAx1.McSetAccel(0, 2000, 0)
Call ComiLxAx1.McMoveTo(0, 3000)
' Move (2)
Call ComiLxAx1.McSetSpeed(0, 1000, 3000)
Call ComiLxAx1.McSetAccel(0, 2000, 2000)
Call ComiLxAx1.McMoveTo(0, 8000)
' Move (3)
Call ComiLxAx1.McSetSpeed(0, 0, 1000)
Call ComiLxAx1.McSetAccel(0, 0, 2000)
Call ComiLxAx1.McMoveTo(0, 3500)
Call ComiLxAx1.McEndList '모션 리스트 등록을 마칩

Call ComiLxAx1.McStartListMotion '리스트 모션 수행
' 리스트 모션이 모두 완료될 때까지 기다림
while not ComiLxAx1.McChekcListMotionDone
wend

Call ComiLxAx1.McBeginList UnloadDevice
    
```

■ McSetScurve

메소드 원형

Sub **McSetScurve** (ByVal Channel, ByVal Svacc As Double, ByVal Svdec Double)

메소드 설명

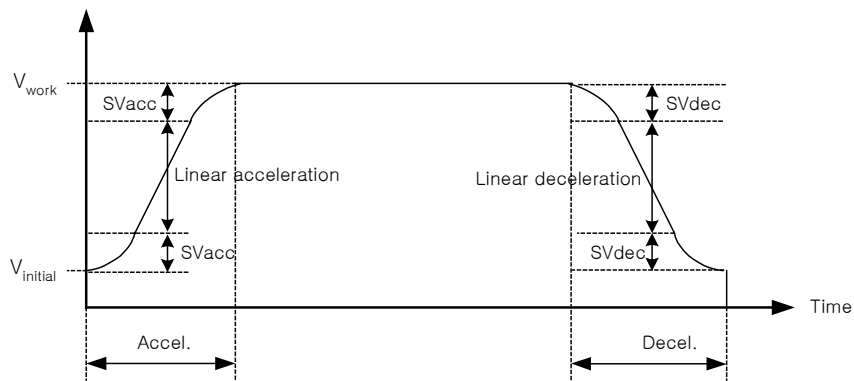
속도모드를 S-curve 속도 패턴으로 설정한 경우에 S-curve Section 의 범위를 속도단위로 설정합니다. 단, 이 메소드는 Motion 에 바로 영향을 주는 것이 아니고 Move, MoveTo 등의 이송 메소드가 수행될 때 설정된 내용이 적용됩니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **SVacc** : 가속구간의 S-curve Section 의 범위를 속도단위로 지정합니다.
- ▶ **SVdec** : 감속구간의 S-curve Section 의 범위를 속도단위로 지정합니다.

참 고

S-curve speed mode 에서는 Motion 을 수행할 때 S 자형 형태로 가속과 감속을 수행합니다. S-curve speed mode 에서 가(감)속 구간은 [그림 #]과 같이 S-curve section 과 Linear acceleration section 으로 구성됩니다.



[그림

#] S-curve speed pattern

※ **S-curve section** : S-curve 형식의 가/감속이 이루어지는 구간. 이 구간은 **McSetScurve** 메소드의 SVacc 와 SVdec 파라미터에 의해 설정됩니다. SVacc 값이 0 이거나 속도 범위(Working speed - Initial speed)의 50%로 설정되면 가속구간은 Linear acceleration section 이 없이 모두 S-curve section 으로 구성됩니다. SVdec 값이 0 이거나 속도 범위(Working speed - Initial speed)의 50%로 설정되면 감속구간은 Linear deceleration

or

section 이 없이 모두 S-curve section 으로 구성됩니다.

※ S-curve speed mode 에서 **McSetAccel** 메소드를 통하여 설정한 가(감)속 값은 S-curve section 을 포함한 전체 가(감)속 시간을 결정하는 파라미터로 사용되며 실제 가(감)속도 또는 Jerk 는 자동으로 계산됩니다. 전체 가속 시간 Tacc 는

$$Tacc = (Vwork - Vinitial)/a$$

여기서,

Tacc : Acceleration time

Vinitial : Initial speed

Vwork : Working speed

a : Acceleration setting value

과 같으며 Deceleration time 또한 위와 같은 계산식이 적용됩니다.

예 제

■ 예제 1

본 예제는 다음과 같은 속도 조건을 가지고 S-Curve 가/감속 모드로 In-Position 작업을 수행하는 예입니다.

```
Vinitial=0
Vwork=10000
Acc Time=0.5 초 => Acc = 10000/0.5 = 20000
Dec Time=0.5 초 => Dec = 10000/0.5 = 20000
SVacc=0 => No linear section in acceleration
SVdec=0 => No linear section in deceleration
```

이 예제는 svacc, svdec 값을 모두 0 으로 하므로써 가/감속시에 완전한 S-Curve 를 그리는 가/감속 모드를 수행합니다.

```
Call ComiLxAx1.LoadDivice

속도 모드를 s-curve 모드로 설정
Call ComiLxAx1.McSetSpeedMode(X_AXIS, 2)

Call ComiLxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiLxAx1.McSetAccel(X_AXIS, 20000, 20000)
Call ComiLxAx1.McSetScurve(X_AXIS, 0, 0)

Call ComiLxAx1.McMove(X_AXIS, 50000)

Call ComiLxAx1.UnloadDevice
```

■ 예제 2

본 예제는 다음과 같은 속도 조건을 가지고 S-Curve 가/감속 모드로 In-Position 작업을 수행하는 예입니다.

```
Vinitial=0
Vwork=10000
```

```
Acc Time=0.5 초 => Acc = 10000/0.5 = 20000
Dec Time=0.5 초 => Dec = 10000/0.5 = 20000
SVacc=2000
SVdec=2000
```

이 예제는 svacc, svdec 값을 각각 2000 으로 설정하므로써 0~2000 과 8000~10000 의 속도 구간은 s-curve 가/감속을, 2000 ~ 8000 의 속도 구간은 Linear 가/감속을 적용하도록 하는 예제입니다.

```
Call ComiLxAx1.LoadDivice

'속도 모드를 s-curve 모드로 설정
Call ComiLxAx1.McSetSpeedMode(X_AXIS, 2)

Call ComiLxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiLxAx1.McSetAccel(X_AXIS, 20000, 20000)
Call ComiLxAx1.McSetScurve(X_AXIS, 2000, 2000)

Call ComiLxAx1.McMove(X_AXIS, 50000)

Call ComiLxAx1.UnloadDevice
```

■ McStartVMove

메소드 원형

Sub **McStartVMove** (ByVal Channel As Long, ByVal Direction As Long)

메소드 설명

작업속도까지 가속한 후에 작업속도를 유지하며 정지메소드가 호출될 때까지 지정한 방향으로의 모션을 계속 수행합니다.

매개 변수

- ▶ *Channel* : 채널(축) 번호, 0 ~ 3
- ▶ *Direction* : 모션의 방향을 설정합니다.

Value	Meaning
0 또는 음수	(-) 방향
양수	(+) 방향

참 고

- ☐ 속도 모드를 Constant Speed Mode 로 지정한 경우에는 가속 구간이 없이 작업속도로 모션을 시작합니다.
- ☐ Velocity Move 를 정지할 때 McStop 또는 McEngStop 을 사용합니다.

예 제

```

Call ComiLxAx1.LoadDivice

Call ComiLxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiLxAx1.McSetAccel(X_AXIS, 20000, 20000)
' (+)방향으로 Velocity Move 수행
Call ComiLxAx1.McStartVMove(X_AXIS, 1)

' 어떤 특정 기간동안 Velocity Move 지속

' 감속 후 정지
Call ComiLxAx1.McStop(X_AXIS)

Call ComiLxAx1.UnloadDevice
    
```

■ McStartMove

메소드 원형

Sub **McStartMove** (ByVal Channel As Long, ByVal Distance As Long)

메소드 설명

하나의 축에 대하여 현재의 위치에서 지정한 거리만큼 이동을 수행합니다. 이 메소드는 모션(Motion)을 시작 시킨 후에 바로 Return 합니다. 속도 패턴은 McSetSpeedMode, McSetSpeed, McSetAccel, McSetScurve 메소드등에 의해 설정된 대로 이루어집니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **Distance** : 이동할 거리를 지정합니다. 이 값은 현재의 위치에 대한 상대 좌표이며 거리에 대한 단위는 McSetUnitDistance 메소드에 의해 결정됩니다. McSetUnitDistance 메소드로 거리의 단위를 변경하지 않았다면 거리의 단위는 Pulse 수가 됩니다. 즉, fDistance 값 1 은 1 Pulse 출력을 의미합니다.

참 고

- McMove 메소드가 모션이 완료될 때까지 Return 되지 않는데 반하여, 이 메소드는 지정한 모션을 시작시킨 후에 바로 Return 하게 됩니다.
- McStartMoveTo 메소드가 절대좌표로의 이동을 수행하는데 반하여, 이 메소드는 현재 위치에서 상대적인 거리를 파라미터로하여 이동을 수행합니다.

예 제

■ 예제 1

본 예제는 Trapezoidal 속도모드를 적용하여 30000 과 40000 으로 나누어 2 회의 Move 작업을 수행하므로써 총 70000 의 거리를 이동하는 예입니다.

```
Call ComiLxAx1.LoadDivice

Call ComiLxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiLxAx1.McSetAccel(X_AXIS, 20000, 20000)

Call ComiLxAx1.McStartMove(X_AXIS, 30000)
While Not(ComiLxAx1.McDone (X_AXIS))
Wend

Call ComiLxAx1.McStartMove(X_AXIS, 40000)
' 모션이 완료될때까지 기다린다
while Not(ComiLxAx1.McDone (X_AXIS))
Wend

Call ComiLxAx1.UnloadDevice
```

■ 예제 2

ur

본 예제는 McSetUnitDistance()메소드와 McSetUnitSpeed() 메소드를 사용하여 논리거리(Logic Distance)와 논리속도(Logic Speed)를 정의하여 Move 작업을 수행하는 예입니다. 1mm 이동하는데 필요한 펄스수가 100 펄스이고 1 회전에 필요한 펄스수가 10000 펄스일때 거리의 단위를 mm 로, 속도의 단위를 rpm 으로 설정하는 예제입니다.

```
Call ComiLxAx1.LoadDivice

` Set 100 pulses for unit distance
` 이 예제에서는 1mm 이동에 필요한 펄스수를 100 펄스로
` 가정하고 단위 거리를 1mm 로 설정한 것이다.
Call ComiLxAx1.McSetUnitDistance(X_AXIS, 100)
` Set 10000/60(=166.7) PPS for unit speed
` 이 예제에서는 1 회전에 필요한 펄스수를 10000
` 펄스로 가정하고 단위 속도를 1rpm 로 설정한 것이다.
Call ComiLxAx1.McSetUnitSpeed(X_AXIS, 10000./60)

Call ComiLxAx1.McSetSpeedMode(X_AXIS, 1)
` Set speed as 60 rpm
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, 60)
Call ComiLxAx1.McSetAccel(X_AXIS, 60, 60)
` 60 rpm 의 속도로 100mm 이동
Call ComiLxAx1.McStartMove(X_AXIS, 100)
While Not(ComiLxAx1.McDone (X_AXIS))
Wend

Call ComiLxAx1.UnloadDevice
```


■ McMove

메소드 원형

Sub **McMove** (ByVal Channel As Long, ByVal Distance As Double)

메소드 설명

하나의 축에 대하여 현재의 위치에서 지정한 거리만큼 이동을 수행합니다. 이 때의 속도 패턴은 McSetSpeedMode, McSetSpeed, McSetAccel, McSetScurve 메소드등에 의해 설정된 대로 이루어집니다. 이 메소드는 지정한 위치로의 이동이 완료되기 전까지 Return 되지 않습니다.

매개 변수

▶ **Channel** : 채널(축) 번호, 0 ~ 3

▶ **Distance** : 이동할 거리를 지정합니다. 이 값은 현재의 위치에 대한 상대 좌표이며 거리에 대한 단위는 McSetUnitDistance 메소드에 의해 결정됩니다. McSetUnitDistance 메소드로 거리의 단위를 변경하지 않았다면 거리의 단위는 Pulse 수가 됩니다. 즉, fDistance 값 1은 1Pulse 출력을 의미합니다.

참 고

□ McStartMove 메소드가 모션이 완료되는 것을 기다리지 않고 바로 Return 하는데 반하여, 이 메소드는 지정한 상대좌표로의 이동이 완료되기 전까지 Return 되지 않고 루프를 돌게 됩니다. 루프를 도는 동안 윈도우 이벤트나 메시지가 처리될 수 있도록 하려면 이 메소드를 수행하기 이전에 McSetBlockingMode 메소드를 사용하여 Blocking 이 일어나지 않도록 설정하여야 합니다.

□ McMoveTo 메소드가 절대좌표로의 이동을 수행하는데 반하여, 이 메소드는 현재 위치에서 상대적인 거리를 파라미터로하여 이동을 수행합니다.

예 제

■ 예제 1

본 예제는 Trapezoidal 속도모드를 적용하여 30000 과 40000 으로 나누어 2 회의 Move 작업을 수행하므로써 총 70000 의 거리를 이동하는 예입니다.

```
Call ComiLxAx1.LoadDevice

Call ComiLxAx1.McSetBlockingMode (FALSE)

Call ComiLxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiLxAx1.McSetAccel(X_AXIS, 20000, 20000)
Call ComiLxAx1.McMove(X_AXIS, 30000)
Call ComiLxAx1.McMove(X_AXIS, 40000)

Call ComiLxAx1.UnloadDevice
```

예제 2

본 예제는 McSetUnitDistance() 메소드와 McSetUnitSpeed() 메소드를 사용하여 논리거리(Logic Distance)와 논리속도(Logic Speed)를 정의하여 Move 작업을 수행하는 예입니다. 1mm 이동하는데 필요한 펄스수가 100 펄스이고 1 회전에 필요한 펄스수가 10000 펄스일때 거리의 단위를 mm 로, 속도의 단위를 rpm 으로 설정하는 예제입니다.

```

Call ComiLxAx1.LoadDivice

` Set 100 pulses for unit distance
` 이 예제에서는 1mm 이동에 필요한 펄스수를 100 펄스로
` 가정하고 단위 거리를 1mm 로 설정한 것이다.
Call ComiLxAx1.McSetUnitDistance(X_AXIS, 100)
` Set 10000/60(=166.7) PPS for unit speed
` 이 예제에서는 1 회전에 필요한 펄스수를 10000
` 펄스로 가정하고 단위 속도를 1rpm 로 설정한 것이다.
Call ComiLxAx1.McSetUnitSpeed(X_AXIS, 10000./60)

Call ComiLxAx1.McSetSpeedMode(X_AXIS, 1)
` Set speed as 60 rpm
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, 60)
Call ComiLxAx1.McSetAccel(X_AXIS, 60, 60)
` 60 rpm 의 속도로 100mm 이동
Call ComiLxAx1.McMove(X_AXIS, 100)

Call ComiLxAx1.UnloadDevice
    
```

■ McStartMoveTo

메소드 원형

Sub **McStartMoveTo** (ByVal Channel As Long, ByVal Position As Double)

메소드 설명

하나의 축에 대하여 지정한 절대좌표로의 이동을 수행합니다. 이 메소드는 모션(Motion)을 시작 시킨 후에 바로 Return 합니다. 속도 패턴은 McSetSpeedMode, McSetSpeed, McSetAccel, McSetScurve 메소드등에 의해 설정된 대로 이루어집니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **Position** : 이동할 절대 좌표 값을 지정합니다. 좌표의 단위는 McSetUnitDistance 메소드에 의해 결정됩니다.

참 고

- McMoveTo 메소드가 모션이 완료될 때까지 Return 되지 않는데 반하여, 이 메소드는 지정한 모션을 시작시킨 후에 바로 Return 하게 됩니다.
- McStartMove 메소드가 현재위치에 대한 상대좌표로의 이동을 수행하는데 반하여, 이 메소드는 절대좌표로의 이동을 수행합니다.

예 제

■ 예제 1

본 예제는 현재 좌표를 0 이라 가정하고 Trapezoidal 속도모드를 적용하여 2 회의 MoveTo 작업을 수행하므로써 절대좌표 70000 으로 이동하는 예입니다.

```
Call ComiLxAx1.LoadDivice

' Command Position 의 현재 좌표를 0 으로 초기화한다.
Call ComiLxAx1.McSetPosition_C (X_AXIS, 0)

Call ComiLxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiLxAx1.McSetAccel(X_AXIS, 20000, 20000)

' 좌표 0 에서 30000 으로 이동
Call ComiLxAx1.McStartMoveTo(X_AXIS, 30000)
While Not(ComiLxAx1.McDone (X_AXIS))
Wend
' 좌표 30000 에서 70000 으로 이동
Call ComiLxAx1.McStartMoveTo(X_AXIS, 70000)
While Not(ComiLxAx1.McDone(X_AXIS))
Wend

Call ComiLxAx1.UnloadDevice
```

 or

■ 예제 2

본 예제는 McSetUnitDistance() 메소드와 McSetUnitSpeed() 메소드를 사용하여 논리거리(Logic Distance)와 논리속도(Logic Speed)를 정의하여 Move 작업을 수행하는 예입니다. 1mm 이동하는데 필요한 펄스수가 100 펄스이고 1 회전에 필요한 펄스수가 10000 펄스일때 거리의 단위를 mm 로, 속도의 단위를 rpm 으로 설정하는 예제입니다.

```
Call ComiLxAx1.LoadDivice

` Set 100 pulses for unit distance
` 이 예제에서는 1mm 이동에 필요한 펄스수를 100 펄스로
` 가정하고 단위 거리를 1mm로 설정한 것이다.
Call ComiLxAx1.McSetUnitDistance(X_AXIS, 100)
` Set 10000/60(=166.7) PPS for unit speed
` 이 예제에서는 1 회전에 필요한 펄스수를 10000
` 펄스로 가정하고 단위 속도를 1rpm로 설정한 것이다.
Call ComiLxAx1.McSetUnitSpeed(X_AXIS, 10000./60)

` Command Position 의 현재 좌표를 0 으로 초기화한다.
Call ComiLxAx1.McSetPosition_C (X_AXIS, 0)

Call ComiLxAx1.McSetSpeedMode(X_AXIS, 1)
` Set speed as 60 rpm
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, 60)
Call ComiLxAx1.McSetAccel(X_AXIS, 60, 60)
` 60 rpm 의 속도로 좌표 100(mm)으로 이동
Call ComiLxAx1.McStartMoveTo(X_AXIS, 100)
While Not(ComiLxAx1.McDone (X_AXIS))
Wend

Call ComiLxAx1.UnloadDevice
```

■ McMoveTo

메소드 원형

Sub **McMoveTo**(ByVal Channel As Long)

메소드 설명

하나의 축에 대하여 지정한 절대좌표로의 이동을 수행합니다. 속도 패턴은 McSetSpeedMode, McSetSpeed, McSetAccel, McSetScurve 메소드등에 의해 설정된대로 이루어집니다. 이 메소드는 지정한 절대좌표로의 이동이 완료되기 전까지 Return 되지 않습니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **Position** : 이동할 절대 좌표 값을 지정합니다. 좌표의 단위는 McSetUnitDistance 메소드에 의해 결정됩니다.

참 고

- McStartMoveTo 메소드가 모션이 완료되는 것을 기다리지 않고 바로 Return 하는데 반하여, 이 메소드는 지정한 절대좌표로의 이동이 완료되기 전까지 Return 되지 않고 루프를 돌게 됩니다. 루프를 도는 동안 윈도우 이벤트나 메시지가 처리될 수 있도록 하려면 이 메소드를 수행하기 이전에 McSetBlockingMode 메소드를 사용하여 Blocking 이 일어나지 않도록 설정하여야 합니다.
- McMoveTo 메소드가 절대좌표로의 이동을 수행하는데 반하여, 이 메소드는 현재 위치에서 상대적인 거리를 파라미터로하여 이동을 수행합니다.

예 제

■ 예제 1

본 예제는 현재 좌표를 0 이라 가정하고 Trapezoidal 속도모드를 적용하여 2 회의 MoveTo 작업을 수행하므로써 절대좌표 70000 으로 이동하는 예입니다.

```
Call ComiLxAx1.LoadDevice

' Command Position 의 현재 좌표를 0 으로 초기화한다.
Call ComiLxAx1.McSetPosition_C (X_AXIS, 0)

Call ComiLxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiLxAx1.McSetAccel(X_AXIS, 20000, 20000)

' 좌표 0 에서 30000 으로 이동
Call ComiLxAx1.McMoveTo(X_AXIS, 30000)
' 좌표 30000 에서 70000 으로 이동
Call ComiLxAx1.McMoveTo(X_AXIS, 70000)

Call ComiLxAx1.UnloadDevice
```

■ 예제 2

본 예제는 `McSetUnitDistance()` 메소드와 `McSetUnitSpeed()` 메소드를 사용하여 논리거리(Logic Distance)와 논리속도(Logic Speed)를 정의하여 Move 작업을 수행하는 예입니다. 1mm 이동하는데 필요한 펄스수가 100 펄스이고 1 회전에 필요한 펄스수가 10000 펄스일때 거리의 단위를 mm 로, 속도의 단위를 rpm 으로 설정하는 예제입니다.

```
Call ComiLxAx1.LoadDivice

` Set 100 pulses for unit distance
` 이 예제에서는 1mm 이동에 필요한 펄스수를 100 펄스로
` 가정하고 단위 거리를 1mm로 설정한 것이다.
Call ComiLxAx1.McSetUnitDistance(X_AXIS, 100)
` Set 10000/60(=166.7) PPS for unit speed
` 이 예제에서는 1 회전에 필요한 펄스수를 10000
` 펄스로 가정하고 단위 속도를 1rpm로 설정한 것이다.
Call ComiLxAx1.McSetUnitSpeed(X_AXIS, 10000./60)

` Command Position 의 현재 좌표를 0 으로 초기화한다.
Call ComiLxAx1.McSetPosition_C (X_AXIS, 0)

Call ComiLxAx1.McSetSpeedMode(X_AXIS, 1)
` Set speed as 60 rpm
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, 60)
Call ComiLxAx1.McSetAccel(X_AXIS, 60, 60)
` 60 rpm 의 속도로 좌표 100(mm)으로 이동
Call ComiLxAx1.McMoveTo(X_AXIS, 100)

Call ComiLxAx1.UnloadDevice
```

■ McStop

메소드 원형

Sub **McStop** (ByVal Channel As Long)

메소드 설명

지정한 축에 대한 모션을 감속 후 정지합니다.

매개 변수

▶ *Channel* : 채널(축) 번호, 0 ~ 3

예 제

```
Call ComiLxAx1.LoadDivice

Call ComiLxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiLxAx1.McSetAccel(X_AXIS, 20000, 20000)
' (+)방향으로 Velocity Move 수행
Call ComiLxAx1.McStartVMove(X_AXIS, 1)

' 일정 동작동안 Velocity Move 지속

' 감속 후 정지
Call ComiLxAx1.McStop(X_AXIS)

Call ComiLxAx1.UnloadDevice
```

or

■ McEmgStop

메소드 원형

Sub **McEmgStop** (ByVal Channel As Long)

메소드 설명

지정한 축에 대한 모션을 감속없이 즉시 정지합니다.

매개 변수

▶ *Channel* : 채널(축) 번호, 0 ~ 3

예 제

· 디바이스는 로딩되어 있는 상태

```
Call ComiLxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiLxAx1.McSetAccel(X_AXIS, 20000, 20000)
' (+)방향으로 Velocity Move 수행
Call ComiLxAx1.McStartVMove(X_AXIS, 1)
```

· 일정 기간동안 Velocity Move 지속

· 감속없이 즉시 정지

```
Call ComiLxAx1.McEmgStop(X_AXIS)
```

```
Call ComiLxAx1.UnloadDevice
```


■ McDone

메소드 원형

Function **McDone** (ByVal Channel As Long) As Boolean

메소드 설명

하나의 축에 대하여 모션이 완료됐는지를 체크합니다.

매개 변수

▶ *Channel* : 채널(축) 번호, 0 ~ 3

Return 값

지정한 축의 모션이 완료됐는지를 알려줍니다.

Value	Meaning
0	모션이 완료되지 않았음
1	모션이 완료됨

예 제

```

Call ComiLxAx1.LoadDivice

Call ComiLxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiLxAx1.McSetAccel(X_AXIS, 20000, 20000)

Call ComiLxAx1.McStartMove(X_AXIS, 30000)
While Not(ComiLxAx1.McDone(X_AXIS))
Wend

Call ComiLxAx1.McStartMove(X_AXIS, 40000)
' 모션이 완료될때까지 기다린다
while Not (ComiLxAx1.McDone(X_AXIS))
Wend

Call ComiLxAx1.UnloadDevice

```

2.6.3 Multi-Axis 동시제어 메소드

이 단원에서는 Multi-Axis 동시 제어에 관련된 메소드들을 소개합니다. Multi-Axis 동시 제어는 여러 개의 축을 완전한 동기를 맞추어 동시에 제어하는 기능을 말합니다. 만일 속도 패턴을 동일하게 설정하였다면 여러 개의 제어 대상 축이 시작 및 종료 시점은 물론이고 가속/감속 구간까지 완전히 동기를 맞추어 제어될 수 있습니다.

Multi-Axis 동시제어 기능은 Velocity Move 와 In-position Move 모두에 적용 가능합니다. 이와 관련된 메소드들은 다음과 같습니다.

메소드 / 설명	페이지
Sub McStartVMoveAll (ByVal NumAxis As Long, ByRef AxisList[] As Long, ByRef DirList[] As Long) 여러 개의 축에 대하여 Velocity Move 작업을 동시에 시작합니다.	131
Sub McStartMoveAll (ByVal NumAxis As Long, ByRef AxisList[] As Long, ByRef DistList[] As Double) 여러 개의 축에 대하여 현재의 위치에서 지정한 거리만큼 이동을 동시에 시작합니다.	133
Sub McMoveAll (ByVal NumAxis As Long, ByRef AxisList[] As Long, ByRef DistList[] As Double) 여러 개의 축에 대하여 현재의 위치에서 지정한 거리만큼 이동을 수행합니다.	135
Sub McStartMoveToAll (ByVal NumAxis As Long, ByRef AxisList[] As Long, ByRef PosList[] As Long) 여러 개의 축에 대하여 지정한 절대좌표로의 이동을 동시에 시작합니다.	137
Sub McMoveToAll (ByVal NumAxis As Long, ByRef AxisList[] As Long, ByRef PosList[] As Long) 여러 개의 축에 대하여 지정한 절대좌표로의 이동을 수행합니다.	139
Sub McStopAll (ByVal NumAxis As Long, ByRef AxisList[] As Long) 여러 개의 축에 대한 모션을 동시에 감속 후 정지합니다.	141
Sub McEmgStopAll (ByVal NumAxis As Long, ByRef AxisList[] As Long) 여러 개의 축에 대한 모션을 동시에 감속없이 즉시 정지합니다.	142
Function McAllDone(ByVal NumAxis As Long, ByRef AxisList[] As Long)As Boolean 여러 개의 축에 대하여 지정한 모든 축의 모션이 완료됐는지를 체크합니다.	143

■ McStartVMoveAll

메소드 원형

```
Sub McStartVMoveAll (ByVal NumAxis As Long, ByRef AxisList[] As Long, ByRef DirList[] As Long)
```

메소드 설명

여러 개의 축에 대하여 Velocity Move 작업을 동시에 시작합니다. Velocity Move 는 작업속도까지 가속한 후에 작업속도를 유지하며 정지메소드가 호출될 때까지 지정된 방향으로의 모션을 계속 수행합니다. 이 메소드를 사용하면 여러 개의 축이 동시에 작업을 시작합니다. 따라서 이 메소드는 여러축이 동기를 맞추어 작업을 시작해야하는 경우에 유용하게 사용될 수 있습니다. 속도 패턴은 각 축에 대하여 McSetSpeedMode, McSetSpeed, McSetAccel, McSetScurve 메소드등에 의해 설정된 대로 이루어집니다.

매개 변수

- ▶ **NumAxis** : 동시에 작업을 수행할 대상 축의 수
- ▶ **AxisList** : 동시에 작업을 수행할 대상 축의 배열. 이 배열의 크기는 NumAxis 값과 일치해야 합니다.
- ▶ **DirList** : 방향을 지시하는 값의 배열. 이 배열의 크기는 NumAxis 값과 일치해야 합니다. 모션의 방향을 지시하는 값은 다음과 같습니다.

Value	Meaning
0 또는 음수	(-) 방향
양수	(+) 방향

예 제

```
Call ComiLxAx1.LoadDivice

Dim AxisList(1)
AxisList(0) = 0   ' X 축 첨가
AxisList(1) = 1   ' Y 축 첨가

Call ComiLxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiLxAx1.McSetAccel(X_AXIS, 20000, 20000)

Call ComiLxAx1.McSetSpeedMode(Y_AXIS, 1)
Call ComiLxAx1.McSetSpeed(Y_AXIS, 0, 10000)
Call ComiLxAx1.McSetAccel(Y_AXIS, 20000, 20000)

Call ComiLxAx1.McStartVMoveAll(2, AxisList(0), DirList(0))
' 일정 시간동안 Velocity Move 지속
' 감속 후 정지
Call ComiLxAx1.McStopAll(2, AxisList(0))
```

ur

Call ComiLxAx1.UnloadDevice

■ McStartMoveAll

메소드 원형

```
Sub McStartMoveAll (ByVal NumAxis As Long, ByRef AxisList[] As Long, ByRef DistList[] As Long)
```

메소드 설명

여러 개의 축에 대하여 현재의 위치에서 지정한 거리만큼 이동을 동시에 시작합니다. 이 메소드를 사용하면 여러 개의 축이 동시에 작업을 시작합니다. 따라서 이 메소드는 여러축이 동기를 맞추어 작업을 시작해야하는 경우에 유용하게 사용될 수 있습니다.

이 메소드는 모션(Motion)을 시작 시킨 후에 바로 Return 합니다. 속도 패턴은 각 축에 대하여 McSetSpeedMode, McSetSpeed, McSetAccel, McSetScurve 메소드등에 의해 설정된 대로 이루어집니다.

매개 변수

- ▶ **NumAxis** : 동시에 작업을 수행할 대상 축의 수
- ▶ **AixsList** : 동시에 작업을 수행할 대상 축의 배열. 이 배열의 크기는 NumAxis 값과 일치해야 합니다.
- ▶ **DistList** : 이동할 거리값의 배열. 이 배열의 크기는 NumAxis 값과 일치해야 합니다. 이동 거리값은 현재의 위치에 대한 상대 좌표이며 거리에 대한 단위는 McSetUnitDistance 메소드에 의해 결정됩니다. McSetUnitDistance 메소드로 거리의 단위를 변경하지 않았다면 거리의 단위는 Pulse 수가 됩니다. 즉, 거리값 1은 1Pulse 출력을 의미합니다.

참 고

□ McMoveAll 메소드가 모션이 완료될 때까지 Return 되지 않는데 반하여, 이 메소드는 지정한 모션을 시작시킨 후에 바로 Return 하게 됩니다.

□ McStartMoveToAll 메소드가 절대좌표로의 이동을 수행하는데 반하여, 이 메소드는 현재 위치에서 상대적인 거리를 파라미터로하여 이동을 수행합니다.

예 제

본 예제는 x 축과 y 축을 동시에 제어하는 것에 대한 예제입니다. 본 예제에서는 x 축과 y 축의 모션은 동시에 시작하되 각 축의 속도와 이동 거리는 다르게 설정할 수 있다는 것을 보여주기 위한 예제입니다.

```
Call ComiLxAx1.LoadDivice

Call ComiLxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiLxAx1.McSetAccel(X_AXIS, 20000, 20000)

Call ComiLxAx1.McSetSpeedMode(Y_AXIS, 1)
```

ur

```
Call ComiLxAx1.McSetSpeed(Y_AXIS, 0, 20000)
Call ComiLxAx1.McSetAccel(Y_AXIS, 40000, 40000)

Call ComiLxAx1.McStartMoveAll (2, AxisList(0), DistList(0))
While Not (ComiLxAx1.McAllDone (2, AxisList(0)))
Wend

Call ComiLxAx1.UnloadDevice
```

■ McMoveAll

메소드 원형

```
Sub McMoveAll (ByVal NumAxis As Long, ByRef AxisList[] As Long, ByRef DistList[] As Long)
```

메소드 설명

여러 개의 축에 대하여 현재의 위치에서 지정한 거리만큼 이동을 수행합니다. 이 메소드를 사용하면 여러 개의 축이 동시에 작업을 시작합니다. 따라서 이 메소드는 여러축이 동기를 맞추어 작업을 시작해야하는 경우에 유용하게 사용될 수 있습니다.

속도 패턴은 각 축에 대하여 McSetSpeedMode, McSetSpeed, McSetAccel, McSetScurve 메소드등에 의해 설정된 대로 이루어집니다.

매개 변수

- ▶ **NumAxis** : 동시에 작업을 수행할 대상 축의 수
- ▶ **AxisList** : 동시에 작업을 수행할 대상 축의 배열. 이 배열의 크기는 NumAxis 값과 일치해야 합니다.
- ▶ **DistList** : 이동할 거리값의 배열. 이 배열의 크기는 NumAxis 값과 일치해야 합니다. 이동 거리값은 현재의 위치에 대한 상대 좌표이며 거리에 대한 단위는 McSetUnitDistance 메소드에 의해 결정됩니다. McSetUnitDistance 메소드로 거리의 단위를 변경하지 않았다면 거리의 단위는 Pulse 수가 됩니다. 즉, Distance 값 1은 1Pulse 출력을 의미합니다.

참 고

□ 이 메소드는 지정된 모든 축의 모션이 완료될 때까지 Return 되지 않고 루프를 돌게 됩니다. 루프를 도는 동안 윈도우 이벤트나 메시지가 처리될 수 있도록 하려면 이 메소드를 수행하기 이전에 McSetBlockingMode 메소드를 사용하여 Blocking 이 일어나지 않도록 설정하여야 합니다.

□ McMoveToAll 메소드가 절대좌표로의 이동을 수행하는데 반하여, 이 메소드는 현재 위치에서 상대적인 거리를 파라미터로하여 이동을 수행합니다.

예 제

본 예제는 x 축과 y 축을 동시에 제어하는 것에 대한 예제입니다. 본 예제에서는 x 축과 y 축의 모션은 동시에 시작하되 각 축의 속도와 이동 거리는 다르게 설정할 수 있다는 것을 보여주기 위한 예제입니다.

```
Call ComiLxAx1.LoadDevice

Call ComiLxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiLxAx1.McSetAccel(X_AXIS, 20000, 20000)
```

ur

```
Call ComiLxAx1.McSetSpeedMode(Y_AXIS, 1)
Call ComiLxAx1.McSetSpeed(Y_AXIS, 0, 20000)
Call ComiLxAx1.McSetAccel(Y_AXIS, 40000, 40000)

Call ComiLxAx1.McMoveAll (2, AxisList(0), DistList(0))

Call ComiLxAx1.UnloadDevice
```


■ McStartMoveToAll

메소드 원형

```
Sub McStartMoveToAll (ByVal NumAxis As Long, ByRef AxisList[] As Long, ByRef  
PosList[] As Long)
```

메소드 설명

여러 개의 축에 대하여 지정한 절대좌표로의 이동을 시작합니다. 이 메소드를 사용하면 여러 개의 축이 동시에 작업을 시작합니다. 따라서 이 메소드는 여러축이 동기를 맞추어 작업을 시작해야하는 경우에 유용하게 사용될 수 있습니다. 속도 패턴은 각 축에 대하여 McSetSpeedMode, McSetSpeed, McSetAccel, McSetScurve 메소드등에 의해 설정된 대로 이루어집니다.

매개 변수

- ▶ **NumAxis** : 동시에 작업을 수행할 대상 축의 수
- ▶ **AixsList** : 동시에 작업을 수행할 대상 축의 배열. 이 배열의 크기는 NumAxis 값과 일치해야 합니다.
- ▶ **PosList** : 절대좌표값의 배열. 이 배열의 크기는 NumAxis 값과 일치해야 합니다. 좌표에 대한 단위는 McSetUnitDistance 메소드에 의해 결정됩니다. McSetUnitDistance 메소드로 거리의 단위를 변경하지 않았다면 좌표의 단위는 Pulse 수가 됩니다. 즉, 좌표값 1은 1Pulse 출력을 의미합니다.

참 고

□ McMoveToAll 메소드가 모션이 완료될 때까지 Return 되지 않는데 반하여, 이 메소드는 지정한 모션을 시작시킨 후에 바로 Return 하게 됩니다.

□ McStartMoveAll 메소드가 현재위치에 대한 상대좌표로의 이동을 수행하는데 반하여, 이 메소드는 절대좌표로의 이동을 수행합니다.

예 제

본 예제는 x 축과 y 축을 동시에 제어하는 것에 대한 예제입니다. 본 예제에서는 x 축과 y 축의 모션은 동시에 시작하되 각 축의 속도와 이동 거리는 다르게 설정할 수 있다는 것을 보여주기 위한 예제입니다.

```
Call ComiLxAx1.LoadDivice

Call ComiLxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiLxAx1.McSetAccel(X_AXIS, 20000, 20000)

Call ComiLxAx1.McSetSpeedMode(Y_AXIS, 1)
Call ComiLxAx1.McSetSpeed(Y_AXIS, 0, 20000)
Call ComiLxAx1.McSetAccel(Y_AXIS, 40000, 40000)
```

ur

```
Call ComiLxAx1.McStartMoveToAll (2, AxisList(0), PosList(0))
While Not (ComiLxAx1.McAllDone (2, AxisList(0)))
Wend

Call ComiLxAx1.UnloadDevice
```

■ McMoveToAll

메소드 원형

Sub **McMoveToAll** (ByVal NumAxis As Long, ByRef AxisList[] As Long, ByRef PosList[] As Long)

메소드 설명

여러 개의 축에 대하여 지정한 절대좌표로의 이동을 수행합니다. 이 메소드를 사용하면 여러 개의 축이 동시에 작업을 시작합니다. 따라서 이 메소드는 여러축이 동기를 맞추어 작업을 시작해야하는 경우에 유용하게 사용될 수 있습니다. 속도 패턴은 각 축에 대하여 McSetSpeedMode, McSetSpeed, McSetAccel, McSetScurve 메소드등에 의해 설정된 대로 이루어집니다.

매개 변수

- ▶ **NumAxis** : 동시에 작업을 수행할 대상 축의 수
- ▶ **AixsList** : 동시에 작업을 수행할 대상 축의 배열. 이 배열의 크기는 NumAxis 값과 일치해야 합니다.
- ▶ **PosList** : 절대좌표값의 배열. 이 배열의 크기는 NumAxis 값과 일치해야 합니다. 좌표에 대한 단위는 McSetUnitDistance 메소드에 의해 결정됩니다. McSetUnitDistance 메소드로 거리의 단위를 변경하지 않았다면 좌표의 단위는 Pulse 수가 됩니다. 즉, 좌표값 1은 1Pulse 출력을 의미합니다.

참 고

□ 이 메소드는 모션이 완료될 때까지 Return 되지 않고 루프를 돌게 됩니다. 루프를 도는 동안 윈도우 이벤트나 메시지가 처리될 수 있도록 하려면 이 메소드를 수행하기 이전에 McSetBlockingMode 메소드를 사용하여 Blocking 이 일어나지 않도록 설정하여야 합니다.

□ McMoveAll 메소드가 현재위치에 대한 상대좌표로의 이동을 수행하는데 반하여, 이 메소드는 절대좌표로의 이동을 수행합니다.

예 제

본 예제는 x 축과 y 축을 동시에 제어하는 것에 대한 예제입니다. 본 예제에서는 x 축과 y 축의 모션은 동시에 시작하되 각 축의 속도와 이동 거리는 다르게 설정할 수 있다는 것을 보여주기 위한 예제입니다.

```
Call ComiLxAx1.LoadDivice

Call ComiLxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiLxAx1.McSetAccel(X_AXIS, 20000, 20000)

Call ComiLxAx1.McSetSpeedMode(Y_AXIS, 1)
```

라이브러리 사용과 지원 method

ur

```
Call ComiLxAx1.McSetSpeed(Y_AXIS, 0, 20000)
Call ComiLxAx1.McSetAccel(Y_AXIS, 40000, 40000)

Call ComiLxAx1.McMoveToAll (2, AxisList(0), PosList(0))

Call ComiLxAx1.UnloadDevice
```

■ McStopAll

메소드 원형

```
Sub McStopAll (ByVal NumAxis As Long, ByRef AxisList[] As Long)
```

메소드 설명

여러 개의 축에 대한 모션을 동시에 감속 후 정지합니다.

매개 변수

- ▶ **NumAxis** : 동시에 작업을 수행할 대상 축의 수
- ▶ **AxisList** : 동시에 작업을 수행할 대상 축의 배열. 이 배열의 크기는 NumAxis 값과 일치해야 합니다.

예 제

```
Call ComiLxAx1.LoadDevice

Call ComiLxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiLxAx1.McSetAccel(X_AXIS, 20000, 20000)

Call ComiLxAx1.McSetSpeedMode(Y_AXIS, 1)
Call ComiLxAx1.McSetSpeed(Y_AXIS, 0, 10000)
Call ComiLxAx1.McSetAccel(Y_AXIS, 20000, 20000)

Call ComiLxAx1.McStartVMoveAll (2, AxisList(0), DirList(0))

' 일정 시간동안 Velocity Move 지속

' 감속 후 정지
Call ComiLxAx1.McStopAll(2, AxisList(0))

Call ComiLxAx1.UnloadDevice
```

■ McEmgStopAll

메소드 원형

Sub **McEmgStopAll** (ByVal NumAxis As Long, ByRef AxisList[] As Long)

메소드 설명

여러 개의 축에 대한 모션을 동시에 감속없이 즉시 정지합니다.

매개 변수

- ▶ **NumAxis** : 동시에 작업을 수행할 대상 축의 수
- ▶ **AixsList** : 동시에 작업을 수행할 대상 축의 배열. 이 배열의 크기는 NumAxis 값과 일치해야 합니다.

예 제

본 예제는 x 축과 y 축을 동시에 Velocity Move 작업을 수행하면서 Digital Input CH0 가 ON 이 되면 즉시 정지하고, 사용자가 키입력을 하면 정상 종료(감속후 정지)를 하는 것에 대한 예제입니다.

```
Call ComiLxAx1.LoadDivice

Call ComiLxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiLxAx1.McSetAccel(X_AXIS, 20000, 20000)

Call ComiLxAx1.McSetSpeedMode(Y_AXIS, 1)
Call ComiLxAx1.McSetSpeed(Y_AXIS, 0, 10000)
Call ComiLxAx1.McSetAccel(Y_AXIS, 20000, 20000)

Call ComiLxAx1.McStartVMoveAll (2, AxisList(0), DirList(0))

' 일정 시간동안 Velocity Move 지속

' 감속 후 정지
Call ComiLxAx1.McEmgStopAll(2, AxisList(0))

Call ComiLxAx1.UnloadDevice
```

■ McAllDone

메소드 원형

```
Function McAllDone(ByVal NumAxis As Long, ByRef AxisList[] As Long)As Boolean
```

메소드 설명

여러 개의 축에 대하여 지정한 모든 축의 모션이 완료됐는지를 체크합니다.

매개 변수

- ▶ **NumAxis** : 동시에 작업을 수행할 대상 축의 수
- ▶ **AxisList** : 동시에 작업을 수행할 대상 축의 배열. 이 배열의 크기는 NumAxis 값과 일치해야 합니다.

Return 값

지정한 모든 축의 모션이 완료됐는지를 알려줍니다.

Value	Meaning
0	모션이 완료되지 않았음
1	모션이 완료됨

예 제

```
Call ComiLxAx1.LoadDevice

Call ComiLxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiLxAx1.McSetAccel(X_AXIS, 20000, 20000)

Call ComiLxAx1.McSetSpeedMode(Y_AXIS, 1)
Call ComiLxAx1.McSetSpeed(Y_AXIS, 0, 20000)
Call ComiLxAx1.McSetAccel(Y_AXIS, 40000, 40000)

Call ComiLxAx1.McStartMoveAll (2, AxisList(0), DistList(0))
While Not (ComiLxAx1.McAllDone (2, AxisList(0)))
Wend

Call ComiLxAx1.UnloadDevice
```

2.6.4 Coordinated Motion 메소드

이 단원에서는 Coordinated Motion 에 관련된 메소드들을 소개합니다. Coordinated Motion 이란 두 축 이상의 축이 연동되어 직선 보간(Linear Interpolation), 원호 보간(Circular Interpolation) 등의 모션을 수행하는 것을 의미합니다.

Multi-Axis 동시 제어 기능도 여러 개의 축을 제어하되 각 축이 서로 연동되어서 모션을 수행하는 것이 아니고 각 축이 개별적으로 모션을 수행하되 동시에 시작하는 것임에 반하여 Coordinated Motion 은 여러 개의 축이 서로 연동되어서 보간 이동을 수행한다는 것이 Multi-Axis 동시 제어와 차이가 있습니다.

Coordinated Motion 에 관련된 메소드들은 다음과 같습니다.

메소드 / 설명	페이지
Function McMapAxes (ByVal MapIndex As Long, ByVal MapMask As Integer) As Boolean Coordinated Motion 을 수행할 축들을 그룹화합니다	147
Sub McSetSpeedModeMx (ByVal MapIndex As Long, ByVal ModeIndex As Long) Coordinated Motion 의 속도 모드를 설정합니다.	149
Sub McSetSpeedMx (ByVal MapIndex As Long, ByVal Speed As Long, ByVal Accel As Long) Coordinated Motion 의 속도 및 가속도를 설정합니다	151
Sub McStartLine(ByVal MapIndex As Long, ByRef DistList[] As Double) 현재 위치로부터의 상대 좌표로의 직선 보간 이동을 수행합니다.	155
Sub McLine (ByVal MapIndex As Long, ByRef DistList[] As Double) 현재 위치로부터의 상대 좌표로의 직선 보간 이동을 수행합니다	157
Sub McStartLineTo(ByVal MapIndex As Long, ByRef PosList[] As Double) 절대 좌표로의 직선 보간 이동을 수행합니다.	159
Sub McLineTo(ByVal MapIndex As Long, ByRef PosList[] As Double) 절대 좌표로의 직선 보간 이동을 수행합니다.	162
Sub McStartArcA(ByVal MapIndex As Long, ByVal XcentOffset As Double, ByVal YcentOffset As Double, ByVal EndAngle As Double) 상대좌표를 파라미터로 하여 원호보간 이동을 수행합니다.	165
Sub McArcA(ByVal MapIndex As Long, ByVal XcentOffset As Double, ByVal YcentOffset As Double, ByVal EndAngle As Double) 상대좌표를 파라미터로 하여 원호보간 이동을 수행합니다.	167
Sub McStartArcP(ByVal MapIndex As Long, ByVal XCentOffset As Double, ByVal	170

<p>YCentOffset As Double, ByVal XEndPoByValDist As Double, ByVal YendPoByValDist As Double, ByVal Dir As Long)</p> <p>상대좌표를 파라미터로 하여 원호보간 이동을 수행합니다.</p>	
<p>Sub McArcP(ByVal MapIndex As Long, ByVal XcentOffset As Double, ByVal YcentOffset As Double, ByVal XendPoByValDist As Double, ByVal YendPoByValDist As Double, ByVal Dir As Long)</p> <p>상대좌표를 파라미터로 하여 원호보간 이동을 수행합니다.</p>	172
<p>Sub McStartArcToA(ByVal MapIndex As Long, ByVal Xcent As Double, ByVal Ycent As Double, ByVal EndAngle As Double)</p> <p>절대좌표를 파라미터로 하여 원호보간 이동을 수행합니다.</p>	175
<p>Sub McArcToA(ByVal MapIndex As Long, ByVal Xcent As Double, ByVal Ycent As Double, ByVal EndAngle As Double)</p> <p>절대좌표를 파라미터로 하여 원호보간 이동을 수행합니다.</p>	177
<p>Sub McStartArcToP(ByVal MapIndex As Long, ByVal Xcent As Double, ByVal Ycent As Double, ByVal XendPos As Double, ByVal YendPos As Double, ByVal Dir As Long)</p> <p>절대좌표를 파라미터로 하여 원호보간 이동을 수행합니다.</p>	181
<p>Sub McArcToP(ByVal MapIndex As Long, ByVal Xcent As Double, ByVal Ycent As Double, ByVal XendPos As Double, ByVal YendPos As Double, ByVal Dir As Long)</p> <p>절대좌표를 파라미터로 하여 원호보간 이동을 수행합니다.</p>	183
<p>sub McCompleteArc (ByVal MapIndex As Long)</p> <p>원호보간 작업을 수행하였을 때 Ending 좌표와 목표 좌표의 차이를 보정해주는 함수입니다.</p>	188
<p>Sub McBuildSpline (ByRef InArray As Double, ByVal NumInArray As Long, ByRef OutArray As Double, ByVal NumOutArray As Long)</p> <p>사용자가 입력한 데이터를 기반으로 Cubic spline 보간을 수행한 후 OutArray 에 그 결과값을 넣어줍니다.</p>	189
<p>Function McStartHelical (ByVal MapIndex As Long, ByVal Zaxis As Long, ByVal Xcenter As Double, ByVal Ycenter As Double, ByVal Direction As Long, ByVal NumCircle As Long, ByVal LastArcAngle As Double, ByVal Zdistance As Double) As Boolean</p> <p>헬리컬 보간 구동을 시작합니다.</p>	190



UI

Function McAbortHelical As Boolean 현재 구동되고 있는 헬리컬 보간 작업을 취소합니다.	192
Function McMxDone(ByVal MapIndex As Long) As Boolean 지정한 축그룹(MapIndex)의 Coordinated Motion 이 완료됐는지를 체크합니다.	193

■ McMapAxes

메소드 원형

Function **McMapAxes** (ByVal MapIndex As Long, ByVal MapMask As Integer) As Boolean

메소드 설명

Coordinated Motion 을 수행할 축들을 그룹화합니다. Coordinated Motion 축 그룹은 최대 2 개(0 과 1)까지 지정할 수 있으며 MapIndex 가 그룹을 지정하는 인덱스값입니다. 각 축 그룹은 최대 4 개의 축을 포함할 수 있습니다.

매개 변수

- ▶ **MapIndex** : 축 그룹 인덱스, 이 값은 0 또는 1 이어야 합니다.
- ▶ **AixsList** : 그룹에 포함할 축들을 지정할 마스크 값. 이 값의 BIT0-BIT3 을 이용하여 그룹에 포함할 축들을 지정합니다. 각 비트의 값이 0 이면 해당 축(비트의 순서와 일치하는 축)은 배제되는 것이며 1 이면 해당 축이 포함되는 것입니다.

참 고

□ Coordinated Motion 에 관련된 모든 메소드들을 사용하기 전에 먼저 이 메소드를 이용하여 축들을 그룹화하여야 합니다. Coordinated Motion 에 관련된 모든 메소드들은 Map Index 를 파라미터로 입력받게 되어 있는데 이 메소드의 MapIndex 에 지정한 값을 입력하면 됩니다.

예 제

■ 예제 1

본 예제는 x 축과 y 축을 그룹화하여 (10000, 20000)의 상대좌표로의 직선보간 이동을 수행하는 예제입니다.

```
Call ComiLxAx1.LoadDivice

Const X_MASK = 1
Const Y_MASK = 2
Const Z_MASK = 4
Const U_MASK = 8

Dim DistList(1)

' Map X&Y axis to MAP0
Call ComiLxAx1.McMapAxes(MAP0, X_MASK or Y_MASK)
' Set speed mode as Trapezoidal
Call ComiLxAx1.McSetSpeedModeMx(MAP0, 1)
' Set speed & accel => V=5000, Acc=8000
Call ComiLxAx1.McSetSpeedMx(MAP0, 5000, 8000)
' Move to relative coord. (10000, 20000)
DistList(0)=10000
DistList(1)=20000

Call ComiLxAx1.McLine(MAP0, DistList(0))

Call ComiLxAx1.UnloadDevice
```

■ 예제 2

본 예제는 2 개의 축그룹을 동시에 Coordinated Motion 을 수행하는 것을 예로 보이기 위한 것입니다. x 축과 y 축을 MAP0 에 맵핑하고 z 축과 u 축을 MAP1 에 맵핑하여 두개의 축 그룹을 동시에 Coordinated Motion 을 수행합니다.

```

Call ComiLxAx1.LoadDivice
const X_MASK = 1
const Y_MASK = 2
Const Z_MASK = 4
Const U_MASK = 8

Const MAP0 = 0
Const MAP1 = 1

Dim DistList1(1), DistList2(1)

DistList1(0)=10000
DistList1(1)=20000
DistList2(0)=8000
DistList2(1)=5000

` Map X&Y axis to MAP0
Call ComiLxAx1.McMapAxes(MAP0, X_MASK or Y_MASK)
` Map Z&U axis to MAP1
Call ComiLxAx1.McMapAxes(MAP1, Z_MASK or U_MASK)

` Set speed mode of MAP0 as Trapezoidal
Call ComiLxAx1.McSetSpeedModeMx(MAP0, 1)
` Set speed & accel of MAP0 => V=5000, Acc=8000
Call ComiLxAx1.McSetSpeedMx(MAP0, 5000, 8000)
` Move to relative coord. (10000, 20000)

` Set speed mode of MAP1 as Trapezoidal
Call ComiLxAx1.McSetSpeedModeMx(MAP1, 1)
` Set speed & accel of MAP1 => V=2000, Acc=5000
Call ComiLxAx1.McSetSpeedMx(MAP1, 2000, 5000)

` Move to relative coord of (X, Y) => (10000, 20000)
Call ComiLxAx1.McStartLine(MAP0, DistList1(0))
` Move to relative coord of (Z, U) => (8000, 5000)
Call ComiLxAx1.McStartLine(MAP1, DistList2(0))

Call ComiLxAx1.UnloadDevice

```

■ McSetSpeedModeMx

메소드 원형

Sub **McSetSpeedModeMx** (ByVal MapIndex As Long, ByVal ModeIndex As Long)

메소드 설명

Coordinated Motion 의 속도 모드를 설정합니다. 단, 이 메소드는 Motion 에 바로 영향을 주는 것이 아니고 Line, Arc 등의 Coordinated Motion 이송 메소드가 수행될 때 설정된 내용이 적용됩니다.

매개 변수

▶ **MapIndex** : 축 그룹 인덱스, 이 값은 0 또는 1 이어야 하며 McMapAxes 메소드를 통하여 제어 대상축들이 맵핑되어 있어야 합니다.

▶ **ModeIndex** : 속도 모드를 지정합니다. 속도 모드는 아래의 표와 같이 3 가지로 설정할 수 있습니다. Coordinated Motion 에서는 S-curve 속도 모드로 설정하면 가/감속 구간에서 Linear Section 이 없는 완전한 S-curve 가/감속 모드로 구성됩니다. 각각의 속도 모드에 대한 자세한 사항은 McSetSpeedMode 메소드를 참조하십시오.

Value	Meaning
0	Constant speed mode
1	Trapezoidal speed mode
2	S-curve speed mode

예 제

본 예제는 x 축과 y 축을 그룹화하여 (10000, 20000)의 상대좌표로의 직선보간 이동을 수행하는 예제입니다.

```

Call ComiLxAxis1.LoadDevice

Const X_MASK = 1
Const Y_MASK = 2
Const Z_MASK = 4
Const U_MASK = 8

Const MAP0 = 0

Dim DistList(1)
DistList(0) = 10000
DistList(1) = 20000

' Map X&Y axis to MAP0
Call ComiLxAxis1.McMapAxes(MAP0, X_MASK or Y_MASK)
' Set speed mode as Trapezoidal
Call ComiLxAxis1.McSetSpeedModeMx(MAP0, 1)
' Set speed & accel => V=5000, Acc=8000
Call ComiLxAxis1.McSetSpeedMx(MAP0, 5000, 8000)
' Move to relative coord. (10000, 20000)

```

ur

```
Call ComiLxAx1.McLine(MAP0, DistList(0))
```

```
Call ComiLxAx1.UnloadDevice
```

■ McSetSpeedMx

메소드 원형

Sub **McSetSpeedMx** (ByVal MapIndexAs Long, ByVal Speed As Long, ByVal Accel As Long)

메소드 설명

Coordinated Motion 의 속도 및 가속도를 설정합니다. 단, 이 메소드는 Motion 에 바로 영향을 주는 것이 아니고 Line, Arc 등의 Coordinated Motion 이송 메소드가 수행될 때 설정된 내용이 적용됩니다.

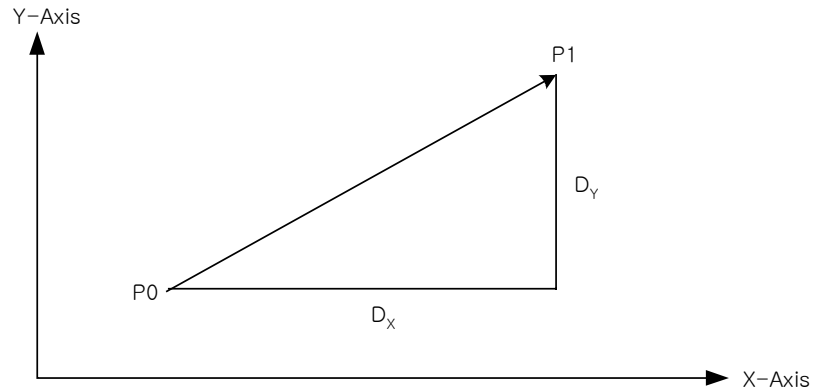
매개 변수

- ▶ **MapIndex** : 축 그룹 인덱스, 이 값은 0 또는 1 이어야 합니다.
- ▶ **Speed** : 작업속도를 벡터 속도값으로 지정합니다. 벡터 속도에 대한 자세한 내용은 “참고” 항목을 참조하십시오.
- ▶ **Accel** : 가속도와 감속도를 지정합니다. Coordinated Motion 에서는 가속도와 감속도가 같은값으로 설정됩니다.

참 고

- Coordinated Motion 에서 초기속도는 자동으로 최저 속도로 설정됩니다.
- 직선 보간 메소드(Line 또는 LineTo 메소드)에서는 Constant, Trapezoidal, S-Curve 의 세 가지 속도 모드를 모두 적용할 수 있습니다. 단, 다음과 같은 제약사항이 있습니다.
 1. 감속도(Deceleration)는 가속도와 같은값으로 자동 설정됩니다.
 2. S-curve 속도모드로 설정하면 가/감속 구간에서 Linear Section 이 없는 완전한 S-curve 가/감속을 수행하게 됩니다.
 3. 원형 보간 메소드(Arc 또는 ArcTo 메소드)에서는 가/감속도가 적용되지 않습니다.
- 직선 보간 이동시의 벡터 속도

그림[#]은 2 축(편의상 X, Y 축으로 가정) 직선 보간 이동을 그래프로 나타낸 것입니다.



[그림#] X, Y 축간의 직선 보간 이송

그래프와 같이 P0 지점에서 P1 으로 이송시에 X 축 이송 거리 D_x 와 Y 축 이송 거리 D_y 사이의 관계는 다음과 같습니다.

$$\Delta P = \sqrt{D_x^2 + D_y^2}$$

각 축의 이송 거리와 각 축의 속도는 정비례하므로 벡터 속도 V , X 축의 속도 V_x 그리고 Y 축의 속도 V_y 간의 관계는 다음과 같이 됩니다.

$$V_x = \frac{D_x \times V}{\sqrt{D_x^2 + D_y^2}}$$

$$V_y = \frac{D_y \times V}{\sqrt{D_x^2 + D_y^2}}$$

마찬가지로 3 축과 4 축 직선 보간 이동에서도 벡터 속도와 각 축의 속도간의 관계는 다음과 같은 관계식이 성립됩니다.

3 축(편의상 X, Y, Z 축으로 가정)의 경우 각 축의 속도는

$$V_i = \frac{D_i \times V}{\sqrt{D_x^2 + D_y^2 + D_z^2}}$$

과 같이 되며 4 축의 경우에는

$$V_i = \frac{D_i \times V}{\sqrt{D_X^2 + D_Y^2 + D_Z^2 + D_U^2}}$$

과 같이 됩니다.

샘플코드를 예를 들어 설명하면 다음과 같습니다.

```
Const X_Axis = 1
Const Y_Axis = 2
Const MAP_IDX = 0
\ 코드의 간결성을 위하여 앞에서 행해져야할 초기화 루틴은 모두 생략
\ .....
\ X 축과 Y 축을 0 번 그룹으로 그룹화함
Call ComiLxAx1.McMapAxes (MAP_IDX, (X_AXIS or Y_AXIS))
\ Trapezoidal 속도모드로 설정
Call ComiLxAx1.McSetSpeedModeMx(MAP_IDX, 1)
\ 벡터속도 1000 PPS, 벡터가속도 2000 PPS/sec 로 설정
Call ComiLxAx1.McSetSpeedMx(MAP_IDX, 1000, 2000)

Dim DistList(1)

DistList(0) = 3000
DistList(1) = 4000

Call ComiLxAx1.McLine(MAP_IDX, DistList(0))
```

위의 코드는 현재 위치가 (0,0)이라고 가정할 때 (3000, 4000)의 좌표로 직선 보간 이동을 수행합니다. 벡터 속도를 1000 으로 지정하였으므로 각 축의 속도를 계산해본다면

$$V_X = \frac{D_X \times V}{\sqrt{D_X^2 + D_Y^2}} = \frac{3000 \times 1000}{\sqrt{3000^2 + 4000^2}} = 600$$

$$V_Y = \frac{D_Y \times V}{\sqrt{D_X^2 + D_Y^2}} = \frac{4000 \times 1000}{\sqrt{3000^2 + 4000^2}} = 800$$

과 같이 됩니다.

예 제

본 예제는 x 축과 y 축을 그룹화하여 (10000, 20000)의 상대좌표로의 직선보간 이동을 수행하는 예제입니다.

```
Call ComiLxAx1.LoadDivice

Const X_MASK = 1
Const Y_MASK = 2
Const Z_MASK = 4
Const U_MASK = 8
```

ur

```
Const MAP0 = 0

` Map X&Y axis to MAP0
Call ComiLxAx1.McMapAxes(MAP0, X_MASK or Y_MASK)
` Set speed mode as Trapezoidal
Call ComiLxAx1.McSetSpeedModeMx(MAP0, 1)
` Set speed & accel => V=5000, Acc=8000
Call ComiLxAx1.McSetSpeedMx(MAP0, 5000, 8000)
` Move to relative coord. (10000, 20000)
Call ComiLxAx1.McLine(MAP0, DistList(0))

Call ComiLxAx1.UnloadDevice
```

■ McStartLine

메소드 원형

```
Sub McStartLine(ByVal MapIndex As Long, ByRef DistList[] As Double)
```

메소드 설명

이 메소드는 현재 위치로부터의 상대 좌표로의 직선 보간 이동을 수행합니다.

매개 변수

▶ **MapIndex** : 축 그룹 인덱스. 이 값은 0 또는 1 이어야 하며 McMapAxes 메소드를 통하여 제어 대상축들이 맵핑되어 있어야 합니다.

▶ **DistList** : 현재 위치로부터의 상대적인 이동 좌표값(각 축의 이동 거리값)의 배열. 이 배열의 크기는 McMapAxes 메소드를 통하여 맵핑된 축의 수와 일치하여야 합니다. 거리에 대한 단위는 McSetUnitDistance 메소드에 의해 결정됩니다. McSetUnitDistance 메소드로 거리의 단위를 변경하지 않았다면 거리의 단위는 Pulse 수가 됩니다. 즉, 거리값 1 은 1 Pulse 출력을 의미합니다.

참 고

□ McLine 메소드가 모션이 완료될 때까지 Return 되지 않는데 반하여, 이 메소드는 지정한 모션을 시작시킨 후에 바로 Return 하게 됩니다.

□ McStartLineTo 메소드가 절대좌표로의 직선 보간 이동을 수행하는데 반하여, 이 메소드는 현재 위치에서 상대적인 거리를 파라미터로하여 직선 보간 이동을 수행합니다.

예 제

■ 예제 1

본 예제는 x 축과 y 축을 그룹화하여 (10000, 20000)의 상대좌표로의 직선보간 이동을 수행하는 예제입니다.

```
Call ComiLxAx1.LoadDivice

Const X_MASK = 1
Const Y_MASK = 2
Const Z_MASK = 4
Const U_MASK = 8

Const MAP0 = 0

Dim DistList(1)
DistList(0)=10000
DistList(1)=20000

` Map X&Y axis to MAP0
Call ComiLxAx1.McMapAxes(MAP0, X_MASK or Y_MASK)
` Set speed mode as Trapezoidal
Call ComiLxAx1.McSetSpeedModeMx(MAP0, 1)
` Set speed & accel => V=5000, Acc=8000
Call ComiLxAx1.McSetSpeedMx(MAP0, 5000, 8000)
```

ur

```
` Move to relative coord. (10000, 20000)
Call ComiLxAx1.McStartLine(MAP0, DistList(0))
` Coordinated Motion 이 완료될때까지 기다린다.
While Not ( ComiLxAx1.McMxDone(MAP0))
Wend

Call ComiLxAx1.UnloadDevice
```

■ McLine

메소드 원형

```
Sub McLine (ByVal MapIndex As Long, ByRef DistList[] As Double)
```

메소드 설명

이 메소드는 현재 위치로부터의 상대 좌표로의 직선 보간 이동을 수행합니다.

매개 변수

▶ **MapIndex** : 축 그룹 인덱스. 이 값은 0 또는 1 이어야 하며 McMapAxes 메소드를 통하여 제어 대상축들이 맵핑되어 있어야 합니다.

▶ **DistList** : 현재 위치로부터의 상대적인 이동 좌표값(각 축의 이동 거리값)의 배열. 이 배열의 크기는 McMapAxes 메소드를 통하여 맵핑된 축의 수와 일치하여야 합니다. 거리에 대한 단위는 McSetUnitDistance 메소드에 의해 결정됩니다. McSetUnitDistance 메소드로 거리의 단위를 변경하지 않았다면 거리의 단위는 Pulse 수가 됩니다. 즉, 거리값 1 은 1 Pulse 출력을 의미합니다.

참 고

□ McStartLine 메소드가 모션이 완료되는 것을 기다리지 않고 바로 Return 하는데 반하여, 이 메소드는 지정한 상대좌표로의 이동이 완료되기 전까지 Return 되지 않고 루프를 돌게 됩니다. 루프를 도는 동안 윈도우 이벤트나 메시지가 처리될 수 있도록 하려면 이 메소드를 수행하기 이전에 McSetBlockingMode 메소드를 사용하여 Blocking 이 일어나지 않도록 설정하여야 합니다.

□ McLineTo 메소드가 절대좌표로의 직선 보간 이동을 수행하는데 반하여, 이 메소드는 현재 위치에서 상대적인 거리를 파라미터로하여 직선 보간 이동을 수행합니다.

예 제

본 예제는 x 축과 y 축을 그룹화하여 (10000, 20000)의 상대좌표로의 직선보간 이동을 수행하는 예제입니다.

```
Call ComiLxAxis1.LoadDevice
```

```
Const X_MASK = 1
Const Y_MASK = 2
Const Z_MASK = 4
Const U_MASK = 8
```

```
Const MAP0 = 0
```

```
Dim DistList(1)
DistList(0)=10000
DistList(1)=20000
```

```
` Map X&Y axis to MAP0
```

ur

```
Call ComiLxAx1.McMapAxes(MAP0, X_MASK or Y_MASK)
` Set speed mode as Trapezoidal
Call ComiLxAx1.McSetSpeedModeMx(MAP0, 1)
` Set speed & accel => V=5000, Acc=8000
Call ComiLxAx1.McSetSpeedMx(MAP0, 5000, 8000)
` Move to relative coord. (10000, 20000)
Call ComiLxAx1.McLine(MAP0, DistList(0))

Call ComiLxAx1.UnloadDevice
```

■ McStartLineTo

메소드 원형

Sub **McStartLineTo**(ByVal MapIndex As Long, ByRef PosList[] As Double)

메소드 설명

이 메소드는 절대 좌표로의 직선 보간 이동을 수행합니다.

매개 변수

▶ **MapIndex** : 축 그룹 인덱스. 이 값은 0 또는 1 이어야 하며 McMapAxes 메소드를 통하여 제어 대상축들이 맵핑되어 있어야 합니다.

▶ **PosList** : 이동할 목표 절대좌표값(각 축의 절대좌표값)의 배열. 이 배열의 크기는 McMapAxes 메소드를 통하여 맵핑된 축의 수와 일치하여야 합니다. 거리에 대한 단위는 McSetUnitDistance 메소드에 의해 결정됩니다. McSetUnitDistance 메소드로 거리의 단위를 변경하지 않았다면 거리의 단위는 Pulse 수가 됩니다. 즉, 거리값 1 은 1 Pulse 출력을 의미합니다.

참 고

□ McLineTo 메소드가 모션이 완료될 때까지 Return 되지 않는데 반하여, 이 메소드는 지정한 모션을 시작시킨 후에 바로 Return 하게 됩니다.

□ McStartLine 메소드가 상대좌표로의 직선 보간 이동을 수행하는데 반하여, 이 메소드는 절대 좌표로의 직선 보간 이동을 수행합니다.

예 제

본 예제는 x 축과 y 축을 그룹화하여 (10000, 20000)의 상대좌표로의 직선보간 이동을 수행하는 예제입니다.

```
Call ComiLxAxis1.LoadDevice

Const X_MASK = 1
Const Y_MASK = 2
Const Z_MASK = 4
Const U_MASK = 8

Const MAP0 = 0

Dim PosList(1)
PosList(0) = 10000
PosList(1) = 20000

' Map X&Y axis to MAP0
Call ComiLxAxis1.McMapAxes(MAP0, X_MASK or Y_MASK)
' Set speed mode as Trapezoidal
Call ComiLxAxis1.McSetSpeedModeMx(MAP0, 1)
' Set speed & accel => V=5000, Acc=8000
Call ComiLxAxis1.McSetSpeedMx(MAP0, 5000, 8000)
' Move to absolute coord. (10000, 20000)
```

```

ur

```

```

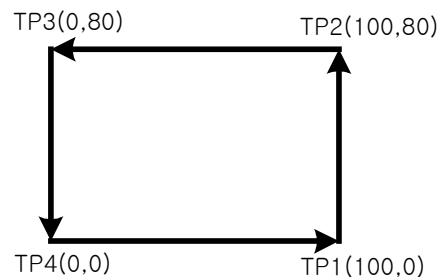
Call ComiLxAx1.McStartLineTo(MAP0, PosList(0))
` Coordinated Motion 이 완료될때까지 기다린다.
While Not (ComiLxAx1.McMxDone(MAP0))
Wend

Call ComiLxAx1.UnloadDevice

```

예제 2

본 예제는 x 축과 y 축을 그룹화하여 아래 그림과 같이 Coordinated Motion 을 수행하는 예제입니다. 그리고 1 회전에 필요한 펄스수가 3600 펄스라 가정하여 거리의 단위를 각도(1°)로, 속도의 단위를 rpm 으로 설정합니다.



```

Call ComiLxAx1.LoadDivice

Const X_AXIS = 0
Const Y_AXIS = 1

Const X_MASK = 1
Const Y_MASK = 2
Const Z_MASK = 4
Const U_MASK = 8

Const MAP0 = 0

Dim PosList(1)

` x 축과 y 축에 대하여 논리거리 및 논리속도 정의
Call ComiLxAx1.McSetUnitDistance(X_AXIS, 10)
Call ComiLxAx1.McSetUnitDistance(Y_AXIS, 10)
Call ComiLxAx1.McSetUnitSpeed(X_AXIS, 3600./60)
Call ComiLxAx1.McSetUnitSpeed(Y_AXIS, 3600./60)

` X&Y 축의 Command Position 의 현재 좌표를 0 으로 초기화한다.
Call ComiLxAx1.McSetPosition_C (X_AXIS, 0)
Call ComiLxAx1.McSetPosition_C (Y_AXIS, 0)

` Map X&Y axis to MAP0
Call ComiLxAx1.McMapAxes(MAP0, X_MASK or Y_MASK)
` Set speed mode as Trapezoidal
Call ComiLxAx1.McSetSpeedModeMx(MAP0, 1)
` Set speed & accel => V=60(RPM), Acc=100(RPM/SEC)
Call ComiLxAx1.McSetSpeedMx(MAP0, 60, 100)

` Move to (100,0)

```



```

PosList(0) = 100
PosList(1) = 0

Call ComiLxAx1.McStartLineTo(MAP0, PosList(0))
While Not ( ComiLxAx1.McMxDone(MAP0))
Wend

` Move to (100,80)
PosList(0) = 100
PosList(1) = 80

Call ComiLxAx1.McStartLineTo(MAP0, PosList(0))
While Not ( ComiLxAx1.McMxDone(MAP0))
Wend

` Move to (0,80)
PosList(0) = 0
PosList(1) = 80
Call ComiLxAx1.McStartLineTo(MAP0, PosList(0))
While Not (ComiLxAx1.McMxDone(MAP0))
Wend

` Move to (0,0)
PosList(0) = 0
PosList(1) = 0
Call ComiLxAx1.McStartLineTo(MAP0, PosList(0))
While Not ( ComiLxAx1.McMxDone(MAP0))
Wend

Call ComiLxAx1.UnloadDevice

```

■ McLineTo

메소드 원형

Sub **McLineTo**(ByVal MapIndex As Long, ByRef PosList[] As Double)

메소드 설명

이 메소드는 절대 좌표로의 직선 보간 이동을 수행합니다.

매개 변수

▶ **MapIndex** : 축 그룹 인덱스. 이 값은 0 또는 1 이어야 하며 McMapAxes 메소드를 통하여 제어 대상축들이 맵핑되어 있어야 합니다.

▶ **PosList** : 이동할 목표 절대좌표값(각 축의 절대좌표값)의 배열 이 배열의 크기는 McMapAxes 메소드를 통하여 맵핑된 축의 수와 일치하여야 합니다. 거리에 대한 단위는 McSetUnitDistance 메소드에 의해 결정됩니다. McSetUnitDistance 메소드로 거리의 단위를 변경하지 않았다면 거리의 단위는 Pulse 수가 됩니다. 즉, 거리값 1 은 1 Pulse 출력을 의미합니다.

참 고

□ McStartLineTo 메소드가 모션이 완료되는 것을 기다리지 않고 바로 Return 하는데 반하여, 이 메소드는 지정한 상대좌표로의 이동이 완료되기 전까지 Return 되지 않고 루프를 돌게 됩니다. 루프를 도는 동안 윈도우 이벤트나 메시지가 처리될 수 있도록 하려면 이 메소드를 수행하기 이전에 McSetBlockingMode 메소드를 사용하여 Blocking 이 일어나지 않도록 설정하여야 합니다.

□ McLine 메소드가 상대좌표로의 직선 보간 이동을 수행하는데 반하여, 이 메소드는 절대 좌표로의 직선 보간 이동을 수행합니다.

예 제

본 예제는 x 축과 y 축을 그룹화하여 (10000, 20000)의 상대좌표로의 직선보간 이동을 수행하는 예제입니다.

```
Call ComiLxAx1.LoadDivice

Const X_MASK = 1
Const Y_MASK = 2
Const Z_MASK = 4
Const U_MASK = 8

Const MAP0 = 0

Dim PosList(1)
PosList(0) = 10000
PosList(1) = 20000

` Map X&Y axis to MAP0
Call ComiLxAx1.McMapAxes(MAP0, X_MASK or Y_MASK)
```

```

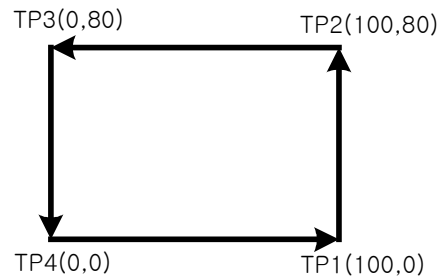
` Set speed mode as Trapezoidal
Call ComiLxAx1.McSetSpeedModeMx(MAP0, 1)
` Set speed & accel => V=5000, Acc=8000
Call ComiLxAx1.McSetSpeedMx(MAP0, 5000, 8000)
` Move to absolute coord. (10000, 20000)
Call ComiLxAx1.McStartLineTo(MAP0, PosList(0))
` Coordinated Motion 이 완료될때까지 기다린다.
While Not ( ComiLxAx1.McMxDone(MAP0))
Wend

Call ComiLxAx1.UnloadDevice

```

▣ 예제 2

본 예제는 x 축과 y 축을 그룹화하여 아래 그림과 같이 Coordinated Motion 을 수행하는 예제입니다. 그리고 1 회전에 필요한 펄스수가 3600 펄스라 가정하여 거리의 단위를 각도(1°)로, 속도의 단위를 rpm으로 설정합니다.



```

Call ComiLxAx1.LoadDivice

Const X_AXIS = 0
Const Y_AXIS = 1

Const X_MASK = 1
Const Y_MASK = 2
Const Z_MASK = 4
Const U_MASK = 8

Const MAP0 = 0

Dim PosList(1)

` x 축과 y 축에 대하여 논리거리 및 논리속도 정의
Call ComiLxAx1.McSetUnitDistance(X_AXIS, 10)
Call ComiLxAx1.McSetUnitDistance(Y_AXIS, 10)
Call ComiLxAx1.McSetUnitSpeed(X_AXIS, 3600./60)
Call ComiLxAx1.McSetUnitSpeed(Y_AXIS, 3600./60)

` X&Y 축의 Command Position 의 현재 좌표를 0 으로 초기화한다.
Call ComiLxAx1.McSetPosition_C (X_AXIS, 0)
Call ComiLxAx1.McSetPosition_C (Y_AXIS, 0)

` Map X&Y axis to MAP0
Call ComiLxAx1.McMapAxes(MAP0, X_MASK or Y_MASK)
` Set speed mode as Trapezoidal
Call ComiLxAx1.McSetSpeedModeMx(MAP0, 1)
` Set speed & accel => V=60(RPM), Acc=100(RPM/SEC)
Call ComiLxAx1.McSetSpeedMx(MAP0, 60, 100)

```

ur

```
Move to (100,0)
PosList(0) = 100
PosList(1) = 0
Call ComiLxAx1.McLineTo(MAP0, PosList(0))

` Move to (100,80)
PosList(0) = 100
PosList(1) = 80
Call ComiLxAx1.McLineTo(MAP0, PosList(0))

` Move to (0,80)
PosList(0) = 0
PosList(1) = 80
Call ComiLxAx1.McLineTo(MAP0, PosList(0))

` Move to (0,0)
PosList(0) = 0
PosList(1) = 0
Call ComiLxAx1.McLineTo(MAP0, PosList(0))

Call ComiLxAx1.UnloadDevice
```

■ McStartArcA

메소드 원형

```
Sub McStartArcA(ByVal MapIndex As Long, ByVal XcentOffset As Double, ByVal YcentOffset As Double, ByVal EndAngle As Double)
```

메소드 설명

이 메소드는 상대좌표를 파라미터로 하여 원호보간 이동을 수행합니다. 이 메소드는 End PoByVal 에 대한 정보를 각도값으로 설정합니다.

매개 변수

- ▶ **MapIndex** : 축 그룹 인덱스. 이 값은 0 또는 1 이어야 하며 McMapAxes 메소드를 통하여 제어 대상축들이 맵핑되어 있어야 합니다.
- ▶ **XCentOffset** : 현재 위치(시작 위치)로부터 원의 중심까지 X 축상 상대 좌표
- ▶ **YCentOffset** : 현재 위치(시작 위치)로부터 원의 중심까지 Y 축상 상대 좌표
- ▶ **EndAngle** : 원호보간 이동을 완료할 목표지점의 현재 위치에 대한 각도값을 Degree(°)값으로 지정합니다. 각도의 부호가 (+)이면 반시계방향, (-)이면 시계방향으로의 이동을 의미합니다.

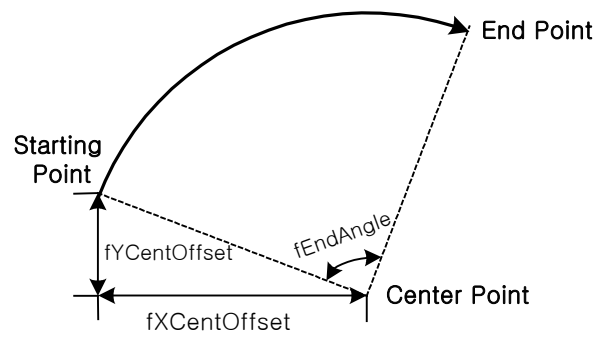
참 고

□ 원호보간 이동은 두 축에 대해서만 적용가능합니다. 따라서 본 단락에서는 맵핑된 두 축을 X, Y 축으로 간주하여 설명합니다. 여기서 X 축이라 함은 맵핑된 두 축 중에서 채널번호(X,Y,Z,U 순)가 낮은 축을 의미하며 Y 축은 채널번호가 높은 축을 의미합니다. 예를 들어 Z 축과 U 축이 맵핑된 두 축이라면 Z 축이 X 축에 해당하며 U 축이 Y 축에 해당합니다.

□ 이 메소드는 원호 보간 이동을 시작시킨후 바로 Return 합니다.

□ McStartArcP 메소드가 원호 보간 이동을 완료할 목표지점의 좌표값을 파라미터로 사용하는데 반하여 이 메소드는 각도값을 파라미터로 사용합니다. 사용자는 편의에 따라 McStartArcP 나 McStartArcA 메소드중에 하나를 사용할 수 있습니다.

□ McStartArcA 메소드를 사용하여 원호보간 이동을 수행할 때 각 파라미터의 의미는 [그림 #]과 같습니다.



[그림 #] McStartArcA 메소드를 사용한 원호 보간 이동

■ McArcA

메소드 원형

Sub **McArcA**(ByVal MapIndex As Long, ByVal XcentOffset As Double, ByVal YcentOffset As Double, ByVal EndAngle As Double)

메소드 설명

이 메소드는 상대좌표를 파라미터로 하여 원호보간 이동을 수행합니다. 이 메소드는 End PoByVal 에 대한 정보를 각도값으로 설정합니다.

매개 변수

- ▶ **MapIndex** : 축 그룹 인덱스. 이 값은 0 또는 1 이어야 하며 McMapAxes 메소드를 통하여 제어 대상축들이 맵핑되어 있어야 합니다.
- ▶ **XCentOffset** : 현재 위치(시작 위치)로부터 원의 중심까지 X 축상 상대 좌표
- ▶ **YCentOffset** : 현재 위치(시작 위치)로부터 원의 중심까지 Y 축상 상대 좌표
- ▶ **EndAngle** : 원호보간 이동을 완료할 목표지점의 현재 위치에 대한 각도값을 Degree(°)값으로 지정합니다. 각도의 부호가 (+)이면 반시계방향, (-)이면 시계방향으로의 이동을 의미합니다.

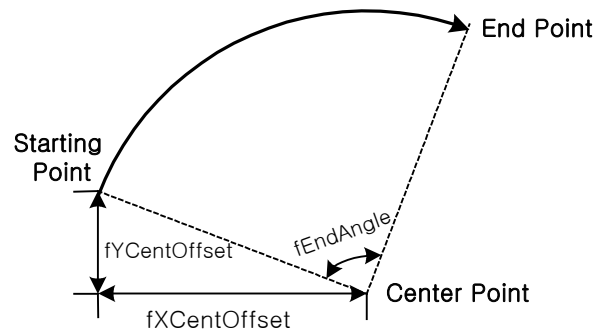
참 고

□ 원호보간 이동은 두 축에 대해서만 적용가능합니다. 따라서 본 단락에서는 맵핑된 두 축을 X, Y 축으로 간주하여 설명합니다. 여기서 X 축이라 함은 맵핑된 두 축 중에서 채널번호(X,Y,Z,U 순)가 낮은 축을 의미하며 Y 축은 채널번호가 높은 축을 의미합니다. 예를 들어 Z 축과 U 축이 맵핑된 두 축이라면 Z 축이 X 축에 해당하며 U 축이 Y 축에 해당합니다.

□ 이 메소드는 원호 보간 이동이 완료되기 전까지 Return 되지 않고 루프를 돌게 됩니다. 루프를 도는 동안 윈도우 이벤트나 메시지가 처리될 수 있도록 하려면 이 메소드를 수행하기 이전에 McSetBlockingMode 메소드를 사용하여 Blocking 이 일어나지 않도록 설정하여야 합니다.

□ McArcP 메소드가 원호 보간 이동을 완료할 목표지점의 좌표값을 파라미터로 사용하는데 반하여 이 메소드는 각도값을 파라미터로 사용합니다. 사용자는 편의에 따라 McArcP 나 McArcA 메소드중에 하나를 사용할 수 있습니다.

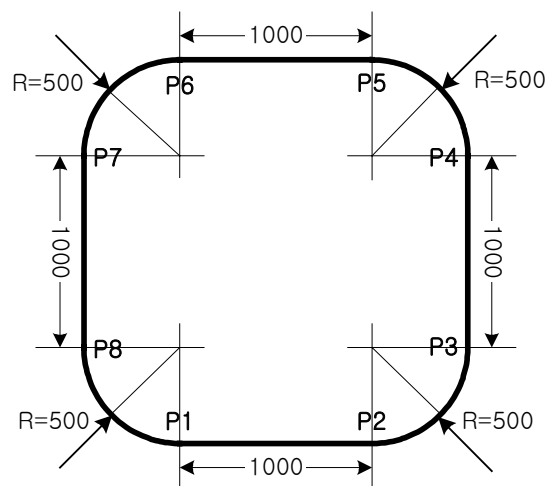
□ McArcA 메소드를 사용하여 원호보간 이동을 수행할 때 각 파라미터의 의미는 [그림 #]과 같습니다.



[그림 #] McArcA 메소드를 사용한 원호 보간 이동

예 제

본 예제는 아래 그림과 같이 직선보간 이동과 원호보간 이동을 조합하는 Coordinated Motion 을 수행하는 예제입니다. p1 점으로부터 출발하여 p8 점을 거쳐 다시 p1 으로 복귀하는 작업입니다.



```
Call ComiLxAx1.LoadDivice

Const X_MASK = 1
Const Y_MASK = 2
Const Z_MASK = 4
Const U_MASK = 8

Const MAP0 = 0

Dim DistList(1)

` Map X&Y axis to MAP0
Call ComiLxAx1.McMapAxes(MAP0, X_MASK or Y_MASK)
` Set speed mode as Trapezoidal
Call ComiLxAx1.McSetSpeedModeMx(MAP0, 1)
` Set speed & accel => V=500, Acc=500
Call ComiLxAx1.McSetSpeedMx(MAP0, 500, 500)
` Move from P1 to P2
```



```

DistList(0) = 1000
DistList(1) = 0
Call ComiLxAx1.McLine(MAP0, DistList(0))

` Move from P2 to P3
Call ComiLxAx1.McArcA(MAP0, 0, 500, 90)
` Move from P3 to P4
DistList(0) = 0
DistList(1) = 1000
Call ComiLxAx1.McLine(MAP0, DistList(0))
` Move from P4 to P5
Call ComiLxAx1.McArcA(MAP0, -500, 0, 90)
` Move from P5 to P6
DistList(0) = -1000
DistList(1) = 0
Call ComiLxAx1.McLine(MAP0, DistList(0))
` Move from P6 to P7
Call ComiLxAx1.McArcA(MAP0, 0, -500, 90)
` Move from P7 to P8

DistList(0) = 0
DistList(1) = -1000

Call ComiLxAx1.McLine(MAP0, DistList(0))

` Move from P8 to P1
Call ComiLxAx1.McArcA(MAP0, 500, 0, 90)

Call ComiLxAx1.UnloadDevice

```

■ McStartArcP

메소드 원형

```
Sub McStartArcP(ByVal MapIndex As Long, ByVal XCentOffset As Double, ByVal YCentOffset As Double, ByVal XEndPoByValDist As Double, ByVal YendPoByValDist As Double, ByVal Dir As Long)
```

메소드 설명

이 메소드는 상대좌표를 파라미터로 하여 원호보간 이동을 수행합니다. 이 메소드는 End PoByVal 에 대한 정보를 상대좌표값으로 설정합니다.

매개 변수

- ▶ **MapIndex** : 축 그룹 인덱스. 이 값은 0 또는 1 이어야 하며 McMapAxes 메소드를 통하여 제어 대상축들이 맵핑되어 있어야 합니다.
- ▶ **XCentOffset** : 현재 위치(시작 위치)로부터 원의 중심까지 X 축상 상대좌표
- ▶ **YCentOffset** : 현재 위치(시작 위치)로부터 원의 중심까지 Y 축상 상대좌표
- ▶ **XEndPoByValDist** : 원호보간 이동을 완료할 목표지점의 현재 위치로부터 X-축상 거리값.
- ▶ **YEndPoByValDist** : 원호보간 이동을 완료할 목표지점의 현재 위치로부터 Y-축상 거리값.
- ▶ **Dir** : 회전 방향을 지정합니다.

Value	Meaning
0 또는 음수	시계 방향(CW)으로 회전
양수	반시계 방향(CCW)으로 회전

참 고

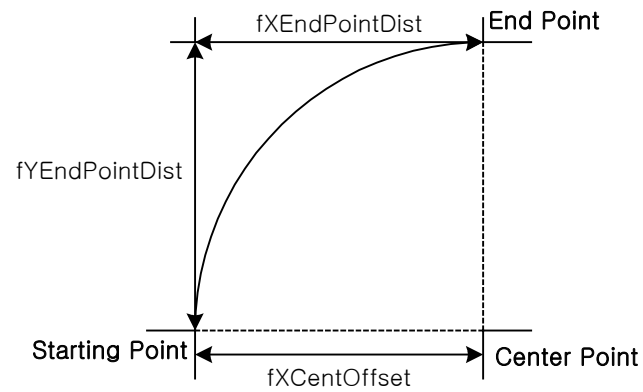
□ 원호보간 이동은 두 축에 대해서만 적용가능합니다. 따라서 본 단락에서는 맵핑된 두 축을 X, Y 축으로 간주하여 설명합니다. 여기서 X 축이라 함은 맵핑된 두 축 중에서 채널번호(X,Y,Z,U 순)가 낮은 축을 의미하며 Y 축은 채널번호가 높은 축을 의미합니다. 예를 들어 Z 축과 U 축이 맵핑된 두 축이라면 Z 축이 X 축에 해당하며 U 축이 Y 축에 해당합니다.

□ 이 메소드는 원호 보간 이동을 시작시킨후 바로 Return 합니다.

□ McStartArcA 메소드가 원호 보간 이동을 완료할 목표지점의 각도를 파라미터로 사용하는데 반하여 이 메소드는 상대 좌표값을 파라미터로 사용합니다. 사용자는 편의에 따라 McStartArcP 나 McStartArcA 메소드중에 하나를 사용할 수 있습니다.

□ XEndPoByValDist 값과 YEndPoByValDist 값이 모두 0 으로 지정되면 완전한 원을 그리게 됩니다.

□ McStartArcP 메소드를 사용하여 원호보간 이동을 수행할 때 각 파라미터의 의미는 [그림 #]과 같습니다.



[그림 #] McStartArcP 메소드를 사용한 원호 보간 이동

■ McArcP

메소드 원형

Sub **McArcP**(ByVal MapIndex As Long, ByVal XcentOffset As Double, ByVal YcentOffset As Double, ByVal XendPoByValDist As Double, ByVal YendPoByValDist As Double, ByVal Dir As Long)

메소드 설명

이 메소드는 상대좌표를 파라미터로 하여 원호보간 이동을 수행합니다. 이 메소드는 End PoByVal 에 대한 정보를 상대좌표값으로 설정합니다.

매개 변수

- ▶ **MapIndex** : 축 그룹 인덱스. 이 값은 0 또는 1 이어야 하며 McMapAxes 메소드를 통하여 제어 대상축들이 맵핑되어 있어야 합니다.
- ▶ **XCenOffset** : 현재 위치(시작 위치)로부터 원의 중심까지 X 축상 상대좌표
- ▶ **YCentOffset** : 현재 위치(시작 위치)로부터 원의 중심까지 Y 축상 상대좌표
- ▶ **XEndPoByValDist** : 원호보간 이동을 완료할 목표지점의 현재 위치로부터 X-축상 거리값.
- ▶ **YEndPoByValDist** : 원호보간 이동을 완료할 목표지점의 현재 위치로부터 Y-축상 거리값.
- ▶ **Dir** : 회전 방향을 지정합니다.

Value	Meaning
0 또는 음수	시계 방향(CW)으로 회전
양수	반시계 방향(CCW)으로 회전

참 고

□ 원호보간 이동은 두 축에 대해서만 적용가능합니다. 따라서 본 단락에서는 맵핑된 두 축을 X, Y 축으로 간주하여 설명합니다. 여기서 X 축이라 함은 맵핑된 두 축 중에서 채널번호(X,Y,Z,U 순)가 낮은 축을 의미하며 Y 축은 채널번호가 높은 축을 의미합니다. 예를 들어 Z 축과 U 축이 맵핑된 두 축이라면 Z 축이 X 축에 해당하며 U 축이 Y 축에 해당합니다.

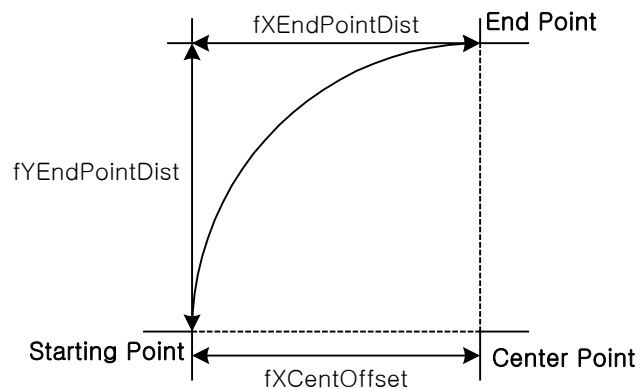
□ 이 메소드는 원호 보간 이동이 완료되기 전까지 Return 되지 않고 루프를 돌게 됩니다. 루프를 도는 동안 윈도우 이벤트나 메시지가 처리될 수 있도록 하려면 이 메소드를 수행하기 이전에 McSetBlockingMode 메소드를 사용하여 Blocking 이 일어나지 않도록 설정하여야 합니다.

□ XEndPoByValDist 값과 YEndPoByValDist 값이 모두 0 으로 지정되면 완전한 원을 그리게

됩니다.

□ McStartArcA 메소드가 원호 보간 이동을 완료할 목표지점의 각도를 파라미터로 사용하는데 반하여 이 메소드는 상대 좌표값을 파라미터로 사용합니다. 사용자는 편의에 따라 McStartArcP 나 McStartArcA 메소드중에 하나를 사용할 수 있습니다.

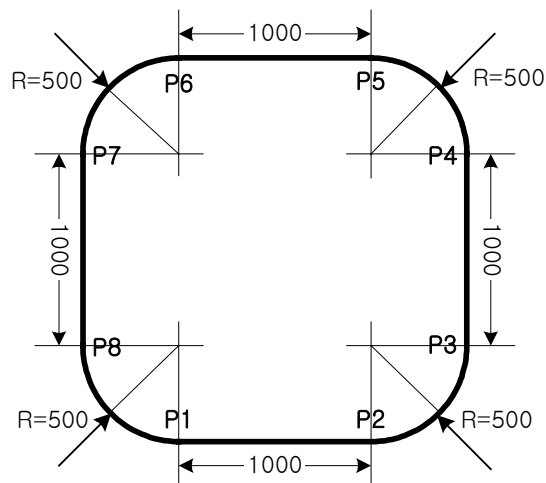
□ McStartArcP 메소드를 사용하여 원호보간 이동을 수행할 때 각 파라미터의 의미는 [그림 #]과 같습니다.



[그림 #] McStartArcP 메소드를 사용한 원호 보간 이동

예 제

본 예제는 아래 그림과 같이 직선보간 이동과 원호보간 이동을 조합하는 Coordinated Motion 을 수행하는 예제입니다. P1 점으로부터 출발하여 P8 점을 거쳐 다시 P1 으로 복귀하는 작업입니다.



```
Call ComiLxAx1.LoadDivice
```

```
Const X_MASK = 1
```

```
Const Y_MASK = 2
```

or

```

Const Z_MASK = 4
Const U_MASK = 8

Const MAP0 = 0

Dim DistList(2)

` Map X&Y axis to MAP0
Call ComiLxAx1.McMapAxes(MAP0, X_MASK or Y_MASK)
` Set speed mode as Trapezoidal
Call ComiLxAx1.McSetSpeedModeMx(MAP0, 1)
` Set speed & accel => V=500, Acc=500
Call ComiLxAx1.McSetSpeedMx(MAP0, 500, 500)

` Move from P1 to P2
DistList(0) = 1000
DistList(1) = 0
Call ComiLxAx1.McLine(MAP0, DistList(0))
` Move from P2 to P3
Call ComiLxAx1.McArcP(MAP0, 0, 500, 500, 500)

` Move from P3 to P4
DistList(0) = 0
DistList(1) = 1000
Call ComiLxAx1.McLine(MAP0, DistList(0))
` Move from P4 to P5
Call ComiLxAx1.McArcP(MAP0, -500, 0, -500, 500)
` Move from P5 to P6
DistList(0) = -1000
DistList(1) = 0
Call ComiLxAx1.McLine(MAP0, DistList(0))
` Move from P6 to P7
Call ComiLxAx1.McArcP(MAP0, 0, -500, -500, -500)
` Move from P7 to P8
DistList(0) = 0
DistList(1) = -1000
Call ComiLxAx1.McLine(MAP0, DistList(0))
` Move from P8 to P1
Call ComiLxAx1.McArcP(MAP0, 500, 0, 500, -500)

Call ComiLxAx1.UnloadDevice
    
```

■ McStartArcToA

메소드 원형

```
Sub McStartArcToA(ByVal MapIndex As Long, ByVal Xcent As Double, ByVal Ycent As Double, ByVal EndAngle As Double)
```

메소드 설명

이 메소드는 원호보간 이동을 수행합니다. 이 메소드는 중심점의 좌표값을 절대좌표값으로 설정하며 원호보간 이동의 완료지점에 대한 정보를 각도로 설정합니다.

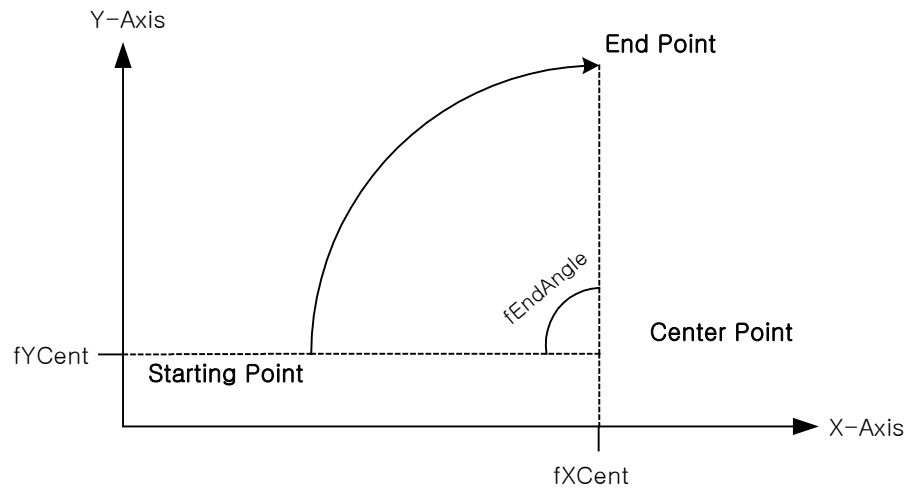
매개 변수

- ▶ **MapIndex** : 축 그룹 인덱스. 이 값은 0 또는 1 이어야 하며 McMapAxes 메소드를 통하여 제어 대상축들이 맵핑되어 있어야 합니다.
- ▶ **XCent** : 중심점의 X 축 절대좌표
- ▶ **YCent** : 중심점의 Y 축 절대좌표
- ▶ **EndAngle** : 원호보간 이동을 완료할 목표지점의 현재 위치에 대한 각도값을 Degree(°)값으로 지정합니다. 각도의 부호가 (+)이면 반시계방향, (-)이면 시계방향으로의 이동을 의미합니다.

참 고

- 원호보간 이동은 두 축에 대해서만 적용가능합니다. 따라서 본 단락에서는 맵핑된 두 축을 X, Y 축으로 간주하여 설명합니다. 여기서 X 축이라 함은 맵핑된 두 축 중에서 채널번호(X,Y,Z,U 순)가 낮은 축을 의미하며 Y 축은 채널번호가 높은 축을 의미합니다. 예를 들어 Z 축과 U 축이 맵핑된 두 축이라면 Z 축이 X 축에 해당하며 U 축이 Y 축에 해당합니다.
- 이 메소드는 원호 보간 이동을 시작시킨후 바로 Return 합니다.
- McStartArcToP 메소드가 원호 보간 이동을 완료할 목표지점의 좌표값을 파라미터로 사용하는데 반하여 이 메소드는 각도값을 파라미터로 사용합니다. 사용자는 편의에 따라 McStartArcP 나 McStartArcA 메소드중에 하나를 사용할 수 있습니다.
- McStartArcToA 메소드를 사용하여 원호보간 이동을 수행할 때 각 파라미터의 의미는 [그림 #]과 같습니다.

UI



[그림 #] McStartArcToA 메소드를 사용한 원호 보간 이동

■ McArcToA

메소드 원형

Sub **McArcToA**(ByVal MapIndex As Long, ByVal Xcent As Double, ByVal Ycent As Double, ByVal EndAngle As Double)

메소드 설명

이 메소드는 원호보간 이동을 수행합니다. 이 메소드는 중심점의 좌표값을 절대좌표값으로 설정하며 원호보간 이동의 완료지점에 대한 정보를 각도로 설정합니다.

매개 변수

- ▶ **MapIndex** : 축 그룹 인덱스. 이 값은 0 또는 1 이어야 하며 McMapAxes 메소드를 통하여 제어 대상축들이 맵핑되어 있어야 합니다.
- ▶ **XCent** : 중심점의 X 축 절대좌표
- ▶ **YCent** : 중심점의 Y 축 절대좌표
- ▶ **EndAngle** : 원호보간 이동을 완료할 목표지점의 현재 위치에 대한 각도값을 Degree(°)값으로 지정합니다. 각도의 부호가 (+)이면 반시계방향, (-)이면 시계방향으로의 이동을 의미합니다.

참 고

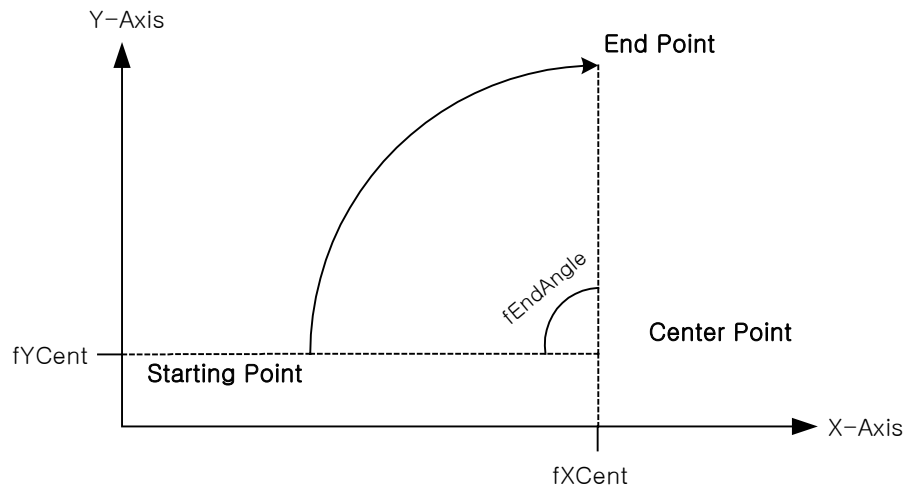
□ 원호보간 이동은 두 축에 대해서만 적용가능합니다. 따라서 본 단락에서는 맵핑된 두 축을 X, Y 축으로 간주하여 설명합니다. 여기서 X 축이라 함은 맵핑된 두 축 중에서 채널번호(X,Y,Z,U 순)가 낮은 축을 의미하며 Y 축은 채널번호가 높은 축을 의미합니다. 예를 들어 Z 축과 U 축이 맵핑된 두 축이라면 Z 축이 X 축에 해당하며 U 축이 Y 축에 해당합니다.

□ 이 메소드는 원호 보간 이동이 완료되기 전까지 Return 되지 않고 루프를 돌게 됩니다. 루프를 도는 동안 윈도우 이벤트나 메시지가 처리될 수 있도록 하려면 이 메소드를 수행하기 이전에 McSetBlockingMode 메소드를 사용하여 Blocking 이 일어나지 않도록 설정하여야 합니다.

□ McArcToP 메소드가 원호 보간 이동을 완료할 목표지점의 좌표값을 파라미터로 사용하는데 반하여 이 메소드는 각도값을 파라미터로 사용합니다. 사용자는 편의에 따라 McArcP 나 McArcA 메소드중에 하나를 사용할 수 있습니다.

□ McStartArcToA 메소드를 사용하여 원호보간 이동을 수행할 때 각 파라미터의 의미는 [그림 #]과 같습니다.

ur

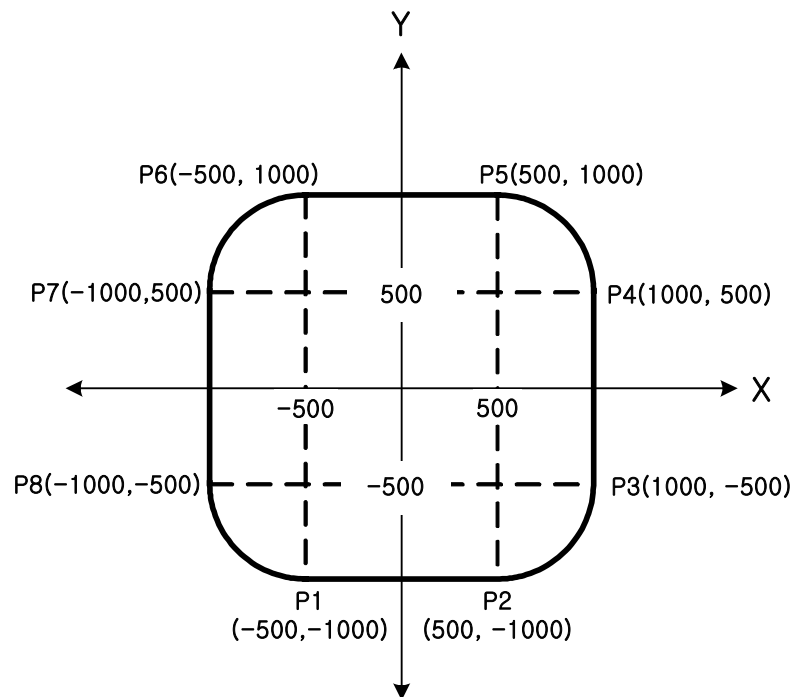


[그림 #] McArcToA 메소드를 사용한 원호 보간 이동

예 제

▣ 예제 1

본 예제는 아래 그림과 같이 직선보간 이동과 원호보간 이동을 조합하는 Coordinated Motion 을 수행하는 예제입니다. p1 점으로부터 출발하여 p8 점을 거쳐 다시 p1 으로 복귀하는 작업입니다. 그리고 현재 위치가 p1 의 위치에 있다고 가정합니다.



```
Call ComiLxAx1.LoadDivice
```

```
Const X_MASK = 1
Const Y_MASK = 2
Const Z_MASK = 4
Const U_MASK = 8
```

```

Const MAP0 = 0

Dim PosList(1)

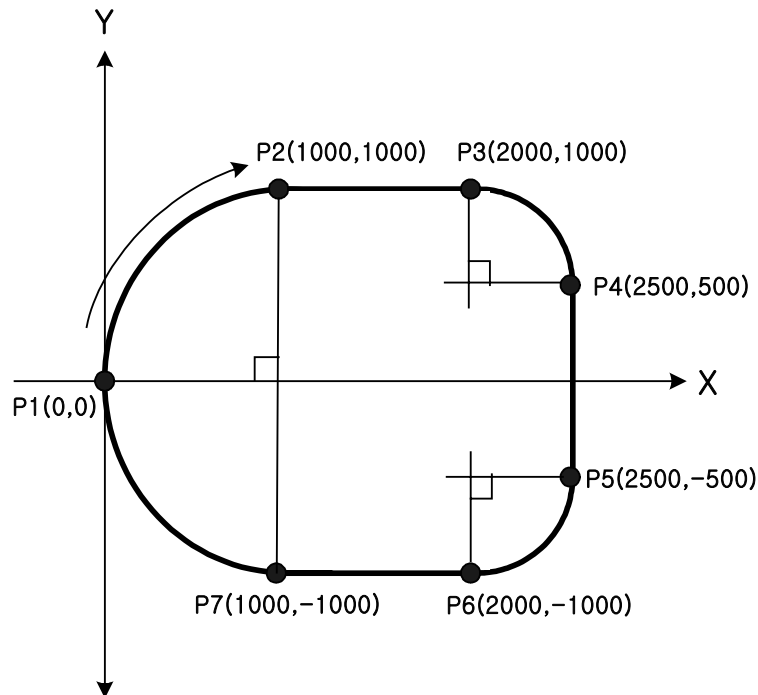
` Map X&Y axis to MAP0
Call ComiLxAx1.McMapAxes(MAP0, X_MASK or Y_MASK)
` Set speed mode as Trapezoidal
Call ComiLxAx1.McSetSpeedModeMx(MAP0, 1)
` Set speed & accel => V=500, Acc=500
Call ComiLxAx1.McSetSpeedMx(MAP0, 500, 500)
` Move from P1 to P2
PosList(0) = 500
PosList(1) = -1000
Call ComiLxAx1.McLineTo(MAP0, PosList(0))
` Move from P2 to P3
Call ComiLxAx1.McArcA(MAP0, 500, -500, 90)
` Move from P3 to P4
PosList(0) = 1000
PosList(1) = 500
Call ComiLxAx1.McLineTo (MAP0, PosList(0))
` Move from P4 to P5
Call ComiLxAx1.McArcA(MAP0, 500, 500, 90)
` Move from P5 to P6
PosList(0) = -500
PosList(1) = 1000
Call ComiLxAx1.McLineTo (MAP0, PosList(0))
` Move from P6 to P7
Call ComiLxAx1.McArcA(MAP0, -500, 500, 90)
` Move from P7 to P8
PosList(0) = -1000
PosList(1) = -500
Call ComiLxAx1.McLineTo (MAP0, PosList(0))
` Move from P8 to P1
Call ComiLxAx1.McArcA(MAP0, -500, -500, 90)

Call ComiLxAx1.UnloadDevice

```

■ 예제 2

본 예제는 아래 그림과 같이 직선보간 이동과 원호보간 이동을 조합하는 Coordinated Motion 을 수행하는 예제입니다. p1 점으로부터 출발하여 p8 점을 거쳐 다시 p1 으로 복귀하는 작업입니다. 그리고 현재 위치가 p1 의 위치에 있다고 가정합니다.

UI


```

Call ComiLxAx1.LoadDivice
Const X_MASK = 1
Const Y_MASK = 2
Const Z_MASK = 4
Const U_MASK = 8
Const MAP0 = 0
Dim PosList(1)
` Map X&Y axis to MAP0
Call ComiLxAx1.McMapAxes(MAP0, X_MASK or Y_MASK)
` Set speed mode as Trapezoidal
Call ComiLxAx1.McSetSpeedModeMx(MAP0, 1)
` Set speed & accel => V=500, Acc=500
Call ComiLxAx1.McSetSpeedMx(MAP0, 500, 500)
` Move from P1 to P2
Call ComiLxAx1.McArcA(MAP0, 1000, 0, 90)
` Move from P2 to P3
PosList(0) = 2000
PosList(1) = 1000
Call ComiLxAx1.McLineTo (MAP0, PosList(0))
` Move from P3 to P4
Call ComiLxAx1.McArcA(MAP0, 2000, 500, 90)
` Move from P4 to P5
PosList(0) = 2500
PosList(1) = -500
Call ComiLxAx1.McLineTo (MAP0, PosList(0))
` Move from P5 to P6
Call ComiLxAx1.McArcA(MAP0, 2000, -500, 90)
` Move from P6 to P7
PosList(0) = 1000
PosList(1) = -1000
Call ComiLxAx1.McLineTo (MAP0, PosList(0))
` Move from P7 to P1
Call ComiLxAx1.McArcA(MAP0, 1000, 0, 90)

Call ComiLxAx1.UnloadDevice
    
```

■ McStartArcToP

메소드 원형

Sub **McStartArcToP**(ByVal MapIndex As Long, ByVal Xcent As Double, ByVal Ycent As Double, ByVal XendPos As Double, ByVal YendPos As Double, ByVal Dir As Long)

메소드 설명

이 메소드는 원호보간 이동을 수행합니다. 이 메소드는 중심점의 좌표값을 절대좌표값으로 설정하며 원호보간 이동의 완료지점(End PoByVal)에 대한 정보 또한 절대좌표값으로 설정합니다.

매개 변수

- ▶ **MapIndex** : 축 그룹 인덱스. 이 값은 0 또는 1 이어야 하며 McMapAxes 메소드를 통하여 제어 대상축들이 맵핑되어 있어야 합니다.
- ▶ **XCent** : 중심점의 X 축 절대좌표값
- ▶ **YCent** : 중심점의 Y 축 절대좌표값
- ▶ **XEndPos** : 원호보간 이동을 완료할 목표지점(End poByVal)의 X 축 절대좌표값
- ▶ **YEndPos** : 원호보간 이동을 완료할 목표지점(End poByVal)의 Y 축 절대좌표값
- ▶ **Dir** : 회전 방향을 지정합니다.

Value	Meaning
0 또는 음수	시계 방향(CW)으로 회전
양수	반시계 방향(CCW)으로 회전

참 고

□ 원호보간 이동은 두 축에 대해서만 적용가능합니다. 따라서 본 단락에서는 맵핑된 두 축을 X, Y 축으로 간주하여 설명합니다. 여기서 X 축이라 함은 맵핑된 두 축 중에서 채널번호(X,Y,Z,U 순)가 낮은 축을 의미하며 Y 축은 채널번호가 높은 축을 의미합니다. 예를 들어 Z 축과 U 축이 맵핑된 두 축이라면 Z 축이 X 축에 해당하며 U 축이 Y 축에 해당합니다.

□ 이 메소드는 원호 보간 이동을 시작시킨후 바로 Return 합니다.

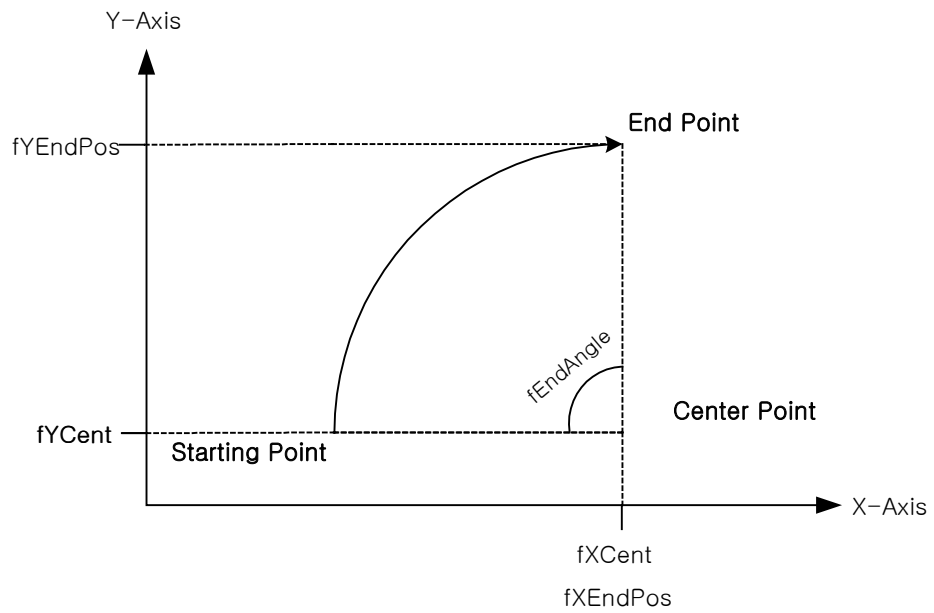
□ XEndPos 값과 YEndPos 값이 현재 위치(Starting PoByVal)의 좌표값과 일치하면 완전한 원을 그리게 됩니다.

□ McStartArcToA 메소드가 원호 보간 이동을 완료할 목표지점의 각도를 파라미터로

ur

사용하는데 반하여 이 메소드는 상대 좌표값을 파라미터로 사용합니다. 사용자는 편의에 따라 `McStartArcToP` 나 `McStartArcToA` 메소드중에 하나를 사용할 수 있습니다.

□ `McStartArcToP` 메소드를 사용하여 원호보간 이동을 수행할 때 각 파라미터의 의미는 [그림 #]과 같습니다.



[그림 #] `McStartArcToP` 메소드를 사용한 원호 보간 이동

■ McArcToP

메소드 원형

Sub **McArcToP**(ByVal MapIndex As Long, ByVal Xcent As Double, ByVal Ycent As Double, ByVal XendPos As Double, ByVal YendPos As Double, ByVal Dir As Long)

메소드 설명

이 메소드는 원호보간 이동을 수행합니다. 이 메소드는 중심점의 좌표값을 절대좌표값으로 설정하며 원호보간 이동의 완료지점(End PoByVal)에 대한 정보 또한 절대좌표값으로 설정합니다.

매개 변수

- ▶ **MapIndex** : 축 그룹 인덱스. 이 값은 0 또는 1 이어야 하며 McMapAxes 메소드를 통하여 제어 대상축들이 맵핑되어 있어야 합니다.
- ▶ **XCent** : 중심점의 X 축 절대좌표값
- ▶ **YCent** : 중심점의 Y 축 절대좌표값
- ▶ **XEndPos** : 원호보간 이동을 완료할 목표지점(End poByVal)의 X 축 절대좌표값
- ▶ **YEndPos** : 원호보간 이동을 완료할 목표지점(End poByVal)의 Y 축 절대좌표값
- ▶ **Dir** : 회전 방향을 지정합니다.

Value	Meaning
0 또는 음수	시계 방향(CW)으로 회전
양수	반시계 방향(CCW)으로 회전

참 고

□ 원호보간 이동은 두 축에 대해서만 적용가능합니다. 따라서 본 단락에서는 맵핑된 두 축을 X, Y 축으로 간주하여 설명합니다. 여기서 X 축이라 함은 맵핑된 두 축 중에서 채널번호(X,Y,Z,U 순)가 낮은 축을 의미하며 Y 축은 채널번호가 높은 축을 의미합니다. 예를 들어 Z 축과 U 축이 맵핑된 두 축이라면 Z 축이 X 축에 해당하며 U 축이 Y 축에 해당합니다.

□ 이 메소드는 원호 보간 이동이 완료되기 전까지 Return 되지 않고 루프를 돌게 됩니다. 루프를 도는 동안 윈도우 이벤트나 메시지가 처리될 수 있도록 하려면 이 메소드를 수행하기 이전에 McSetBlockingMode 메소드를 사용하여 Blocking 이 일어나지 않도록 설정하여야 합니다.

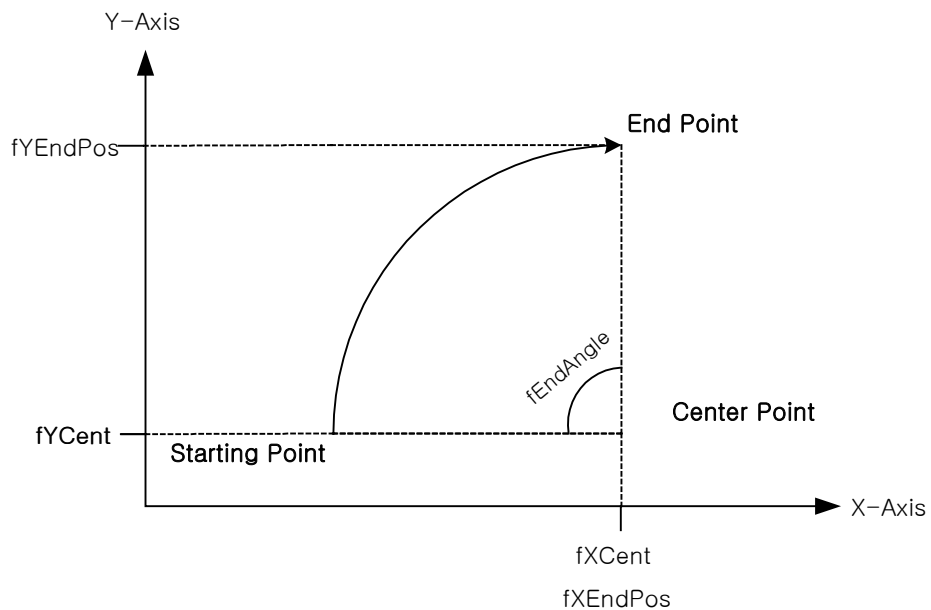
□ XEndPos 값과 YEndPos 값이 현재 위치(Starting PoByVal)의 좌표값과 일치하면 완전한

ur

원을 그리게 됩니다.

□ McArcToA 메소드가 원호 보간 이동을 완료할 목표지점의 각도를 파라미터로 사용하는데 반하여 이 메소드는 상대 좌표값을 파라미터로 사용합니다. 사용자는 편의에 따라 McArcToP 나 McArcToA 메소드중에 하나를 사용할 수 있습니다.

□ McArcToP 메소드를 사용하여 원호보간 이동을 수행할 때 각 파라미터의 의미는 [그림 #]과 같습니다.

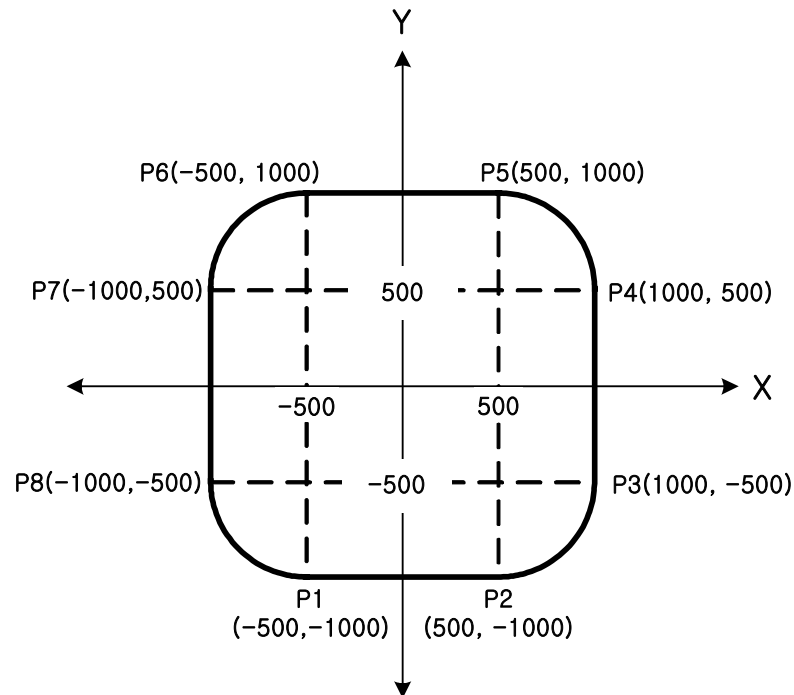


[그림 #] McArcToP 메소드를 사용한 원호 보간 이동

예 제

■ 예제 1

본 예제는 아래 그림과 같이 직선보간 이동과 원호보간 이동을 조합하는 Coordinated Motion 을 수행하는 예제입니다. p1 점으로부터 출발하여 p8 점을 거쳐 다시 p1 으로 복귀하는 작업입니다. 그리고 현재 위치가 p1 의 위치에 있다고 가정합니다.



```

Call ComiLxAx1.LoadDivice

Const X_MASK = 1
Const Y_MASK = 2
Const Z_MASK = 4
Const U_MASK = 8

Const MAP0 = 0

Dim PosList(1)

' Map X&Y axis to MAP0
Call ComiLxAx1.McMapAxes(MAP0, X_MASK or Y_MASK)
' Set speed mode as Trapezoidal
Call ComiLxAx1.McSetSpeedModeMx(MAP0, 1)
' Set speed & accel => V=500, Acc=500
Call ComiLxAx1.McSetSpeedMx(MAP0, 500, 500)
' Move from P1 to P2
PosList(0) = 500
PosList(1) = -1000
Call ComiLxAx1.McLineTo(MAP0, PosList(0))
' Move from P2 to P3
Call ComiLxAx1.McArcP(MAP0, 500, -500, 1000, -500)
' Move from P3 to P4
PosList(0) = 1000
PosList(1) = 500
Call ComiLxAx1.McLineTo(MAP0, PosList(0))
' Move from P4 to P5
Call ComiLxAx1.McArcP(MAP0, 500, 500, 500, 1000)
' Move from P5 to P6
PosList(0) = -500
PosList(1) = 1000
Call ComiLxAx1.McLineTo(MAP0, PosList(0))
' Move from P6 to P7
Call ComiLxAx1.McArcP(MAP0, -500, 500, -1000, 500)
' Move from P7 to P8

```

ur

```

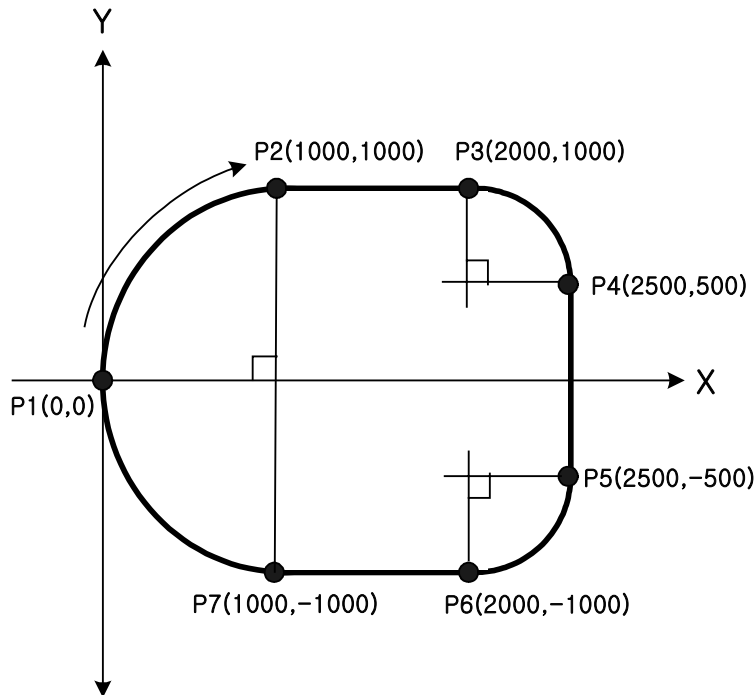
PosList(0) = -1000
PosList(1) = -500
Call ComiLxAx1.McLineTo (MAP0, PosList(0))
` Move from P8 to P1
Call ComiLxAx1.McArcP(MAP0, -500, -500, -500, -1000)

Call ComiLxAx1.UnloadDevice

```

예제 2

본 예제는 아래 그림과 같이 직선보간 이동과 원호보간 이동을 조합하는 Coordinated Motion 을 수행하는 예제입니다. P1 점으로부터 출발하여 P8 점을 거쳐 다시 P1 으로 복귀하는 작업입니다. 그리고 현재 위치가 P1 의 위치에 있다고 가정합니다.



```

Call ComiLxAx1.LoadDivice

Const X_MASK = 1
Const Y_MASK = 2
Const Z_MASK = 4
Const U_MASK = 8

Const MAP0 = 0

Dim PosList(1)

` Map X&Y axis to MAP0
Call ComiLxAx1.McMapAxes(MAP0, X_MASK or Y_MASK)
` Set speed mode as Trapezoidal
Call ComiLxAx1.McSetSpeedModeMx(MAP0, 1)
` Set speed & accel => V=500, Acc=500
Call ComiLxAx1.McSetSpeedMx(MAP0, 500, 500)
` Move from P1 to P2
Call ComiLxAx1.McArcP(MAP0, 1000, 0, 1000, 1000)
` Move from P2 to P3
PosList(0) = 2000
PosList(1) = 1000

```

```

Call ComiLxAx1.McLineTo (MAP0, PosList(0))
` Move from P3 to P4
Call ComiLxAx1.McArcP(MAP0, 2000, 500, 2500, 500)
` Move from P4 to P5
PosList(0) = 2500
PosList(1) = -500
Call ComiLxAx1.McLineTo (MAP0, PosList(0))
` Move from P5 to P6
Call ComiLxAx1.McArcP(MAP0, 2000, -500, 2000, -1000)
` Move from P6 to P7
PosList(0) = 1000
PosList(1) = -1000
Call ComiLxAx1.McLineTo (MAP0, PosList(0))
` Move from P7 to P1
Call ComiLxAx1.McArcP(MAP0, 1000, 0, 0, 0)

Call ComiLxAx1.UnloadDevice
    
```

or

■ McCompleteArc

함수 원형

```
sub McCompleteArc ( ByVal MapIndex As Long)
```

함수 설명

이 함수는 McStartArcA, McStartArcP, McArcToA, McArcToP 함수를 이용하여 원호보간 작업을 수행하였을 때 Ending 좌표가 목표 좌표와 약간의 차이를 가질 수 있는데 이를 보정해주는 함수입니다. 따라서 앞에서 언급된 4 개의 함수를 이용하여 원호보간 작업을 수행한 경우에는 원호보간이 완료된 후에 이 함수를 한번 수행해주면 해당 오차를 보정해줍니다.

McArcA 함수와 같이 “Start” 가 붙지않은 원호보간 함수를 사용한 경우에는 이 함수를 사용할 필요가 없습니다.

매개 변수

▶ *MapIndex* : 축 그룹 인덱스. 이 값은 0 또는 1 이어야 하며 McMapAxes 함수를 통하여 제어 대상축들이 맵핑되어 있어야 합니다.

■ McBuildSpline

함수 원형

```
Sub McBuildSpline (ByRef InArray As Double, ByVal NumInArray As Long, ByRef OutArray
    As Double, ByVal NumOutArray As Long)
```

함수 설명

사용자가 입력한 데이터를 기반으로 Cubic spline 보간을 수행한 후 OutArray 에 그 결과값을 넣어줍니다. 이 함수는 시간축을 매개변수로 하여 Cubic spline 보간을 수행합니다.

매개 변수

- ▶ *InArray* : 스플라인 보간을 수행할 샘플 데이터 배열
- ▶ *NumInPoints* : 샘플 데이터의 수.
- ▶ *OutArray* : 스플라인 보간 결과를 저장할 데이터 배열
- ▶ *NumOutPoint* : 스플라인 보간을 수행하여 자동 생성할 데이터 수. 이 값은 전체 곡선을 몇 개의 데이터 포인트로 곡선화할 것인지를 결정합니다.

참 고

□ 이 함수는 스플라인 보간 기법을 이용하여 데이터를 생성해주는 역할만 하며 실제로 모션을 구동하지는 않습니다. 스플라인 보간 구동을 하기 위해서는 여기서 생성된 각 데이터 포인트를 McLineTo 함수 등을 이용하여 구동해주어야 합니다. 이 때 리스트모션(Listed Motion) 기법과 함께 사용하면 효과적으로 구동할 수 있습니다.

□ 이 함수를 이용하면 2 차원 스플라인 보간뿐만 아니라 3 차원, 4 차원 스플라인 보간도 가능합니다.

■ McStartHelical

함수 원형

Function **McStartHelical** (ByVal MapIndex As Long, ByVal Zaxis As Long, ByVal Xcenter As Double, ByVal Ycenter As Double, ByVal Direction As Long, ByVal NumCircle As Long, ByVal LastArcAngle As Double, ByVal Zdistance As Double) As Boolean

함수 설명

헬리컬 보간 구동을 시작합니다.

매개 변수

- ▶ **MapIndex** : 원호보간으로 사용할 Axis map index
- ▶ **Zaxis** : 직선보간으로 사용될 축 번호를 지정합니다. 이 변수의 이름은 편의상 Z 축으로 되어있으나 꼭 Z축일 필요는 없습니다.
- ▶ **Xcenter** : 현재 좌표로부터 원호보간의 중심점까지의 X 축 상대 좌표값. 여기서 X 축은 편의상 X 축으로 표현한 것이며, 실제로는 원호보간으로 사용되는 두 축 중에서 축 번호가 낮은 축을 의미합니다.
- ▶ **Ycenter** : 현재 좌표로부터 원호보간의 중심점까지의 Y 축 상대 좌표값. 여기서 Y 축은 편의상 Y 축으로 표현한 것이며, 실제로는 원호보간으로 사용되는 두 축 중에서 축 번호가 높은 축을 의미합니다.
- ▶ **Direction** : 원호보간의 구동 방향을 설정합니다. 양수 - 반시계방향(CCW), 0 또는 음수-시계방향(CW)
- ▶ **NumCircle** : 전체 구동 중에 포함되어야 할 원의 수(마지막 ARC 포함). 여기서 지정한 수의 원(마지막 ARC 포함)이 그려지면 Z축에 지정한 거리에 도달하지 못하였더라도 헬리컬 보간은 자동으로 종료됩니다.
- ▶ **MLastArcAngle** : 마지막 ARC 의 각도입니다. 이 각도는 시작점을 기준으로 X-Y 평면상의 각도를 의미합니다. 마지막 원호 보간은 완전한 원이 아니도록 할 수 있습니다. 예를 들어 처음 좌표를 기준으로 90 도 회전한 위치에서 헬리컬보간을 완료하도록 하고자한다면 이 값을 90 또는 -90 으로 해주면 됩니다.
- ▶ **Zdistance** : Z 축이 이동해야 할 거리(상대좌표)를 지정합니다. Z 축이 지정한 거리만큼 이동이 완료되면 헬리컬보간은 자동으로 종료됩니다.

참 고

- 여기서 각 변수의 이름또는 설명에서 언급하는 X,Y,Z 축은 개념상의 X,Y,Z 축입니다.

여기서 X,Y 축은 원호보간에 사용되는 두 축을 의미하며, Z 축은 직선보간으로 사용되는 축을 의미합니다.

□ 헬리컬 보간이 종료되는 조건은 Z 축이 Zdistance 만큼 이동하거나 Z 축이 강제 종료된 경우, 또는 NumCircle 에서 지정한 수만큼 원호보간을 완료하면 자동종료됩니다. 세가지 조건 중에서 하나만 만족되면 자동 종료됩니다. 따라서 만일 원호보간의 수나 마지막 ARC 의 각도가 중요한 경우에는 처음에는 Zdistance 를 충분히 하여 시뮬레이션해보는 것이 좋습니다.

UI

■ McAbortHelical

함수 원형

Function **McAbortHelical** As Boolean

함수 설명

현재 구동되고 있는 헬리컬 보간 작업을 취소합니다.

■ McMxDone

메소드 원형

```
Function McMxDone(ByVal MapIndex As Long) As Boolean
```

메소드 설명

지정한 축그룹(MapIndex)의 Coordinated Motion 이 완료됐는지를 체크합니다.

매개 변수

▶ **MapIndex** : 축 그룹 인덱스. 이 값은 0 또는 1 이어야 하며 McMapAxes 메소드를 통하여 제어 대상축들이 맵핑되어 있어야 합니다.

Return 값

지정한 축그룹(MapIndex)에 맵핑되어 있는 모든 축이 모션을 완료하였으면 1 을 그렇지 않으면 0 을 반환합니다.

Value	Meaning
0	모션이 완료되지 않았음
1	모션이 완료됨

2.6.5 속도 및 위치 오버라이딩(Overriding) 메소드

이 단원에서는 속도 및 위치 오버라이딩 메소드들을 소개합니다. 속도 오버라이딩은 모션이 진행되고 있는 중에 작업 속도를 변경하는 것을 의미합니다. 위치 오버라이딩은 Single Axis 모션 중에서 Move 나 MoveTo 와 같이 In-Position 모션을 수행하고 있는 중에 목표 거리 또는 목표 좌표를 수정하는 것을 의미 합니다. 속도 및 위치 오버라이딩에 관련된 메소드는 다음과 같습니다.

메소드 / 설명	페이지
Sub McOverrideSpeedSet (ByVal Channel As Long) Single Axis 모션이 진행되고 있는 중에 속도를 변경	195
Sub McOverrideSpeedSetAll (ByVal NumAxis As Long, ByVal AxisList[] As Long) 여러 축에 대하여 동시에 속도를 변경.	197
Sub McOverrideMove (ByVal Channel As Long, ByVal NewDistance As Double) McStartMove 메소드를 통하여 수행되는 상대좌표 In-position 모션에 대하여 상대좌표값, 즉 목표 거리값을 수정	199
Sub McOverrideMoveTo (ByVal Channel As Long, ByVal NewPosition As Double) McStartMoveTo 메소드를 통하여 수행되는 절대좌표 In-position 모션에 대하여 목표 절대좌표값을 수정.	200

■ McOverrideSpeedSet

메소드 원형

Sub **McOverrideSpeedSet** (ByVal Channel As Long)

메소드 설명

이 메소드는 Single Axis 모션이 진행되고 있는 중에 속도를 변경(오버라이딩, Overriding)하고자 할 때 사용하는 메소드입니다. 속도를 오버라이딩(Overriding)하기 위해서는 먼저 McSetSpeedMode, McSetSpeed, McSetAccel, McSetScurve 등의 속도 패턴 설정 메소드를 통하여 변경하고자 하는 속도 또는 가속도값을 설정하고 McOverrideSpeedSet 메소드를 통하여 설정된 속도 또는 가속도값을 실제 모션에 적용합니다.

매개 변수

▶ *Channel* : 채널(축) 번호, 0 ~ 3

참 고

□ 이 메소드는 변경된 속도 패턴 설정을 실제 모션에 적용하는 역할을 합니다. 속도를 오버라이딩(Overriding)하기 위해서는 이 메소드를 사용하기전에 McSetSpeedMode, McSetSpeed, McSetAccel, McSetScurve 등을 통하여 필요한 변경값을 설정하여야 합니다.

□ 여러축을 동시에 속도 오버라이딩(Overriding)하고자 한다면 이 메소드 대신에 McOverrideSpeedSetAll 메소드를 사용하십시오.

□ Line 이나 Arc 와 같은 Interpolation(또는 Coordinated Motion) 메소드를 사용한 경우에는 속도 오버라이딩을 사용할 수 없습니다.

예 제

본 예제는 McOverrideSpeedSet()메소드를 사용하여 속도를 오버라이딩하는 것을 예로 보여주는 코드입니다. 본 예제는 일정 시간을 구분으로 하여 3 단계의 속도로 변경을 하면서 모션을 수행하는 예제입니다.

```
Call ComiLxAx1.LoadDivice

Const X_AXIS = 0
Dim Speed(2)
Speed(0) = 10000
Speed(1) = 20000
Speed(2) = 30000

Call ComiLxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiLxAx1.McSetAccel(X_AXIS, 20000, 20000)
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, Speed(0) )
' (+)방향으로 Velocity Move 수행
Call ComiLxAx1.McStartVMove(X_AXIS, 1)
```

· 일정 시간동안 운행

ur

```
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, Speed(1) )
Call ComiLxAx1.McOverrideSpeedSet(X_AXIS)

` 일정 시간동안 운행
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, Speed(2))
Call ComiLxAx1.McOverrideSpeedSet(X_AXIS)

` 일정 시간동안 운행

` 감속 후 정지
Call ComiLxAx1.McStop(X_AXIS)

Call ComiLxAx1.UnloadDevice
```

■ McOverrideSpeedSetAll

메소드 원형

Sub **McOverrideSpeedSetAll** (ByVal NumAxis As Long, ByRef AxisList[] As Long)

메소드 설명

이 메소드는 Multi-Axis 동시 제어 모션이 진행되고 있는 중에 여러 축에 대하여 동시에 속도를 변경(오버라이딩, Overriding)하고자할 때 사용하는 메소드입니다. 이 메소드는 속도 오버라이딩을 여러축에 대하여 동시에 수행합니다. 속도를 오버라이딩(Overriding)하기 위해서는 먼저 McSetSpeedMode, McSetSpeed, McSetAccel, McSetScurve 등의 속도 패턴 설정 메소드를 통하여 각 축에 대하여 변경하고자 하는 속도 또는 가속도값을 설정하고 McOverrideSpeedSetAll 메소드를 통하여 설정된 속도 또는 가속도값을 실제 모션에 적용합니다.

매개 변수

- ▶ **NumAxis** : 동시에 작업을 수행할 대상 축의 수
- ▶ **AixsList** : 동시에 작업을 수행할 대상 축의 배열. 이 배열의 크기는 NumAxis 값과 일치해야 합니다.

참 고

- 이 메소드는 변경된 속도 패턴 설정을 실제 모션에 적용하는 역할을 합니다. 속도를 오버라이딩(Overriding)하기 위해서는 이 메소드를 사용하기전에 각 축에 대하여 McSetSpeedMode, McSetSpeed, McSetAccel, McSetScurve 등을 통하여 필요한 변경값을 설정하여야 합니다.
- 하나의 축에 대하여 속도 오버라이딩(Overriding)하고자 한다면 이 메소드 대신에 McOverrideSpeedSet 메소드를 사용하십시오.
- Line 이나 Arc 와 같은 Interpolation(또는 Coordinated Motion) 메소드를 사용한 경우에는 속도 오버라이딩을 사용할 수 없습니다.

예 제

본 예제는 McOverrideSpeedSetAll()메소드를 사용하여 속도를 오버라이딩하는 것을 예로 보여주는 코드입니다. 본 예제는 x,y,z 축을 동시 제어하는 것으로써 일정 시간을 구분으로하여 미리 지정된 3 단계의 속도로 변경을 하면서 모션을 수행하는 예제입니다.

```
Call ComiLxAxis1.LoadDivice

Const X_AXIS = 0
Const Y_AXIS = 1
Const Z_AXIS = 2
Dim AxisList(2)
AxisList(0) = X_AXIS
```

ur

```

AxisList(0) = Y_AXIS
AxisList(0) = Z_AXIS

Dim DirList(2)
DirList(0) = 1
DirList(1) = 1
DirList(2) = 1

Dim Speed(2)
Speed(0) = 10000
Speed(1) = 20000
Speed(2) = 30000

Call ComiLxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiLxAx1.McSetAccel(X_AXIS, 20000, 20000)
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, Speed(0) )

Call ComiLxAx1.McSetSpeedMode(Y_AXIS, 1)
Call ComiLxAx1.McSetAccel(Y_AXIS, 20000, 20000)
Call ComiLxAx1.McSetSpeed(Y_AXIS, 0, Speed(0) )

Call ComiLxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiLxAx1.McSetAccel(Z_AXIS, 20000, 20000)
Call ComiLxAx1.McSetSpeed(Z_AXIS, 0, Speed(0) )

' (+)방향으로 Velocity Move 수행
Call ComiLxAx1.McStartVMove(X_AXIS, 1)
Call ComiLxAx1.McStartVMoveAll(3, AxisList(0), DirList(0))

' 일정 시간동안 운행

Call ComiLxAx1.McSetSpeed(X_AXIS, 0, Speed(1) )
Call ComiLxAx1.McSetSpeed(Y_AXIS, 0, Speed(1) )
Call ComiLxAx1.McSetSpeed(Z_AXIS, 0, Speed(1) )
Call ComiLxAx1.McOverrideSpeedSetAll(3, AxisList(0))

' 일정 시간동안 운행

Call ComiLxAx1.McSetSpeed(X_AXIS, 0, Speed(2))
Call ComiLxAx1.McSetSpeed(Y_AXIS, 0, Speed(2))
Call ComiLxAx1.McSetSpeed(Z_AXIS, 0, Speed(2))
Call ComiLxAx1.McOverrideSpeedSetAll(3, AxisList(0))

' 일정 시간동안 운행

' 감속 후 정지
Call ComiLxAx1.McStopAll(3, AxisList(0))

Call ComiLxAx1.UnloadDevice
    
```

■ McOverrideMove

메소드 원형

Sub **McOverrideMove** (ByVal Channel As Long, ByVal NewDistance As Double)

메소드 설명

이 메소드는 McStartMove 메소드를 통하여 수행되는 상대좌표 In-position 모션에 대하여 상대좌표값, 즉 목표 거리값을 수정(오버라이딩, Overriding)하는 메소드입니다. 이 메소드는 McStartMove 메소드를 사용하여 모션을 수행하고 있는 경우에만 사용가능한 메소드입니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **NewDistance** : 새로운 목표 거리값을 지정합니다. 이 값의 기준 위치는 McStartMove 메소드에서 사용한 기준과 같습니다. 즉 McStartMove 를 실행하기 바로직전의 위치가 좌표값 0 으로 간주하여 NewDistance 값을 설정하여야 합니다.

참 고

- McStartMoveTo 메소드에 시작된 모션의 목표 위치(절대좌표)값을 오버라이딩하려면 McOverrideMoveTo 메소드를 사용하십시오.

or

■ McOverrideMoveTo

메소드 원형

Sub **McOverrideMoveTo** (ByVal Channel As Long, ByVal NewPosition As Double)

메소드 설명

이 메소드는 McStartMoveTo 메소드를 통하여 수행되는 절대좌표 In-position 모션에 대하여 목표 절대좌표값을 수정(오버라이딩, Overriding)하는 메소드입니다. 이 메소드는 McStartMoveTo 메소드를 사용하여 모션을 수행하고 있는 경우에만 사용가능한 메소드입니다.

매개 변수

- ▶ *Channel* : 채널(축) 번호, 0 ~ 3
- ▶ *NewPosition* : 새로운 목표 절대좌표값을 지정합니다.

참 고

□ McStartMove 메소드에 시작된 모션의 목표 거리값을 오버라이딩하려면 McOverrideMove 메소드를 사용하십시오.

2.6.6 원점 복귀(Home Return) 메소드

이 단원에서는 원점 복귀(Home Return)에 관련된 메소드들을 소개합니다. 원점 복귀는 모션제어의 대상이 되는 구조물이 원점 위치로 자동 복귀하도록 하는 작업입니다. 원점 복귀 작업이 완료되면 Command Counter, Feedback Counter, Deviation Counter 는 자동으로 0 으로 초기화됩니다.

원점 복귀 작업을 수행하기 위해서는 ORG(HOME), EZ 및 EL 신호가 참조되는데 이 신호들의 의미 및 작용은 다음과 같습니다.

□ ORG 신호

ORG 신호는 구조물이 원점에 복귀했는지를 센서로부터 입력받는 신호입니다. 일반적으로는 근접 센서와 같은 센서들을 이용하여 원점 복귀 여부를 감지하게 됩니다. 이 신호는 터미널 보드의 'HOME' 단자를 통하여 COMI-LX50x 보드에 입력되어야 합니다.

□ EZ 신호

엔코더의 제로 펄스 신호를 의미합니다. 이 신호는 원점 복귀 모드에 따라 ORG 신호 또는 EL 신호와 함께 사용되어 보다 정밀한 원점복귀 작업을 수행할 수 있도록 해줍니다. 이 신호는 터미널 보드의 'EZ+' 단자와 'EZ-' 단자를 통하여 COMI-LX50x 보드에 입력되어야 합니다.

□ EL 신호

기계적인 리미트(Limit) 신호를 의미합니다. 이 신호는 일반적으로 구조물이 움직일 수 있는 한계점을 감지하기 위해 사용되나 원점 복귀 모드에 따라 ORG 신호의 대용으로도 사용될 수 있습니다. EL 신호는 (+)방향 리미트 신호와 (-)방향 리미트 신호의 두 가지 신호가 있습니다. (+)방향 리미트 신호는 터미널 보드의 '+EL' 단자, 그리고 (-)방향 리미트 신호는 '-EL' 단자를 통하여 COMI-LX50x 보드에 입력되어야 합니다.

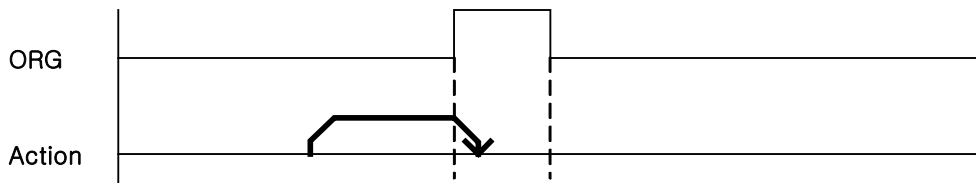
원점복귀모드

COMI-LX50x 모션제어보드는 다음과 같이 13 가지의 다양한 원점 복귀 모드를 제공합니다. 원점 복귀 모드는 McSetHomeConfig 메소드를 통하여 설정됩니다. 아래의 그림은 모두 속도모드를 Trapezoidal 모드로 설정한 상태임을 가정하여 그려진 것이며, 만일 Constant 속도 모드로 설정된 경우에는 감속이 없이 즉시 정지하게 됩니다.

□ MODE 0 : ORG -> Slow down -> Stop

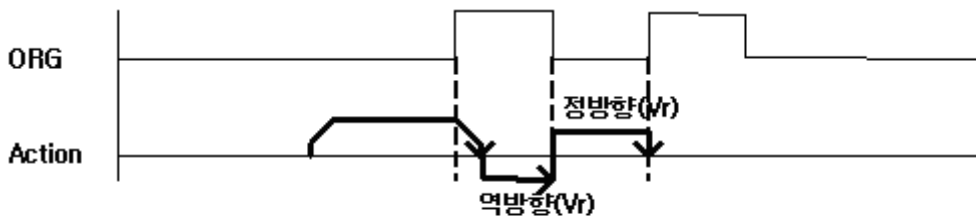
MODE 0 에서는 ORG 신호가 OFF 에서 ON 으로 바뀌는 순간에 모션을 감속 후 정지하고 복귀 작업을 종료합니다.

ur



□ MODE 1 : ORG → Slow down → Go back → Go forward → Stop

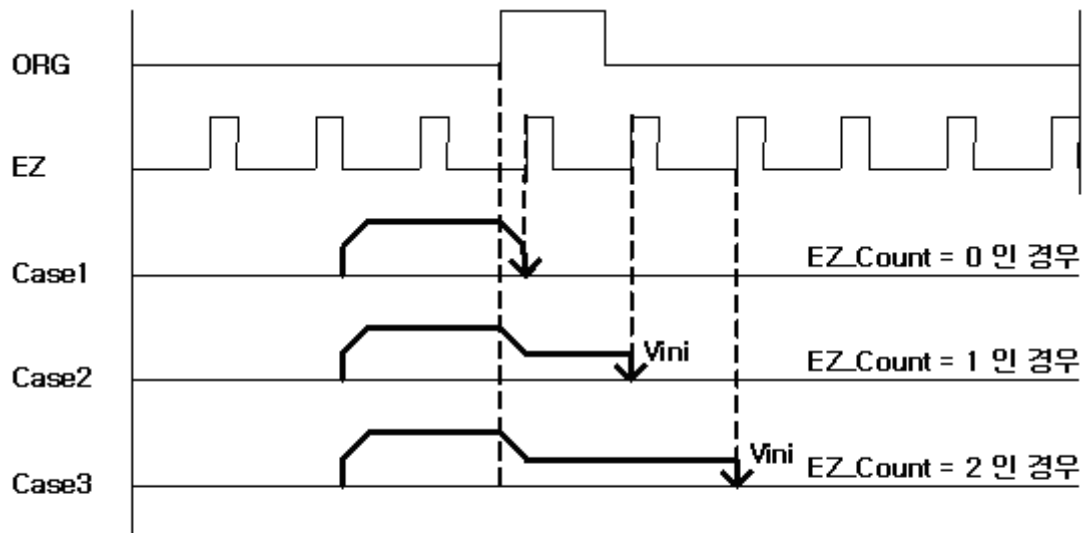
MODE 1에서는 ORG 신호가 OFF에서 ON으로 바뀌는 순간에 모션을 감속 후 정지한 후 ORG 신호가 OFF가 될때까지 V_r (Reverse Speed)의 속도로 역방향 회전을 수행합니다. ORG 신호가 OFF되는 순간에 다시 V_r 의 속도로 정방향 회전을 수행하다가 ORG 신호가 다시 ON되는 순간에 복귀작업을 종료합니다.



V_r : Reverse Speed를 의미하며 McHomeMove 함수 참조

□ MODE 2 : ORG → Slow down → Stop on EZ signal

MODE 2에서는 ORG 신호가 OFF에서 ON으로 바뀌는 순간에 모션을 감속한 후 초기속도값으로 모션을 진행하다가 EZ 신호에 따라 복귀작업을 종료합니다. McSetHomeConfig 메소드를 통하여 미리 설정된 EzCount 값에 따라 종료하는 시점은 아래와같이 달라집니다.



Case1 : EzCount = 0 인 경우, McSetHomeConfig 함수 참조

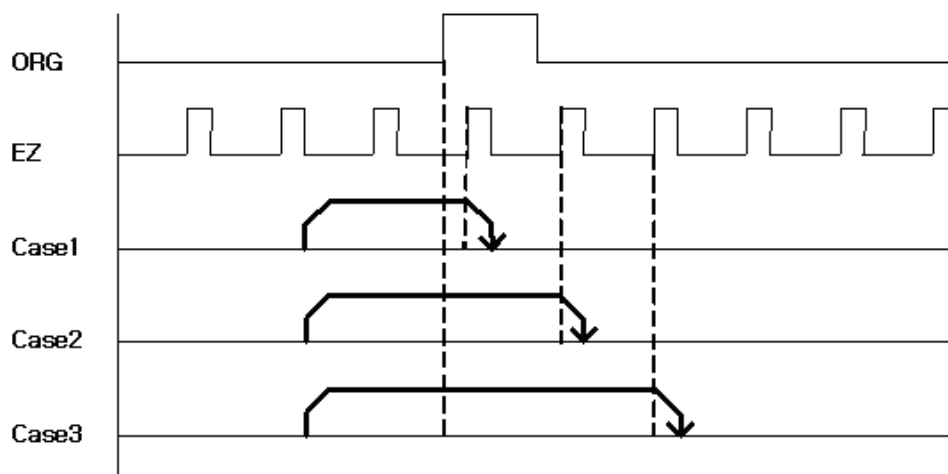
Case2 : EzCount = 1 인 경우, McSetHomeConfig 함수 참조

Case3 : EzCount = 2 인 경우, McSetHomeConfig 함수 참조

Vini : 초기속도를 의미하며, McSetSpeed 함수 참조

□ MODE 3 : ORG → EZ → Slow down → Stop

MODE 3에서는 ORG 신호가 OFF에서 ON으로 바뀌고 난 후 발생하는 EZ 신호에 따라 감속 후 정지합니다. McSetHomeConfig 메소드를 통하여 미리 설정된 EzCount 값에 따라 감속을 시작하는 시점은 아래와 같이 달라집니다.



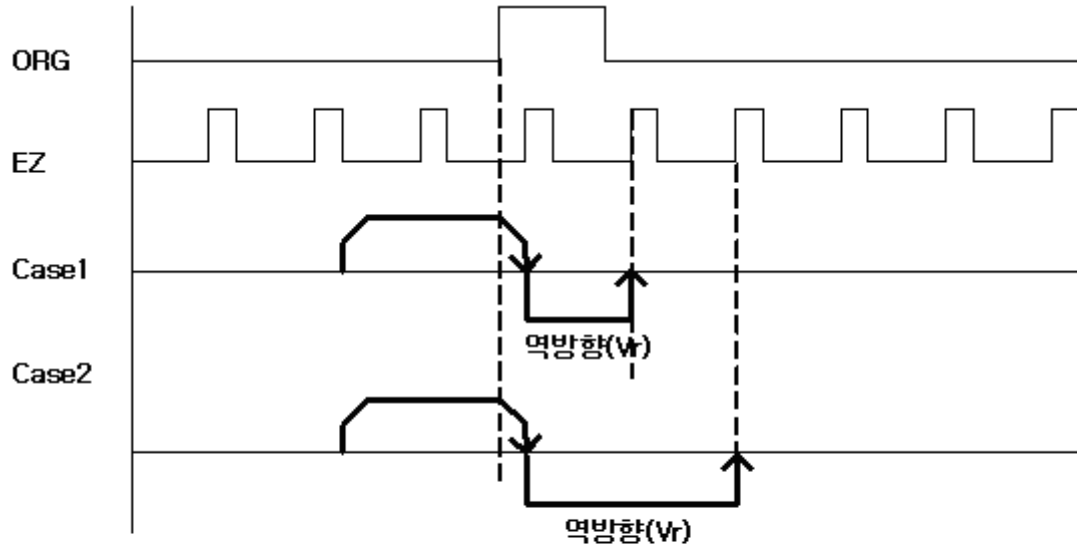
Case1 : EzCount = 0 인 경우, McSetHomeConfig 함수 참조

Case2 : EzCount = 1 인 경우, McSetHomeConfig 함수 참조

Case3 : EzCount = 2 인 경우, McSetHomeConfig 함수 참조

□ MODE 4 : ORG → Slow down → Go back at Vr → Stop on EZ signal

MODE 4에서는 ORG 신호가 OFF에서 ON으로 바뀌는 순간 감속 후 정지합니다. 그리고 Vr의 속도로 역방향 회전한 후 EZ 신호에 따라 복귀 작업을 종료합니다.



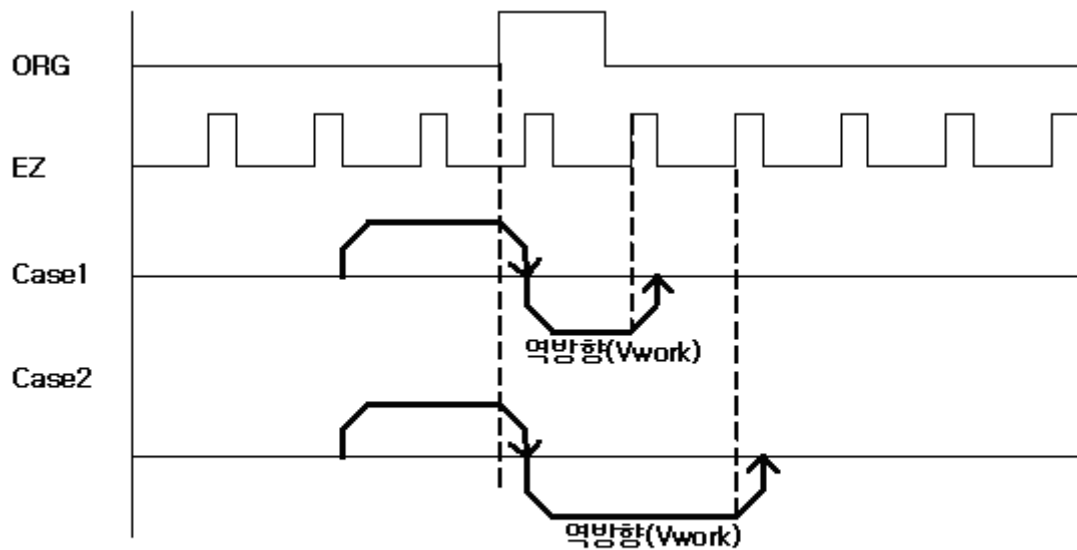
Case1 : EzCount = 1 인 경우, McSetHomeConfig 함수 참조

Case2 : EzCount = 2 인 경우, McSetHomeConfig 함수 참조

Vr : Reverse Speed를 의미하며 McSetHomeConfig 함수 참조

□ MODE 5 : ORG → Slow down → Go back → Accelerate to Vwork → EZ → Slow down → Stop

MODE 5에서는 ORG 신호가 OFF에서 ON으로 바뀌는 순간 감속 후 정지합니다. 그리고 작업속도까지 가속하여 역방향 회전한 후 EZ 신호에 따라 감속 후 복귀 작업을 종료합니다.

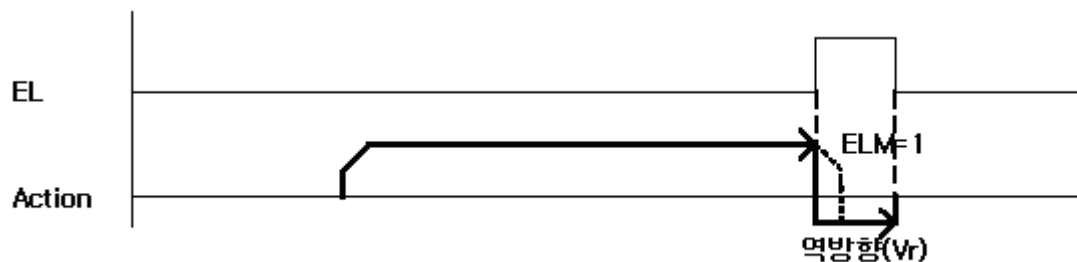


Case1 : EzCount = 1 인 경우, McSetHomeConfig 함수 참조

Case2 : EzCount = 2 인 경우, McSetHomeConfig 함수 참조

□ MODE 6 : EL ON → Stop → Go back at Vr → EL OFF → Stop

MODE 6에서는 EL 신호가 ON으로 바뀌는 순간 즉시 정지(또는 ELM=1인 경우에 감속후 정지)합니다. 그리고 반대 방향으로 Vr 속도로 회전하다가 EL 신호가 OFF되는 순간에 복귀작업을 종료합니다. 여기서 ELM=1은 McSetMioCfgEL 메소드에서 nElMode를 1로 설정했음을 의미합니다.

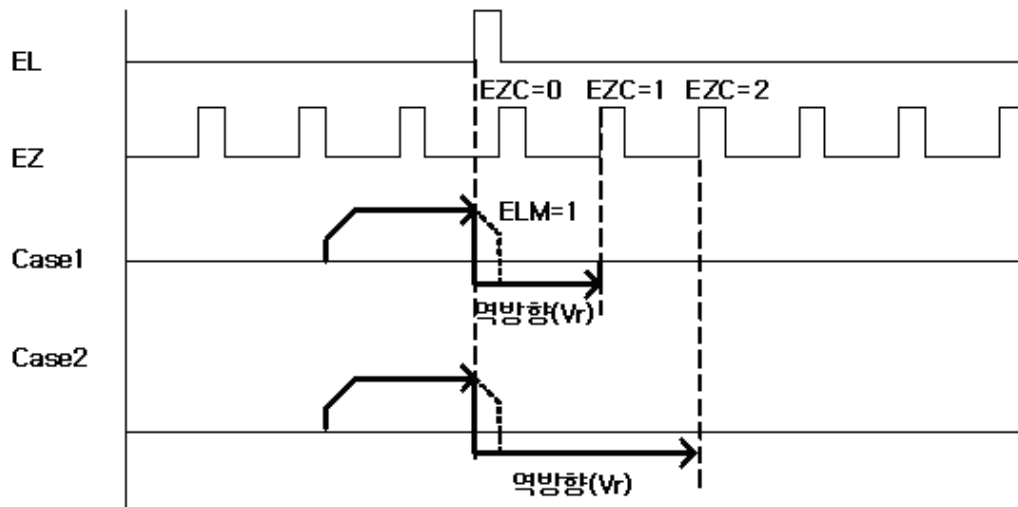


Vr: Reverse Speed를 의미하며 McHomeMove 함수 참조

□ MODE 7 : EL ON → Go back at Vr → Stop on EZ signal

MODE 7에서는 EL 신호가 ON으로 바뀌는 순간 즉시 정지(또는 ELM=1인 경우에 감속후 정지)합니다. 그리고 반대 방향으로 Vr 속도로 회전하다가 EZ_Count에 따라 복귀작업을 종료합니다. 여기서 ELM=1은 McSetMioCfgEL 메소드에서 nElMode를 1로 설정했음을 의미합니다.

UT



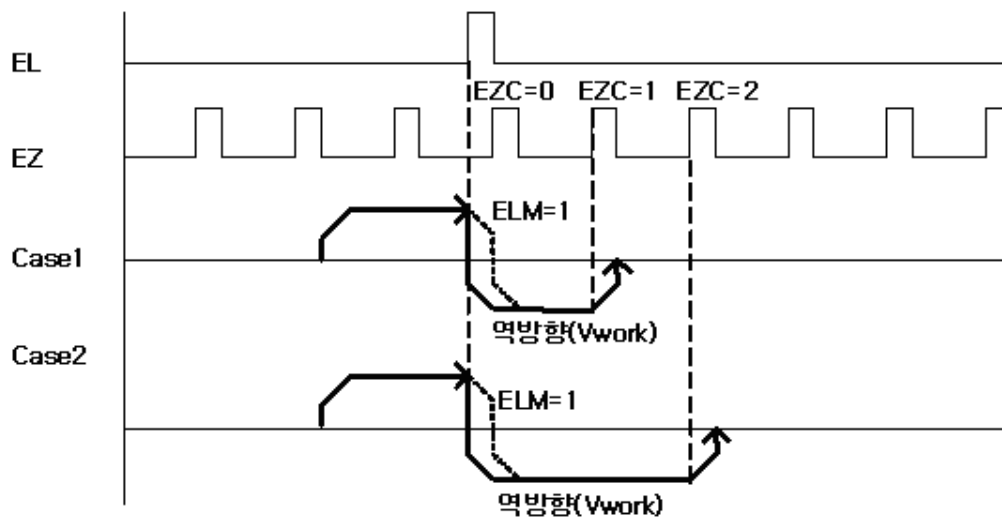
Case1 : EzCount = 1 인 경우, McSetHomeConfig 함수 참조

Case2 : EzCount = 2 인 경우, McSetHomeConfig 함수 참조

Vr : Reverse Speed를 의미하며 McHomeMove 함수 참조

□ MODE 8 : EL ON → Accelerate to Vwork → EZ → Slow down → Stop

MODE 8에서는 EL 신호가 ON으로 바뀌는 순간 즉시 정지(또는 ELM=1인 경우에 감속후 정지)합니다. 그리고 작업속도까지 가속하여 반대 방향으로 회전한 후 EZ 신호에 따라 감속 후 복귀 작업을 종료합니다. 여기서 ELM=1은 McSetMioCfgEL 메소드에서 nElMode를 1로 설정했음을 의미합니다.



Case1 : EzCount = 1 인 경우, McSetHomeConfig 함수 참조

Case2 : EzCount = 2 인 경우, McSetHomeConfig 함수 참조

Vwork : 작업속도를 의미하며 McSetSpeed 함수 참조

□ **MODE 9 : EL ON -> Accelerate to Vwork -> EZ -> Slow down -> Stop**

MODE 9 에서는 MODE 0 에서와 똑같은 복귀 작업을 수행한다. 그리고 난후 Feedback Counter 가 0 이 되도록 모션을 다시 취한 후에 복귀 작업을 종료한다.

□ **MODE 10 : EL ON -> Accelerate to Vwork -> EZ -> Slow down -> Stop**

MODE 10 에서는 MODE 3 에서와 똑같은 복귀 작업을 수행한다. 그리고 난후 Feedback Counter 가 0 이 되도록 모션을 다시 취한 후에 복귀 작업을 종료한다.

□ **MODE 11 : EL ON -> Accelerate to Vwork -> EZ -> Slow down -> Stop**

MODE 11 에서는 MODE 5 에서와 똑같은 복귀 작업을 수행한다. 그리고 난후 Feedback Counter 가 0 이 되도록 모션을 다시 취한 후에 복귀 작업을 종료한다.

□ **MODE 12 : EL ON -> Accelerate to Vwork -> EZ -> Slow down -> Stop**

MODE 12 에서는 MODE 8 에서와 똑같은 복귀 작업을 수행한다. 그리고 난후 Feedback Counter 가 0 이 되도록 모션을 다시 취한 후에 복귀 작업을 종료한다.

■ McSetHomeConfig

메소드 원형

```
sub McSetHomeConfig (ByVal Channel As Long, ByVal OrgMode As Long, ByVal OrgLogic As Long, ByVal EzCount As Long, ByVal EzLogic As Long, ByVal ErcOut As Long)
```

메소드 설명

이 메소드는 원점 복귀 작업을 수행하기 위한 여러가지 환경설정을 수행합니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **OrgMode** : 원점 복귀 모드 번호를 설정합니다. COMI-LX50x 모션 제어보드는 13 가지(0 ~ 12)의 다양한 복귀 모드를 제공합니다. 각 복귀 모드에 대한 자세한 사항은 앞 페이지를 참조하십시오.
- ▶ **OrgLogic** : ORG 신호의 Action Level 을 설정합니다. 즉 ORG 신호의 레벨(Level)이 High 상태일때 ON 인지, Low 상태일때 ON 인지를 설정합니다.

Value	Meaning
0	Low active, 즉 ORG 신호레벨이 Low 일때 Logic 1 이 된다.
1	High active, 즉 ORG 신호레벨이 High 일때 Logic 1 이 된다.

- ▶ **EzCount** : 이 값은 ORG 신호 또는 EL 신호가 ON 이 된 후 실제로 복귀 작업을 완료하는데 필요한 EZ Count 값을 0 ~ 15 사이의 값으로 설정합니다. 이 값의 참조 여부는 복귀 모드에 따라서 다릅니다.
- ▶ **EzLogic** : EZ 신호의 Action Level 을 설정합니다. 즉 EZ 신호의 레벨(Level)이 High 상태일때 ON 인지, Low 상태일때 ON 인지를 설정합니다.

Value	Meaning
0	Low active, 즉 EZ 신호레벨이 Low 일때 Logic 1 이 된다.
1	High active, 즉 EZ 신호레벨이 High 일때 Logic 1 이 된다.

- ▶ **ErcOut** : 원점 복귀 작업이 끝나는 시점에서 ERC 펄스를 출력할 것인지를 결정합니다. ERC 신호는 서보모터 드라이버의 Deviation Counter 를 리셋하는 기능을 제공합니다.

예 제

본 예제는 x 축에 대하여 원점복귀 작업을 수행하는 예제입니다. 본 예제에서는 원점 복귀 모드 4 번을 설정하여 작업을 수행합니다.


```
const X_AXIS = 0
const MODE4 = 4
const LOW_ACTIVE = 0

Call ComiLxAx1.LoadDevice

Call ComiLxAx1.McSetHomeConfig(X_AXIS,MODE4,LOW_ACTIVE,0,LOW_ACTIVE,0)

Call ComiLxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiLxAx1.McSetSpeed(X_AXIS, 200, 5000)
Call ComiLxAx1.McSetAccel(X_AXIS, 5000, 5000)

Call ComiLxAx1.McHomeMove(X_AXIS, 1, 500)
While ( Not ComiLxAx1.McDone (X_AXIS))
Wend

Call ComiLxAx1.UnloadDevice
```

■ McHomeMove

메소드 원형

```
sub McHomeMove (ByVal Channel As Long, ByVal Direction As Long, ByVal RvsVel As Double)
```

메소드 설명

이 메소드는 원점 복귀 작업을 수행하기 위한 여러가지 환경설정을 수행합니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **OrgMode** : 원점 복귀 모션을 수행할 방향을 지정합니다.

Value	Meaning
0 또는 음수	(-) 방향
양수	(+) 방향

- ▶ **RvsVel**: Reverse Speed 를 설정합니다. 복귀 모드에 따라 Reverse Speed 를 필요로 하는 모드가 있습니다. 앞의 복귀 모드 설명에서 Reverse Speed 는 Vr 로 표기되었습니다.

참 고

- 이 메소드는 원점 복귀 작업을 시작시킨 후에 바로 리턴(Return)합니다. 따라서 사용자는 McDone() 메소드를 사용하여 복귀 작업이 완료되었는지를 체크하여야 합니다.

예 제

본 예제는 x 축에 대하여 원점복귀 작업을 수행하는 예제입니다. 본 예제에서는 원점 복귀 모드 4 번을 설정하여 작업을 수행합니다.

```
const X_AXIS = 0
const MODE4 = 4
const LOW_ACTIVE = 0

Call ComiLxAx1.LoadDevice

Call ComiLxAx1.McSetHomeConfig(X_AXIS,MODE4,LOW_ACITVE,0,LOW_ACTIVE,0)

Call ComiLxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiLxAx1.McSetSpeed(X_AXIS, 200, 5000)
Call ComiLxAx1.McSetAccel(X_AXIS, 5000, 5000)

Call ComiLxAx1.McHomeMove(X_AXIS, 1, 500)
While Not( ComiLxAx1.McDone(X_AXIS))
Wend

Call ComiLxAx1.COMICX_UnloadDevice
}
```

2.6.7 Manual Pulser 모드 모션 제어 메소드

이 단원에서는 Manual Pulser 모드 모션에 관련된 메소드들을 소개합니다. Manual Pulser 모드는 로터리 엔코더(Rotary Encoder)와 같은 장치를 이용하여 수동으로 모션을 제어하는 모드를 의미합니다. COMI-SD501 모션제어 보드는 PA/PB 단자를 통하여 Plus 펄스와 Minus 펄스(CW/CCW) 또는 90° 위상차를 갖는 AB Phase 펄스를 입력받아 수동으로 모션을 제어할 수 있습니다. PA/PB 단자에 입력되는 신호의 형태는 Pulser 입력모드 설정에 따라 달라지며 이는 McSetPulserInputMode()메소드에 의해 설정됩니다.

Manual Pulser 모드 모션에서의 속도는 입력 펄스의 주파수에 의해 결정됩니다. 따라서 속도모드나 가/감속 등은 설정할 필요가 없습니다. 그러나 Manual Pulser 입력신호의 최대 주파수는 McSetSpeed()메소드에 의해 설정되는 작업 속도에 의하여 제한됩니다. 따라서 Manual Pulser 모션을 수행하기 전에 McSetSpeed()메소드를 이용하여 작업속도를 적절한 값으로 설정하여야 합니다.

Manual Pulser 모드 모션은 Velocity Motion 은 물론 상대좌표/절대좌표 In-Position 모션에도 적용가능합니다. 그러나 Coordinated Motion 에는 적용할 수 없습니다.

Manual Pulser 모드 모션에 관련된 메소드들은 다음과 같습니다.

메소드 / 설명	페이지
sub McSetPE (ByVal Channel As Long, ByVal Enable As Short) Manual Pulser 나 External Switch Operation 모드를 Enable/Disable 시킵니다.	213
Sub McSetPulserInputMode (ByVal Channel As Long, ByVal InputMode As Long, ByVal Inverse As Integer) Pulser 입력신호 대한 환경을 설정합니다.	214
Sub McPulserHomeMove (ByVal Channel As Long, ByVal HomeType As Long) Pulser Input 에 의한 원점 복귀 작업을 수행합니다.	215
Sub McStartPulserVMove (ByVal Channel As Long) Stop 메소드가 호출될 때까지 Pulser 신호 입력에 맞추어 계속적인 모션을 수행합니다.	216
Sub McStartPulserMove (ByVal Channel As Long, ByVal Distance As Double) Pulser 신호 입력에 맞추어 지정한 거리(상대좌표)만큼 이동을 수행합니다.	217
Sub McPulserMove (ByVal Channel As Long, ByVal Distance As Double) Pulser 신호 입력에 맞추어 지정한 거리(상대좌표)만큼 이동을 수행합니다.	218
Sub McStartPulserMoveTo (ByVal Channel As Long,By Val Position As Double) Pulser 신호 입력에 맞추어 지정한 위치(절대좌표)로 이동을 수행합니다.	219

UI

Sub McPulserMoveTo (ByVal Channel As Long, ByVal Position As Double) Pulser 신호 입력에 맞추어 지정한 위치(절대좌표)로 이동을 수행합니다.	220
---	-----

■ McSetPE

함수 원형

sub **McSetPE** (ByVal Channel As Long, ByVal Enable As Short)

함수 설명

이 함수는 Manual Pulser 모드나 External Switch Operation 모드를 Enable/Disable 시킵니다. Manual Pulser 모드를 사용하기 위해서는 먼저 이 함수를 이용하여 Manual Pulser 모드를 Enable 시켜야 합니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **Enable** : Manual Pulser 모드와 External Switch Operation 모드를 Enable/Disable 시킵니다.

Value	Meaning
0	Manual Pulser 모드와 External Switch Operation 모드 Disable
1	Manual Pulser 모드와 External Switch Operation 모드 Enable

or

■ McSetPulserInputMode

메소드 원형

Sub **McSetPulserInputMode** (ByVal Channel As Long, ByVal InputMode As Long, ByVal Inverse As Integer)

메소드 설명

이 메소드는 Pulser 입력 신호에 대한 환경을 설정합니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **InputMode** : PA 와 PB 입력 단자를 통하여 입력되는 Pulser 입력 신호의 입력모드를 설정합니다. 설정가능한 값은 다음과 같습니다.

Value	Meaning
0	1X A/B (1 채배 Phase type 입력 모드)
1	2X A/B (2 채배 Phase type 입력 모드)
2	4X A/B (4 채배 Phase type 입력 모드)
3	CW/CCW (PA - Plus direction move, PB - Minus direction move)

- ▶ **Inverse** : Pulser 입력 신호에 의해 결정되는 방향(Direction)을 모션에 반대로 적용할 지를 결정합니다. 설정가능한 값은 다음과 같습니다.

Value	Meaning
0	Pulser 입력 신호가 나타내는 방향과 모션의 방향 일치
1	Pulser 입력 신호가 나타내는 방향과 모션의 방향을 반대로 적용

■ McPulserHomeMove

메소드 원형

Sub **McPulserHomeMove** (ByVal Channel As Long, ByVal HomeType As Long)

메소드 설명

이 메소드는 Pulser Input 에 의한 원점 복귀 작업을 수행합니다. 원점 복귀 모드(Home Type)에 따라 원점 복귀가 완료되거나 McStop()메소드 또는 McEmgStop()메소드가 호출되면 모션을 종료합니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **HomeType** : Pulser Input 에 의해 원점 복귀를 수행하는 모드를 설정합니다. 이 값은 다음 중 하나의 값이어야 합니다.

Value	Meaning
0	Command 카운터가 0 이 될때까지 원점복귀를 종료합니다.
1	ORG 신호가 ON 이 되면 원점복귀를 종료합니다.

■ McStartPulserVMove

메소드 원형

Sub **McStartPulserVMove** (ByVal Channel As Long)

메소드 설명

이 메소드는 Stop 메소드가 호출될 때까지 Pulser 신호 입력에 맞추어 계속적인 모션을 수행합니다. 모션의 속도는 Pulser 신호의 주파수에 따라 결정됩니다.

매개 변수

▶ *Channel* : 채널(축) 번호, 0 ~ 3

참 고

□ Manual Pulser 모드 모션에서의 속도는 입력 펄스의 주파수에 의해 결정됩니다. 따라서 속도모드나 가/감속 등은 설정할 필요가 없습니다. 그러나 Manual Pulser 입력신호의 최대 주파수는 McSetSpeed()메소드에 의해 설정되는 작업 속도에 의하여 제한됩니다. 따라서 Manual Pulser 모션을 수행하기 전에 McSetSpeed()메소드를 이용하여 작업속도를 적절한 값으로 설정하여야 합니다. 만일 McSetUnitSpeed()메소드를 이용하여 논리속도를 1PPS 단위가 아닌 다른 값으로 설정하였다면 Pulser 입력속도의 단위는 PPS 단위임에 유의하여야 합니다.

□ Manual Pulser 모드 모션을 중지하기 위해서는 McStop() 메소드대신 McEmgStop()메소드를 사용하여야 합니다.

예 제

```
Call ComiLxAx1.LoadDivice

Const X_AXIS = 0
Const PHASE_1X = 0 ' Pulser input mode => 1X A/B (1 채널 Phase type
입력 모드)

Call ComiLxAx1.McReset
' Pulser 입력신호의 최대 주파수 제한 설정(650000PPS)
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, 650000)
Call ComiLxAx1.McSetPulserInputMode (X_AXIS, PHASE_1X, FALSE)
' Manual Pulser 모드에서 Velocity Move 모션을 시작한다.
Call ComiLxAx1.McStartPulserVMove (X_AXIS)

' 일정시간동안 운행

Call ComiLxAx1.McEmgStop(X_AXIS)
Call ComiLxAx1.UnloadDevice
```


■ McStartPulserMove

메소드 원형

Sub **McStartPulserMove** (ByVal Channel As Long, ByVal Distance As Double)

메소드 설명

이 메소드는 Pulser 신호 입력에 맞추어 지정한 거리(상대좌표)만큼 이동을 수행합니다.

이 메소드는 모션을 시작시킨 후에 바로 Return 합니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **Distance** : 이동할 거리를 지정합니다. 거리에 대한 단위는 McSetUnitDistance 메소드에 의해 결정됩니다. McSetUnitDistance 메소드로 거리의 단위를 변경하지 않았다면 거리의 단위는 Pulse 수가 됩니다. 즉, 거리값 1은 1 Pulse 출력을 의미합니다.

참 고

□ Manual Pulser 모드 모션에서의 속도는 입력 펄스의 주파수에 의해 결정됩니다. 따라서 속도모드나 가/감속 등은 설정할 필요가 없습니다. 그러나 Manual Pulser 입력신호의 최대 주파수는 McSetSpeed()메소드에 의해 설정되는 작업 속도에 의하여 제한됩니다. 따라서 Manual Pulser 모션을 수행하기 전에 McSetSpeed()메소드를 이용하여 작업속도를 적정한 값으로 설정하여야 합니다.

□ Manual Pulser 모드 모션을 취소하기 위해서는 McStop() 메소드대신 McEmgStop()메소드를 사용하여야 합니다.

예 제

```
Call ComiLxAx1.LoadDevice

Const X_AXIS = 0
Const PHASE_1X = 0 ' Pulser input mode => 1X A/B (1 채널 Phase type 입력 모드)

Call ComiLxAx1.McReset
' Pulser 입력신호의 최대 주파수 제한 설정(650000PPS)
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, 650000)
Call ComiLxAx1.McSetPulserInputMode(X_AXIS, PHASE_1X, FALSE)
Call ComiLxAx1.McStartPulserMove(X_AXIS, 1000)
While Not (ComiLxAx1.McDone())
Wend
Call ComiLxAx1.UnloadDevice
```

or

■ McPulserMove

메소드 원형

Sub **McPulserMove** (ByVal Channel As Long, ByVal Distance As Double)

메소드 설명

이 메소드는 Pulser 신호 입력에 맞추어 지정한 거리(상대좌표)만큼 이동을 수행합니다.

이메소드는 모션이 완료될 때까지 Return 되지 않고 루프를 돌게 됩니다.

매개 변수

▶ **Channel** : 채널(축) 번호, 0 ~ 3

▶ **Distance** : 이동할 거리를 지정합니다. 거리에 대한 단위는 McSetUnitDistance 메소드에 의해 결정됩니다. McSetUnitDistance 메소드로 거리의 단위를 변경하지 않았다면 거리의 단위는 Pulse 수가 됩니다. 즉, 거리값 1은 1 Pulse 출력을 의미합니다.

참 고

□ 이 메소드는 모션이 완료되기 전까지 Return 되지 않고 루프를 돌게 됩니다. 루프를 도는 동안 윈도우 이벤트나 메시지가 처리될 수 있도록 하려면 이 메소드를 수행하기 이전에 McSetBlockingMode 메소드를 사용하여 Blocking 이 일어나지 않도록 설정하여야 합니다.

□ Manual Pulser 모드 모션에서의 속도는 입력 펄스의 주파수에 의해 결정됩니다. 따라서 속도모드나 가/감속 등은 설정할 필요가 없습니다. 그러나 Manual Pulser 입력신호의 최대 주파수는 McSetSpeed()메소드에 의해 설정되는 작업 속도에 의하여 제한됩니다. 따라서 Manual Pulser 모션을 수행하기 전에 McSetSpeed()메소드를 이용하여 작업속도를 적절한 값으로 설정하여야 합니다.

예 제

```
Call ComiLxAx1.LoadDevice
```

```
Const X_AXIS = 0
```

```
Const PHASE_1X = 0 ` Pulser input mode => 1X A/B (1 채널 Phase type 입력 모드)
```

```
Call ComiLxAx1.McReset
```

```
` Pulser 입력신호의 최대 주파수 제한 설정(650000PPS)
```

```
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, 650000)
```

```
Call ComiLxAx1.McSetPulserInputMode(X_AXIS, PHASE_1X, FALSE)
```

```
Call ComiLxAx1.McPulserMove(X_AXIS, 1000)
```

```
Call ComiLxAx1.UnloadDevice
```

■ McStartPulserMoveTo

메소드 원형

Sub **McStartPulserMoveTo** (ByVal Channel As Long,By Val Position As Double)

메소드 설명

이 메소드는 Pulser 신호 입력에 맞추어 지정한 위치(절대좌표)로 이동을 수행합니다. 단 이 메소드는 모션을 시작시킨 후에 바로 Return 합니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **Position** : 이동할 목표 위치(절대좌표값)를 지정합니다. 위치에 대한 단위는 McSetUnitDistance 메소드에 의해 결정됩니다. McSetUnitDistance 메소드로 거리의 단위를 변경하지 않았다면 거리의 단위는 Pulse 수가 됩니다. 즉, 거리값 1 은 1 Pulse 출력을 의미합니다.

참 고

□ Manual Pulser 모드 모션에서의 속도는 입력 펄스의 주파수에 의해 결정됩니다. 따라서 속도모드나 가/감속 등은 설정할 필요가 없습니다. 그러나 Manual Pulser 입력신호의 최대 주파수는 McSetSpeed()메소드에 의해 설정되는 작업 속도에 의하여 제한됩니다. 따라서 Manual Pulser 모션을 수행하기 전에 McSetSpeed()메소드를 이용하여 작업속도를 적절한 값으로 설정하여야 합니다.

□ Manual Pulser 모드 모션을 취소하기 위해서는 McStop() 메소드대신 McEmgStop()메소드를 사용하여야 합니다.

예 제

```
Call ComiLxAx1.LoadDevice
Const X_AXIS = 0
Const PHASE_1X = 0 ` Pulser input mode => 1X A/B (1 채배 Phase type 입력 모드)

Call ComiLxAx1.McReset
` Pulser 입력신호의 최대 주파수 제한 설정(650000PPS)
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, 650000)
Call ComiLxAx1.McSetPulserInputMode(X_AXIS, PHASE_1X, FALSE)
Call ComiLxAx1.McStartPulserMoveTo(X_AXIS, 1000)
While Not ( ComiLxAx1.McDone())
Wend

Call ComiLxAx1.UnloadDevice
```

or

■ McPulserMoveTo

메소드 원형

Sub **McPulserMoveTo** (ByVal Channel As Long, ByVal Position As Double)

메소드 설명

이 메소드는 Pulser 신호 입력에 맞추어 지정한 위치(절대좌표)로 이동을 수행합니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **Position** : 이동할 목표 위치(절대좌표값)를 지정합니다. 위치에 대한 단위는 McSetUnitDistance 메소드에 의해 결정됩니다. McSetUnitDistance 메소드로 거리의 단위를 변경하지 않았다면 거리의 단위는 Pulse 수가 됩니다. 즉, 거리값 1 은 1 Pulse 출력을 의미합니다.

참 고

□ 이 메소드는 모션이 완료되기 전까지 Return 되지 않고 루프를 돌게 됩니다. 루프를 도는 동안 윈도우 이벤트나 메시지가 처리될 수 있도록 하려면 이 메소드를 수행하기 이전에 McSetBlockingMode 메소드를 사용하여 Blocking 이 일어나지 않도록 설정하여야 합니다.

□ Manual Pulser 모드 모션에서의 속도는 입력 펄스의 주파수에 의해 결정됩니다. 따라서 속도모드나 가/감속 등은 설정할 필요가 없습니다. 그러나 Manual Pulser 입력신호의 최대 주파수는 McSetSpeed()메소드에 의해 설정되는 작업 속도에 의하여 제한됩니다. 따라서 Manual Pulser 모션을 수행하기 전에 McSetSpeed()메소드를 이용하여 작업속도를 적절한 값으로 설정하여야 합니다.

예 제

```
Call ComiLxAx1.LoadDevice

Const X_AXIS = 0
Const PHASE_1X = 0 ` Pulser input mode => 1X A/B (1 채널 Phase type 입력 모드)

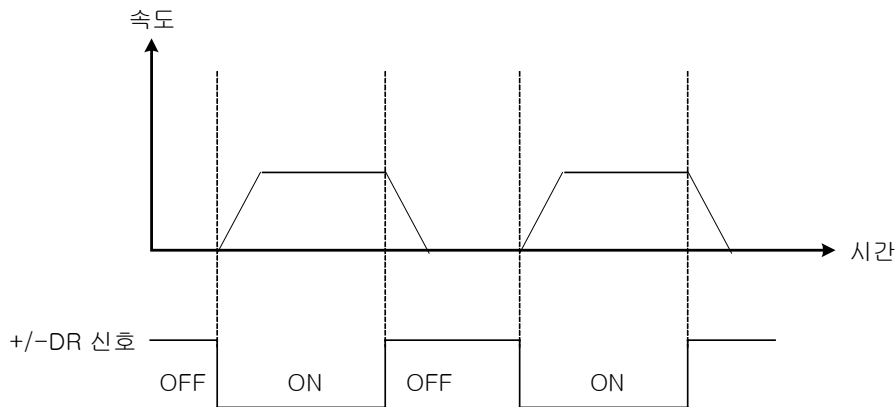
Call ComiLxAx1.McReset
` Pulser 입력신호의 최대 주파수 제한 설정(650000PPS)
Call ComiLxAx1.McSetSpeed(X_AXIS, 0, 650000)
Call ComiLxAx1.McSetPulserInputMode(X_AXIS, PHASE_1X, FALSE)
Call ComiLxAx1.McPulserMoveTo(X_AXIS, 1000)

Call ComiLxAx1.UnloadDevice
```

2.6.8 External Switch Operation 모드 모션 제어 함수

이 단원에서는 External Switch Operation 모드에 관련된 함수들을 소개합니다. 여기서 External Switch 는 +DR 과 -DR 신호를 의미합니다. External Switch Operation 모드는 +DR 또는 -DR 신호가 ON 상태인 경우에만 모션을 구동하고 OFF 인 상태에서는 모션을 정지하는 기능을 말합니다. 이 때의 속도 패턴은 Single Axis Motion 과 같이 McSetSpeedMode, McSetSpeed, McSetAccel 의 함수에 의해 설정된 속도 패턴을 사용합니다. +DR 신호는 정방향으로 모션을 구동시키고 -DR 신호는 역방향으로 모션을 구동시킵니다.

+DR 또는 -DR 신호와 모션 구동의 관계는 다음 그림과 같습니다.



[그림 5-19] External Switch Operation

External Switch Operation 모드 모션에 관련된 함수들은 다음과 같습니다.

메소드 / 설명	페이지
sub McSetPE (ByVal Channel As Long, ByVal Enable As Short) Manual Pulser 나 External Switch Operation 모드를 Enable/Disable 시킵니다.	222
sub McStartVMoveEx (ByVal Channel As Long) +/-DR 신호가 ON 상태가 되면 Velocity Move 모션을 수행합니다	223
sub McStartMoveEx (ByVal Channel As Long, ByVal Distance As Double) External switch operation 모드하에 Relative In-Position(상대좌표 위치결정) 모션을 시작합니다.	224

■ McSetPE

함수 원형

sub **McSetPE** (ByVal Channel As Long, ByVal Enable As Short)

함수 설명

이 함수는 Manual Pulser 모드나 External Switch Operation 모드를 Enable/Disable 시킵니다. Manual Pulser 모드를 사용하기 위해서는 먼저 이 함수를 이용하여 Manual Pulser 모드를 Enable 시켜야 합니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **Enable** : Manual Pulser 모드와 External Switch Operation 모드를 Enable/Disable 시킵니다.

Value	Meaning
0	Manual Pulser 모드와 External Switch Operation 모드 Disable
1	Manual Pulser 모드와 External Switch Operation 모드 Enable

■ McStartVMoveEx

함수 원형

sub **McStartVMoveEx** (ByVal Channel As Long)

함수 설명

+/-DR 신호가 ON 상태가 되면 Velocity Move 모션을 수행합니다. +/-DR 신호가 OFF 가 되면 모션을 정지합니다. 이 때의 구동 방향은 +DR 신호와 -DR 신호에 의해 결정됩니다.

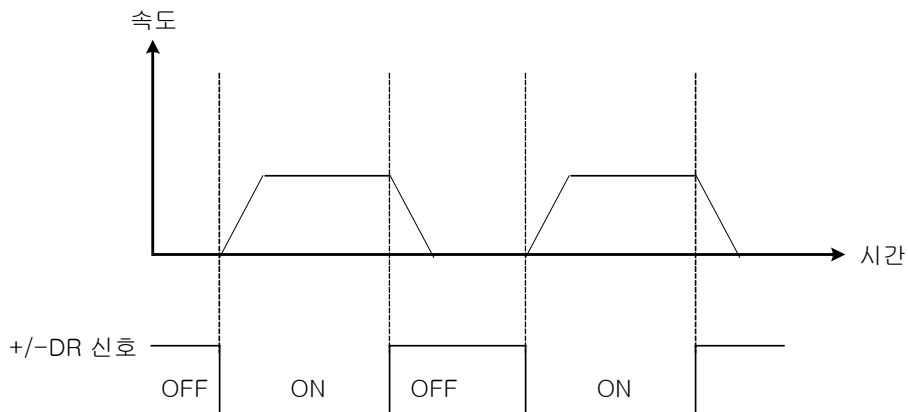
매개 변수

▶ *Channel* : 채널(축) 번호, 0 ~ 3

참 고

□ 속도 모드를 Constant Speed Mode 로 지정한 경우에는 가속 구간이 없이 작업속도로 모션을 시작합니다.

□ +DR 또는 -DR 신호와 모션 구동의 관계는 다음 그림과 같습니다.



■ McStartMoveEx

함수 원형

sub **McStartMoveEx** (ByVal Channel As Long, ByVal Distance As Double)

함수 설명

External switch operation 모드하에 Relative In-Position(상대좌표 위치결정) 모션을 시작합니다. +/-DR 신호가 ON 상태일 때에만 모터를 구동합니다. +DR 신호가 ON 이 되면 정방향으로 회전하고 -DR 신호가 ON 이 되면 역방향으로 회전합니다.

지정한 거리만큼 이동이 완료되면 +/-DR 신호와 관계없이 모션을 종료합니다.

매개 변수

▶ *Channel* : 채널(축) 번호, 0 ~ 3

참 고

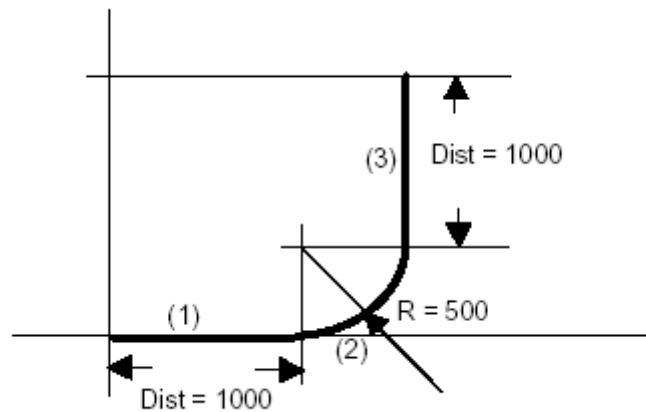
□ 속도 모드를 Constant Speed Mode 로 지정한 경우에는 가속 구간이 없이 작업속도로 모션을 시작합니다.

□ +DR 또는 -DR 신호와 모션 구동의 관계는 McStartVMoveEx 때와 같습니다. 단, 지정한 거리만큼 이동이 완료되면 +/-DR 신호가 ON 상태이어도 모션을 정지합니다.

2.6.9 리스트 모션(Listed Motion) 메쏘드

이 단원에서는 리스트 모션(Listed Motion)제어에 관련된 메쏘드들을 소개합니다. 리스트 모션은 수행해야할 여러 단계의 작업을 리스트로 등록시킨 후에 일괄적으로 처리하는 기능을 말합니다. 리스트 모션의 장점은 하나의 작업과 그 다음 작업간에 지연시간(Delay)이 없이 연속적인 작업을 수행할 수 있도록 한다는 것입니다. 또한 리스트 모션을 사용하면 Move 나 MoveTo 메쏘드와 같은 In-Position 메쏘드를 사용할 때에도 작업 속도의 연속성을 확보할 수 있습니다(적용예 2 참조).

■ 리스트 모션의 적용 예 1



[그림 #] 3 회의 Coordinated Motion 으로 이루어지는 작업의 예

[그림 #]과 같이 3 회의 Coordinated Motion 을 연속적으로 수행하고자할 때 다음과 같이 리스트 모션을 적용하면 각 작업간의 Delay 가 최소화되어 연속적인 작업을 수행할 수 있습니다.

```
Dim DistList(1)
```

```
Call ComiLxAx1.McBeginList ' 모션 리스트 등록 시작
Call ComiLxAx1.McMapAxes(0, 0x3) ' Map X & Y axis
' Set Trapezoidal Speed Mode
Call ComiLxAx1.McSetSpeedModeMx(0, 1)
' Set work speed=500, Acceleration=1000
Call ComiLxAx1.McSetSpeedMx(0, 500, 1000)
' (1000,0)만큼 직선 보간 이동
DistList(0) = 1000
DistList(1) = 0

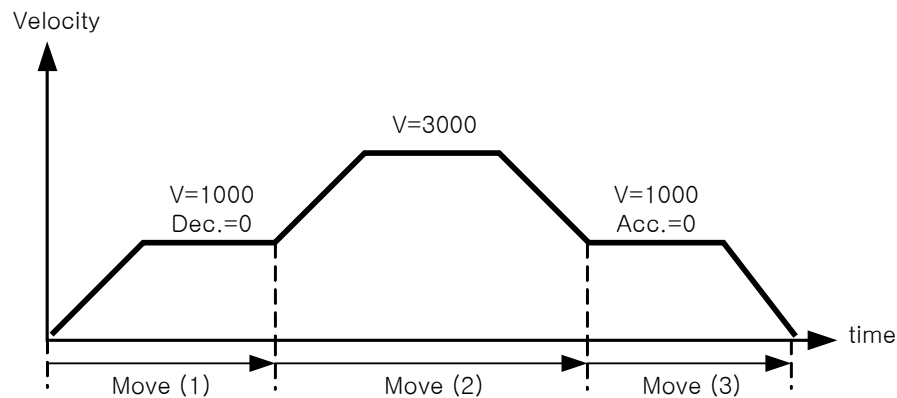
Call ComiLxAx1.McLine(0, DistList(0))
' 중심점 Offset = (0, 500), End Angle = 90 DEG
' 의 조건으로 원호보간 이동
Call ComiLxAx1.McArcA(0, 0, 500, 90)
```

ur

```

` (0,1000)만큼 직선 보간 이동
DistList(0) = 0
DistList(1) = 1000
Call ComiLxAx1.McLine(0, DistList(0))
Call ComiLxAx1.McEndList ` 모션 리스트 등록을 마칩
Call ComiLxAx1.McStartListMotion ` 리스트 모션 수행
` 리스트 모션이 모두 완료될 때까지 기다림
While Not ( ComiLxAx1.McChekcListMotionDone)
Wend
    
```

■ 리스트 모션의 적용 예 2



작업	초기속도	작업속도	Acceleration	Deceleration
Move (1)	0	1000	2000	0
Move (2)	1000	3000	2000	2000
Move (3)	0	1000	0	2000

[그림 #] 속도의 연속성을 가지는 연속적인 In-Position 모션

리스트 모션을 이용하면 [그림 #]과 같이 In-Position 의 경우에도 작업 속도의 연속성을 확보할 수 있습니다. 다음의 코드는 [그림 #]의 작업을 리스트 모션을 적용하여 구현하는 예입니다.

```

Call ComiLxAx1.McBeginList ` 모션 리스트 등록 시작
` Set Trapezoidal Speed Mode
Call ComiLxAx1.McSetSpeedMode(0, 1)
` Move (1)
Call ComiLxAx1.McSetSpeed(0, 0, 1000)
Call ComiLxAx1.McSetAccel(0, 2000, 0)
Call ComiLxAx1.McMoveTo(0, 3000)
` Move (2)
Call ComiLxAx1.McSetSpeed(0, 1000, 3000)
Call ComiLxAx1.McSetAccel(0, 2000, 2000)
Call ComiLxAx1.McMoveTo(0, 8000)
` Move (3)
Call ComiLxAx1.McSetSpeed(0, 0, 1000)
Call ComiLxAx1.McSetAccel(0, 0, 2000)
    
```

```

Call ComiLxAx1.McMoveTo(0, 11000)
Call ComiLxAx1.McEndList   ' 모션 리스트 등록을 마칩

Call ComiLxAx1.McStartListMotion ' 리스트 모션 수행
' 리스트 모션이 모두 완료될 때까지 기다림
While Not ( ComiLxAx1.McCheckListMotionDone)
Wend
    
```

리스트 모션에 관련된 메소드들은 다음과 같습니다.

메소드 / 설명	페이지
sub McSetListMotionAxes (short MapMask) 리스트모션에서 사용되는 모든 축을 등록하는 함수입니다.	228
Sub McBeginList List Motion 에서 수행할 작업들의 등록을 시작합니다.	229
Sub McEndList List Motion 에서 수행할 작업들의 등록을 종료합니다.	230
Sub McStartListMotion List Motion 으로 등록된 작업들에 대하여 실제로 모션을 시작합니다.	231
Sub McAbortListMotion 리스트 모션을 수행하고 있는 중에 리스트 모션을 중지합니다.	232
Function McCheckListMotionDone As Boolean 리스트 모션 수행이 완료됐는지를 알려줍니다.	233

■ McSetListMotionAxes

함수 원형

```
sub McSetListMotionAxes (short MapMask)
```

함수 설명

이 함수는 리스트모션에서 사용되는 모든 축을 등록하는 함수입니다. 리스트모션을 등록하기 전에 먼저 리스트모션에서 수행되는 작업과 관련된 모든축을 먼저 등록하여야 합니다.

매개 변수

- ▶ *Channel* : 채널(축) 번호(0 부터 시작합니다).
- ▶ *MapMask* : 그룹에 포함할 축들을 지정할 마스크 값. 이 값의 BIT0~BIT3 을 이용하여 그룹에 포함할 축들을 지정합니다. 각 비트의 값이 0 이면 해당 축(비트의 순서와 일치하는 축)은 배제되는 것이며 1 이면 해당 축이 포함되는 것입니다.

■ McBeginList

메소드 원형

Sub **McBeginList**

메소드 설명

이 메소드는 List Motion 에서 수행할 작업들의 등록을 시작합니다.

or

■ McEndList

메소드 원형

Sub McEndList

메소드 설명

이 메소드는 List Motion 에서 수행할 작업들의 등록을 종료합니다.

■ McStartListMotion

메소드 원형

Sub **McStartListMotion**

메소드 설명

이 메소드는 List Motion 으로 등록된 작업들에 대하여 실제로 모션을 시작합니다.

■ McAbortListMotion

메소드 원형

Sub **McAbortListMotion**

메소드 설명

이 메소드는 McStartListMotion()을 이용하여 리스트 모션을 수행하고 있는 중에 리스트 모션을 중지하고자 할 때 사용합니다. 리스트 모션을 시작하면 등록된 작업들이 모두 수행되면 자동으로 중지됩니다. McAbortListMotion() 메소드는 리스트 모션이 진행되고 있는 중에 리스트 모션을 중지해야할 때 사용합니다.

■ McCheckListMotionDone

메소드 원형

Function **McCheckListMotionDone** As Boolean

메소드 설명

이 메소드는 리스트 모션 수행이 완료됐는지를 알려주는 메소드입니다.

Return 값

Value	Meaning
0	리스트 모션이 완료되지 않음
1	등록된 모든 리스트 모션이 완료됨

2.6.10 상태 감시 및 제어 메소드

이 단원에서는 상태 감시 및 제어 메소드에 관하여 설명합니다. 상태 감시 및 제어 메소드들은 모션의 상태를 감시하는데 필요한 메소드들을 그룹화한 것입니다. 상태 감시에는 모션의 속도, 위치 등을 체크하는 것을 포함하며 이외에도 현재 모션이 진행되고 있는지를 체크하고, 현재 진행되고 있는 모션의 단계 및 각 I/O 상태들을 체크하는 기능도 포함합니다.

상태 감시 및 제어 메소드에 관련된 메소드들은 다음과 같습니다.

메소드 / 설명	페이지
Function McGetCurSpeed (ByVal Channel As Long) As Double 현재 출력되고 있는 Command 속도를 반환합니다.	236
Sub McEnableActSpdChk (ByVal Interval As Long) 실제 모션의 속도를 체크하는 기능을 Enable 시킵니다.	237
Sub McDisableActSpdChk 실제 모션의 속도(Actual Speed)를 체크하는 기능을 Disable 시킵니다.	238
Function McGetActualSpeed (ByVal Channel As Long) As Double EA/EB 단자로 입력되는 Feedback 펄스를 계측하여 실제 모션의 속도를 반환합니다.	239
Function McGetPositionA (ByVal Channel As Long) As Double 현재의 실제 위치(Actual Position)를 논리 단위로 반환합니다.	240
Sub McSetPositionA (ByVal Channel As Long, ByVal Pos As Double) 현재의 실제 위치(Actual Position)값을 설정하며 단위는 논리단위입니다.	241
Function McGetPositionC (ByVal Channel As Long) As Double 현재의 명령 위치(Command Position)를 논리 단위로 반환합니다.	242
Sub McSetPositionC (ByVal Channel As Long, ByVal Pos As Double) 명령 위치(Command Position)값을 설정하며 단위는 논리단위입니다.	243
Function McGetCountA (ByVal Channel As Long) As Long 현재의 실제 위치(Actual Position) 카운트값을 반환합니다.	244
Sub McSetCountA (ByVal Channel As Long, ByVal Count As Long) 현재의 실제 위치(Actual Position) 카운트값을 설정합니다.	245
Function McGetCountC (ByVal Channel As Long) As Long 현재의 명령 위치(Command Position) 카운트값을 반환합니다.	246
Sub McSetCountC (ByVal Channel As Long, ByVal Count As Long) 현재의 명령 위치(Command Position) 카운트값을 설정합니다.	247

Function McGetCountD (ByVal Channel As Long) As Long Deviation Counter 의 값을 읽어서 반환합니다.	248
Function McSetCountD (ByVal Channel As Long, ByVal Count As Long) As Long Deviation Counter 의 값을 새로이 설정합니다.	249
Function McGetCountG (ByVal Channel As Long) As Long General Counter 의 값을 읽어서 반환합니다.	250
Sub McSetCountG (ByVal Channel As Long, ByVal Count As Long) General Counter 의 값을 새로이 설정합니다.	251
Function McGetCountR (ByVal Channel As Long) As Long Remained Counter 의 값을 읽어서 반환합니다.	252
Sub McSetCountR (ByVal Channel As Long, ByVal Count As Long) Remained Counter 의 값을 새로이 설정합니다.	253
Function McGetLatchState (ByVal Channel As Long) As Boolean 현재의 래치 상태를 반환합니다. 사용자는 이 함수를 이용하여 래치상태를 체크한 후 래치되었으면 카운트값을 읽어야 합니다.	254
Function McReadLatchCounter (ByVal Channel As Long, ByVal Counter As Long) As Long 래치된 카운트값을 반환합니다.	255
Function McGetMotionStatus (ByVal Channel As Long) As Long 현재 모션의 동작 상태를 반환합니다.	256
Function McGetMioStatus (ByVal Channel As Long) As Long 현재 모션과 관련된 여러가지 I/O 상태를 반환합니다.	257

■ McGetCurSpeed

메소드 원형

Function **McGetCurSpeed** (ByVal Channel As Long) As Double

메소드 설명

이 메소드는 현재 출력되고 있는 Command 속도를 반환합니다.

매개 변수

▶ *Channel* : 채널(축) 번호, 0 ~ 3

Return 값

지정한 축의 Command 속도를 반환합니다. 이 값은 모터의 실제속도가 아니며 스텝모터 또는 서보모터 드라이버에 전달되는 Command 속도입니다. 또한 이 값의 단위는 단위는 McSetUnitSpeed() 메소드에 의하여 결정되며 기본적으로는 Pulses/sec 입니다.

■ McEnableActSpdChk

메소드 원형

Sub **McEnableActSpdChk** (ByVal Interval As Long)

메소드 설명

이 메소드는 실제 모션의 속도를 체크하는 기능을 Enable 시킵니다. 모션의 실제속도는 Feedback 펄스수를 주기적으로 체크하여 펄스수의 변화량을 시간으로 나누어 계산됩니다. McGetActualSpeed()메소드를 사용하기전에 먼저 이메소드를 사용하여 Actual Speed 체크 기능을 Enable 시켜야합니다.

매개 변수

▶ **Channel** : 채널(축) 번호, 0 ~ 3

▶ **Interval** : Feedback 펄스의 수를 체크하는 주기를 msec 단위로 설정합니다. Feedback 펄스의 주파수는 다음과 같은 식에 의해 계산됩니다.

$$V_a = \frac{\Delta C}{\Delta T} \times \frac{1}{R_u \times R_{io}}$$

여기서,

V_a : Feedback 펄스를 통하여 계측되는 모션의 실제 속도

ΔC : 체크 주기동안에 변화된 Feedback counter 값

ΔT : 체크주기, dwInterval 이 msec 단위로 설정되기 때문에 dwInterval/1000 를 의미합니다.

R_u : McSetUnitSpeed()메소드를 통하여 설정된 단위속도당 PPS 비

R_{io} : McSetInOutRatio()메소드를 통하여 설정된 Command 펄스와 Feedback 펄스의 분해능 비율(Resolution ratio)

■ McDisableActSpdChk

메소드 원형

Sub **McDisableActSpdChk**

메소드 설명

이 메소드는 실제 모션의 속도(Actual Speed)를 체크하는 기능을 Disable 시킵니다. Actual Speed 를 체크할 필요가 없는 경우에는 Actual Speed Check 기능을 Disable 시키는 것이 좋습니다. 컴퓨터가 부팅되면 Actual Speed Check 기능은 기본적으로 Disable 됩니다.

■ McGetActualSpeed

메소드 원형

Function **McGetActualSpeed** (ByVal Channel As Long) As Double

메소드 설명

이 메소드는 EA/EB 단자로 입력되는 Feedback 펄스를 계측하여 실제 모션의 속도를 반환합니다.

매개 변수

▶ *Channel* : 채널(축) 번호, 0 ~ 3

Return 값

지정한 축의 Command 속도를 반환합니다. 이 값은 모터의 실제속도가 아니며 스텝모터 또는 서보모터 드라이버에 전달되는 Command 속도입니다. 또한 이 값의 단위는 단위는 **McSetUnitSpeed()** 메소드에 의하여 결정되며 기본적으로 Pulses/sec 입니다.

참 고

□ 이 메소드를 사용하기 전에 McEnableActSpdChk()메소드를 이용하여 Actual Speed 체크 기능을 Enable 시켜야합니다.

□ Feedback 펄스의 주파수는 다음과 같은 식에 의해 계산됩니다.

$$V_a = \frac{\Delta C}{\Delta T} \times \frac{1}{R_u \times R_{io}}$$

여기서,

V_a : Feedback 펄스를 통하여 계측되는 모션의 실제 속도

ΔC : 체크 주기동안에 변화된 Feedback counter 값

ΔT : 체크주기, dwInterval 이 msec 단위로 설정되기 때문에 dwInterval/1000 를 의미합니다.

R_u : McSetUnitSpeed()메소드를 통하여 설정된 단위속도당 PPS 비

R_{io} : McSetInOutRatio()메소드를 통하여 설정된 Command 펄스와 Feedback 펄스의 비

■ McGetPositionA

메소드 원형

Function **McGetPositionA** (ByVal Channel As Long) As Double

메소드 설명

이 메소드는 현재의 실제 위치(Actual Position)를 논리 단위로 반환합니다.

매개 변수

▶ *Channel* : 채널(축) 번호, 0 ~ 3

Return 값

현재의 실제 위치(Actual Position)를 논리 단위로 반환합니다. 논리 단위는 McSetUnitDistance()메소드에 의해 결정됩니다.

참 고

□ 실제 위치는 EA/EB 단자로 입력되는 Feedback 펄스를 카운트하여 계산됩니다.

□ Feedback 펄스의 카운트값과 논리적 실제 위치의 관계는 다음과 같습니다..

$$P_a = \frac{C}{R_u \times R_{io}}$$

여기서,

P_a : Actual Position

C : Feedback Count

R_u : McSetUnitSpeed()메소드를 통하여 설정된 단위속도당 PPS 비

R_{io} : McSetInOutRatio()메소드를 통하여 설정된 Command 펄스와 Feedback 펄스의 분해능 비율(Resolution ratio)

■ McSetPositionA

메소드 원형

Sub **McSetPositionA** (ByVal Channel As Long, ByVal Pos As Double)

메소드 설명

이 메소드는 현재의 실제 위치(Actual Position)값을 설정하며 단위는 논리단위입니다.

매개 변수

- ▶ *Channel* : 채널(축) 번호, 0 ~ 3
- ▶ *Pos* : 새로이 설정할 실제 위치(Actual Positon)값. 이 값의 단위는 논리 단위이며 McSetUnitDistance 메소드에 의해 결정됩니다.

or

■ McGetPositionC

메소드 원형

Function **McGetPositionC** (ByVal Channel As Long) As Double

메소드 설명

이 메소드는 현재의 명령 위치(Command Position)를 논리 단위로 반환합니다.

매개 변수

▶ *Channel* : 채널(축) 번호, 0 ~ 3

Return 값

현재의 명령 위치(Command Position)를 논리 단위로 반환합니다. 논리 단위는 McSetUnitDistance()메소드에 의해 결정됩니다.

■ McSetPositionC

메소드 원형

Sub **McSetPositionC** (ByVal Channel As Long, ByVal Pos As Double)

메소드 설명

이 메소드는 명령 위치(Command Position)값을 설정하며 단위는 논리단위입니다.

매개 변수

- ▶ *Channel* : 채널(축) 번호, 0 ~ 3
- ▶ *Pos* : 새로이 설정할 명령 위치(Actual Positon)값. 이 값의 단위는 논리 단위이며 McSetUnitDistance 메소드에 의해 결정됩니다.

or

■ McGetCountA

메소드 원형

Function **McGetCountA** (ByVal Channel As Long) As Long

메소드 설명

이 메소드는 현재의 실제 위치(Actual Position) 카운트값을 반환합니다. 이 값은 논리단위가 아닌 실제 펄스 카운트값입니다.

매개 변수

▶ *Channel* : 채널(축) 번호, 0 ~ 3

Return 값

현재의 실제 위치(Actual Position) 카운트값을 반환합니다. 이 값은 논리단위가 아닌 실제 펄스 카운트값입니다.

참 고

□ McGetPositionA 메소드가 McSetUnitDistance 메소드에 의해 결정되는 논리단위값을 기준으로 위치값을 반환하는데 반하여 McGetCountA 메소드는 순수한 펄스 카운트값을 반환합니다.

■ McSetCountA

메소드 원형

Sub **McSetCountA** (ByVal Channel As Long, ByVal Count As Long)

메소드 설명

이 메소드는 현재의 실제 위치(Actual Position) 카운트값을 설정합니다. 이 값은 논리단위가 아닌 실제 펄스 카운트값입니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **Count** : 새로이 설정할 실제 위치(Actual Position) 카운트값. 이 값은 논리단위가 아닌 실제 펄스 카운트값입니다.

or

■ McGetCountC

메소드 원형

Function **McGetCountC** (ByVal Channel As Long) As Long

메소드 설명

이 메소드는 현재의 명령 위치(Command Position) 카운트값을 반환합니다. 이 값은 논리단위가 아닌 실제 펄스 카운트값입니다.

매개 변수

▶ *Channel* : 채널(축) 번호, 0 ~ 3

Return 값

현재의 명령 위치(Command Position) 카운트값을 반환합니다. 이 값은 논리단위가 아닌 실제 펄스 카운트값입니다.

참 고

□ McGetPositionC 메소드가 McSetUnitDistance 메소드에 의해 결정되는 논리단위값을 기준으로 위치값을 반환하는데 반하여 McGetCountC 메소드는 순수한 펄스 카운트값을 반환합니다.

■ McSetCountC

메소드 원형

Sub **McSetCountC** (ByVal Channel As Long, ByVal Count As Long)

메소드 설명

이 메소드는 현재의 명령 위치(Command Position) 카운트값을 설정합니다. 이 값은 논리단위가 아닌 실제 펄스 카운트값입니다.

매개 변수

- ▶ *Channel* : 채널(축) 번호, 0 ~ 3
- ▶ *Count* : 새로이 설정할 명령 위치(Command Position) 카운트값. 이 값은 논리단위가 아닌 실제 펄스 카운트값입니다.

■ McGetCountD

메소드 원형

Function **McGetCountD** (ByVal Channel As Long) As Long

메소드 설명

이 메소드는 Deviation Counter 의 값을 읽어서 반환합니다. Deviation Counter 는 명령 위치(Command Position)와 실제 위치(Actual Position)간의 차이(Difference)를 카운트하는 카운터입니다.

매개 변수

▶ *Channel* : 채널(축) 번호, 0 ~ 3

Return 값

현재의 명령 위치(Command Position)와 실제 위치(Actual Position)간의 차이(Difference)를 카운트하는 카운터의 값을 읽어서 반환합니다. 이 값은 논리단위가 아니며 순수한 펄스 수의 차를 나타냅니다. 만일 Command Pulse 와 Feedback Pulse 간의 분해능(Resolution)이 다르다면 이에 대한 보상은 이루어지지 않으므로 이 메소드를 사용하는 것은 바람직하지 않습니다.

■ McSetCountD

메소드 원형

Function **McSetCountD** (ByVal Channel As Long, ByVal Count As Long) As Long

메소드 설명

이 메소드는 Deviation Counter 의 값을 새로이 설정합니다. Deviation Counter 는 명령 위치(Command Position)와 실제 위치(Actual Position)간의 차이(Difference)를 카운트하는 카운터입니다.

매개 변수

- ▶ *Channel* : 채널(축) 번호, 0 ~ 3
- ▶ *Count* : 새로이 설정할 Deviation Counter 의 카운트값. 이 값은 논리단위가 아닌 실제 펄스 카운트값입니다.

or

■ McGetCountG

메소드 원형

Function **McGetCountG** (ByVal Channel As Long) As Long

메소드 설명

이 메소드는 General Counter 의 값을 읽어서 반환합니다.

매개 변수

▶ *Channel* : 채널(축) 번호, 0 ~ 3

Return 값

General Counter 의 카운트값.

■ McSetCountG

메소드 원형

Sub **McSetCountG** (ByVal Channel As Long, ByVal Count As Long)

메소드 설명

이 메소드는 General Counter 의 값을 새로이 설정합니다.

매개 변수

- ▶ *Channel* : 채널(축) 번호, 0 ~ 3
- ▶ *Count* : 새로이 설정할 General Counter 의 카운트값.

or

■ McGetCountR

메소드 원형

Function **McGetCountR** (ByVal Channel As Long) As Long

메소드 설명

이 메소드는 Remained Counter 의 값을 읽어서 반환합니다. Remained Counter 는 In-Position 작업에서 현재 남은 출력 펄스의 수를 알려주는 카운터입니다.

매개 변수

▶ *Channel* : 채널(축) 번호, 0 ~ 3

Return 값

Remained Counter 의 값을 읽어서 반환합니다.

■ McSetCountR

메소드 원형

Sub **McSetCountR** (ByVal Channel As Long, ByVal Count As Long)

메소드 설명

이 메소드는 Remained Counter 의 값을 새로이 설정합니다. Remained Counter 는 In-Position 작업에서 현재 남은 출력 펄스의 수를 알려주는 카운터입니다.

매개 변수

- ▶ *Channel* : 채널(축) 번호, 0 ~ 3
- ▶ *Count* : 새로이 설정할 General Counter 의 카운트값.

■ McGetLatchState

함수 원형

Function **McGetLatchState** (ByVal Channel As Long) As Boolean

함수 설명

현재의 래치 상태를 반환합니다. 사용자는 이 함수를 이용하여 래치상태를 체크한 후 래치 되었으면 카운트값을 읽어야 합니다.

매개 변수

▶ *Channel* : 채널(축) 번호(0 부터 시작합니다)

반환값

현재 래치 상태를 반환합니다.

참 고

- 이 함수가 제대로 사용되기 위해서는 McMaskInterrupt 함수를 이용하여 BIT14 를 마스크 (1 로 만들어줌) 해주어야 합니다.
- 래치상태를 읽으면 이전의 래치상태는 리셋(0)됩니다.
- McGetIntStatus 함수를 이용해서도 래치상태를 체크할 수 있으며, McGetIntStatus 함수를 수행하면 마찬가지로 이전의 래치상태는 리셋(0)됩니다.

■ McReadLatchCouner

함수 원형

Function **McReadLatchCouner** (ByVal Channel As Long, ByVal Counter As Long) As Long

함수 설명

래치된 카운트값을 반환합니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호(0 부터 시작합니다)
- ▶ **Counter** : 읽을 래치 카운터를 지정합니다. 이 값은 다음과 같습니다.

Value	Meaning
0	명령위치 카운터(Command position counter)
1	실제위치 카운터(Feedback position counter)
2	Deviation 또는 펄스 출력 속도.
3	General Counter

반환값

지정한 래치 카운터의 카운트값을 반환합니다.

참 고

- LTC2 카운터는 McSetMioCfgrLTC 함수에서의 설정에 따라 Deviation counter 또는 펄스 출력 속도가 될 수 있습니다.

■ McGetMotionStatus

메소드 원형

Function **McGetMotionStatus** (ByVal Channel As Long) As Long

메소드 설명

이 메소드는 현재 모션의 동작 상태를 반환합니다.

매개 변수

► *Channel* : 채널(축) 번호, 0 ~ 3

Return 값

현재 모션의 동작 상태를 반환합니다. 반환 값의 의미는 다음과 같습니다.

Value	Meaning
0	Stop
1	Wait for DR input
2	Wait for STA input
3	Reserved
4	Wait other axis
5	Wait ERC finished
6	Wait DIR change
7	Reserved
8	Wait PA/PB
9	In home special speed motion
10	In start velocity motion
11	In acceleration
12	In working velocity
13	In deceleration
14	Wait INP
15	Reserved

■ McGetMioStatus

메소드 원형

Function **McGetMioStatus**(ByVal Channel As Long) As Long

메소드 설명

이 메소드는 현재 모션과 관련된 여러가지 I/O 상태를 반환합니다.

매개 변수

▶ *Channel* : 채널(축) 번호, 0 ~ 3

Return 값

모션과 관련된 여러가지 I/O 상태를 32 비트 값으로 반환합니다. 반환되는 값의 각 비트의 값은 다음의 표와 같이 특정 I/O 핀의 상태를 나타냅니다.

Bit	Name	
BIT0	RDY	Servo ready signal input status(1=ON), 단 COMI-LX501에서는 지원되지 않음
BIT1	ALM	Alarm signal status(1=ON)
BIT2	+EL	Positive limit switch status(1=ON)
BIT3	-EL	Negative limit switch status(1=ON)
BIT4	ORG	Orgin switch status(1=ON)
BIT5	DIR	Operating direction status(1=ON)
BIT6	Reserved	
BIT7	PCS	PCS signal input status(1=ON)
BIT8	ERC	ERC pin output status(1=ON)
BIT9	EZ	Index signal status(1=ON)
BIT10	CLR	Clear input status(1=ON)
BIT11	Latch	Latch signal input status(1=ON)
BIT12	SD	Slow Down signal input status(1=ON)
BIT13	INP	In-Position signal input status(1=ON)
BIT14	DRP	+DR input signal status(1=ON)
BIT15	DRN	-DR input signal status(1=ON)
BIT16	STA	STA input signal status(1=ON)
BIT17	STP	STP input signal status(1=ON)
BIT16 ~BIT31	Reserved	

2.6.11 I/O(입출력) 환경설정 메소드

이 단원에서는 I/O(입출력) 환경설정 메소드에 관하여 설명합니다. COMI-SD501 모션제어 보드는 General Digital Input/Output 이외에 여러가지 모션 제어에 필요한 I/O 핀을 제공합니다. 여기에는 알람 신호(Alarm, ALM), 리미트 신호(Limit, EL) 등이 포함되며 서보 모터의 경우에는 INP, ERC 신호가 포함됩니다. 그리고 여러가지 Comparator(비교기) 또한 여기에 포함됩니다.

I/O(입출력) 환경설정에 관련된 메소드들은 다음과 같습니다.

메소드 / 설명	페이지
sub McSetFilterLogic (ByVal Channel As Long, ByVal Enable As Short) 각종 I/O 신호에 필터 로직을 적용할지를 설정하는 함수입니다.	261
sub McSetELL (ByVal Channel As Long, ByVal Logic As Long) +EL, -EL 신호의 입력 로직을 설정합니다.	262
Sub McSetMioCfgALM (ByVal Channel As Long, AlarmLogic As Long, AlarmMode As Long) Alarm 신호에 대한 입력모드를 설정합니다.	263
Sub McGetMioCfgALM (ByVal Channel As Long, ByRef AlarmLogic As Long, ByRef AlarmMode As Long) 현재 설정된 Alarm 신호에 대한 입력모드를 읽어옵니다.	264
sub McSetMioCfgCLR (ByVal Channel As Long, ByVal CntrSel As Long, ByVal SignalType As Long) 이 함수는 CLR 신호에 대한 환경을 설정합니다	265
sub McGetMioCfgCLR (ByVal Channel As Long, ByRef CntrSel As Long, ByRef SignalType As Long) 이 함수는 CLR 신호에 대한 모드 설정값을 읽어옵니다.	266
sub McSetMioCfgCMP (ByVal Channel As Long, ByVal CmpPulseWidth As Long, ByVal CmpLogic As Long) 위치비교 출력 펄스(CMP 신호)의 출력 방식을 설정합니다.	267
sub McGetMioCfgCMP (ByVal Channel As Long, ByRef CmpPulseWidth Long, ByRef CmpLogic As Long) 위치비교 출력 펄스(CMP 신호)의 출력 설정값을 얻어옵니다.	268
sub McSetMioCfgDR (ByVal Channel As Long, ByVal Logic As Long) 이 함수는 +/-DR 신호에 대한 환경을 설정합니다.	269
sub McGetMioCfgDR (ByVal Channel As Long, ByRef Logic As Long)	270

이 함수는 +/-DR 신호에 대한 모드 설정값을 읽어옵니다.	
Sub McSetMioCfgEL (ByVal Channel As Long, ByVal EIMode As Long) EL 신호에 대한 모드를 설정합니다.	271
Sub McGetMioCfgEL (ByVal Channel As Long, ByRef EIMode As Long) EL 신호에 대한 모드 설정값을 읽어옵니다.	272
Sub McSetMioCfgINP (ByVal Channel As Long, ByVal InpEnable As Integer, ByVal InpLogic As Long) INP 기능 및 신호에 대한 환경을 설정합니다.	273
Sub McGetMioCfgINP (ByVal Channel As Long, ByRef pinpEnable As Long, ByRef InpLogic As Long) INP 기능 및 신호에 대한 환경 설정값을 읽어옵니다.	274
Sub McSetMioCfgERC (ByVal Channel As Long, ByVal ErcLogic As Long, ByVal ErcOnTime As Long) ERC 신호의 입력 로직(Logic) 및 ON time 을 설정합니다.	275
Sub McGetMioCfgERC (ByVal Channel As Long, ByRef ErcLogic As Long, ByRef ErcOnTime As Long) ERC 신호의 입력 로직(Logic) 및 ON time 에 대한 설정값을 읽어옵니다.	276
Sub McSetMioCfgLTC (ByVal Channel As Long, ByVal LtcLogic As Long, ByVal Ltc2Src As Long) LATCH 입력 신호의 유형 및 LTC2 대상 카운터를 선택합니다.	277
Sub McSetMioCfgLTC (ByVal Channel As Long, ByRef LtcLogic As Long, ByRef Ltc2Src As Long) 현재 설정된 LATCH 입력 신호의 유형 및 LTC2 대상 카운터의 설정값을 읽어옵니다.	278
Sub McSetMioCfgSD (ByVal Channel As Long, ByVal SdEnable As Integer, ByVal SdLogic As Long, ByVal SdLatch As Long, ByVal SdMode As Long) SD 신호에 대한 환경을 설정합니다. SD 신호는 Deceleration 시작점을 외부에서 입력하는 신호입니다.	279
Sub McGetMioCfgSD (ByVal Channel As Long, ByRef SdEnable As Integer, ByRef SdLogic As Long, ByRef SdLatch As Long, ByRef SdMode As Long) SD 신호에 대한 환경 설정값을 읽어옵니다.	280
sub McSetMioCfgSTA (ByVal Channel As Long, ByVal Mode As Long, byVal InputType As Long) STA 신호에 대한 환경을 설정합니다.	281
sub McSetMioCfgSTP (ByVal Channel As Long, ByVal Mode As Long) STP 신호에 대한 환경을 설정합니다.	282

Sub McSetSoftLimit (ByVal Channel As Long, ByVal LimitP As Double, ByVal LimitN As Double) 소프트웨어적인 Limit 를 설정합니다.	283
Sub McEnableSoftLimit (ByVal Channel As Long) 소프트웨어적인 Limit 의 적용을 Enable 시킵니다.	284
Sub McDisableSoftLimit (ByVal Channel As Long) 소프트웨어적인 Limit 의 적용을 Disable 시킵니다.	285
Sub McSetErrorCompare (ByVal Channel As Long, ByVal Tol As Double, ByVal Enable As Long) 명령 펄스(Command Pulse)와 Feedback 펄스간의 에러를 체크하는 기능을 설정합니다.	286
Sub McSetGeneralCompare (ByVal Channel As Long, ByVal CmpSrc As Long, ByVal CmpMethod As Long, ByVal CmpAction As Long, ByVal Data As Double) General Comparator 의 비교 대상, 비교값 등을 설정합니다.	287
sub McSetTriggerCompare (ByVal Channel As Long, ByVal CmpSrc As Long, ByVal CmpMethod As Long, ByVal Data As Double) 위치비교기를 설정합니다.	288
Function McRegTableCCMP (ByVal Channel As Long, ByRef DataBuffer As Double, ByVal NumData As Long) As Boolean 연속적인 위치 비교 출력 기능을 사용하기 위해서 임의의 연속적인 위치 데이터를 등록합니다.	289
Function McBuildTableCCMP (ByVal Channel As Long, ByVal StartData As Double, ByVal Interval As Double, ByVal NumData As Long) As Boolean 연속적인 위치 비교 출력 기능을 사용하기 위해서 일정한 위치 간격을 가지는 연속적인 위치 데이터를 자동으로 생성하여 등록하도록 합니다.	290
Function McStartCCMP (ByVal Channel As Long, ByVal CmpSrc As Long , ByVal CmpMethod As Long) As Boolean 연속적인 위치 비교 출력 기능을 시작합니다.	291
Function McStopCCMP (ByVal Channel As Long) As Boolean 연속적인 위치 비교 출력 기능을 종료합니다.	293

■ McSetFilterLogic

함수 원형

sub **McSetFilterLogic** (ByVal Channel As Long, ByVal Enable As Short)

함수 설명

이 함수는 각종 I/O 신호에 필터 로직을 적용할지를 설정하는 함수입니다. 필터 로직이 적용되면 펄스폭이 너무 짧은 입력 신호는 무시되게 됩니다. 필터 로직이 적용되는 I/O 신호 및 펄스폭은 아래의 참고 항목을 참조하십시오.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **Enable** : 필터로직을 적용할지를 결정합니다.

Value	Meaning
0	Filter disable
1	Filter enable

참 고

□ 필터 로직 적용이 Enable 됐을 때 필터 로직이 적용되는 신호 및 기준 펄스 폭은 다음과 같습니다.

필터 적용 I/O	필터 기준
+EL, -EL, SD, ORG, ALM, INP	펄스폭이 4 μ s 미만은 무시
+DR, -DR	펄스폭이 3.2ms 미만은 무시

■ McSetELL

함수 원형

```
sub McSetELL ( ByVal Channel As Long, ByVal Logic As Long)
```

함수 설명

+EL, -EL 신호의 입력 로직을 설정합니다.

매개 변수

- ▶ *Channel* : 채널(축) 번호(0 부터 시작합니다).
- ▶ *Logic* : +EL, -EL 신호의 입력 로직 설정

Value	Meaning
0	Low active 또는 Normal Open
1	High active 또는 Normal Close

■ McSetMioCfgALM

메소드 원형

Sub **McSetMioCfgALM** (ByVal Channel As Long, AlarmLogic As Long, AlarmMode As Long)

메소드 설명

이 메소드는 Alarm 신호에 대한 입력모드를 설정합니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **AlarmLogic** : Alarm 신호 입력 펄스의 로직(Logic)을 설정합니다.

Value	Meaning
0	Active low
1	Active high

- ▶ **AlarmMode** : Alarm 신호가 Logic ON 될때 모터가 반응하는 방식을 결정합니다.

Value	Meaning
0	Alarm 신호가 ON 되면 모션을 즉시 정지합니다.
1	Alarm 신호가 ON 되면 모션을 감속후에 정지합니다.

■ McGetMioCfgALM

메소드 원형

Sub **McGetMioCfgALM** (ByVal Channel As Long, ByRef AlarmLogic As Long, ByRef AlarmMode As Long)

메소드 설명

이 메소드는 현재 설정된 Alarm 신호에 대한 입력모드를 읽어옵니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **AlarmLogic** : Alarm 신호 입력 펄스의 로직(Logic) 설정값을 받아들이는 변수. 이 변수에 반환되는 값의 의미는 다음과 같습니다.

Value	Meaning
0	Active low
1	Active high

- ▶ **AlarmMode** : Alarm 신호에 대한 모터의 반응방식을 결정하는 모드 설정값을 받아들이는 변수. 이 변수에 반환되는 값의 의미는 다음과 같습니다.

Value	Meaning
0	Alarm 신호가 ON 되면 모션을 즉시 정지합니다.
1	Alarm 신호가 ON 되면 모션을 감속후에 정지합니다.

■ McSetMioCfgCLR

함수 원형

```
sub McSetMioCfgCLR ( ByVal Channel As Long, ByVal CntrSel As Long, ByVal SignalType As Long )
```

함수 설명

이 함수는 CLR 신호에 대한 환경을 설정합니다. CLR 신호는 COMI-LX50x 의 각종 카운터의 카운트값을 리셋합니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **CntrSel** : CLR 신호에 의해 카운트값을 리셋하는 카운터를 지정합니다. COMI-LX50x 은 축당 4 개의 카운터를 제공합니다. CntrSel 파라미터의 각 비트는 다음과 같이 각 카운터의 Clear Enable/Disable 을 결정합니다.

BIT	Meaning
BIT0	Command position counter 의 클리어 여부를 결정합니다.(1=Enable)
BIT1	Feedback position counter 의 클리어 여부를 결정합니다.(1=Enable)
BIT2	Deviation counter 의 클리어 여부를 결정합니다.(1=Enable)
BIT3	General counter 의 클리어 여부를 결정합니다.(1=Enable)

- ▶ **SignalType** : CLR 신호의 신호 유형을 결정합니다.

Value	Meaning
0	Clears at the falling edge
1	Clears at the rising edge
2	Clears at low level
3	Clears at high level

■ McGetMioCfgCLR

함수 원형

```
sub McGetMioCfgCLR ( ByVal Channel As Long, ByRef CntrSel As Long, ByRef SignalType  
As Long )
```

함수 설명

이 함수는 CLR 신호에 대한 모드 설정값을 읽어옵니다.

매개 변수

- ▶ *Channel* : 채널(축) 번호, 0 ~ 3
- ▶ *CntrSel* : CLR 적용 카운터 설정값을 받아올 변수의 주소값.
- ▶ *SignalType* : CLR 신호의 신호 형태 설정값을 받아올 변수의 주소값

■ McSetMioCfgCMP

함수 원형

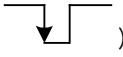

```
sub McSetMioCfgCMP ( ByVal Channel As Long, ByVal CmpPulseWidth As Long, ByVal CmpLogic As Long )
```

함수 설명

위치비교 출력 펄스(CMP 신호)의 출력 방식을 설정합니다. 연속적인 위치비교 출력 기능을 사용하는 경우에는 위치조건이 만족되면 One shot 펄스가 출력이 됩니다. 이 때의 펄스의 폭 및 펄스의 출력 로직(Logic)을 설정합니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호(0 부터 시작합니다)
- ▶ **CmpPulseWidth** : One shot 펄스의 펄스폭을 설정합니다. 실제 펄스폭은 CmpPulseWidth * 1.5(us) 가 됩니다. 즉, 이 값을 1 로 하면 1.5us, 2 로 하면 3us...와 같이 됩니다.
- ▶ **CmpLogic** : CMP 펄스의 출력 로직(Logic)을 설정합니다.

Value	Meaning
0	Low active ()
1	High active ()

참 고

□ CmpPulseWidth 파라미터값을 0 으로 하면 CMP 출력 신호의 펄스폭은 COMMAND 펄스의 펄스폭과 같게 됩니다.

□ McSetTriggerCompare 또는 McStartCCMP 함수에서 CmpMethod 파라미터를 4 또는 5 의 값으로 설정하면 CMP 출력 신호는 One shot 펄스가 아니고, 조건이 만족되는한 Active 상태를 유지하게 됩니다. 자세한 사항은 McSetTriggerCompare 함수 설명을 참조하시기 바랍니다.

■ McGetMioCfgCMP

함수 원형

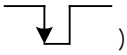

```
sub McGetMioCfgCMP ( ByVal Channel As Long, ByRef CmpPulseWidth Long, ByRef CmpLogic
    As Long )
```

함수 설명

위치비교 출력 펄스(CMP 신호)의 출력 설정값을 얻어옵니다.

매개 변수

- ▶ *Channel* : 채널(축) 번호(0 부터 시작합니다)
- ▶ *CmpPulseWidth* : One shot 펄스의 펄스폭 설정값을 반환받을 변수의 주소값.
- ▶ *CmpLogic* : CMP 펄스의 출력 로직(Logic) 설정값을 반환받을 변수의 주소값

Value	Meaning
0	Low active ()
1	High active ()

■ McSetMioCfgDR

함수 원형

sub **McSetMioCfgDR** (ByVal Channel As Long, ByVal Logic As Long)

함수 설명

이 함수는 +/-DR 신호에 대한 환경을 설정합니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **Logic** : +/-DR 신호 입력 펄스의 로직(Logic)을 설정합니다.

Value	Meaning
0	Active low
1	Active high

■ McGetMioCfgDR

함수 원형

```
sub McGetMioCfgDR ( ByVal Channel As Long, ByRef Logic As Long )
```

함수 설명

이 함수는 +/-DR 신호에 대한 모드 설정값을 읽어옵니다.

매개 변수

- ▶ *Channel* : 채널(축) 번호, 0 ~ 3
- ▶ *Logic* : +/-DR 신호의 입력 로직 설정값을 받아올 변수의 주소값. 이 변수에 반환되는 값의 의미는 다음과 같습니다.

Value	Meaning
0	Active low
1	Active high

■ McSetMioCfgEL

메소드 원형

Sub **McSetMioCfgEL** (ByVal Channel As Long, ByVal ElMode As Long)

메소드 설명

이 메소드는 EL 신호에 대한 모드를 설정합니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **ElMode** : EL 신호가 Logic ON 될때 모터가 반응하는 방식을 결정합니다.

Value	Meaning
0	EL 신호가 ON 되면 모션을 즉시 정지합니다.
1	EL 신호가 ON 되면 모션을 감속후에 정지합니다.

■ McGetMioCfgEL

메소드 원형

Sub **McGetMioCfgEL** (ByVal Channel As Long, ByRef ElMode As Long)

메소드 설명

이 메소드는 EL 신호에 대한 모드 설정값을 읽어옵니다.

매개 변수

- ▶ *Channel* : 채널(축) 번호, 0 ~ 3
- ▶ *ElMode* : EL 신호에 대한 모드 설정값을 받아들이는 변수. 이 변수에 반환되는 값의 의미는 다음과 같습니다.

Value	Meaning
0	EL 신호가 ON 되면 모션을 즉시 정지합니다.
1	EL 신호가 ON 되면 모션을 감속후에 정지합니다.

■ McSetMioCfgINP

메소드 원형

Sub **McSetMioCfgINP** (ByVal Channel As Long, ByVal InpEnable As Integer, ByVal InpLogic As Long)

메소드 설명

이 메소드는 INP 기능 및 신호에 대한 환경을 설정합니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **InpEnable** : INP 기능을 Enable/Disable 시킵니다.

Value	Meaning
0	INP 기능을 Disable 시킵니다.
1	INP 기능을 Enable 시킵니다.

- ▶ **InpLogic** : INP 입력 신호의 로직(Logic)을 설정합니다.

Value	Meaning
0	Active low
1	Active high

or

■ McGetMioCfgINP

메소드 원형

Sub **McGetMioCfgINP** (ByVal Channel As Long, ByRef pInpEnable As Long, ByRef InpLogic As Long)

메소드 설명

이 메소드는 INP 기능 및 신호에 대한 환경 설정값을 읽어옵니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **InpEnable** : INP 기능 설정값을 읽어올 변수. 이 변수에 전달되는 값의 의미는 다음과 같습니다.

Value	Meaning
0	INP 기능을 Disable 시킵니다.
1	INP 기능을 Enable 시킵니다.

- ▶ **InpLogic** : INP 입력 신호의 로직(Logic)을 설정합니다.

Value	Meaning
0	Active low
1	Active high

■ McSetMioCfgERC

메소드 원형

Sub **McSetMioCfgERC** (ByVal Channel As Long, ByVal ErcLogic As Long, ByVal ErcOnTime As Long)

메소드 설명

이 메소드는 ERC 신호의 입력 로직(Logic) 및 ON time 을 설정합니다.

매개 변수

▶ **Channel** : 채널(축) 번호, 0 ~ 3

▶ **ErcLogic** : ERC 입력 신호의 로직(Logic)을 설정합니다.

Value	Meaning
0	Active low
1	Active high

▶ **ErcOnTime** : INP 입력 신호의 로직(Logic)을 설정합니다.

Value	ON time of ERC signal
0	12 us
1	102us
2	409us
3	1.6ms
4	13ms
5	52ms
6	104ms

■ McGetMioCfgERC

메소드 원형

Sub **McGetMioCfgERC** (ByVal Channel As Long, ByRef ErcLogic As Long, ByRef ErcOnTime As Long)

메소드 설명

이 메소드는 ERC 신호의 입력 로직(Logic) 및 ON time 에 대한 설정값을 읽어옵니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **ErcLogic** : ERC 입력 신호의 로직(Logic) 설정값을 받아들이는 변수. 이 변수에 전달되는 값의 의미는 다음과 같습니다.

Value	Meaning
0	Active low
1	Active high

- ▶ **ErcOnTime** : INP 입력 신호의 로직(Logic) 설정값을 받아들이는 변수. 이 변수에 전달되는 값의 의미는 다음과 같습니다.

Value	ON time of ERC signal
0	12 us
1	102us
2	409us
3	1.6ms
4	13ms
5	52ms
6	104ms

■ McSetMioCfgLTC

함수 원형

Sub **McSetMioCfgLTC** (ByVal Channel As Long, ByVal LtcLogic As Long, ByVal Ltc2Src As Long)

함수 설명

LATCH 입력 신호의 유형 및 LTC2 대상 카운터를 선택합니다.

매개 변수

▶ **Channel** : 채널(축) 번호(0 부터 시작합니다)

▶ **LtcLogic** : LATCH 신호의 유형을 설정합니다.

Value	Meaning
0	Falling edge
1	Rising edge

▶ **Ltc2Src** : LTC2 대상 카운터를 선택합니다. 래치되는 카운터는 총 4 가지인데 이 중에서 3 번째 카운터(LTC2)는 다음 둘 중 하나를 선택할 수 있습니다.

Value	Meaning
0	편차카운터
1	현재의 펄스 출력 속도(PPS)

■ McGetMioCfgLTC

함수 원형

Sub **McGetMioCfgLTC** (ByVal Channel As Long, ByRef LtcLogic As Long, ByRef Ltc2Src As Long)

함수 설명

현재 설정된 LATCH 입력 신호의 유형 및 LTC2 대상 카운터의 설정값을 얻어옵니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호(0 부터 시작합니다)
- ▶ **LtcLogic** : LATCH 신호의 유형 설정값을 반환받을 변수. 반환되는 값의 의미는 다음과 같습니다.

Value	Meaning
0	Falling edge
1	Rising edge

- ▶ **Ltc2Src** : LTC2 대상 카운터의 설정값을 반환받을 변수. 반환되는 값의 의미는 다음과 같습니다.

Value	Meaning
0	편차카운터
1	현재의 펄스 출력 속도(PPS)

■ McSetMioCfgSD

메소드 원형

Sub **McSetMioCfgSD** (ByVal Channel As Long, ByVal SdEnable As Integer, ByVal SdLogic As Long, ByVal SdLatch As Long, ByVal SdMode As Long)

메소드 설명

이 메소드는 SD 신호에 대한 환경을 설정합니다. SD 신호는 Deceleration 시작점을 외부에서 입력하는 신호입니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **SdEnable** : SD 신호를 Enable/Disable 시킵니다.

Value	Meaning
0	SD 신호 Disable => Deceleration 시작점이 자동으로 결정됩니다.
1	SD 신호 Enable => 외부에서 입력되는 SD 신호에 의해 Deceleration 시작점이 결정됩니다.

- ▶ **SdLogic** : SD 입력 신호의 로직(Logic)을 설정합니다.

Value	Meaning
0	Active low
1	Active high

- ▶ **SdLatch** : SD 신호에 대한 Latch 여부를 결정합니다.

Value	Meaning
0	SD 신호를 Latch 하지 않음
1	SD 신호를 Latch 함

- ▶ **SdMode** : SD 입력 신호에 대한 모터의 반응을 결정합니다.

Value	Meaning
0	Deceleration 만 적용
1	Deceleration 후 정지

McGetMioCfgSD

메소드 원형

Sub **McGetMioCfgSD** (ByVal Channel As Long, ByRef SdEnable As Integer, ByRef SdLogic As Long, ByRef SdLatch As Long, ByRef SdMode As Long)

메소드 설명

이 메소드는 SD 신호에 대한 환경 설정값을 읽어옵니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **SdEnable** : SD 신호의 Enable/Disable 설정값을 읽어들이 변수. 이 변수에 전달되는 값의 의미는 다음과 같습니다.

Value	Meaning
0	SD 신호 Disable => Deceleration 시작점이 자동으로 결정됩니다.
1	SD 신호 Enable => 외부에서 입력되는 SD 신호에 의해 Deceleration 시작점이 결정됩니다.

- ▶ **SdLogic** : SD 입력 신호의 로직(Logic) 설정값을 읽어들이 변수. 이 변수에 전달되는 값의 의미는 다음과 같습니다.

Value	Meaning
0	Active low
1	Active high

- ▶ **SdLatch** : SD 신호의 Latch 설정값을 읽어들이 변수. 이 변수에 전달되는 값의 의미는 다음과 같습니다.

Value	Meaning
0	SD 신호를 Latch 하지 않음
1	SD 신호를 Latch 함

- ▶ **SdMode** : SD 모드 설정값을 읽어들이 변수. 이 변수에 전달되는 값의 의미는 다음과 같습니다.

Value	Meaning
0	Deceleration 만 적용
1	Deceleration 후 정지

■ McSetMioCfgSTA

함수 원형

```
sub McSetMioCfgSTA ( ByVal Channel As Long, ByVal Mode As Long, ByVal InputType As Long )
```

함수 설명

STA 신호에 대한 환경을 설정합니다. STA 신호는 외부에서 입력되는 모션 시작 신호입니다. 이 함수에서 Mode 를 0 으로 하면 STA 입력 신호는 무시됩니다. Mode 를 1 로 하면 McMove, McMoveAll 과 같은 모든 이동 명령이 외부에서 입력되는 STA 입력 신호와 동기되어 시작됩니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호(0 부터 시작합니다).
- ▶ **Mode** : STA 신호의 적용 여부를 결정합니다.

Value	Meaning	Description
0	Start by software	STA 입력 신호는 무시됩니다. 즉, 모든 이동 명령이 호출되면 바로 이동을 시작합니다.
1	Start by STA hardware signal input	McMove, McMoveAll, McStartVmoveAll 등과 이동 명령이 호출되면 바로 이동을 시작하지 않고 STA 입력핀에 트리거 신호가 입력되면 이동을 시작합니다.

- ▶ **InputType** : STA 핀에 입력되는 트리거 신호 유형을 설정합니다.

Value	Meaning	
0	Level trigger(Low)	STA 입력핀이 Low level 이 되면 모션을 시작합니다.
1	Edge trigger(Falling)	STA 입력핀에 Falling edge 신호가 입력되면 모션을 시작합니다.

참 고

□ Mode 값을 1 로 하면 모든 이동 명령이 STA 입력이 ON 되기전까지 시작되지 않습니다. 따라서 STA 동기 구동이 필요없는 경우에는 Mode 값을 0 으로 하여주어야 합니다.

□ STA 입력핀은 모든 축에 대해 공통으로 사용됩니다. 따라서 STA 입력을 각 축별로 따로 입력할 수는 없습니다. 그러나 STA 입력 환경은 각 축별로 다르게 설정할 수 있습니다.

McSetMioCfgSTP

함수 원형

```
sub McSetMioCfgSTP ( ByVal Channel As Long, ByVal Mode As Long )
```

함수 설명

STP 신호에 대한 환경을 설정합니다. STP 신호는 외부에서 입력되는 모션 종료 신호입니다. 이 함수에서 Mode 를 0 으로 하면 STP 입력 신호는 무시됩니다. Mode 를 1 로 하면 외부에서 STP 에 신호를 입력하여 모션을 종료합니다.

STP 신호는 외부에서 하드웨어적으로 모션 구동을 종료할 수 있도록 하기 위해 마련된 입력 신호입니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호(0 부터 시작합니다).
- ▶ **Mode** : STA 신호의 적용 여부를 결정합니다.

Value	Meaning	Description
0	Ignore STP	STP 입력 신호는 무시됩니다.
1	Immediate stop by STP	STP 입력핀에 Low level 신호가 입력되면 모션을 즉시 정지합니다
2	Stop after deceleration by STP	STP 입력핀에 Low level 신호가 입력되면 모션을 감속 후 정지합니다

참 고

□ Mode 를 1 또는 2 로 하더라도 McStop 과 같은 소프트웨어적인 정지 명령은 여전히 유효합니다. 즉, STP 신호가 입력되지 않아도 소프트웨어적인 정지 명령으로 모션을 정지할 수 있습니다.

□ STP 신호는 STA 신호와 달리 Level trigger 방식만 지원합니다. 그리고 입력 로직은 Low active(low level 일때 ON)입니다.

□ STP 입력핀은 모든 축에 대해 공통으로 사용됩니다. 따라서 STP 입력을 각 축별로 따로 입력할 수는 없습니다. 그러나 STP 입력 환경은 각 축별로 다르게 설정할 수 있습니다.

■ McSetSoftLimit

메소드 원형

Sub **McSetSoftLimit** (ByVal Channel As Long, ByVal LimitP As Double, ByVal LimitN As Double)

메소드 설명

이 메소드는 소프트웨어적인 Limit 를 설정합니다.

매개 변수

- ▶ *Channel* : 채널(축) 번호, 0 ~ 3
- ▶ *LimitP* : (+)방향 Limit 값을 설정합니다.
- ▶ *LimitN* : (-)방향 Limit 값을 설정합니다.

or

■ McEnableSoftLimit

메소드 원형

Sub **McEnableSoftLimit** (ByVal Channel As Long)

메소드 설명

이 메소드는 소프트웨어적인 Limit 의 적용을 Enable 시킵니다.

매개 변수

▶ *Channel* : 채널(축) 번호, 0 ~ 3

■ McDisableSoftLimit

메소드 원형

Sub **McDisableSoftLimit** (ByVal Channel As Long)

메소드 설명

이 메소드는 소프트웨어적인 Limit 의 적용을 Disable 시킵니다.

매개 변수

▶ *Channel* : 채널(축) 번호, 0 ~ 3

■ McSetErrorCompare

메소드 원형

Sub **McSetErrorCompare** (ByVal Channel As Long, ByVal Tol As Double, ByVal Enable As Long)

메소드 설명

이 메소드는 명령 펄스(Command Pulse)와 Feedback 펄스간의 에러를 체크하는 기능을 설정합니다. 에러 체크 기능을 Enable 시키면 Command 펄스 수와 Feedback 펄스 수의 차가 지정한 Tolerance 보다 크면 인터럽트(Event Interrupt 의 BIT10)를 발생시킵니다. Event Interrupt 에 관해서는 McGetIntStatus() 메소드를 참조하십시오.

매개 변수

- ▶ *Channel* : 채널(축) 번호, 0 ~ 3
- ▶ *Tol* : Position error tolerance.
- ▶ *Enable* : 에러 체크 기능을 Enable 또는 Disable 시킵니다.

■ McSetGeneralCompare

메소드 원형

Sub **McSetGeneralCompare** (ByVal Channel As Long, ByVal CmpSrc As Long, ByVal CmpMethod As Long, ByVal CmpAction As Long, ByVal Data As Double)

메소드 설명

이 메소드는 General Comparator 의 비교 대상, 비교값 등을 설정합니다. 비교 대상 카운터값과 지정한 데이터값이 비교 조건을 만족할 때 인터럽트(Event Interrupt 의 BIT11)를 발생시킵니다. Event Interrupt 에 관해서는 McGetIntStatus() 메소드를 참조하십시오.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **CmpSrc** : 비교 대상 카운터를 설정합니다.

Value	Comapare Source
0	Command Counter
1	Feedback Counter
2	Error Counter
3	General Counter

- ▶ **CmpMethod** : 비교 조건을 설정합니다.

Value	Comapare Source
0	비교를 하지 않습니다. => Disable
1	fData=Counter(Directionless) 이면 Action 을 취합니다.
2	fData=Counter(+ Direction) 이면 Action 을 취합니다.
3	fData=Counter(- Direction) 이면 Action 을 취합니다.
4	fData>Counter 이면 Action 을 취합니다.
5	fData<Counter 이면 Action 을 취합니다.

- ▶ **CmpAction** : 비교 조건이 성립되었을 때 취할 Action 을 설정합니다.

Value	Comapare Source
0	인터럽트만 발생
1	즉시 정지 및 인터럽트 발생
2	감속 후 정지 및 인터럽트 발생
3	속도 변경 및 인터럽트 발생

- ▶ **Data** : 비교 기준 값.

■ McSetTriggerCompare

함수 원형

```
sub McSetTriggerCompare ( ByVal Channel As Long, ByVal CmpSrc As Long, ByVal CmpMethod As Long, ByVal Data As Double )
```

함수 설명

위치비교기를 설정합니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호(0 부터 시작합니다)
- ▶ **CmpSrc** : 비교 대상 카운터를 설정합니다.

Value	Meaning
0	Command counter
1	Feedback counter
2	Deviation counter
3	General counter

- ▶ **CmpMethod** : CMP 신호 출력 조건을 설정합니다. 즉, 비교 대상 카운트값과 사용자가 지정한 데이터 값을 어떻게 비교하여 CMP 신호 출력을 내보낼지를 결정합니다.

Value	Meaning
0	위치비교 출력 기능 Disable
1	Counting direction 과 관계없이 Count = fData 이면 One shot 펄스 출력.
2	카운트가 증가(Counting up)하는 시점에서 Count=fData 이면 One shot 펄스 출력.
3	카운트가 감소(Counting down)하는 시점에서 Count=fData 이면 One shot 펄스 출력.
4	Count < fData 이면 CMP 출력을 내보냅니다. 이때에는 One shot 펄스가 아니며, Count < fData 인 동안에는 CMP 출력이 Active 가 됩니다.
5	Count > fData 이면 CMP 출력을 내보냅니다. 이때에는 One shot 펄스가 아니며, Count > fData 인 동안에는 CMP 출력이 Active 가 됩니다.

- ▶ **Data** : 비교 대상 레퍼런스(Reference)값. 비교기는 지정한 카운트값을 이 값과 비교하여 CMP 출력을 내보낸다.

■ McRegTableCCMP

함수 원형

```
Function McRegTableCCMP ( ByVal Channel As Long, ByRef DataBuffer As Double, ByVal  
NumData As Long ) As Boolean
```

함수 설명

연속적인 위치 비교 출력 기능을 사용하기 위해서 임의의 연속적인 위치 데이터를 등록합니다.

매개 변수

- ▶ *Channel* : 채널(축) 번호(0 부터 시작합니다)
- ▶ *DataBuffer* : 연속적인 위치 데이터를 담고 있는 버퍼(배열).
- ▶ *NumData* : 버퍼에 담겨있는 위치 데이터 수

■ McBuildTableCCMP

함수 원형

```
Function McBuildTableCCMP ( ByVal Channel As Long, ByVal StartData As Double, ByVal  
Interval As Double, ByVal NumData As Long ) As Boolean
```

함수 설명

연속적인 위치 비교 출력 기능을 사용하기 위해서 일정한 위치 간격을 가지는 연속적인 위치 데이터를 자동으로 생성하여 등록하도록 합니다.

이 함수는 일정한 위치 간격으로 CMP 출력을 내보낼 때 McRegTableCCMP 함수 대신에 사용할 수 있습니다.

매개 변수

- ▶ *Channel* : 채널(축) 번호(0 부터 시작합니다)
- ▶ *StartData* : 시작 위치값
- ▶ *Interval* : 위치 간격
- ▶ *NumData* : 총 데이터 수

■ McStartCCMP

함수 원형

Function **McStartCCMP** (ByVal Channel As Long, ByVal CmpSrc As Long , ByVal CmpMethod As Long) As Boolean

함수 설명

연속적인 위치 비교 출력 기능을 시작합니다. McRegTableCCMP 또는 McBuildTableCCMP 함수를 이용하여 등록된 연속적인 위치데이터를 비교기에 순차적으로 자동로딩하면서 연속적인 위치비교 출력기능을 수행합니다.

연속적인 위치비교 출력 기능은 현재 비교기에 로드된 비교조건이 만족되어 CMP 출력이 나가게 되면 인터럽트가 발생되어 드라이버 프로그램에서 사용자가 등록한 다음 데이터를 비교기에 자동으로 로드하도록 하는 기능입니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호(0 부터 시작합니다)
- ▶ **CmpSrc** : 비교 대상 카운터를 설정합니다.

Value	Meaning
0	Command counter
1	Feedback counter
2	Deviation counter
3	General counter

- ▶ **CmpMethod** : CMP 신호 출력 조건을 설정합니다. 즉, 비교 대상 카운트값과 사용자가 지정한 데이터 값을 어떻게 비교하여 CMP 신호 출력을 내보낼지를 결정합니다.

Value	Meaning
0	위치비교 출력 기능 Disable
1	Counting direction 과 관계없이 Count = fData 이면 One shot 펄스 출력.
2	카운트가 증가(Counting up)하는 시점에서 Count=fData 이면 One shot 펄스 출력.
3	카운트가 감소(Counting down)하는 시점에서 Count=fData 이면 One shot 펄스 출력.

참 고

□ 연속적인 위치비교 출력 기능이 시작된 이후에 McMaskInterrupt 함수를 이용하여 BIT12를 언마스크(0으로 만듦)하면 인터럽트를 받을 수 없으므로 연속적인 위치비교 출력이 기능이 진행되지 않습니다.

□ 사용자가 등록한 모든 위치 데이터에 대하여 CMP 출력이 완료되었으면 McStopCCMP 함수를 호출하여 중지하지 않는한 처음 데이터부터 다시 비교기에 로드되어 연속적인 비교출력이 자동으로 재개됩니다. 그러나 현재 로드된 비교조건이 만족되기 전까지는 다음 위치데이터가 로드되지 않는다는 점에 유의하시기 바랍니다. 즉, 1000, 5000, 10000, 15000의 연속적인 위치데이터를 등록한 후 모션이 10000 위치까지 이동 후 다시 0 위치로 복귀하고 다시 10000 위치로 이동했을 때 처음 10000 위치로 이동시에는 1000, 5000, 10000의 위치에서 CMP 출력이 나가지만 두번째 10000 위치로 이동시에는 CMP 출력이 나가지 않습니다. 이유는 현재 비교기에 로드된 위치데이터가 15000인데 이 조건이 만족되지 않았으므로 계속해서 비교기에 이 데이터가 남아있기 때문입니다.

■ McStopCCMP

함수 원형

Function **McStopCCMP** (ByVal Channel As Long) As Boolean

함수 설명

연속적인 위치 비교 출력 기능을 종료합니다.

매개 변수

▶ *Channel* : 채널(축) 번호(0 부터 시작합니다)

2.6.12 인터럽트 관련 메소드

이 단원에서는 인터럽트에 관련된 메소드들을 소개합니다. 인터럽트는 특정 상황이 발생되었을 때 사용자(또는 프로그래머)에게 이 것을 알려주기 위한 것입니다. 윈도우 환경에서는 일반 Application 레벨에서 인터럽트를 처리할 수 없으므로 이벤트를 통하여 Application 에게 인터럽트가 발생하였음을 알려줍니다.

사용자 프로그램에서 인터럽트를 처리하기 위해서는 먼저 `CreateEvent()` 등의 표준 윈도우 API 메소드를 통하여 이벤트를 생성하고 `McMaskInterrupt()` 메소드를 이용하여 이벤트를 등록하여야 합니다. 이벤트가 등록된 후 사용자는 스레드(Thread) 또는 타이머 콜백 메소드에서 이벤트가 발생했는지를 체크하고 이벤트가 발생하였으면 `McGetAxisIntState()` 메소드와 `McGetIntStatus()` 메소드를 통하여 인터럽트 Status 를 확인하여 각 Status 에 따라 적절한 Action 을 취하여야 합니다.

인터럽트 처리에 관련된 메소드는 다음과 같습니다.

메소드 / 설명	페이지
Sub McMaskInterrupt (ByVal Channel As Long, ByVal Mask As Long) 어떠한 조건의 인터럽트를 받아들일 것인지를 설정합니다.	295
Sub McEnableInterrupt (ByVal EventHandle As Long) 인터럽트 이벤트 통지 기능을 Enable 시킵니다.	296
Sub McDisableInterrupt 인터럽트 이벤트 통지 기능을 Disable 시킵니다.	297
Function McGetAxisIntState (ByVal Channel As Long) As Boolean 각 축에 대하여 인터럽트가 발생하였는지를 알려주는 메소드입니다.	298
Sub McGetIntStatus (ByVal Channel As Long, ByRef ErrorStatus As Long, ByRef EventStatus As Long) 각 축의 인터럽트가 발생한 원인을 알려주는 메소드입니다.	299

■ McMaskInterrupt

메소드 원형

Sub **McMaskInterrupt** (ByVal Channel As Long, ByVal Mask As Long)

메소드 설명

이 메소드는 어떠한 조건의 인터럽트를 받아들일 것인지를 설정합니다. 인터럽트를 발생시킬 조건을 Mask 파라미터를 통하여 설정하십시오.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3.
- ▶ **Mask** : 인터럽트를 발생시킬 조건을 설정합니다. 이 값의 각 비트는 다음의 표와 같이 인터럽트 발생 조건을 설정합니다.

Bit	인터럽트 발생 조건
BIT0	Normal Stop
BIT1	Succesive start of the next operation
BIT2	Reserved
BIT3	Reserved
BIT4	Start of acceleration
BIT5	End of acceleration
BIT6	Start of deceleration
BIT7	End of deceleration
BIT8	Reserved
BIT9	Reserved
BIT10	Position error tolerance exceed (McSetErrorCompare 메소드 참조)
BIT11	General Comparator (McSetGeneralCompare 메소드 참조)
BIT12	CMP trigger output
BIT13	CLR signal input resetting counter value
BIT14	LTC input making counter value latched
BIT15	ORG input making counter value latched
BIT16	SD input ON
BIT17	±DR input change
BIT18 ~ BIT31	Reserved

■ McEnableInterrupt

메소드 원형

Sub **McEnableInterrupt** (ByVal Reserved As Long)

메소드 설명

이 메소드는 인터럽트 이벤트 통지 기능을 Enable 시킵니다. 인터럽트가 Enable 되면, Error interrupt 또는 Event interrupt 에 의해 인터럽트가 발생되면 ComLxAc OCX 의 InterruptEvent 이벤트핸들러(콜백 함수)가 호출됩니다.

매개 변수

▶ *Reserved* : 이 파라미터는 사용되지 않습니다. 이전 버전과의 라이브러리 호환성 때문에 의미없이 사용되는 파라미터입니다.

■ McDisableInterrupt

메소드 원형

Sub **McDisableInterrupt**

메소드 설명

이 메소드는 인터럽트 이벤트 통지 기능을 Disable 시킵니다. 기본적으로 이벤트 통지 기능은 Disable 된 상태입니다.

■ McGetAxisIntState

메소드 원형

Function **McGetAxisIntState** (ByVal Channel As Long) As Boolean

메소드 설명

이 메소드는 각 축에 대하여 인터럽트가 발생하였는지를 알려주는 메소드입니다. 사용자는 인터럽트 이벤트가 발생하면 먼저 이 메소드를 이용하여 어느 축에서 인터럽트가 발생하였는지를 체크한 후 McGetIntStatus()메소드를 이용하여 인터럽트의 종류를 파악하여 적절한 대응을 합니다.

매개 변수

▶ *Channel* : 채널(축) 번호, 0 ~ 3.

Return 값

각 축의 인터럽트 발생 상태

Value	Meaning
0	해당축에서 인터럽트가 발생되지 않음
1	해당축에서 인터럽트가 발생됨

■ McGetIntStatus

메소드 원형

Sub **McGetIntStatus** (ByVal Channel As Long, ByRef ErrorStatus As Long, ByRef EventStatus As Long)

메소드 설명

이 메소드는 각 축의 인터럽트가 발생한 원인을 알려주는 메소드입니다. 사용자는 인터럽트 이벤트가 발생하면 먼저 McGetAxisIntState() 메소드를 이용하여 어느 축에서 인터럽트가 발생하였는지를 체크한 후 이 메소드를 이용하여 인터럽트의 종류를 파악하여 적절한 대응을 합니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3. 채널별로 각각 다른 인터럽트 조건을 설정할 수 있습니다.
- ▶ **ErrorStatus** : 에러에 관련된 인터럽트의 상태를 나타내는 값을 받아들이는 변수. 이 변수에 전달되는 값은 32 비트 정수값이며 각 비트의 값의 의미는 다음과 같습니다.

Bit	인터럽트 발생 조건
BIT0	Stop by +SL(Software limit) stop motion
BIT1	Stop by -SL(Software limit) stop motion
BIT2	Reserved
BIT3	Stop by General Comparator stop motion
BIT4	Reserved
BIT5	+EL signal is turning ON stop motion
BIT6	-EL signal is turning ON stop motion
BIT7	ALM signal is turning ON stop motion
BIT8	Reserved
BIT9	Reserved
BIT10	SD signal turning ON after deceleration
BIT11	Abnormal operation data stop motion
BIT12	Reserved
BIT13	Reserved
BIT14	PA/PB input buffer counter overflows
BIT15	In-position counter counts beyond the range at the time of ByValerpolation
BIT16 ~	Reserved

ur

BIT31	
-------	--

▶ **EventStatus** : 에러 인터럽트 이외의 인터럽트(이벤트 인터럽트)의 상태를 나타내는 값을 받아들이는 변수. 이벤트 인터럽트는 McMaskInterrupt () 메소드를 통하여 마스크(Mask) 가능합니다. 이 변수에 전달되는 값은 32 비트 정수값이며 각 비트의 값의 의미는 다음과 같습니다.

Bit	인터럽트 발생 조건
BIT0	Normal Stop
BIT1	Succesive start of the next operation
BIT2	Reserved
BIT3	Reserved
BIT4	Start of acceleration
BIT5	End of acceleration
BIT6	Start of deceleration
BIT7	End of deceleration
BIT8	Reserved
BIT9	Reserved
BIT10	Position error tolerance exceed (McSetErrorCompare 메소드 참조)
BIT11	General Comparator (McSetGeneralCompare 메소드 참조)
BIT12	CMP output triggered
BIT13	Reserved
BIT14	Latched
BIT15	ORG input signal ON
BIT16	SD input signal ON

Appendix A

라이브러리 메소드 리스트

본 부록에서는 COMI-LX 시리즈 라이브러리를 사용하는데 있어서 각 메소드의 이해 및 검색에 도움을 주고자 라이브러리 메소드에 대한 다양한 리스트를 제공합니다. 본 부록에서는 각 기능별로 메소드를 구분하여 수록한 리스트와 각 메소드별로 지원 가능한 디바이스에 관한 리스트를 수록하였습니다.

Appendix A 라이브러리 메소드 리스트

A.1 기능별 메소드 색인

라이브러리 및 디바이스 시작/종료	Page
Function LoadDevice As Boolean	10
Sub UnloadDevice	11

아날로그 입력	Page
Sub AdSetInputType (ByVal InputMode As Long)	14
Function AdSetRange (ByVal ch As Long, ByVal vmin As Single, ByVal vmax As Single) As Boolean	15
Function AdGetDigit (ByVal ch As Long) As Long	16
Function AdGetVolt (ByVal ch As Long) As Single	17
Function Us1Start (ByVal NumChannel As Long, ByRef ChanList As Long, ByVal ScanFreq As Long, ByVal BufSize As Long, ByVal TrsMethod As Long) As Long	20
Sub Us1Stop (ByVal ReleaseBuf As Integer)	22
Function Us1RetrvOne (ByVal ChOrder As Long, ByVal scanCount As Long) As Integer	23
Function Us1RetrvChannel (ByVal chOrder As Long, ByVal startCount, ByVal maxNumData, ByRef DestBuf As Double) As Long	24
Function Us1RetrvBlock (ByVal startCount As Long, ByVal maxNumScan As Long, ByRef DestBuf As Double) As Long	26
Function Us1ReleaseBuf As Boolean	28
Sub Us2SetTriggerEvent (ByVal InputSource As Long, ByVal EdgeType As Long, ByVal TrgMode As Long, ByVal AiRef As Single, ByVal AiRefBand As Single)	36
Function Us2Start (ByVal NumChannel As Long, ByRef ChanList As Long, ByVal ScanFreq As Long, ByVal BufSize As Integer , ByVal PauseAtBufFull As Integer) As Double	40
Sub Us2Resume	42
Function Us2IsBufFull As Boolean	43
Function Us2ChangeScanFreq (ByVal ScanFreq As Long) As Double	44
Function Us2DmaCount As Long	45
Function Us2RetrvChannel (ByVal ChanOrder As Long, ByVal StartCount As Long, ByVal MaxNumData As Long, ByRef DestBuf As Double) As Long	46

Appendix A 라이브러리 메소드 리스트

Sub	Us2Stop (ByVal ReleaseBuf As Integer)	48
Function	Us2ReleaseBuf As Boolean	49

아날로그 출력		Page
Function	DA_Out (ByVal ch As Long, ByVal volt As Single) As Boolean	50
Function	WfmStart (ByVal ch As Long, ByRef DataBuffer As Single, ByVal NumData As Long, ByVal PPS As Long, ByVal MaxLoops As Long) As Long	52
Function	WfmReload (ByVal ch As Long, ByRef DataBuffer As Single, ByVal NumData As Long) As Integer	54
Function	WfmRateChange (ByVal ch As Long, ByVal PPS As Long) As Long	55
Function	WfmGetCurPos (ByVal ch As Long) As Long	56
Function	WfmGetCurLoops (ByVal ch As Long) As Long	57
Sub	WfmStop (ByVal ch As Long)	58

디지털 입출력		Page
Sub	DioSetUsage (ByVal usage As TCmDiDoUsage)	61
Function	DiGetOne (ByVal ch As Long) As Long	63
Function	DiGetAll As Long	64
Sub	DoPutOne (ByVal ch As Long, ByVal status As Long)	65
Sub	DoPutAll (ByVal Statuses As Long)	66
Function	SdioInitComm As Boolean	68
Function	SdioCheckModule (ByVal ModuleNo As Long) As Boolean	69
Function	SdioSetDioUsage (ByVal ModuleNo As Long, ByVal Usage As TCmDiDoUsage) As Boolean	70
Function	SdioReadLowByte (ByVal ModuleNo As Long) As Integer	72
Function	SdioReadHighByte (ByVal ModuleNo As Long) As Integer	74
Function	SdioWriteLowByte (ByVal ModuleNo As Long, ByVal LowByte As Integer) As Boolean	76
Function	SdioWriteHighByte (ByVal ModuleNo As Long, ByVal HighByte As Integer) As Boolean	78

카운터		Page
Function	ReadCounter32 (ByVal ch As Long) As Long	81
Function	ClearCounter32 (ByVal ch As Long) As Long	82

모션 제어(모션 초기화 및 환경설정)		Page
Sub	McReset	86
Sub	McServoOn (ByVal Channel As Long, ByVal Enable As Short)	87
Sub	McGetServoOn (ByVal Channel As Long) As Long	88
Sub	McSetBlockingMode (ByVal Blocking As Integer)	89
Sub	McSetOutputMode (ByVal Channel As Long, ByVal OutputMode As Long)	90
Function	McGetOutputMode (ByVal Channel As Long) As Long	91
Sub	McSetInputMode (ByVal Channel As Long, ByVal InputMode As Long, ByVal PulseLogic As Long)	92
Sub	McGetInputMode (ByVal Channel As Long, ByRef InputMode As Long, ByRef PulseLogic As Long)	93
Sub	McSetSpeedRange (ByVal Channel As Long, ByVal MaxSpeed As Long)	94
Sub	McSetUnitDistance (ByVal Channel As Long, ByVal UnitDist As Double)	96
Function	McGetUnitDistance (ByVal Channel As Long) As Double	98
Sub	McSetUnitSpeed (ByVal Channel As Long, ByVal UnitSpeed As Double)	99
Function	McGetUnitSpeed (ByVal Channel As Long) As Double	101
Sub	McSetInOutRatio (ByVal Channel As Long, ByVal Ratio As Double)	102
모션 제어(Single Axis 모션)		Page
Sub	McSetSpeedMode (ByVal Channel As Long, ByVal ModeIndex As Long)	105
Sub	McSetSpeed (ByVal Channel, ByVal IniSpeed As Double, ByVal WorkSpeed As Double)	109
Sub	McSetAccel (ByVal Channel, ByVal Accel As Double, ByVal Decel As Double)	112
Sub	McSetScurve (ByVal Channel, ByVal Svacc As Double, ByVal Svdec As Double)	115
Sub	McStartVMove (ByVal Channel As Long, ByVal Direction As Long)	118
Sub	McStartMove (ByVal Channel As Long, ByVal Distance As Long)	119
Sub	McMove (ByVal Channel As Long, ByVal Distance As Double)	121
Sub	McStartMoveTo (ByVal Channel As Long, ByVal Position As Double)	123
Sub	McMoveTo (ByVal Channel As Long)	125
Sub	McStop (ByVal Channel As Long)	127
Sub	McEmgStop (ByVal Channel As Long)	128
Function	McDone (ByVal Channel As Long) As Boolean	129
모션 제어(Multi-Axis 동시제어)		Page
Sub	McStartVMoveAll (ByVal NumAxis As Long, ByRef AxisList[] As Long, ByRef DirList[] As Long)	131

Appendix A 라이브러리 메소드 리스트

Sub	McStartMoveAll (ByVal NumAxis As Long, ByRef AxisList[] As Long, ByRef DistList[] As Long)	133
Sub	McMoveAll (ByVal NumAxis As Long, ByRef AxisList[] As Long, ByRef DistList[] As Long)	135
Sub	McStartMoveToAll (ByVal NumAxis As Long, ByRef AxisList[] As Long, ByRef PosList[] As Long)	137
Sub	McMoveToAll (ByVal NumAxis As Long, ByRef AxisList[] As Long, ByRef PosList[] As Long)	139
Sub	McStopAll (ByVal NumAxis As Long, ByRef AxisList[] As Long)	141
Sub	McEmgStopAll (ByVal NumAxis As Long, ByRef AxisList[] As Long)	142
Function	McAllDone (ByVal NumAxis As Long, ByRef AxisList[] As Long)As Boolean	143

모션 제어(Coordinated Motion)		Page
Function	McMapAxes (ByVal MapIndex As Long, ByVal MapMask As Integer) As Boolean	147
Sub	McSetSpeedModeMx (ByVal MapIndex As Long, ByVal ModelIndex As Long)	149
Sub	McSetSpeedMx (ByVal MapIndex As Long, ByVal Speed As Long, ByVal Accel As Long)	151
Sub	McStartLine (ByVal MapIndex As Long, ByRef DistList[] As Double)	155
Sub	McLine (ByVal MapIndex As Long, ByRef DistList[] As Double)	157
Sub	McStartLineTo (ByVal MapIndex As Long, ByRef PosList[] As Double)	159
Sub	McLineTo (ByVal MapIndex As Long, ByRef PosList[] As Double)	162
Sub	McStartArcA (ByVal MapIndex As Long, ByVal XcentOffset As Double, ByVal YcentOffset As Double, ByVal EndAngle As Double)	165
Sub	McArcA (ByVal MapIndex As Long, ByVal XcentOffset As Double, ByVal YcentOffset As Double, ByVal EndAngle As Double)	167
Sub	McStartArcP (ByVal MapIndex As Long, ByVal XCentOffset As Double, ByVal YCentOffset As Double, ByVal XEndPointDist As Double, ByVal YEndPointDist As Double, ByVal Dir As Long)	170
Sub	McArcP (ByVal MapIndex As Long, ByVal XcentOffset As Double, ByVal YcentOffset As Double, ByVal XEndPointDist As Double, ByVal YEndPointDist As Double, ByVal Dir As Long)	172
Sub	McStartArcToA (ByVal MapIndex As Long, ByVal Xcent As Double, ByVal Ycent As Double, ByVal EndAngle As Double)	175
Sub	McArcToA (ByVal MapIndex As Long, ByVal Xcent As Double, ByVal Ycent As Double, ByVal EndAngle As Double)	177
Sub	McStartArcToP (ByVal MapIndex As Long, ByVal Xcent As Double, ByVal Ycent As	181

	Double, ByVal XendPos As Double, ByVal YendPos As Double, ByVal Dir As Long)	
Sub	McArcToP (ByVal MapIndex As Long, ByVal Xcent As Double, ByVal Ycent As Double, ByVal XendPos As Double, ByVal YendPos As Double, ByVal Dir As Long)	183
Sub	McCompleteArc (ByVal MapIndex As Long)	188
Sub	McBuildSpline (ByRef InArray As Double, ByVal NumInArray As Long, ByRef OutArray As Double, ByVal NumOutArray As Long)	189
Function	McStartHelical (ByVal MapIndex As Long, ByVal Zaxis As Long, ByVal Xcenter As Double, ByVal Ycenter As Double, ByVal Direction As Long, ByVal NumCircle As Long, ByVal LastArcAngle As Double, ByVal Zdistance As Double) As Boolean	190
Function	McAbortHelical As Boolean	192
Function	McMxDone (ByVal MapIndex As Long) As Boolean	193

모션제어(속도 및 위치 오버라이딩(Overriding))

Sub	McOverrideSpeedSet (ByVal Channel As Long)	195
Sub	McOverrideSpeedSetAll (ByVal NumAxis As Long, ByRef AxisList[] As Long)	197
Sub	McOverrideMove (ByVal Channel As Long, ByVal NewDistance As Double)	199
Sub	McOverrideMoveTo (ByVal Channel As Long, ByVal NewPosition As Double)	200

모션제어(원점 복귀(Home Return))

Page

Sub	McSetHomeConfig (ByVal Channel As Long, ByVal OrgMode As Long, ByVal OrgLogic As Long, ByVal EzCount As Long, ByVal EzLogic As Long, ByVal ErcOut As Long)	208
Sub	McHomeMove (ByVal Channel As Long, ByVal Direction As Long, ByVal RvsVel As Double)	210

모션제어(Manual Pulser 모드 모션)

Sub	McSetPE (ByVal Channel As Long, ByVal Enable As Short)	213
Sub	McSetPulserInputMode (ByVal Channel As Long, ByVal InputMode As Long, ByVal Inverse As Integer)	214
Sub	McPulserHomeMove (ByVal Channel As Long, ByVal HomeType As Long)	215
Sub	McStartPulserVMove (ByVal Channel As Long)	216
Sub	McStartPulserMove (ByVal Channel As Long, ByVal Distance As Double)	217
Sub	McPulserMove (ByVal Channel As Long, ByVal Distance As Double)	218
Sub	McStartPulserMoveTo (ByVal Channel As Long,By Val Position As Double)	219
Sub	McPulserMoveTo (ByVal Channel As Long, ByVal Position As Double)	220

모션제어(External Switch Operation 모드 모션 제어)		Page
Sub	McSetPE (ByVal Channel As Long, ByVal Enable As Short)	222
Sub	McStartVMoveEx (ByVal Channel As Long)	223
Sub	McStartMoveEx (ByVal Channel As Long, ByVal Distance As Double)	224

모션제어(리스트 모션(Listed Motion))		Page
Sub	McSetListMotionAxes (short MapMask)	228
Sub	McBeginList	229
Sub	McEndList	230
Sub	McStartListMotion	231
Sub	McAbortListMotion	232
Function	McCheckListMotionDone As Boolean	233

모션제어(상태 감시 및 제어)		
Function	McGetCurSpeed (ByVal Channel As Long) As Double	236
Sub	McEnableActSpdChk (ByVal Interval As Long)	237
Sub	McDisableActSpdChk	238
Function	McGetActualSpeed (ByVal Channel As Long) As Double	239
Function	McGetPositionA (ByVal Channel As Long) As Double	240
Sub	McSetPositionA (ByVal Channel As Long, ByVal Pos As Double)	241
Function	McGetPositionC (ByVal Channel As Long) As Double	242
Sub	McSetPositionC (ByVal Channel As Long, ByVal Pos As Double)	243
Function	McGetCountA (ByVal Channel As Long) As Long	244
Sub	McSetCountA (ByVal Channel As Long, ByVal Count As Long)	245
Function	McGetCountC (ByVal Channel As Long) As Long	246
Sub	McSetCountC (ByVal Channel As Long, ByVal Count As Long)	247
Function	McGetCountD (ByVal Channel As Long) As Long	248
Function	McSetCountD (ByVal Channel As Long, ByVal Count As Long) As Long	249
Function	McGetCountG (ByVal Channel As Long) As Long	250
Sub	McSetCountG (ByVal Channel As Long, ByVal Count As Long)	251
Function	McGetCountR (ByVal Channel As Long) As Long	252
Sub	McSetCountR (ByVal Channel As Long, ByVal Count As Long)	253
Function	McGetLatchState (ByVal Channel As Long) As Boolean	254
Function	McReadLatchCounter (ByVal Channel As Long, ByVal Counter As Long) As Long	255
Function	McGetMotionStatus (ByVal Channel As Long) As Long	256

Function **McGetMioStatus**(ByVal Channel As Long) As Long 257

모션제어(I/O(입출력) 환경설정)		Page
Sub	McSetFilterLogic (ByVal Channel As Long, ByVal Enable As Short)	261
Sub	McSetELL (ByVal Channel As Long, ByVal Logic As Long)	262
Sub	McSetMioCfgALM (ByVal Channel As Long, AlarmLogic As Long, AlarmMode As Long)	263
Sub	McGetMioCfgALM (ByVal Channel As Long, ByRef AlarmLogic As Long, ByRef AlarmMode As Long)	264
Sub	McSetMioCfgCLR (ByVal Channel As Long, ByVal CntrSel As Long, ByVal SignalType As Long)	265
Sub	McGetMioCfgCLR (ByVal Channel As Long, ByRef CntrSel As Long, ByRef SignalType As Long)	266
Sub	McSetMioCfgCMP (ByVal Channel As Long, ByVal CmpPulseWidth As Long, ByVal CmpLogic As Long)	267
Sub	McGetMioCfgCMP (ByVal Channel As Long, ByRef CmpPulseWidth Long, ByRef CmpLogic As Long)	268
Sub	McSetMioCfgDR (ByVal Channel As Long, ByVal Logic As Long)	269
Sub	McGetMioCfgDR (ByVal Channel As Long, ByRef Logic As Long)	270
Sub	McSetMioCfgEL (ByVal Channel As Long, ByVal EIMode As Long)	271
Sub	McGetMioCfgEL (ByVal Channel As Long, ByRef EIMode As Long)	272
Sub	McSetMioCfgINP (ByVal Channel As Long, ByVal InpEnable As Integer, ByVal InpLogic As Long)	273
Sub	McGetMioCfgINP (ByVal Channel As Long, ByRef plnpEnable As Long, ByRef InpLogic As Long)	274
Sub	McSetMioCfgERC (ByVal Channel As Long, ByVal ErcLogic As Long, ByVal ErcOnTime As Long)	275
Sub	McGetMioCfgERC (ByVal Channel As Long, ByRef ErcLogic As Long, ByRef ErcOnTime As Long)	276
Sub	McSetMioCfgLTC (ByVal Channel As Long, ByVal LtcLogic As Long, ByVal Ltc2Src As Long)	277
Sub	McSetMioCfgLTC (ByVal Channel As Long, ByRef LtcLogic As Long, ByRef Ltc2Src As Long)	278
Sub	McSetMioCfgSD (ByVal Channel As Long, ByVal SdEnable As Integer, ByVal SdLogic As Long, ByVal SdLatch As Long, ByVal SdMode As Long)	279
Sub	McGetMioCfgSD (ByVal Channel As Long, ByRef SdEnable As Integer, ByRef SdLogic As Long)	280

Appendix A 라이브러리 메소드 리스트

	As Long, ByRef SdLatch As Long, ByRef SdMode As Long)	
Sub	McSetMioCfgSTA (ByVal Channel As Long, ByVal Mode As Long, ByVal InputType As Long)	281
Sub	McSetMioCfgSTP (ByVal Channel As Long, ByVal Mode As Long)	282
Sub	McSetSoftLimit (ByVal Channel As Long, ByVal LimitP As Double, ByVal LimitN As Double)	283
Sub	McEnableSoftLimit (ByVal Channel As Long)	284
Sub	McDisableSoftLimit (ByVal Channel As Long)	285
Sub	McSetErrorCompare (ByVal Channel As Long, ByVal Tol As Double, ByVal Enable As Long)	286
Sub	McSetGeneralCompare (ByVal Channel As Long, ByVal CmpSrc As Long, ByVal CmpMethod As Long, ByVal CmpAction As Long, ByVal Data As Double)	287
Sub	McSetTriggerCompare (ByVal Channel As Long, ByVal CmpSrc As Long, ByVal CmpMethod As Long, ByVal Data As Double)	288
Sub	McRegTableCCMP (ByVal Channel As Long, ByRef DataBuffer As Double, ByVal NumData As Long) As Boolean	289
Sub	McBuildTableCCMP (ByVal Channel As Long, ByVal StartData As Double, ByVal Interval As Double, ByVal NumData As Long) As Boolean	290
Sub	McStartCCMP (ByVal Channel As Long, ByVal CmpSrc As Long , ByVal CmpMethod As Long) As Boolean	291
Sub	McStopCCMP (ByVal Channel As Long) As Boolean	293
모션 제어 (인터럽트 관련 메소드)		Page
Sub	McMaskInterrupt (ByVal Channel As Long, ByVal Mask As Long)	295
Sub	McEnableInterrupt (ByVal EventHandle As Long)	296
Sub	McDisableInterrupt	297
Function	McGetAxisIntState (ByVal Channel As Long) As Boolean	298
Sub	McGetIntStatus (ByVal Channel As Long, ByRef ErrorStatus As Long, ByRef EventStatus As Long)	299

A.2 메쏘드별 지원 가능 디바이스 리스트

메쏘드명	각 보드별 지원 여부									
	LX101	LX102	LX103	LX201	LX202	LX203	LX301	LX401	LX402	LX50x
LoadDevice	V	V	V	V	V	V	V	V	V	V
UnLoadDevice	V	V	V	V	V	V	V	V	V	V
AdSetInputType	V	V	V							
AdSetRange	V	V	V	V	V	V				
AdGetDigit	V	V	V							
AdGetVolt	V	V	V							
Us1Start	V	V	V							
Us1Stop	V	V	V							
Us1RetrvOne	V	V	V							
Us1RetrvChannel	V	V	V							
Us1RetrvBlock	V	V	V							
Us1ReleaseBuf	V	V	V							
Us2SetTriggerEvent				V	V	V				
Us2Start				V	V	V				
Us2Resume				V	V	V				
Us2IsBufFull				V	V	V				
Us2ChangeScanFreq				V	V	V				
Us2DmaCount				V	V	V				
Us2RetrvChannel				V	V	V				
Us2Stop				V	V	V				
Us2ReleaseBuf				V	V	V				
DaOut	V	V	V				V			
WfmStart	V	V	V				V			
WfmReload	V	V	V				V			
WfmRateChange	V	V	V				V			
WfmGetCurPos	V	V	V				V			
WfmGetCurLoops	V	V	V				V			
WfmStop	V	V	V				V			
DioSetUsage	V	V	V	V	V	V	V	V		V
DiGetOne	V	V	V	V	V	V	V	V		V

Appendix A 라이브러리 메소드 리스트

DiGetAll	V	V	V	V	V	V	V	V		V
DoPutOne	V	V	V	V	V	V	V	V		V
DoPutAll	V	V	V	V	V	V	V	V		V
SdioInitComm									V	
SdioCheckModule									V	
SdioSetDioUsage									V	
SdioReadLowByte									V	
SdioReadHighByte									V	
SdioWriteLowByte									V	
SdioWriteHighByte									V	
ReadCounter32	V	V	V				V	V		
ClearCounter32	V	V	V				V	V		

※ 모션에 관련된 메소드는 COMI-LX50x에만 적용 가능하며 본 리스트에는 생략되었습니다.