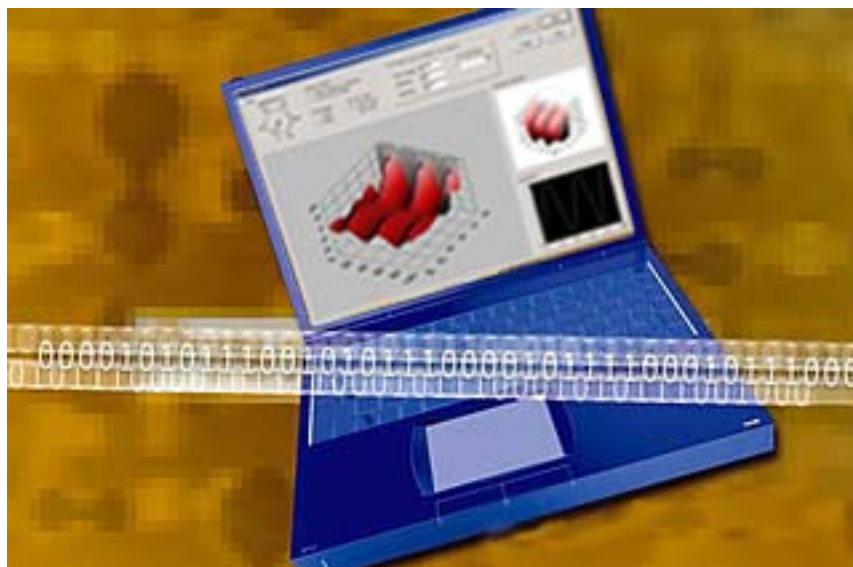

COMIZOA DAQ System (ST) Visual Basic



*COMputer Innovation
is Zoomed by Our Affection!*



저작권자 : ㈜커미조아

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

2001 년 11 월 20 일 중판 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

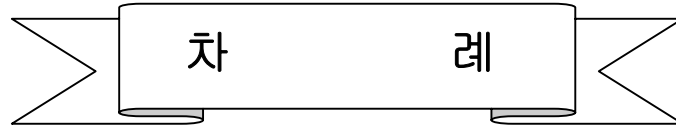
www.comizoa.co.kr

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	10
2.2 아날로그 입력(Analog Input)	13
2.3 아날로그 출력(Analog Output)	18
2.4 디지털 입출력(Digital Input/Output)	21
2.4.1 일반적인 디지털 입출력	22
2.4.2 시리얼 통신을 이용한 디지털 입출력	27
2.5 모션 제어(Motion Control)	40
2.5.1 모션 초기화 및 환경설정 메소드	41
2.5.2 Single Axis 모션 제어 메소드	56
2.5.3 Multi-Axis 동시제어 메소드	83
2.5.4 Coordinated Motion 메소드	97
2.5.5 속도 및 위치 오버라이딩(Overriding) 메소드	141
2.5.6 원점 복귀(Home Return) 메소드	148
2.5.7 Manual Pulser 모드 모션 제어 메소드	158
2.5.8 리스트 모션(Listed Motion) 메소드	166
2.5.9 상태 감시 및 제어 메소드	174
2.5.10 I/O(입출력) 환경설정 메소드	196
2.5.11 인터럽트 관련 메소드	213
Appendix A 라이브러리 메소드 리스트	221
A.1 기능별 메소드 색인	221
A.2 메소드별 지원 가능 디바이스 리스트	227

CHAPTER 1

본 장에서는 COMI-ST 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-ST 디바이스 시리즈를 제어하는 Visual Basic 으로 작성된 프로그램을 제작, 배포하실 때에도, 라이브러리를 함께 배포하셔야 합니다.

CHAPTER 1

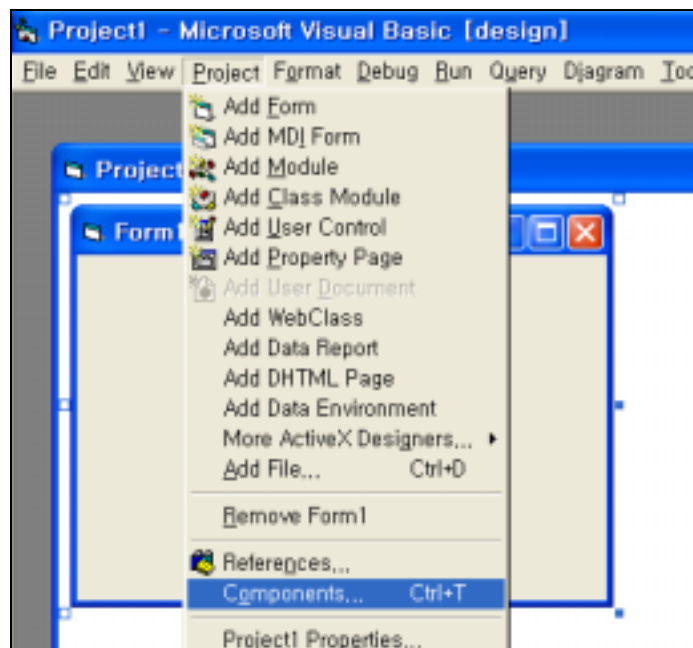
본 장에서는 사용자가 COMI-ST 시리즈 디바이스를 제어하는 Visual Basic 프로그램을 구현할 때 유용하게 사용될 수 있는 라이브러리의 설치와 배포방법을 설명합니다. ㈜커미조사에서 제공하는 COMI-ST 시리즈용 Visual Basic 라이브러리는 모든 종류의 COMI-ST 시리즈 디바이스에 적용 가능한 통합 라이브러리입니다.

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

즉, Visual Basic 에서 COMI-ST 시리즈 디바이스를 제어하려면, 디바이스 드라이버와 ComidasCx.dll 파일 그리고, ComiCxAx.ocx 파일이 시스템에 설치되어야 합니다.

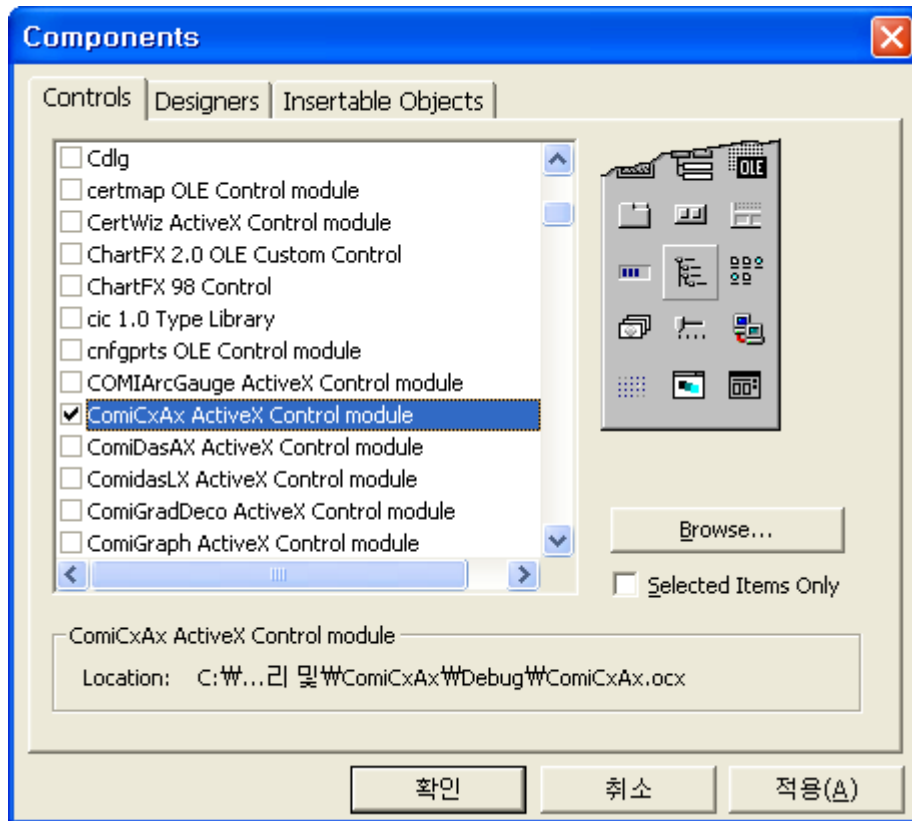
1.1 라이브러리 설치

‘ST 소프트웨어 매뉴얼’ 을 참조하여, 디바이스의 드라이버와 응용프로그램을 설치하면, C:\WProgram Files\COMIZOAW\Comidas-STWVB 폴더내에, Visual Basic 용 라이브러리, 예제, 매뉴얼이 복사되어집니다. 라이브러리인 ComiCxAx.ocx 는, 설치마법사가 자동으로 윈도우에 등록하여주므로 사용자는 Visual Basic 에서 컴포넌트를 추가하여 주기만 하면 됩니다.



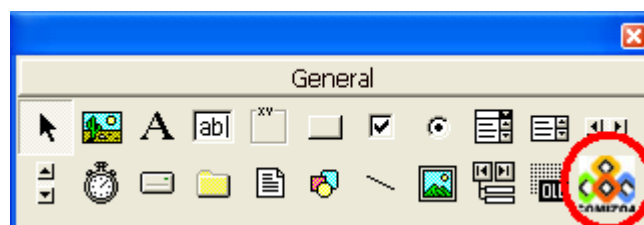
[그림 1.1] Components 창 열기

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



[그림 1.2] Components 창에서의 ComiCxAx.ocx 콘트롤 선택

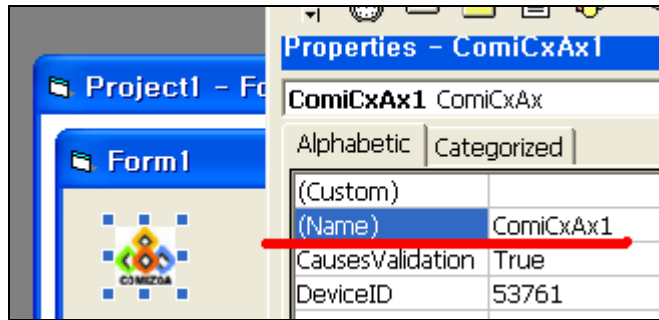
‘ComiCxAx ActiveX Control module’ 을 선택하시어, 체크표시를 하시고 ‘확인’ 을 클릭합니다. (윈도우에 등록되어 있지 않은 ocx 인 경우에는 ‘Browse(찾아보기)’ 를 클릭하여, 그 위치를 설정해 주어야 하지만, ComiCxAx.ocx 는 드라이버 및 응용프로그램 설치시에 자동으로 윈도우의 시스템 폴더에 복사되어진 후, 레지스트리에 자동 등록됩니다.)



[그림 1.3] toolbox 에 등록된 ComiCxAx.ocx

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

롤들을 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, ComiCxAx 컨트롤은 내부적으로 COMI-ST 디바이스 시리즈를 제어하고 디바이스로부터 데이터를 전달받을 때만 사용되어집니다.



[그림 1.4] 폼에 삽입된 ComiCxAx 컨트롤의 인스턴스와 인스턴스명 설정

하나의 폼에는 여러 ComiCxAx 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 'ComiCxAx1' 로 설정하고 프로그래밍을 하는것으로 간주하고 설명하겠습니다.

1.2 프로그램의 배포 방법

프로그램 배포시에도 디바이스 드라이버와 ComidasCx.dll, ComiCxAx.ocx 파일을 같이 배포하여야 합니다.

'ST 소프트웨어 매뉴얼' 을 참조하여, 디바이스의 드라이버와 응용프로그램을 설치하시면, C:\Program Files\COMIZOAWComidas-STWDriver 폴더내에 디바이스 드라이버가 담겨있습니다. 배포시에는 이 드라이버 파일을 같이 배포하셔서 최종사용자로 하여금 디바이스 설치시 드라이버를 설치하도록 하셔야 합니다.

또한, C:\Program Files\COMIZOAWComidas-STWC_CPPWLibrary 폴더에 ComidasCx.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 2000, XP 인 경우 C:\WINDOWS\system32 입니다.)

마지막으로, C:\Program Files\COMIZOA\Comidas-STWVBW\Library 폴더에 ComiCxAx.ocx 파일이 담겨있습니다. 배포시에는 이 파일이 대상 시스템의 OS에 등록되어야 합니다. 즉, 이 파일의 위치는 상관없이 OS의 레지스트리에 등록이 되어야하는데, 등록방법으로는 DOS-command 창에서 'regsvr32 ComiCxAx.ocx' 라는 명령을 실행시켜주는 방법과 인스톨 셸드와 같은 셋업 프로그램을 생성해주는 유틸리티를 이용하여, 최종사용자가 배포된 프로그램을 셋업이 자동으로 동시에 ocx 파일이 OS에 등록되도록 하는 방법이 있습니다

Chapter 2

method

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

3.1 라이브러리 및 디바이스 시작/종료	6
3.2 아날로그 입력(Analog Input)	11
3.3 아날로그 출력(Analog Output)	11
3.4 디지털 입출력(Digital Input/Output)	11
3.5 모션 제어(Motion Control)	112

CHAPTER 2

method

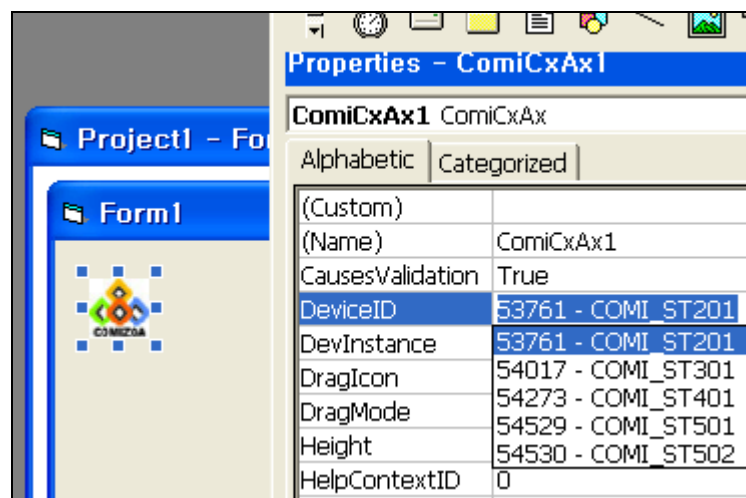
하나의 폼에는 다수의 ComiCxAx 컨트롤의 인스턴스가 삽입될 수 있고, 이들은 각각 자신만의 디바이스 ID (DeviceID Property)와 디바이스 Instance (DeviceInstance Property)로 시스템에 연결된 디바이스를 가르킵니다. 각각의 인스턴스마다 자신이 가르키고 있는 디바이스를, ComiCxAx 컨트롤이 제공하는 메소드들을 이용하여 계측하고 또한 제어하게 됩니다. Visual Basic 의 다른 컴포넌트와 마찬가지로, COMI-ST 디바이스 시리즈를 이용한 프로그램을 잘 구현하려면, ComiCxAx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

ComiCxAx 는 DeviceID 와 DeviceInstance 라는 두 개의 Property 를 가지고 있습니다. DeviceID 는 'COMI-ST201' , 'COMI-ST301' 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

```
ComiCxAx1.DeviceID = COMI-ST201
```

과 같이 설정하여 줄 수도 있습니다.



[그림 2.1] 폼에 삽입된 ComiCxAx 컨트롤의 DeviceID 설정

DeviceInstance 속성은, 같은 종류의 대바이스가 여러 개 설치되었을 때, 이를 구분하는 속성입니다. 예를 들어, 각각 하나씩의 COMI-ST201 과 COMI-ST301 이 있다면 이는

```
ComiCxAx1.DeviceID = COMI-ST201
ComiCxAx2.DeviceID = COMI-ST301
```

라고 설정하여 주면 되지만 같은 COMI-ST201 디바이스가 두 개 이상 설치되어 있다면,

```
ComiCxAx1.DeviceInstance = 0  
ComiCxAx2.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 적용되어져야 할 메소드들입니다. 이에 관련된 메소드들의 리스트는 다음과 같습니다.

메소드명	각 보드별 지원 여부			
	ST201	ST301	ST401	ST502
LoadDevice	V	V	V	V
UnloadDevice	V	V	V	V

[표 3-1] 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.

사용예

2 대의 COMI-ST201 디바이스를 이용하여 프로그램 할 때의 코드 예

```
ComiCxAx1.DeviceID = COMI_ST201
ComiCxAx2.DeviceID = COMI_ST201

ComiCxAx1.DevInstance = 0
ComiCxAx2.DevInstance = 1

Success = ComiCxAx1.LoadDevice

If (Success = True) Then
    '
Else
    Call MsgBox("Can't load first COMI-ST201 device!",vbOKOnly)
End If

Success = ComiCxAx2.LoadDevice

If (Success = True) Then
    '
Else
    Call MsgBox("Can't load second COMI-ST201 device!",vbOKOnly)
End If

Call ComiCxAx1.UnloadDevice
Call ComiCxAx2.UnloadDevice
```

■ UnloadDevice

메소드 원형

```
sub UnloadDevice
```

메소드 설명

이 메소드는 하나의 COMIDAS 디바이스를 언로드(unload)합니다.

사용예

```
ComiCxAx1.DeviceID = COMI-ST201
Success = ComiCxAx1.LoadDevice

If (Success = True) Then
    '
Else
    Call MsgBox("Can't load COMI-ST201 device!",vbOKOnly)
End If

Call ComiCxAx1.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 point A/D 방식이며 두 번째는 A/D Scan 방식입니다.

Single point A/D 는 사용자가 원하는 시점에서 소프트웨어적으로 A/D trigger 를 하여 A/D 데이터를 획득하는 방식입니다.

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 point A/D 방식에 비해 속도가 빠르고 정확한 샘플링 주기를 보장할 수 있습니다.

메소드명	각 보드별 적용 여부			
	ST201	ST301	ST401	ST502
AdSetInputType	V			
AdSetRange	V			
AdGetDigit	V			
AdGetVolt	V			

[표 3-2] 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 Differential 로 설정합니다.
1 또는 AI_SINGLE	아날로그 입력 신호의 연결 형식을 Single Ended 로 설정합니다.

■ 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. 유효한 vmin 값은 0, -1, -2, -5, -10 입니다.
- ▶ **vmax** : A/D 범위의 최대값을 지정합니다. 유효한 vmax 값은 1, 2, 5, 10 입니다.

Return 값

메소드 수행의 성공 여부

Value	Meaning
FALSE	메소드 수행 실패
TRUE	메소드 수행 성공

사용예

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

```
Call ComiCxAx1.LoadDevice
Call ComiCxAx1.AdSetRange(0, -10, 10)
Call ComiCxAx1.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 다릅니다. COM1-ST201 의 경우, 12 Bit 의 분해능을 가지고 있어 0~4095 범위의 정수값을 반환합니다.

□ 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

D_{\min} : 정수값의 최소 범위값

사용예

A/D CH0 의 A/D 값을 정수값으로 받는 예

```
Call ComiCxAx1.LoadDevice
```

```
Call ComiCxAx1.AdSetInputType(AI_SINGLE)
```

```
Call ComiCxAx1.AdSetRange ( 0 , VMIN, VMAX)
```

```
Result = ComiCxAx1.AdGetDigit (0)
```

스캔된 값 처리부..

```
Call ComiCxAx1.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.

사용예

A/D CH0 의 A/D 값을 Voltage 값으로 받는 예

```
Call ComiCxAx1.LoadDevice

Call ComiCxAx1.AdSetInputType(AI_SINGLE)
Call ComiCxAx1.AdSetRange ( 0 , VMIN, VMAX)

Result = ComiCxAx1.AdGetVolt (0)
'스캔된 값 처리부..

Call ComiCxAx1.UnLoaddevice
}
```

2.3 아날로그 출력(Analog Output)

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

COMI-ST301 디바이스는 일반적인 Analog Output 기능으로써 사용자가 지정한 전압을 출력하는 기능을 가지고 있습니다.

메소드명	각 보드별 지원 여부			
	ST201	ST301	ST401	ST502
DaSetRange		V		
DaOut		V		

[표 3-5] Analog Output 관련 메소드 리스트 및 각 보드별 지원 여부

DaSetRange

메소드 원형

```
sub DA_Out (ByVal ch As Long, ByVal vmin As Long, ByVal vmax As Long)
```

메소드 설명

이 메소드는 지정한 Analog Output 채널에 지정한 출력범위를 설정합니다.

매개 변수

- ▶ *ch* : Analog Output 채널번호. 채널 번호는 0 부터 시작합니다.77
- ▶ *vmin* : Analog Output 출력 범위의 최소값
- ▶ *vmax* : Analog Output 출력 범위의 최대값

DaOut

메소드 원형

Function **DA_Out** (ByVal ch As Long, ByVal volt As Single) As Boolean

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
FALSE	메소드 수행 실패
TRUE	메소드 수행 성공

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.

㈜커미조아에서 제공하는 COMI-ST 시리즈 디바이스는 크게 두 가지로 구분되는 디지털 입출력 기능을 제공합니다.

하나는 일반적인 디지털 입출력 기능입니다. 일반적인 디지털 입출력은 PC 에 장착된 Compact PCI 보드에서 디지털 입출력을 직접제어하는 것을 의미하며 이와 관련된 메소드들은 [2.4.1 일반적인 디지털 입출력] 단원에서 설명됩니다. 현재 출시된 COMI-ST 시리즈 디바이스중에서 일반적인 디지털 입출력 기능을 제공하는 디바이스는 COMI-ST502 모션제어 보드가 있습니다.

다른 하나는 시리얼 통신(RS-422)을 이용한 디지털 입출력 기능입니다. 이 기능은 PC 에 장착되어 시리얼 통신을 관장하는 마스터보드(COMI-ST401 보드)와 외부에 설치되어 실제 디지털 입출력을 제어하는 터미널 모듈(COMI-STM4A)이 RS-422 시리얼 통신으로 연결되어 제어되는 방식입니다. 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 Compact PCI 보드에서 디지털 입출력을 직접제어하는 것을 의미합니다. 현재 출시된 COMI-ST 시리즈 디바이스중에서 일반적인 디지털 입출력 기능을 제공하는 디바이스는 COMI-ST502 모션제어 보드가 있습니다.

메소드명	각 보드별 적용 여부			
	ST201	ST301	ST401	ST-502
DiGetOne				V
DiGetAll				V
DoPutOne				V
DoPutAll				V

[표 3-6] Digital Input/Output 에 관련된 메소드 리스트 및 각 보드별 지원 여부

■ 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

■ 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.

■ DoPutOne

메소드 원형 :

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

메소드 설명 :

이 메소드는 지정한 Digital Output 채널에 지정한 Status로 출력을 내보냅니다.

매개 변수 :

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

■ 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.

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 PC 에 장착되어 시리얼 통신을 관장하는 마스터보드(COMI-ST401 보드)와 외부에 설치되어 실제 디지털 입출력을 제어하는 터미널 모듈(COMI-STM4A)이 RS-422 시리얼 통신으로 연결되어 제어되는 방식입니다.

각 COMI-STM4A 터미널 모듈은 16 채널의 디지털 입출력 채널을 제공하며, 하나의 COMI-ST401 마스터 보드에 16 개의 COMI-STM4A 터미널 모듈이 확장되어 연결될 수 있어서 하나의 마스터 보드가 총 256 개의 디지털 입출력 채널을 제어할 수 있습니다.

메소드명	각 보드별 적용 여부			
	ST201	ST301	ST401	ST502
SdioInitComm			V	
SdioCheckModule			V	
SdioSetDioUsage			V	
SdioReadLowByte			V	
SdioReadHighByte			V	
SdioWriteLowByte			V	
SdioWriteHighByte			V	

[표 3-6] Sireal Digital I/O 에 관련된 메소드 리스트 및 각 보드별 지원 여부

■ SdioInitComm

메소드 원형 :

Function **SdioInitComm** As Boolean

메소드 설명 :

이 메소드는 COMI-ST401 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
0	실패
1	성공

예 제

```
Call ComiCxAx1.LoadDevice

` RS-422 통신 초기화
Call ComiCxAx1.SdioInitComm
` 여기에서 필요한 SDIO 메소드들을 수행한다.
*****

Call ComiCxAx1.UnloadDevice
```

■ SdioCheckModule

메소드 원형 :

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

메소드 설명 :

이 메소드는 지정한 주소값(모듈 번호)을 가지는 COMI-STM4A 디지털 입출력 터미널 모듈이 COMI-ST401 마스터 보드에 현재 연결되어 있는지를 체크하는 메소드입니다.

매개 변수 :

▶ **ModuleNo** : 검색하고자 하는 COMI-STM4A 디지털 입출력 터미널 모듈 번호를 지정합니다. 모듈 번호는 0 ~ 15 까지 지정할 수 있으며 해당 COMI-STM4A 터미널 모듈에서 점퍼로 설정되는 모듈번호와 일치하여야 합니다.

Return

지정한 COMI-STM4A 터미널 모듈이 연결되었는지를 나타내는 값

Value	Meaning
0	연결되어 있지 않음
1	연결되어 있음

예 제

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

```

Call ComiCxAx1.LoadDevice

` RS-422 통신 초기화
Call ComiCxAx1.SdioInitComm

` 여기에서 필요한 SDIO 메소드들을 수행한다.
For i=0 to 15
    If ComiCxAx1.SdioCheckModule(i) Then
        Call MsgBox("연결됨",vbOkonly)
    Else
        Call MsgBox("연결 안됨",vbOkonly)
    End if
Next i

Call ComiCxAx1.UnloadDevice
    
```

SdioSetDioUsage

메소드 원형 :

```
Function SdioSetDioUsage ( ByVal ModuleNo As Long, ByVal Usage As TCmDiDoUsage) As Boolean
```

메소드 설명 :

이 메소드는 지정한 COM1-STM4A 터미널 모듈의 Digital Input/Output 단자의 용도를 설정합니다. 각 COM1-STM4A 터미널 모듈은 16 개의 디지털 입출력 채널을 제공하며 16 채널의 입출력 모드를 4 가지 방법으로 설정할 수 있습니다.

매개 변수

▶ **ModuleNo** : 설정하고자 하는 COM1-STM4A 디지털 입출력 터미널 모듈 번호를 지정합니다. 모듈 번호는 0 ~ 15 까지 지정할 수 있으며 해당 COM1-STM4A 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_DO	CH0~CH7 : 디지털 입력, CH8~CH15 : 디지털 출력
2 또는 DO_DI	CH0~CH7 : 디지털 출력, CH8~CH15 : 디지털 입력
3 또는 DO_ONLY	전 채널을 디지털 출력 채널로 사용합니다.

Return

메소드 수행의 성공 여부

Value	Meaning
0	실패
1	성공

예 제

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

Call ComiCxAx1.LoadDevice 이며 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 ComiCxAx1.SdioSetDioUsage(MOD0, DI_DO);

di_byte = ComiCxAx1.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 ComiCxAx1.SdioWriteHighByte( MOD0, do_byte);

Call ComiCxAx1.UnloadDevice
    
```

■ SdioReadLowByte

메소드 원형 :

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

메소드 설명 :

이 메소드는 지정한 COM1-STM4A 터미널 모듈의 0 번 채널부터 7 번 채널까지의 현재 입력 또는 출력 상태를 읽어들이합니다. 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-STM4A 디지털 입출력 터미널 모듈 번호를 지정합니다. 모듈 번호는 0 ~ 15 까지 지정할 수 있으며 해당 COM1-STM4A 터미널 모듈에서 점퍼로 설정되는 모듈번호와 일치하여야 합니다.

Return

지정한 COM1-STM4A 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 ComiCxAx1.LoadDevice 이며 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 ComiCxAx1.SdioSetDioUsage(MOD0, DI_DO);
```

```
di_byte = ComiCxAx1.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 ComiCxAx1.SdioWriteHighByte(MOD0, do_byte);

Call ComiCxAx1.UnloadDevice
    
```

■ SdioReadHighByte

메소드 원형 :

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

메소드 설명 :

이 메소드는 지정한 COM1-STM4A 터미널 모듈의 8 번 채널부터 15 번 채널까지의 현재 입력 또는 출력 상태를 읽어들이니다. 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

- ▶ **Device** : 디바이스 핸들 값입니다. 이 값은 COM1LX_LoadDevice 메소드에 의해 얻어진 값이어야 합니다.
- ▶ **ModuleNo** : 설정하고자 하는 COM1-STM4A 디지털 입출력 터미널 모듈 번호를 지정합니다. 모듈 번호는 0 ~ 15 까지 지정할 수 있으며 해당 COM1-STM4A 터미널 모듈에서 점퍼로 설정되는 모듈번호와 일치하여야 합니다.

Return

지정한 COM1-STM4A 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 ComiCxAx1.LoadDevice 이며 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 ComiCxAx1.SdioSetDioUsage(MOD0, DI_DO);
```



```

di_byte = ComiCxAx1.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 ComiCxAx1.SdioWriteLowByte(MOD0, do_byte);

Call ComiCxAx1.UnloadDevice
    
```

■ SdioWriteLowByte

메소드 원형 :

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

메소드 설명 :

이 메소드는 지정한 COMI-STM4A 터미널 모듈의 CH0 ~ CH7 에 디지털 출력을 내보냅니다.
이 메소드를 사용하기 전에 SdioSetDioUsage 메소드를 사용하여 CH0 ~ CH7 을 디지털 출력 채널로 설정하여야 합니다.

매개 변수

- ▶ **Device** : 디바이스 핸들 값입니다. 이 값은 COMILX_LoadDevice 메소드에 의해 얻어진 값이어야 합니다.
- ▶ **ModuleNo** : 설정하고자 하는 COMI-STM4A 디지털 입출력 터미널 모듈 번호를 지정합니다. 모듈 번호는 0 ~ 15 까지 지정할 수 있으며 해당 COMI-STM4A 터미널 모듈에서 점퍼로 설정되는 모듈번호와 일치하여야 합니다.
- ▶ **LowByte** : 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
0	실패
1	성공

참고

특정 채널의 상태만 변경하고자 하는 경우 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 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
```

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 ComiCxAx1.LoadDevice 이며 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 ComiCxAx1.SdioSetDioUsage(MOD0, DO_DI);
```

```
di_byte = ComiCxAx1.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 ComiCxAx1.SdioWriteLowByte(MOD0, do_byte);
```

```
Call ComiCxAx1.UnloadDevice
```

■ SdioWriteHighByte

메소드 원형 :

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

메소드 설명 :

이 메소드는 지정한 COM1-STM4A 터미널 모듈의 CH8 ~ CH15 에 디지털 출력을 내보냅니다.
이 메소드를 사용하기 전에 SdioSetDioUsage 메소드를 사용하여 CH8 ~ CH15 를 디지털 출력 채널로 설정하여야 합니다.

매개 변수

- ▶ **Device** : 디바이스 핸들 값입니다. 이 값은 COM1LX_LoadDevice 메소드에 의해 얻어진 값이어야 합니다.
- ▶ **ModuleNo** : 설정하고자 하는 COM1-STM4A 디지털 입출력 터미널 모듈 번호를 지정합니다. 모듈 번호는 0 ~ 15 까지 지정할 수 있으며 해당 COM1-STM4A 터미널 모듈에서 점퍼로 설정되는 모듈번호와 일치하여야 합니다.
- ▶ **HighByte** : 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 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 Function writeBitToByte(Status, index, bit) As Byte
    If (bit = 1) Then
        writeBitToByte = Status Or (2 ^ (index-8))
    Else
        writeBitToByte = Status And Not (2 ^ (index-8))
    End If
End Function
```

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 ComiCxAx1.LoadDevice 이며 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 ComiCxAx1.SdioSetDioUsage(MOD0, DI_DO);
```

```
di_byte = ComiCxAx1.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 ComiCxAx1.SdioWriteHighByte(MOD0, do_byte);
```

```
Call ComiCxAx1.UnloadDevice
```

2.5 모션 제어(Motion Control)

이 단원에서 설명하는 모션제어 기능은 (주)커미조아에서 개발한 COMI-ST502 모션제어 (Motion Control) 전용 보드에서 제공하는 기능입니다. 따라서 이 단원에서 소개되는 모든 메소드는 COMI-ST502 보드에만 적용가능합니다.

COMI-ST502 모션제어 보드는 펄스 구동 방식에 의하여 스텝모터나 서보 모터의 정밀 위치 제어 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.

2.5.1 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.	
Sub McSetBlockingMode (ByVal Blocking As Integer) 모션이 완료될때까지 루프를 돌 때 윈도우 또는 시스템 이벤트를 처리할 수 있도록 할지를 결정하는 메소드	
Sub McSetOutputMode (ByVal Channel As Long, ByVal OutputMode As Long) 현재 설정된 Command 펄스의 출력 모드를 얻어옵니다.	
Function McGetOutputMode (ByVal Channel As Long) As Long 현재 설정된 Command 펄스의 출력 모드를 반환합니다.	
Sub McSetInputMode (ByVal Channel As Long, ByVal InputMode As Long, ByVal PulseLogic As Long) Feedback 펄스의 입력 모드를 설정합니다.	
Sub McSetSpeedRange(ByVal Channel, ByVal MaxSpeed) 모션에 적용할 수 있는 최저/최고 속도를 제한합니다.	
Sub McGetInputMode (ByVal Channel As Long, ByRef InputMode As Long, ByVal PulseLogic As Long) 현재 설정된 Feedback 펄스의 입력 모드를 얻어옵니다.	
Sub McSetUnitDistance (ByVal Channel As Long, ByVal UnitDist As Double) 논리적 단위 거리에 대한 펄스 수를 설정합니다.	
Function McGetUnitDistance (ByVal Channel As Long) As Double 현재 설정된 논리적 단위 거리에 대한 펄스 수를 반환합니다.	
Sub McSetUnitSpeed (ByVal Channel As Long, ByVal UnitSpeed As Long) 논리적 단위 속도에 대한 펄스 출력 속도(PPS)를 설정합니다.	
double McGetUnitSpeed (ByVal Channel) 현재 설정된 논리적 단위 속도에 대한 실제 펄스 출력 속도(PPS)를 반환합니다.	

■ 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 ComiCxAx1.LoadDevice  
Call ComiCxAx1.McReset  
Call ComiCxAx1.UnloadDevice
```


■ 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 ComiCxAx1.LoadDevice

Call ComiCxAx1.McSetBlockingMode ( 0 )
' Set constant speed mode
Call ComiCxAx1.McSetSpeedMode( 0, 0)
' Set speed as 5000 PPS
Call ComiCxAx1.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 ComiCxAx1.McMove(0, 5000)

Call ComiCxAx1.UnloadDevice
```

■ McSetOutputMode

메소드 원형

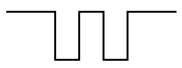











Sub **McSetOutputMode** (ByVal Channel As Long, ByVal OutputMode As Long)

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 입력 모드를 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

■ 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 설정 가능하며 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 속도

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

■ 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 0~3의 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 0~1의 정수이며 각 값의 의미는 McSetInputMode() 메소드를 참조하십시오. 이 값이 NULL 이면 입력 로직을 반환하지 않습니다.

■ 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 1로 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 10으로 하면 됩니다.

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

예제

1mm 이동하는데 필요한 펄스수가 100 펄스이고 1 회전에 필요한 펄스수가 10000 펄스일때 거리의 단위를 mm로, 속도의 단위를 rpm으로 설정하는 예제입니다.

Call ComiCxAx1.**LoadDevice**

` Set 100 pulses for unit distance
 ` 이 예제에서는 1mm 이동에 필요한 펄스수를 100 펄스로
 ` 가정하고 단위 거리를 1mm로 설정한 것이다.
 Call ComiCxAx1.**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 ComiCxAx1.**McSetUnitSpeed**(0, 10000./60)


```
Call ComiCxAx1.McSetSpeed(0, 0, 60) ` Set speed as 60 rpm  
Call ComiCxAx1.McSetSpeedMode(0, 0) ` Set contant speed mode  
  
Call ComiCxAx1.McMove(0, 100) `60 rpm 의 속도로 100mm 이동  
  
Call ComiCxAx1.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 속도와 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.

Call ComiCxAx1.**LoadDivice**

```

` Set 10 pulses for unit distance
` 이 예제에서는 1 회전에 필요한 펄스 수를 3600 펄스로
` 가정하고 단위 거리를 1°로 설정한 것이다.
Call ComiCxAx1.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 ComiCxAx1.McSetUnitSpeed(0, 3600./60)
` Set trapezoidal speed mode
Call ComiCxAx1.McSetSpeedMode(0, 1)
` Set speed as 100 rpm
Call ComiCxAx1.McSetSpeed(0, 0, 100)
`가속도와 감속도를 각각 200rpm/s로 설정한다. 이렇게 하면 작업속도가
` 100rpm이므로 가속 및 감속 시간은 각각 0.5 초 걸린다.
Call ComiCxAx1.McSetAccel(0, 200, 200)
` 모터를 720°회전한다. 실제로는 720*10 펄스가 출력된다.
Call ComiCxAx1.McMove(0, 720)

Call ComiCxAx1.UnloadDevice
    
```

■ 예제 2

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

```

Call ComiCxAx1.LoadDivice

` Set 1000 pulses for unit distance
` 이 예제에서는 1cm 이동에 필요한 펄스수를 1000 펄스로
` 가정하고 단위 거리를 1cm로 설정한 것이다.
Call ComiCxAx1.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 ComiCxAx1.McSetUnitSpeed(0, 1000)

Call ComiCxAx1.McSetSpeed(0, 0, 50) ` Set speed as 50 cm/sec/
Call ComiCxAx1.McSetSpeedMode(0, 0) ` Set constant speed mode

Call ComiCxAx1.McMove(0, 10) ` 50 cm/sec 의 속도로 10cm 이동 . 실제로는
10*1000=10000 펄스가 출력된다.

Call ComiCxAx1.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.

2.5.2 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 ModelIndex As Long) Motion의 속도 모드를 설정합니다.	
Sub McSetSpeed (ByVal Channel, ByVal IniSpeed As Double, ByVal WorkSpeed As Double) Motion의 속도를 설정합니다.	
Sub McSetAccel(ByVal Channel, ByVal Accel As Double, ByVal Decel As Double) Motion의 가/감속도를 설정합니다.	
Sub McSetScurve (ByVal Channel, ByVal Svacc As Double, ByVal Svdec Double) 속도모드를 S-curve 속도 패턴으로 설정한 경우에 S-curve Section의 범위를 속도단위로 설정합니다.	
Sub McStartVMove (ByVal Channel As Long, ByVal Direction As Long) 작업속도까지 가속한 후에 작업속도를 유지하며 정지메소드가 호출될 때까지 지정한 방향으로의 모션을 계속 수행합니다.	
Sub McStartMove (ByVal Channel As Long, ByVal Distance As Long) 현재의 위치에서 지정한 거리만큼 이동을 수행합니다. 이 메소드는 모션을 시작시킨 후 바로 Return 합니다.	
Sub McMove (ByVal Channel As Long, ByVal Distance As Double) 현재의 위치에서 지정한 거리만큼 이동을 수행합니다. 이 메소드는 모션이 완료될 때까지 Return 되지 않습니다.	
Sub McStartMoveTo (ByVal Channel As Long, ByVal Position As Double) 지정한 절대좌표로의 이동을 수행합니다. 이 메소드는 모션(Motion)을 시작시킨 후에 바로 Return 합니다.	
Sub McMoveTo (ByVal Channel As Long, ByVal Position As Double) 지정한 절대좌표로의 이동을 수행합니다. 이 메소드는 모션이 완료될때까지 Return 되지 않습니다.	
Sub McStop (ByVal Channel As Long) 지정한 축에 대한 모션을 감속 후 정지합니다.	
Sub McEngStop (ByVal Channel As Long)	

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

■ 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

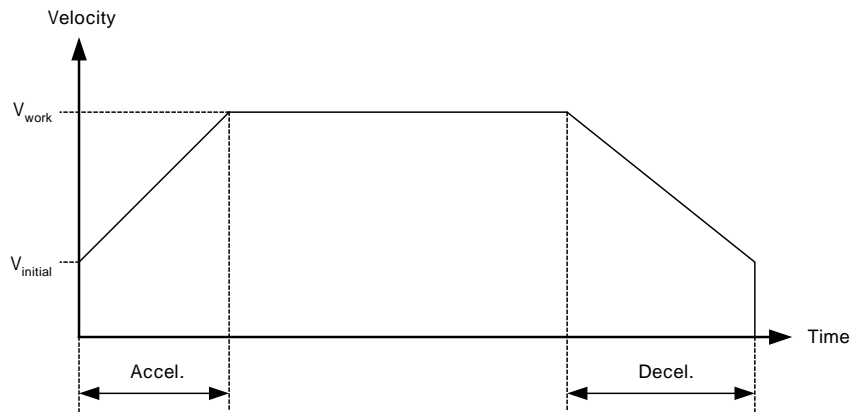
참 고

☐ 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

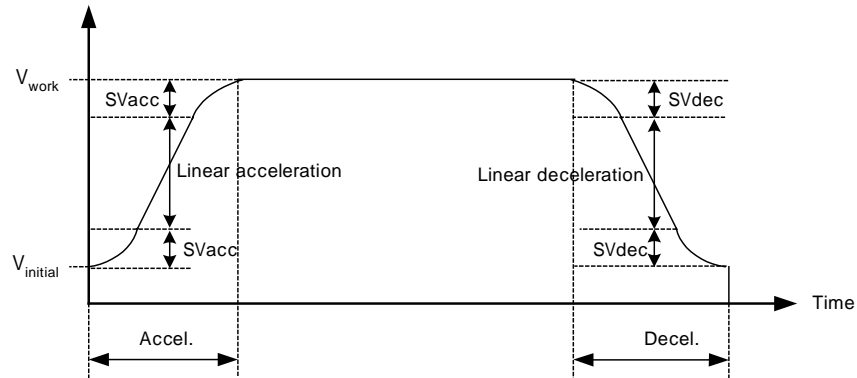
V_{work} : Working speed

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

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

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

다음의 예제는 Trapezoidal 속도 모드를 설정하는 예제입니다.

```
Call ComiCxAx1.LoadDivice
```

```

` Set trapezoidal speed mode
Call ComiCxAx1.McSetSpeedMode(X_AXIS, 1)
` Set speed as 1000
Call ComiCxAx1.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 ComiCxAx1.McSetAccel(X_AXIS, 2000, 2000)
` 현재의 위치로부터 5000 만큼 이동
Call ComiCxAx1.McMove(X_AXIS, 5000)

Call ComiCxAx1.UnloadDevice

```

■ 예제 2

다음의 예제는 S-Curve 속도 모드를 설정하는 예제입니다.

```

Call ComiCxAx1.LoadDevice

` Set S-Curve speed mode
Call ComiCxAx1.McSetSpeedMode(X_AXIS, 2)
` Set speed as 1000
Call ComiCxAx1.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 ComiCxAx1.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 ComiCxAx1.McSetScurve(X_AXIS, 0, 0)
` 현재의 위치로부터 5000 만큼 이동
Call ComiCxAx1.McMove(X_AXIS, 5000)

Call ComiCxAx1.UnloadDevice

```

■ McSetSpeed

메소드 원형

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

메소드 설명

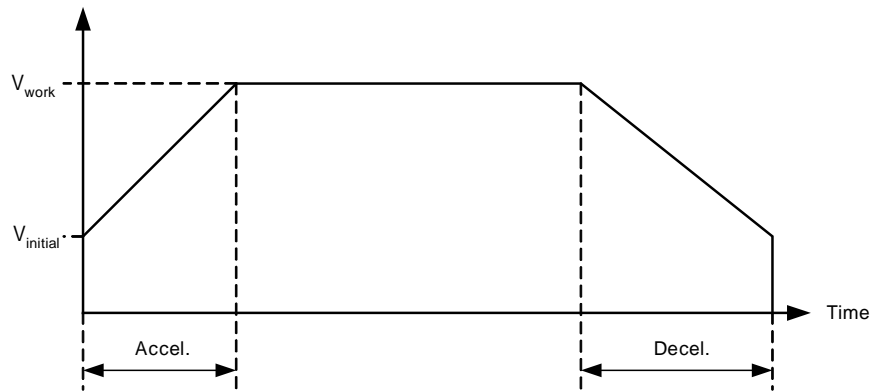
Motion 의 속도를 설정합니다. 단, 이 메소드는 Motion 에 바로 영향을 주는 것이 아니고 Move, MoveTo 등의 이송 메소드가 수행될 때 설정된 내용이 적용됩니다.

매개 변수

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

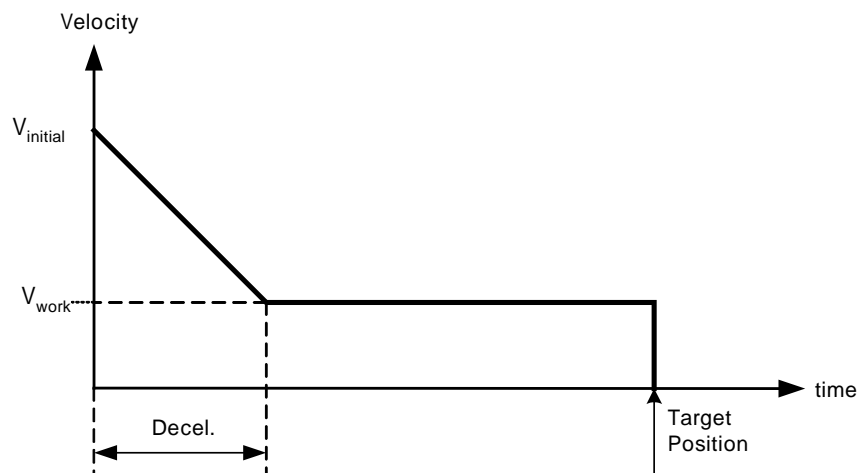
참 고

- 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 (fWorkSpeed)가 초기속도(fIniSpeed)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 ComiCxAx1.LoadDivice
Call ComiCxAx1.McSetSpeedMode(X_AXIS, 0)
Call ComiCxAx1.McSetSpeed(X_AXIS, 0, 1000)
Call ComiCxAx1.McSetStartVMove(X_AXIS, 1)

Call ComiCxAx1.McEmgStop(X_AXIS)

Call ComiCxAx1.UnloadDevice
```

■ 예제 2

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

```
Call ComiCxAx1.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 ComiCxAx1.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 ComiCxAx1.McSetUnitSpeed(0, 3600./60)
` Set trapezoidal speed mode
Call ComiCxAx1.McSetSpeedMode(0, 1)
` Set speed as 100 rpm
Call ComiCxAx1.McSetSpeed(0, 0, 100)
` 가속도와 감속도를 각각 200rpm/s로 설정한다. 이렇게 하면 작업속도가
` 100rpm이므로 가속 및 감속 시간은 각각 0.5 초 걸린다.
Call ComiCxAx1.McSetAccel(0, 200, 200)
` 모터를  $720^{\circ}$  회전한다. 실제로는  $720 \times 10$  펄스가 출력된다.
Call ComiCxAx1.McMove(0, 720)

Call ComiCxAx1.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 ComiCxAx1.LoadDivice

Call ComiCxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiCxAx1.McSetSpeed(X_AXIS, 0, 1000)
Call ComiCxAx1.McSetAccel(X_AXIS, 2000, 2000)
` V=1000, Acc=2000, Dec=2000 의 속도 패턴으로
` 4000 만큼 이동
Call ComiCxAx1.McMove(X_AXIS, 4000)

Call ComiCxAx1.UnloadDevice
```

■ 예제 2

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

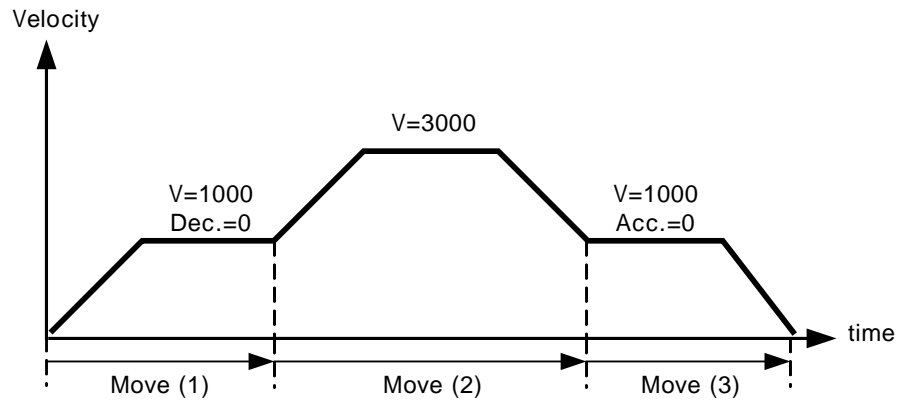
```
Call ComiCxAx1.LoadDivice

` Set 10 pulses for unit distance
` 이 예제에서는 1 회전에 필요한 펄스 수를 3600 펄스로
` 가정하고 단위 거리를  $1^{\circ}$ 로 설정한 것이다.
Call ComiCxAx1.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 ComiCxAx1.McSetUnitSpeed(0, 3600./60)
` Set trapezoidal speed mode
Call ComiCxAx1.McSetSpeedMode(0, 1)
` Set speed as 100 rpm
Call ComiCxAx1.McSetSpeed(0, 0, 100)
` 가속도와 감속도를 각각 200rpm/s로 설정한다. 이렇게 하면 작업속도가
` 100rpm이므로 가속 및 감속 시간은 각각 0.5 초 걸린다.
Call ComiCxAx1.McSetAccel(0, 200, 200)
` 모터를  $720^{\circ}$  회전한다. 실제로는  $720 \times 10$  펄스가 출력된다.
Call ComiCxAx1.McMove(0, 720)

Call ComiCxAx1.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.



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 ComiCxAx1.LoadDivice

Call ComiCxAx1.McBeginList '모션 리스트 등록 시작
' Set Trapezoidal Speed Mode
Call ComiCxAx1.McBeginList McSetSpeedMode(0, 1)
' Move (1)
Call ComiCxAx1.McSetSpeed(0, 0, 1000)
Call ComiCxAx1.McSetAccel(0, 2000, 0)
Call ComiCxAx1.McMoveTo(0, 3000)
' Move (2)
Call ComiCxAx1.McSetSpeed(0, 1000, 3000)
Call ComiCxAx1.McSetAccel(0, 2000, 2000)
Call ComiCxAx1.McMoveTo(0, 8000)
' Move (3)
Call ComiCxAx1.McSetSpeed(0, 0, 1000)
Call ComiCxAx1.McSetAccel(0, 0, 2000)
Call ComiCxAx1.McMoveTo(0, 3500)
Call ComiCxAx1.McEndList '모션 리스트 등록을 마칩

Call ComiCxAx1.McStartListMotion '리스트 모션 수행
' 리스트 모션이 모두 완료될 때까지 기다림
while not ComiCxAx1.McChekcListMotionDone
wend

Call ComiCxAx1.McBeginList UnloadDevice
    
```

■ McSetScurve

메소드 원형

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

메소드 설명

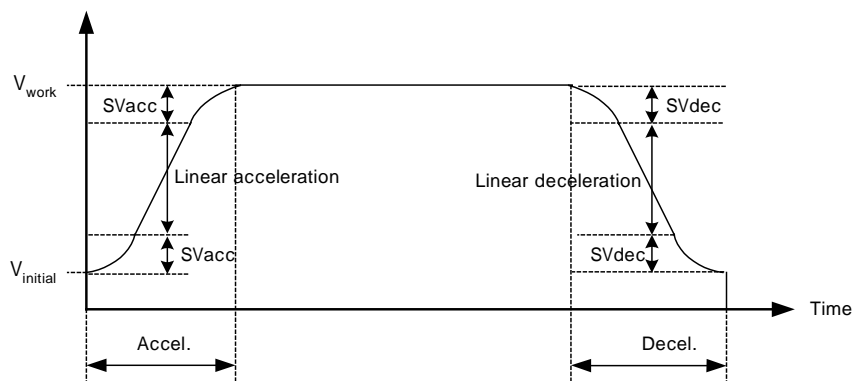
속도모드를 S-curve 속도 패턴으로 설정한 경우에 S-curve Section 의 범위를 속도단위로 설정합니다. 단, 이 메소드는 Motion 에 바로 영향을 주는 것이 아니고 Move, MoveTo 등의 이송 메소드가 수행될 때 설정된 내용이 적용됩니다.

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

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 ComiCxAx1.LoadDivice

속도 모드를 s-curve 모드로 설정
Call ComiCxAx1.McSetSpeedMode(X_AXIS, 2)

Call ComiCxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiCxAx1.McSetAccel(X_AXIS, 20000, 20000)
Call ComiCxAx1.McSetScurve(X_AXIS, 0, 0)

Call ComiCxAx1.McMove(X_AXIS, 50000)

Call ComiCxAx1.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 ComiCxAx1.LoadDivice  
  
`속도 모드를 S-curve 모드로 설정  
Call ComiCxAx1.McSetSpeedMode(X_AXIS, 2)  
  
Call ComiCxAx1.McSetSpeed(X_AXIS, 0, 10000)  
Call ComiCxAx1.McSetAccel(X_AXIS, 20000, 20000)  
Call ComiCxAx1.McSetScurve(X_AXIS, 2000, 2000)  
  
Call ComiCxAx1.McMove(X_AXIS, 50000)  
  
Call ComiCxAx1.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 McEmgStop 을 사용합니다.

예 제

```

Call ComiCxAx1.LoadDivice

Call ComiCxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiCxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiCxAx1.McSetAccel(X_AXIS, 20000, 20000)
' (+)방향으로 Velocity Move 수행
Call ComiCxAx1.McStartVMove(X_AXIS, 1)

' 어떤 특정 기간동안 Velocity Move 지속

' 감속 후 정지
Call ComiCxAx1.McStop(X_AXIS)

Call ComiCxAx1.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 1은 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 ComiCxAx1.LoadDivice

Call ComiCxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiCxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiCxAx1.McSetAccel(X_AXIS, 20000, 20000)

Call ComiCxAx1.McStartMove(X_AXIS, 30000)
While Not(ComiCxAx1.McDone (X_AXIS))
Wend

Call ComiCxAx1.McStartMove(X_AXIS, 40000)
' 모션이 완료될때까지 기다린다
while Not(ComiCxAx1.McDone (X_AXIS))
Wend

Call ComiCxAx1.UnloadDevice
```

■ 예제 2

본 예제는 McSetUnitDistance()메쏘드와 McSetUnitSpeed() 메쏘드를 사용하여 논리 거리(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 ComiCxAx1.LoadDivice

` Set 100 pulses for unit distance
` 이 예제에서는 1mm 이동에 필요한 펄스수를 100 펄스로
` 가정하고 단위 거리를 1mm 로 설정한 것이다.
Call ComiCxAx1.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 ComiCxAx1.McSetUnitSpeed(X_AXIS, 10000./60)

Call ComiCxAx1.McSetSpeedMode(X_AXIS, 1)
` Set speed as 60 rpm
Call ComiCxAx1.McSetSpeed(X_AXIS, 0, 60)
Call ComiCxAx1.McSetAccel(X_AXIS, 60, 60)
` 60 rpm 의 속도로 100mm 이동
Call ComiCxAx1.McStartMove(X_AXIS, 100)
While Not(ComiCxAx1.McDone (X_AXIS))
Wend

Call ComiCxAx1.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 ComiCxAx1.LoadDevice

Call ComiCxAx1.McSetBlockingMode (FALSE)

Call ComiCxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiCxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiCxAx1.McSetAccel(X_AXIS, 20000, 20000)
Call ComiCxAx1.McMove(X_AXIS, 30000)
Call ComiCxAx1.McMove(X_AXIS, 40000)

Call ComiCxAx1.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 ComiCxAx1.LoadDivice

` Set 100 pulses for unit distance
` 이 예제에서는 1mm 이동에 필요한 펄스수를 100 펄스로
` 가정하고 단위 거리를 1mm 로 설정한 것이다.
Call ComiCxAx1.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 ComiCxAx1.McSetUnitSpeed(X_AXIS, 10000./60)

Call ComiCxAx1.McSetSpeedMode(X_AXIS, 1)
` Set speed as 60 rpm
Call ComiCxAx1.McSetSpeed(X_AXIS, 0, 60)
Call ComiCxAx1.McSetAccel(X_AXIS, 60, 60)
` 60 rpm 의 속도로 100mm 이동
Call ComiCxAx1.McMove(X_AXIS, 100)

Call ComiCxAx1.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 ComiCxAx1.LoadDivice

' Command Position 의 현재 좌표를 0 으로 초기화한다.
Call ComiCxAx1.McSetPosition_C (X_AXIS, 0)

Call ComiCxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiCxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiCxAx1.McSetAccel(X_AXIS, 20000, 20000)

' 좌표 0 에서 30000 으로 이동
Call ComiCxAx1.McStartMoveTo(X_AXIS, 30000)
While Not(ComiCxAx1.McDone (X_AXIS))
Wend

' 좌표 30000 에서 70000 으로 이동
Call ComiCxAx1.McStartMoveTo(X_AXIS, 70000)
While Not(ComiCxAx1.McDone(X_AXIS))
Wend

Call ComiCxAx1.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 ComiCxAx1.LoadDivice

' Set 100 pulses for unit distance
' 이 예제에서는 1mm 이동에 필요한 펄스수를 100 펄스로
' 가정하고 단위 거리를 1mm 로 설정한 것이다.
Call ComiCxAx1.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 ComiCxAx1.McSetUnitSpeed(X_AXIS, 10000./60)

' Command Position 의 현재 좌표를 0 으로 초기화한다.
Call ComiCxAx1.McSetPosition_C (X_AXIS, 0)

Call ComiCxAx1.McSetSpeedMode(X_AXIS, 1)
' Set speed as 60 rpm
Call ComiCxAx1.McSetSpeed(X_AXIS, 0, 60)
Call ComiCxAx1.McSetAccel(X_AXIS, 60, 60)
' 60 rpm 의 속도로 좌표 100(mm)으로 이동
Call ComiCxAx1.McStartMoveTo(X_AXIS, 100)
While Not(ComiCxAx1.McDone (X_AXIS))
Wend

Call ComiCxAx1.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 ComiCxAx1.LoadDivice

' Command Position 의 현재 좌표를 0 으로 초기화한다.
Call ComiCxAx1.McSetPosition_C (X_AXIS, 0)

Call ComiCxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiCxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiCxAx1.McSetAccel(X_AXIS, 20000, 20000)

' 좌표 0 에서 30000 으로 이동
Call ComiCxAx1.McMoveTo(X_AXIS, 30000)
' 좌표 30000 에서 70000 으로 이동
Call ComiCxAx1.McMoveTo(X_AXIS, 70000)

Call ComiCxAx1.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 ComiCxAx1.LoadDivice

` Set 100 pulses for unit distance
` 이 예제에서는 1mm 이동에 필요한 펄스수를 100 펄스로
` 가정하고 단위 거리를 1mm 로 설정한 것이다.
Call ComiCxAx1.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 ComiCxAx1.McSetUnitSpeed(X_AXIS, 10000./60)

` Command Position 의 현재 좌표를 0 으로 초기화한다.
Call ComiCxAx1.McSetPosition_C (X_AXIS, 0)

Call ComiCxAx1.McSetSpeedMode(X_AXIS, 1)
` Set speed as 60 rpm
Call ComiCxAx1.McSetSpeed(X_AXIS, 0, 60)
Call ComiCxAx1.McSetAccel(X_AXIS, 60, 60)
` 60 rpm 의 속도로 좌표 100(mm)으로 이동
Call ComiCxAx1.McMoveTo(X_AXIS, 100)

Call ComiCxAx1.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 ComiCxAx1.LoadDivice

Call ComiCxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiCxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiCxAx1.McSetAccel(X_AXIS, 20000, 20000)
' (+)방향으로 Velocity Move 수행
Call ComiCxAx1.McStartVMove(X_AXIS, 1)

' 일정 동작동안 Velocity Move 지속

' 감속 후 정지
Call ComiCxAx1.McStop(X_AXIS)

Call ComiCxAx1.UnloadDevice
```

■ 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 ComiCxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiCxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiCxAx1.McSetAccel(X_AXIS, 20000, 20000)
· (+)방향으로 Velocity Move 수행
Call ComiCxAx1.McStartVMove(X_AXIS, 1)
```

· 일정 기간동안 Velocity Move 지속

· 감속없이 즉시 정지

```
Call ComiCxAx1.McEmgStop(X_AXIS)

Call ComiCxAx1.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 ComiCxAx1.LoadDivice

Call ComiCxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiCxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiCxAx1.McSetAccel(X_AXIS, 20000, 20000)

Call ComiCxAx1.McStartMove(X_AXIS, 30000)
While Not(ComiCxAx1.McDone(X_AXIS))
Wend

Call ComiCxAx1.McStartMove(X_AXIS, 40000)
' 모션이 완료될때까지 기다린다
while Not (ComiCxAx1.McDone(X_AXIS))
Wend

Call ComiCxAx1.UnloadDevice
    
```


2.5.3 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.	
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.	
Sub McMoveAll (ByVal NumAxis As Long, ByRef AxisList[] As Long, ByRef DistList[] As Double) 여러 개의 축에 대하여 현재의 위치에서 지정한 거리만큼 이동을 수행합니다.	
Sub McStartMoveToAll (ByVal NumAxis As Long, ByRef AxisList[] As Long, ByRef PosList[] As Long) 여러 개의 축에 대하여 지정한 절대좌표로의 이동을 동시에 시작합니다.	
Sub McMoveToAll (ByVal NumAxis As Long, ByRef AxisList[] As Long, ByRef PosList[] As Long) 여러 개의 축에 대하여 지정한 절대좌표로의 이동을 수행합니다.	
Sub McStopAll (ByVal NumAxis As Long, ByRef AxisList[] As Long) 여러 개의 축에 대한 모션을 동시에 감속 후 정지합니다.	
Sub McEngStopAll (ByVal NumAxis As Long, ByRef AxisList[] As Long) 여러 개의 축에 대한 모션을 동시에 감속없이 즉시 정지합니다.	
Function McAllDone(ByVal NumAxis As Long, ByRef AxisList[] As Long)As Boolean 여러 개의 축에 대하여 지정한 모든 축의 모션이 완료됐는지를 체크합니다.	

■ 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 지정한 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 ComiCxAx1.LoadDevice

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

Call ComiCxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiCxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiCxAx1.McSetAccel(X_AXIS, 20000, 20000)

Call ComiCxAx1.McSetSpeedMode(Y_AXIS, 1)
Call ComiCxAx1.McSetSpeed(Y_AXIS, 0, 10000)
Call ComiCxAx1.McSetAccel(Y_AXIS, 20000, 20000)

Call ComiCxAx1.McStartVMoveAll(2, AxisList(0), DirList(0))
' 일정 시간동안 Velocity Move 지속
' 감속 후 정지
Call ComiCxAx1.McStopAll(2, AxisList(0))
```

Call ComiCxAx1.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 ComiCxAx1.LoadDivice

Call ComiCxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiCxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiCxAx1.McSetAccel(X_AXIS, 20000, 20000)
```

```
Call ComiCxAx1.McSetSpeedMode(Y_AXIS, 1)
Call ComiCxAx1.McSetSpeed(Y_AXIS, 0, 20000)
Call ComiCxAx1.McSetAccel(Y_AXIS, 40000, 40000)

Call ComiCxAx1.McStartMoveAll (2, AxisList(0), DistList(0))
While Not (ComiCxAx1.McAllDone (2, AxisList(0)))
Wend

Call ComiCxAx1.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 메소드들에 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, 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 ComiCxAx1.LoadDivice

Call ComiCxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiCxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiCxAx1.McSetAccel(X_AXIS, 20000, 20000)
```

```
Call ComiCxAx1.McSetSpeedMode(Y_AXIS, 1)
Call ComiCxAx1.McSetSpeed(Y_AXIS, 0, 20000)
Call ComiCxAx1.McSetAccel(Y_AXIS, 40000, 40000)

Call ComiCxAx1.McMoveAll (2, AxisList(0), DistList(0))

Call ComiCxAx1.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
- ▶ **AxisList** : 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 1 은 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 ComiCxAx1.LoadDevice

Call ComiCxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiCxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiCxAx1.McSetAccel(X_AXIS, 20000, 20000)

Call ComiCxAx1.McSetSpeedMode(Y_AXIS, 1)
Call ComiCxAx1.McSetSpeed(Y_AXIS, 0, 20000)
Call ComiCxAx1.McSetAccel(Y_AXIS, 40000, 40000)
```



```
Call ComiCxAx1.McStartMoveToAll (2, AxisList(0), PosList(0))
While Not (ComiCxAx1.McAllDone (2, AxisList(0)))
Wend

Call ComiCxAx1.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
- ▶ **AxisList** : 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 1 은 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 ComiCxAx1.LoadDivice

Call ComiCxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiCxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiCxAx1.McSetAccel(X_AXIS, 20000, 20000)

Call ComiCxAx1.McSetSpeedMode(Y_AXIS, 1)
```

```
Call ComiCxAx1.McSetSpeed(Y_AXIS, 0, 20000)
Call ComiCxAx1.McSetAccel(Y_AXIS, 40000, 40000)

Call ComiCxAx1.McMoveToAll (2, AxisList(0), PosList(0))

Call ComiCxAx1.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 ComiCxAx1.LoadDivice

Call ComiCxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiCxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiCxAx1.McSetAccel(X_AXIS, 20000, 20000)

Call ComiCxAx1.McSetSpeedMode(Y_AXIS, 1)
Call ComiCxAx1.McSetSpeed(Y_AXIS, 0, 10000)
Call ComiCxAx1.McSetAccel(Y_AXIS, 20000, 20000)

Call ComiCxAx1.McStartVMoveAll (2, AxisList(0), DirList(0))

' 일정 시간동안 Velocity Move 지속

' 감속 후 정지
Call ComiCxAx1.McStopAll(2, AxisList(0))

Call ComiCxAx1.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 ComiCxAx1.LoadDivice

Call ComiCxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiCxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiCxAx1.McSetAccel(X_AXIS, 20000, 20000)

Call ComiCxAx1.McSetSpeedMode(Y_AXIS, 1)
Call ComiCxAx1.McSetSpeed(Y_AXIS, 0, 10000)
Call ComiCxAx1.McSetAccel(Y_AXIS, 20000, 20000)

Call ComiCxAx1.McStartVMoveAll (2, AxisList(0), DirList(0))

' 일정 시간동안 Velocity Move 지속

' 감속 후 정지
Call ComiCxAx1.McEmgStopAll(2, AxisList(0))

Call ComiCxAx1.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 ComiCxAx1.LoadDivice

Call ComiCxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiCxAx1.McSetSpeed(X_AXIS, 0, 10000)
Call ComiCxAx1.McSetAccel(X_AXIS, 20000, 20000)

Call ComiCxAx1.McSetSpeedMode(Y_AXIS, 1)
Call ComiCxAx1.McSetSpeed(Y_AXIS, 0, 20000)
Call ComiCxAx1.McSetAccel(Y_AXIS, 40000, 40000)

Call ComiCxAx1.McStartMoveAll (2, AxisList(0), DistList(0))
While Not (ComiCxAx1.McAllDone (2, AxisList(0)))
Wend

Call ComiCxAx1.UnloadDevice
```

2.5.4 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	
Sub McSetSpeedModeMx (ByVal MapIndex As Long, ByVal ModeIndex As Long) Coordinated Motion 의 속도 모드를 설정합니다.	
Sub McSetSpeedMx (ByVal MapIndex As Long, ByVal Speed As Long, ByVal Accel As Long) Coordinated Motion 의 속도 및 가속도를 설정합니다	
Sub McStartLine(ByVal MapIndex As Long, ByRef DistList[] As Double) 현재 위치로부터의 상대 좌표로의 직선 보간 이동을 수행합니다.	
Sub McLine (ByVal MapIndex As Long, ByRef DistList[] As Double) 현재 위치로부터의 상대 좌표로의 직선 보간 이동을 수행합니다	
Sub McStartLineTo(ByVal MapIndex As Long, ByRef PosList[] As Double) 절대 좌표로의 직선 보간 이동을 수행합니다.	
Sub McLineTo(ByVal MapIndex As Long, ByRef PosList[] As Double) 절대 좌표로의 직선 보간 이동을 수행합니다.	
Sub McStartArcA(ByVal MapIndex As Long, ByVal XcentOffset As Double, ByVal YcentOffset As Double, ByVal EndAngle As Double) 상대좌표를 파라미터로 하여 원호보간 이동을 수행합니다.	
Sub McArcA(ByVal MapIndex As Long, ByVal XcentOffset As Double, ByVal YcentOffset As Double, ByVal EndAngle As Double) 상대좌표를 파라미터로 하여 원호보간 이동을 수행합니다.	
Sub McStartArcP(ByVal MapIndex As Long, ByVal XCentOffset As Double, ByVal	

<p>YCentOffset As Double, ByVal XEndPointDist As Double, ByVal YEndPointDist 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 XendPointDist As Double, ByVal YendPointDist As Double, ByVal Dir As Long)</p> <p>상대좌표를 파라미터로 하여 원호보간 이동을 수행합니다.</p>	
<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>	
<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>	
<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>	
<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>	
<p>Function McMxDone(ByVal MapIndex As Long) As Boolean</p> <p>지정한 축그룹(MapIndex)의 Coordinated Motion 이 완료됐는지를 체크합니다.</p>	

■ 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 ComiCxAx1.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 ComiCxAx1.McMapAxes(MAP0, X_MASK or Y_MASK)
` Set speed mode as Trapezoidal
Call ComiCxAx1.McSetSpeedModeMx(MAP0, 1)
` Set speed & accel => V=5000, Acc=8000
Call ComiCxAx1.McSetSpeedMx(MAP0, 5000, 8000)
` Move to relative coord. (10000, 20000)
DistList(0)=10000
DistList(1)=20000

Call ComiCxAx1.McLine(MAP0, DistList(0))

Call ComiCxAx1.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 ComiCxAx1.LoadDevice
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 ComiCxAx1.McMapAxes(MAP0, X_MASK or Y_MASK)
` Map Z&U axis to MAP1
Call ComiCxAx1.McMapAxes(MAP1, Z_MASK or U_MASK)

` Set speed mode of MAP0 as Trapezoidal
Call ComiCxAx1.McSetSpeedModeMx(MAP0, 1)
` Set speed & accel of MAP0 => V=5000, Acc=8000
Call ComiCxAx1.McSetSpeedMx(MAP0, 5000, 8000)
` Move to relative coord. (10000, 20000)

` Set speed mode of MAP1 as Trapezoidal
Call ComiCxAx1.McSetSpeedModeMx(MAP1, 1)
` Set speed & accel of MAP1 => V=2000, Acc=5000
Call ComiCxAx1.McSetSpeedMx(MAP1, 2000, 5000)

` Move to relative coord of (X, Y) => (10000, 20000)
Call ComiCxAx1.McStartLine(MAP0, DistList1(0))
` Move to relative coord of (Z, U) => (8000, 5000)
Call ComiCxAx1.McStartLine(MAP1, DistList2(0))

Call ComiCxAx1.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 ComiCxAx1.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 ComiCxAx1.McMapAxes(MAP0, X_MASK or Y_MASK)
' Set speed mode as Trapezoidal
Call ComiCxAx1.McSetSpeedModeMx(MAP0, 1)
' Set speed & accel => V=5000, Acc=8000
Call ComiCxAx1.McSetSpeedMx(MAP0, 5000, 8000)
' Move to relative coord. (10000, 20000)
```

```
Call ComiCxAx1.McLine(MAP0, DistList(0))
```

```
Call ComiCxAx1.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.

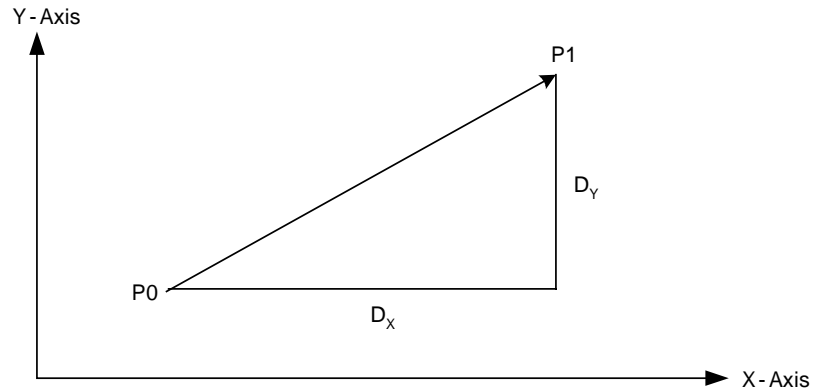
매개 변수

- ▶ **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 ComiCxAx1.McMapAxes (MAP_IDX, (X_AXIS or Y_AXIS))
' Trapezoidal 속도모드로 설정
Call ComiCxAx1.McSetSpeedModeMx(MAP_IDX, 1)
' 벡터속도 1000 PPS, 벡터가속도 2000 PPS/sec 로 설정
Call ComiCxAx1.McSetSpeedMx(MAP_IDX, 1000, 2000)

Dim DistList(1)

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

Call ComiCxAx1.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 계산해보면

$$V_x = \frac{D_x \times V}{\sqrt{D_x^2 + D_y^2}} = \frac{3000 \times 1000}{\sqrt{3000^2 + 4000^2}} = 600$$

$$V_y = \frac{D_y \times V}{\sqrt{D_x^2 + D_y^2}} = \frac{4000 \times 1000}{\sqrt{3000^2 + 4000^2}} = 800$$

과 같이 됩니다.

예 제

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

```
Call ComiCxAx1.LoadDivice

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

```
Const MAP0 = 0

` Map X&Y axis to MAP0
Call ComiCxAx1.McMapAxes(MAP0, X_MASK or Y_MASK)
` Set speed mode as Trapezoidal
Call ComiCxAx1.McSetSpeedModeMx(MAP0, 1)
` Set speed & accel => V=5000, Acc=8000
Call ComiCxAx1.McSetSpeedMx(MAP0, 5000, 8000)
` Move to relative coord. (10000, 20000)
Call ComiCxAx1.McLine(MAP0, DistList(0))

Call ComiCxAx1.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 1은 1 Pulse 출력을 의미합니다.

참 고

□ McLine 메소드가 모션이 완료될 때까지 Return 되지 않는데 반하여, 이 메소드는 지정된 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 ComiCxAx1.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 ComiCxAx1.McMapAxes(MAP0, X_MASK or Y_MASK)
' Set speed mode as Trapezoidal
Call ComiCxAx1.McSetSpeedModeMx(MAP0, 1)
' Set speed & accel => V=5000, Acc=8000
Call ComiCxAx1.McSetSpeedMx(MAP0, 5000, 8000)
```

```
` Move to relative coord. (10000, 20000)
Call ComiCxAx1.McStartLine(MAP0, DistList(0))
` Coordinated Motion 이 완료될때까지 기다린다.
While Not ( ComiCxAx1.McMxDone(MAP0))
Wend

Call ComiCxAx1.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 1은 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 ComiCxAx1.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 ComiCxAx1.McMapAxes(MAP0, X_MASK or Y_MASK)
` Set speed mode as Trapezoidal
Call ComiCxAx1.McSetSpeedModeMx(MAP0, 1)
` Set speed & accel => V=5000, Acc=8000
Call ComiCxAx1.McSetSpeedMx(MAP0, 5000, 8000)
` Move to relative coord. (10000, 20000)
Call ComiCxAx1.McLine(MAP0, DistList(0))

Call ComiCxAx1.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 ComiCxAx1.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

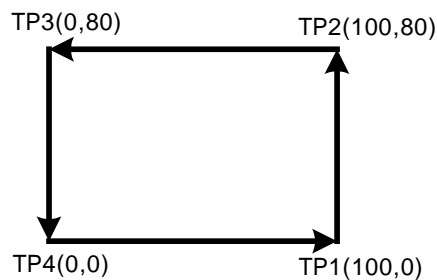
' Map X&Y axis to MAP0
Call ComiCxAx1.McMapAxes(MAP0, X_MASK or Y_MASK)
' Set speed mode as Trapezoidal
Call ComiCxAx1.McSetSpeedModeMx(MAP0, 1)
' Set speed & accel => V=5000, Acc=8000
Call ComiCxAx1.McSetSpeedMx(MAP0, 5000, 8000)
' Move to absolute coord. (10000, 20000)
```

```
Call ComiCxAx1.McStartLineTo(MAP0, PosList(0))
` Coordinated Motion 이 완료될때까지 기다린다.
While Not (ComiCxAx1.McMxDone(MAP0))
Wend

Call ComiCxAx1.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 ComiCxAx1.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 ComiCxAx1.McSetUnitDistance(X_AXIS, 10)
Call ComiCxAx1.McSetUnitDistance(Y_AXIS, 10)
Call ComiCxAx1.McSetUnitSpeed(X_AXIS, 3600./60)
Call ComiCxAx1.McSetUnitSpeed(Y_AXIS, 3600./60)

` X&Y 축의 Command Position 의 현재 좌표를 0 으로 초기화한다.
Call ComiCxAx1.McSetPosition_C (X_AXIS, 0)
Call ComiCxAx1.McSetPosition_C (Y_AXIS, 0)

` Map X&Y axis to MAP0
Call ComiCxAx1.McMapAxes(MAP0, X_MASK or Y_MASK)
` Set speed mode as Trapezoidal
Call ComiCxAx1.McSetSpeedModeMx(MAP0, 1)
` Set speed & accel => V=60(RPM), Acc=100(RPM/SEC)
Call ComiCxAx1.McSetSpeedMx(MAP0, 60, 100)

` Move to (100,0)
```

```

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

Call ComiCxAx1.McStartLineTo(MAP0, PosList(0))
While Not ( ComiCxAx1.McMxDone(MAP0))
Wend

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

Call ComiCxAx1.McStartLineTo(MAP0, PosList(0))
While Not ( ComiCxAx1.McMxDone(MAP0))
Wend

` Move to (0,80)
PosList(0) = 0
PosList(1) = 80
Call ComiCxAx1.McStartLineTo(MAP0, PosList(0))
While Not (ComiCxAx1.McMxDone(MAP0))
Wend

` Move to (0,0)
PosList(0) = 0
PosList(1) = 0
Call ComiCxAx1.McStartLineTo(MAP0, PosList(0))
While Not ( ComiCxAx1.McMxDone(MAP0))
Wend

Call ComiCxAx1.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 ComiCxAx1.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 ComiCxAx1.McMapAxes(MAP0, X_MASK or Y_MASK)
```



```

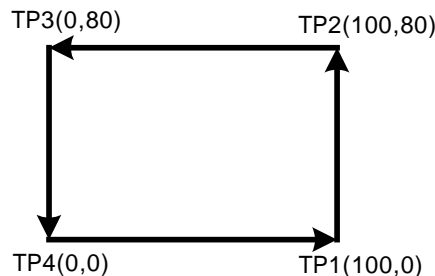
` Set speed mode as Trapezoidal
Call ComiCxAx1.McSetSpeedModeMx(MAP0, 1)
` Set speed & accel => V=5000, Acc=8000
Call ComiCxAx1.McSetSpeedMx(MAP0, 5000, 8000)
` Move to absolute coord. (10000, 20000)
Call ComiCxAx1.McStartLineTo(MAP0, PosList(0))
` Coordinated Motion 이 완료될때까지 기다린다.
While Not ( ComiCxAx1.McMxDone(MAP0))
Wend

Call ComiCxAx1.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 ComiCxAx1.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 ComiCxAx1.McSetUnitDistance(X_AXIS, 10)
Call ComiCxAx1.McSetUnitDistance(Y_AXIS, 10)
Call ComiCxAx1.McSetUnitSpeed(X_AXIS, 3600./60)
Call ComiCxAx1.McSetUnitSpeed(Y_AXIS, 3600./60)

` X&Y 축의 Command Position 의 현재 좌표를 0 으로 초기화한다.
Call ComiCxAx1.McSetPosition_C (X_AXIS, 0)
Call ComiCxAx1.McSetPosition_C (Y_AXIS, 0)

` Map X&Y axis to MAP0
Call ComiCxAx1.McMapAxes(MAP0, X_MASK or Y_MASK)
` Set speed mode as Trapezoidal
Call ComiCxAx1.McSetSpeedModeMx(MAP0, 1)
` Set speed & accel => V=60(RPM), Acc=100(RPM/SEC)
Call ComiCxAx1.McSetSpeedMx(MAP0, 60, 100)

```

```
Move to (100,0)
PosList(0) = 100
PosList(1) = 0
Call ComiCxAx1.McLineTo(MAP0, PosList(0))

` Move to (100,80)
PosList(0) = 100
PosList(1) = 80
Call ComiCxAx1.McLineTo(MAP0, PosList(0))

` Move to (0,80)
PosList(0) = 0
PosList(1) = 80
Call ComiCxAx1.McLineTo(MAP0, PosList(0))

` Move to (0,0)
PosList(0) = 0
PosList(1) = 0
Call ComiCxAx1.McLineTo(MAP0, PosList(0))

Call ComiCxAx1.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 Point 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 축상의 거리
- ▶ **YCentOffset** : 현재 위치(시작 위치)로부터 원의 중심까지 Y 축상의 거리
- ▶ **EndAngle** : 원호보간 이동을 완료할 목표지점의 현재 위치에 대한 각도값을 Degree(°)값으로 지정합니다. 각도의 부호가 (+)이면 반시계방향, (-)이면 시계방향으로의 이동을 의미합니다.

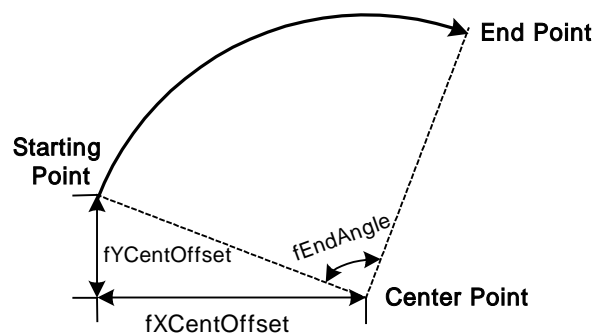
참 고

□ 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 Point 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 축상의 거리
- ▶ **YCentOffset** : 현재 위치(시작 위치)로부터 원의 중심까지 Y 축상의 거리
- ▶ **EndAngle** : 원호보간 이동을 완료할 목표지점의 현재 위치에 대한 각도값을 Degree(°)값으로 지정합니다. 각도의 부호가 (+)이면 반시계방향, (-)이면 시계방향으로의 이동을 의미합니다.

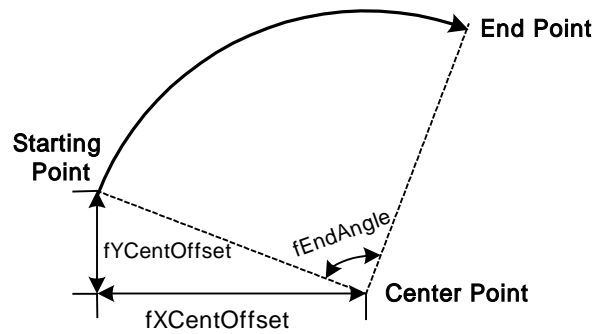
참 고

□ 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.

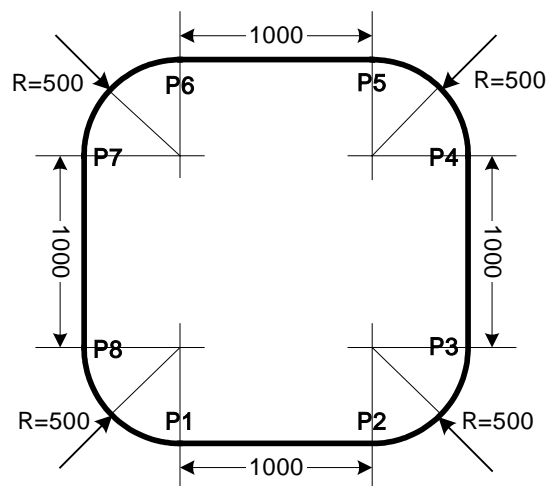
□ McArcA 메소드를 사용하여 원호보간 이동을 수행할 때 각 파라미터의 의미는 [그림 #] 과 같습니다.



[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 ComiCxAx1.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 ComiCxAx1.McMapAxes(MAP0, X_MASK or Y_MASK)
` Set speed mode as Trapezoidal
Call ComiCxAx1.McSetSpeedModeMx(MAP0, 1)
` Set speed & accel => V=500, Acc=500
Call ComiCxAx1.McSetSpeedMx(MAP0, 500, 500)
```

```

` Move from P1 to P2
DistList(0) = 1000
DistList(1) = 0
Call ComiCxAx1.McLine(MAP0, DistList(0))

` Move from P2 to P3
Call ComiCxAx1.McArcA(MAP0, 0, 500, 90)
` Move from P3 to P4
DistList(0) = 0
DistList(1) = 1000
Call ComiCxAx1.McLine(MAP0, DistList(0))
` Move from P4 to P5
Call ComiCxAx1.McArcA(MAP0, -500, 0, 90)
` Move from P5 to P6
DistList(0) = -1000
DistList(1) = 0
Call ComiCxAx1.McLine(MAP0, DistList(0))
` Move from P6 to P7
Call ComiCxAx1.McArcA(MAP0, 0, -500, 90)
` Move from P7 to P8

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

Call ComiCxAx1.McLine(MAP0, DistList(0))

` Move from P8 to P1
Call ComiCxAx1.McArcA(MAP0, 500, 0, 90)

Call ComiCxAx1.UnloadDevice

```

■ McStartArcP

메소드 원형

```
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)
```

메소드 설명

이 메소드는 상대좌표를 파라미터로 하여 원호보간 이동을 수행합니다. 이 메소드는 End Point 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 축상의 거리
- ▶ *YCentOffset* : 현재 위치(시작 위치)로부터 원의 중심까지 Y 축상의 거리
- ▶ *XEndPointDist* : 원호보간 이동을 완료할 목표지점의 현재 위치로부터 X-축상 거리값.
- ▶ *YEndPointDist* : 원호보간 이동을 완료할 목표지점의 현재 위치로부터 Y-축상 거리값.

참 고

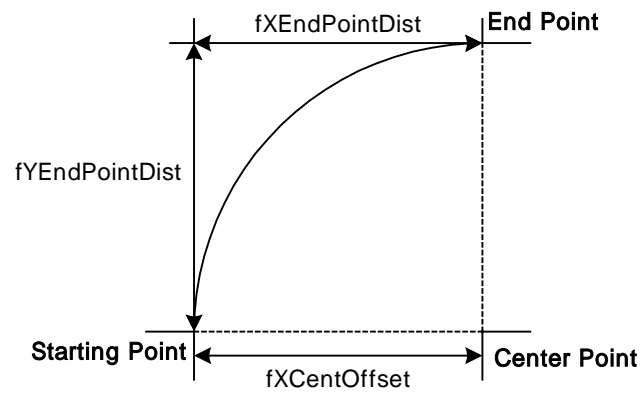
□ 원호보간 이동은 두 축에 대해서만 적용가능합니다. 따라서 본 단락에서는 맵핑된 두 축을 X, Y 축으로 간주하여 설명합니다. 여기서 X 축이라 함은 맵핑된 두 축 중에서 채널번호(X,Y,Z,U 순)가 낮은 축을 의미하며 Y 축은 채널번호가 높은 축을 의미합니다. 예를 들어 Z 축과 U 축이 맵핑된 두 축이라면 Z 축이 X 축에 해당하며 U 축이 Y 축에 해당합니다.

□ 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.

□ XEndPointDist 값과 YEndPointDist 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 XendPointDist As Double, ByVal YendPointDist As Double, ByVal Dir  
As Long)
```

메소드 설명

이 메소드는 상대좌표를 파라미터로 하여 원호보간 이동을 수행합니다. 이 메소드는 End Point 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 축상의 거리.
- ▶ **YCentOffset** : 현재 위치(시작 위치)로부터 원의 중심까지 Y 축상의 거리
- ▶ **XEndPointDist** : 원호보간 이동을 완료할 목표지점의 현재 위치로부터 X-축상 거리값.
- ▶ **YEndPointDist** : 원호보간 이동을 완료할 목표지점의 현재 위치로부터 Y-축상 거리값.

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.

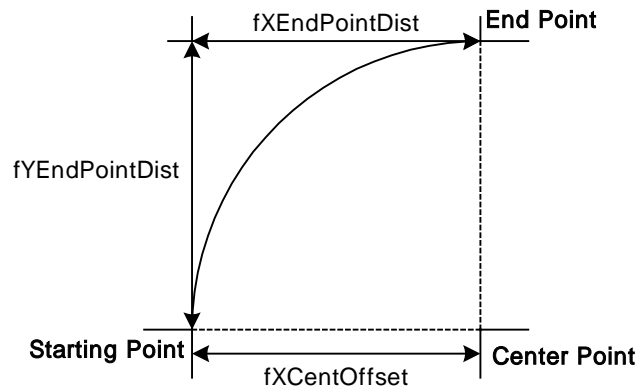
□ 이 메소드는 원호 보간 이동이 완료되기 전까지 Return 되지 않고 루프를 돌게 됩니다. 루프를 도는 동안 윈도우 이벤트나 메시지가 처리될 수 있도록 하려면 이 메소드를 수행하기 이전에 McSetBlockingMode 메소드를 사용하여 Blocking 이 일어나지 않도록 설정하여야 합니다.

□ XEndPointDist 값과 YEndPointDist 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 [그

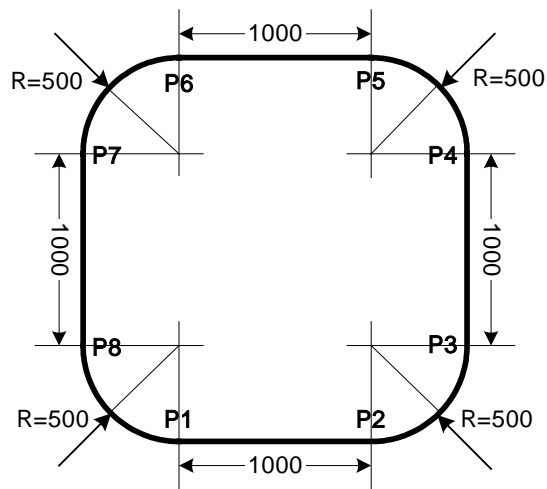
림 #]과 같습니다.



[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 ComiCxAx1.LoadDivice

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

Const MAP0 = 0

Dim DistList(2)

` Map X&Y axis to MAP0
Call ComiCxAx1.McMapAxes(MAP0, X_MASK or Y_MASK)
` Set speed mode as Trapezoidal
Call ComiCxAx1.McSetSpeedModeMx(MAP0, 1)
    
```

```
` Set speed & accel => V=500, Acc=500
Call ComiCxAx1.McSetSpeedMx(MAP0, 500, 500)

` Move from P1 to P2
DistList(0) = 1000
DistList(1) = 0
Call ComiCxAx1.McLine(MAP0, DistList(0))
` Move from P2 to P3
Call ComiCxAx1.McArcP(MAP0, 0, 500, 500, 500)

` Move from P3 to P4
DistList(0) = 0
DistList(1) = 1000
Call ComiCxAx1.McLine(MAP0, DistList(0))
` Move from P4 to P5
Call ComiCxAx1.McArcP(MAP0, -500, 0, -500, 500)
` Move from P5 to P6
DistList(0) = -1000
DistList(1) = 0
Call ComiCxAx1.McLine(MAP0, DistList(0))
` Move from P6 to P7
Call ComiCxAx1.McArcP(MAP0, 0, -500, -500, -500)
` Move from P7 to P8
DistList(0) = 0
DistList(1) = -1000
Call ComiCxAx1.McLine(MAP0, DistList(0))
` Move from P8 to P1
Call ComiCxAx1.McArcP(MAP0, 500, 0, 500, -500)

Call ComiCxAx1.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.

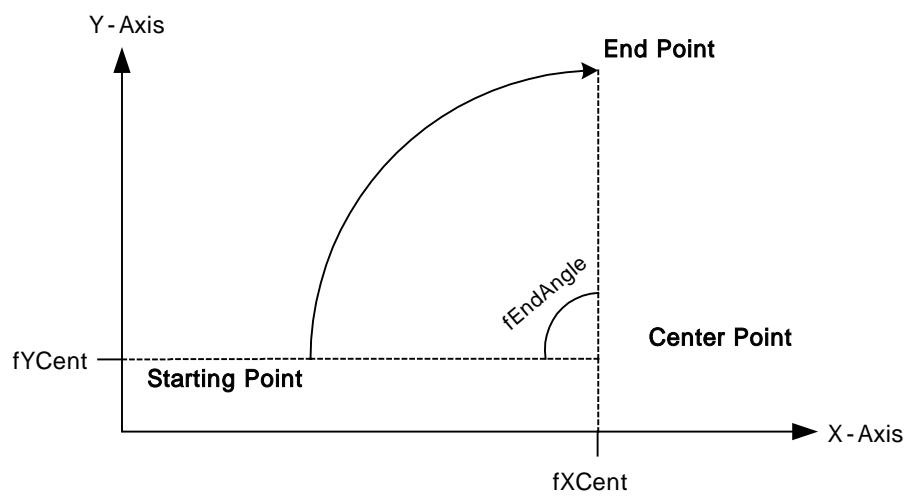
참 고

□ 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.



[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.

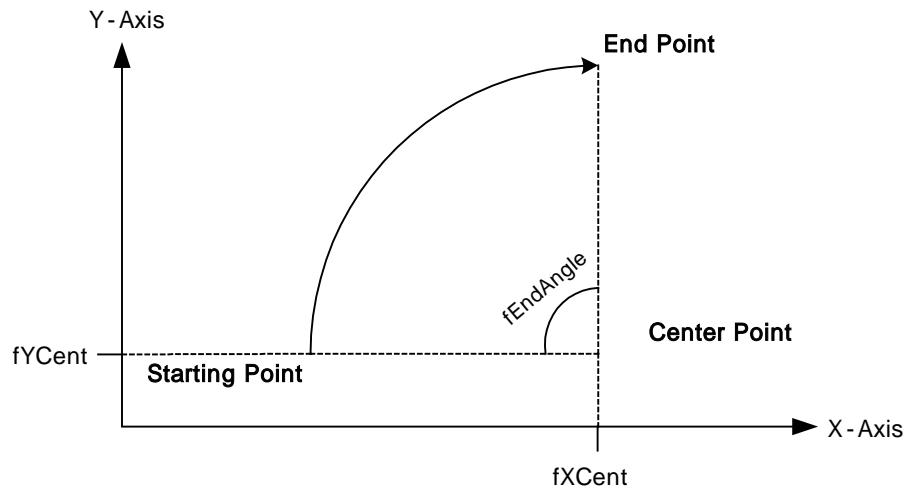
참 고

□ 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.

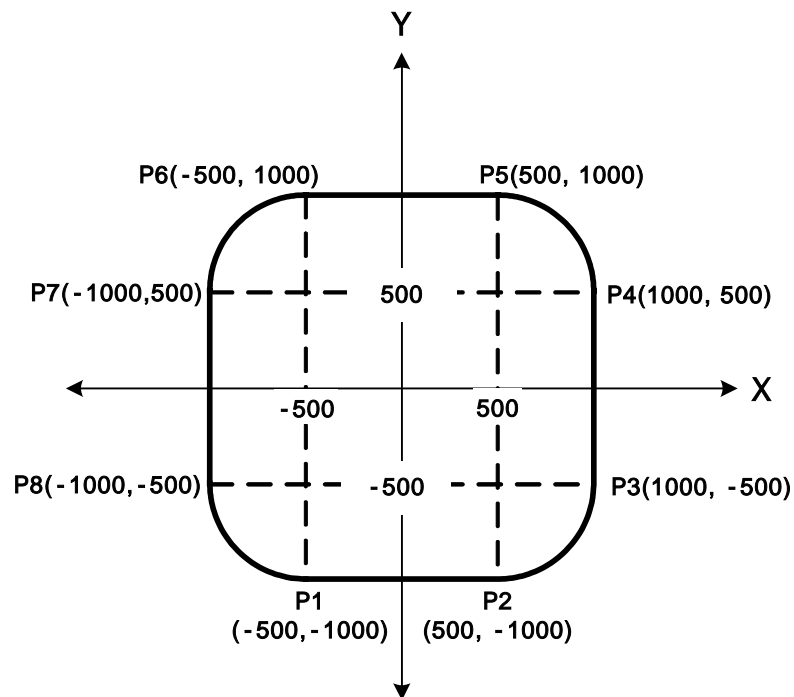


[그림 #] McArcToA 메소드를 사용한 원호 보간 이동

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 ComiCxAx1.LoadDivice
```

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



```

Const MAP0 = 0

Dim PosList(1)

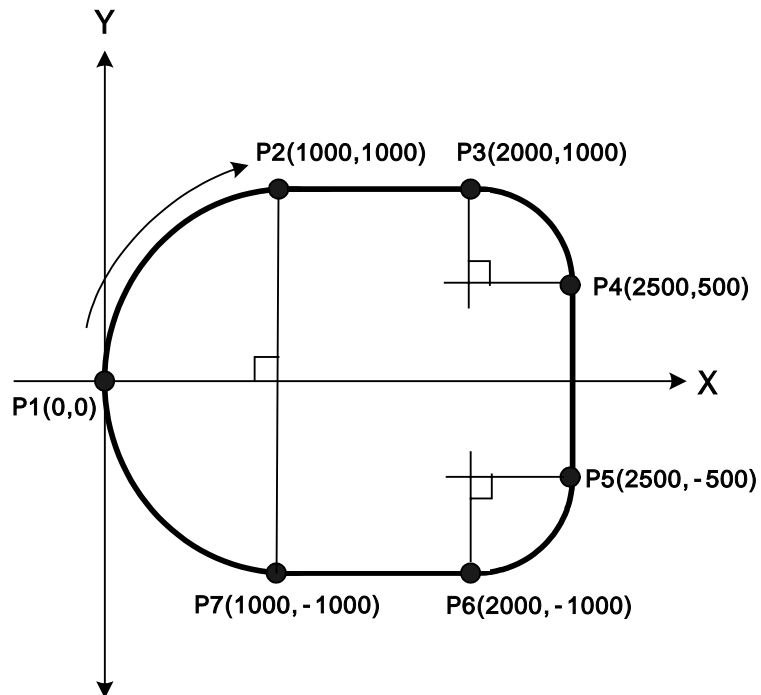
` Map X&Y axis to MAP0
Call ComiCxAx1.McMapAxes(MAP0, X_MASK or Y_MASK)
` Set speed mode as Trapezoidal
Call ComiCxAx1.McSetSpeedModeMx(MAP0, 1)
` Set speed & accel => V=500, Acc=500
Call ComiCxAx1.McSetSpeedMx(MAP0, 500, 500)
` Move from P1 to P2
PosList(0) = 500
PosList(1) = -1000
Call ComiCxAx1.McLineTo(MAP0, PosList(0))
` Move from P2 to P3
Call ComiCxAx1.McArcA(MAP0, 500, -500, 90)
` Move from P3 to P4
PosList(0) = 1000
PosList(1) = 500
Call ComiCxAx1.McLineTo (MAP0, PosList(0))
` Move from P4 to P5
Call ComiCxAx1.McArcA(MAP0, 500, 500, 90)
` Move from P5 to P6
PosList(0) = -500
PosList(1) = 1000
Call ComiCxAx1.McLineTo (MAP0, PosList(0))
` Move from P6 to P7
Call ComiCxAx1.McArcA(MAP0, -500, 500, 90)
` Move from P7 to P8
PosList(0) = -1000
PosList(1) = -500
Call ComiCxAx1.McLineTo (MAP0, PosList(0))
` Move from P8 to P1
Call ComiCxAx1.McArcA(MAP0, -500, -500, 90)

Call ComiCxAx1.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 ComiCxAx1.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 ComiCxAx1.McMapAxes(MAP0, X_MASK or Y_MASK)
' Set speed mode as Trapezoidal
Call ComiCxAx1.McSetSpeedModeMx(MAP0, 1)
' Set speed & accel => V=500, Acc=500
Call ComiCxAx1.McSetSpeedMx(MAP0, 500, 500)
' Move from P1 to P2
Call ComiCxAx1.McArcA(MAP0, 1000, 0, 90)
' Move from P2 to P3
PosList(0) = 2000
PosList(1) = 1000
Call ComiCxAx1.McLineTo (MAP0, PosList(0))
' Move from P3 to P4
Call ComiCxAx1.McArcA(MAP0, 2000, 500, 90)
' Move from P4 to P5
PosList(0) = 2500
PosList(1) = -500
Call ComiCxAx1.McLineTo (MAP0, PosList(0))
' Move from P5 to P6
Call ComiCxAx1.McArcA(MAP0, 2000, -500, 90)
' Move from P6 to P7
PosList(0) = 1000
PosList(1) = -1000
Call ComiCxAx1.McLineTo (MAP0, PosList(0))
' Move from P7 to P1
Call ComiCxAx1.McArcA(MAP0, 1000, 0, 90)

Call ComiCxAx1.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 Point)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 point)의 X 축 절대좌표값
- ▶ **YEndPos** : 원호보간 이동을 완료할 목표지점(End point)의 Y 축 절대좌표값

참 고

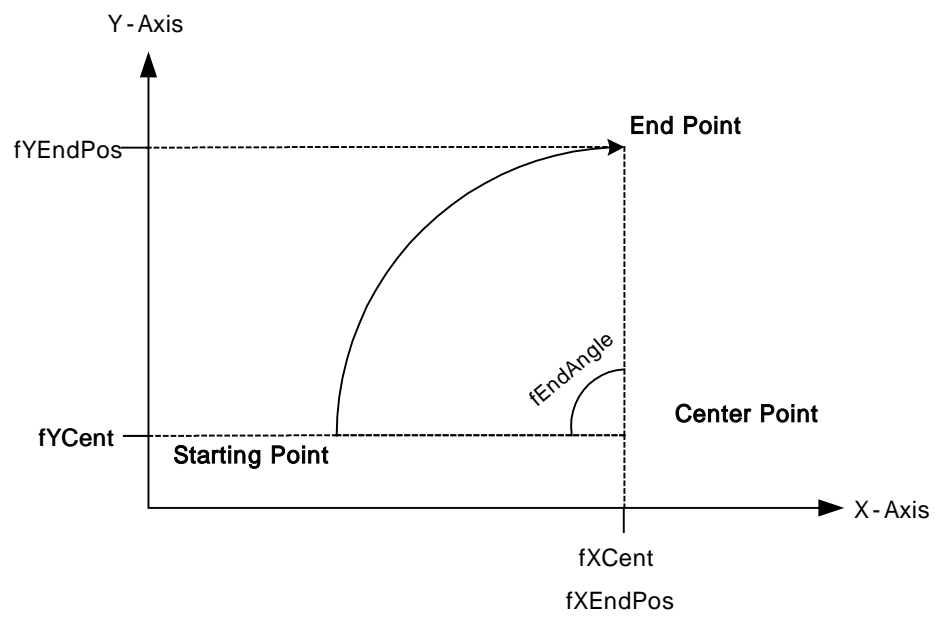
□ 원호보간 이동은 두 축에 대해서만 적용가능합니다. 따라서 본 단락에서는 맵핑된 두 축을 X, Y 축으로 간주하여 설명합니다. 여기서 X 축이라 함은 맵핑된 두 축 중에서 채널번호(X,Y,Z,U 순)가 낮은 축을 의미하며 Y 축은 채널번호가 높은 축을 의미합니다. 예를 들어 Z 축과 U 축이 맵핑된 두 축이라면 Z 축이 X 축에 해당하며 U 축이 Y 축에 해당합니다.

□ 이 메소드는 원호 보간 이동을 시작시킨후 바로 Return 합니다.

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

□ McStartArcToA 메소드가 원호 보간 이동을 완료할 목표지점의 각도를 파라미터로 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 Point)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 point)의 X 축 절대좌표값
- ▶ **YEndPos** : 원호보간 이동을 완료할 목표지점(End point)의 Y 축 절대좌표값

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.

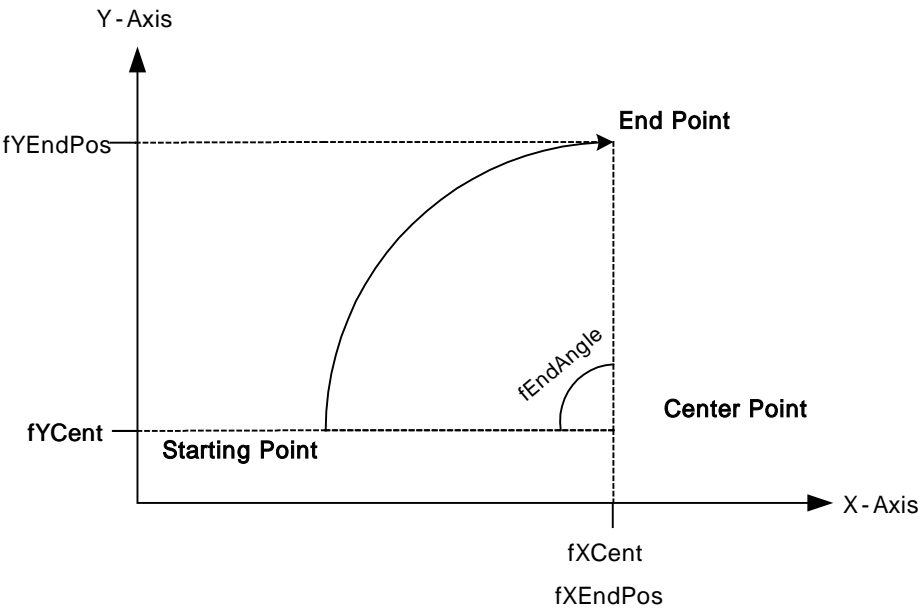
□ 이 메소드는 원호 보간 이동이 완료되기 전까지 Return 되지 않고 루프를 돌게 됩니다. 루프를 도는 동안 윈도우 이벤트나 메시지가 처리될 수 있도록 하려면 이 메소드를 수행하기 이전에 McSetBlockingMode 메소드를 사용하여 Blocking 이 일어나지 않도록 설정하여야 합니다.

□ XEndPos 값과 YEndPos 값이 현재 위치(Starting Point)의 좌표값과 일치하면 완전한 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.

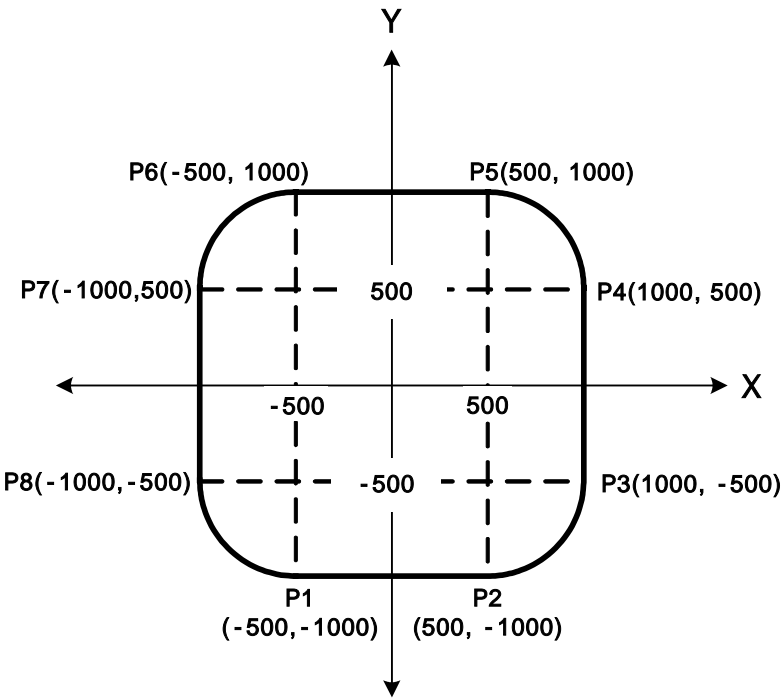


[그림 #] McArcToP 메소드를 사용한 원호 보간 이동

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 ComiCxAx1.LoadDevice

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

Const MAP0 = 0

Dim PosList(1)

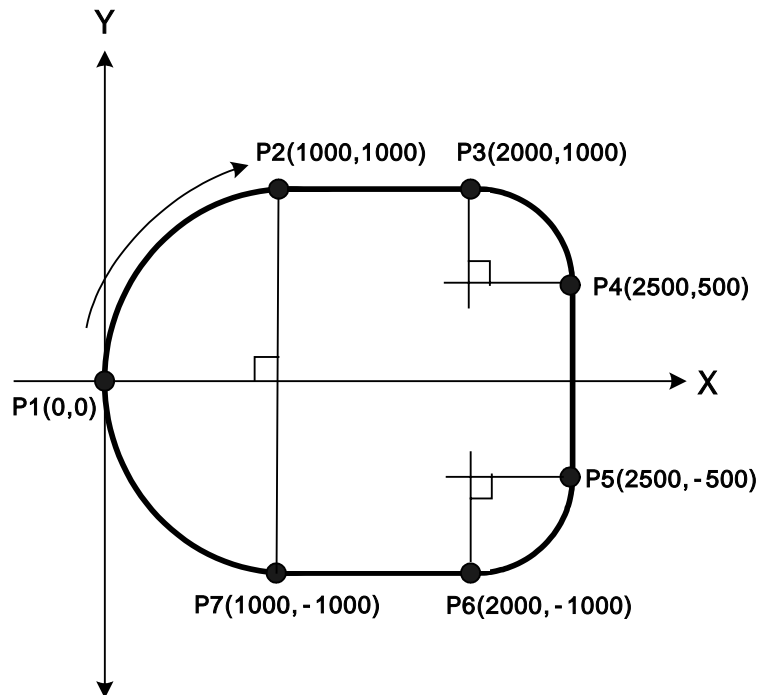
` Map X&Y axis to MAP0
Call ComiCxAx1.McMapAxes(MAP0, X_MASK or Y_MASK)
` Set speed mode as Trapezoidal
Call ComiCxAx1.McSetSpeedModeMx(MAP0, 1)
` Set speed & accel => V=500, Acc=500
Call ComiCxAx1.McSetSpeedMx(MAP0, 500, 500)
` Move from P1 to P2
PosList(0) = 500
PosList(1) = -1000
Call ComiCxAx1.McLineTo(MAP0, PosList(0))
` Move from P2 to P3
Call ComiCxAx1.McArcP(MAP0, 500, -500, 1000, -500)
` Move from P3 to P4
PosList(0) = 1000
PosList(1) = 500
Call ComiCxAx1.McLineTo (MAP0, PosList(0))
` Move from P4 to P5
Call ComiCxAx1.McArcP(MAP0, 500, 500, 500, 1000)
` Move from P5 to P6
PosList(0) = -500
PosList(1) = 1000
Call ComiCxAx1.McLineTo (MAP0, PosList(0))
` Move from P6 to P7
Call ComiCxAx1.McArcP(MAP0, -500, 500, -1000, 500)
` Move from P7 to P8
PosList(0) = -1000
PosList(1) = -500
Call ComiCxAx1.McLineTo (MAP0, PosList(0))
` Move from P8 to P1
Call ComiCxAx1.McArcP(MAP0, -500, -500, -500, -1000)

Call ComiCxAx1.UnloadDevice

```

■ 예제 2

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



```
Call ComiCxAx1.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 ComiCxAx1.McMapAxes(MAP0, X_MASK or Y_MASK)
` Set speed mode as Trapezoidal
Call ComiCxAx1.McSetSpeedModeMx(MAP0, 1)
` Set speed & accel => V=500, Acc=500
Call ComiCxAx1.McSetSpeedMx(MAP0, 500, 500)
` Move from P1 to P2
Call ComiCxAx1.McArcP(MAP0, 1000, 0, 1000, 1000)
` Move from P2 to P3
PosList(0) = 2000
PosList(1) = 1000
Call ComiCxAx1.McLineTo (MAP0, PosList(0))
` Move from P3 to P4
Call ComiCxAx1.McArcP(MAP0, 2000, 500, 2500, 500)
` Move from P4 to P5
PosList(0) = 2500
PosList(1) = -500
Call ComiCxAx1.McLineTo (MAP0, PosList(0))
` Move from P5 to P6
Call ComiCxAx1.McArcP(MAP0, 2000, -500, 2000, -1000)
` Move from P6 to P7
PosList(0) = 1000
PosList(1) = -1000
Call ComiCxAx1.McLineTo (MAP0, PosList(0))
` Move from P7 to P1
```



```
Call ComiCxAx1.McArcP(MAP0, 1000, 0, 0, 0)
```

```
Call ComiCxAx1.UnloadDevice
```

■ 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

2.5.5 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	
Sub McOverrideSpeedSetAll (ByVal NumAxis As Long, ByVal AxisList[] As Long) 여러 축에 대하여 동시에 속도를 변경.	
Sub McOverrideMove (ByVal Channel As Long, ByVal NewDistance As Double) McStartMove 메소드를 통하여 수행되는 상대좌표 In-position 모션에 대하여 상대 좌표값, 즉 목표 거리값을 수정	
Sub McOverrideMoveTo (ByVal Channel As Long, ByVal NewPosition As Double) McStartMoveTo 메소드를 통하여 수행되는 절대좌표 In-position 모션에 대하여 목표 절대좌표값을 수정.	

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 ComiCxAx1.LoadDivice

Const X_AXIS = 0
Dim Speed(2)
Speed(0) = 10000
Speed(1) = 20000
Speed(2) = 30000

Call ComiCxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiCxAx1.McSetAccel(X_AXIS, 20000, 20000)
Call ComiCxAx1.McSetSpeed(X_AXIS, 0, Speed(0) )
' (+)방향으로 Velocity Move 수행
Call ComiCxAx1.McStartVMove(X_AXIS, 1)

' 일정 시간동안 운행

Call ComiCxAx1.McSetSpeed(X_AXIS, 0, Speed(1) )
    
```

```
Call ComiCxAx1.McOverrideSpeedSet(X_AXIS)
```

‣ 일정 시간동안 운행

```
Call ComiCxAx1.McSetSpeed(X_AXIS, 0, Speed(2))
```

```
Call ComiCxAx1.McOverrideSpeedSet(X_AXIS)
```

‣ 일정 시간동안 운행

‣ 감속 후 정지

```
Call ComiCxAx1.McStop(X_AXIS)
```

```
Call ComiCxAx1.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)하고자 할 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 ComiCxAx1.LoadDivice
```

```
Const X_AXIS = 0
Const Y_AXIS = 1
Const Z_AXIS = 2
Dim AxisList(2)
AxisList(0) = X_AXIS
```

```

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 ComiCxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiCxAx1.McSetAccel(X_AXIS, 20000, 20000)
Call ComiCxAx1.McSetSpeed(X_AXIS, 0, Speed(0) )

Call ComiCxAx1.McSetSpeedMode(Y_AXIS, 1)
Call ComiCxAx1.McSetAccel(Y_AXIS, 20000, 20000)
Call ComiCxAx1.McSetSpeed(Y_AXIS, 0, Speed(0) )

Call ComiCxAx1.McSetSpeedMode(X_AXIS, 1)
Call ComiCxAx1.McSetAccel(Z_AXIS, 20000, 20000)
Call ComiCxAx1.McSetSpeed(Z_AXIS, 0, Speed(0) )

' (+)방향으로 Velocity Move 수행
Call ComiCxAx1.McStartVMove(X_AXIS, 1)
Call ComiCxAx1.McStartVMoveAll(3, AxisList(0), DirList(0))

' 일정 시간동안 운행

Call ComiCxAx1.McSetSpeed(X_AXIS, 0, Speed(1) )
Call ComiCxAx1.McSetSpeed(Y_AXIS, 0, Speed(1) )
Call ComiCxAx1.McSetSpeed(Z_AXIS, 0, Speed(1) )
Call ComiCxAx1.McOverrideSpeedSetAll(3, AxisList(0))

' 일정 시간동안 운행

Call ComiCxAx1.McSetSpeed(X_AXIS, 0, Speed(2))
Call ComiCxAx1.McSetSpeed(Y_AXIS, 0, Speed(2))
Call ComiCxAx1.McSetSpeed(Z_AXIS, 0, Speed(2))
Call ComiCxAx1.McOverrideSpeedSetAll(3, AxisList(0))

' 일정 시간동안 운행

' 감속 후 정지
Call ComiCxAx1.McStopAll(3, AxisList(0))

Call ComiCxAx1.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.

■ 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.

2.5.6 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-ST502 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-ST502 보드에 입력되어야 합니다.

□ EL 신호

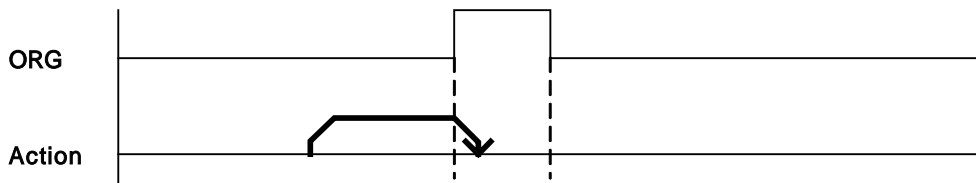
기계적인 리미트(Limit) 신호를 의미합니다. 이 신호는 일반적으로 구조물이 움직일 수 있는 한계점을 감지하기 위해 사용되나 원점 복귀 모드에 따라 ORG 신호의 대용으로도 사용될 수 있습니다. EL 신호는 (+)방향 리미트 신호와 (-)방향 리미트 신호의 두 가지 신호가 있습니다. (+)방향 리미트 신호는 터미널 보드의 ‘+EL’ 단자, 그리고 (-)방향 리미트 신호는 ‘-EL’ 단자를 통하여 COMI-ST502 보드에 입력되어야 합니다.

원점복귀모드

COMI-ST502 모션제어보드는 다음과 같이 13 가지의 다양한 원점 복귀 모드를 제공합니다. 원점 복귀 모드는 McSetHomeConfig 메소드를 통하여 설정됩니다. 아래의 그림은 모두 속도 모드를 Trapezoidal 모드로 설정한 상태임을 가정하여 그려진 것이며, 만일 Constant 속도 모드로 설정된 경우에는 감속이 없이 즉시 정지하게 됩니다.

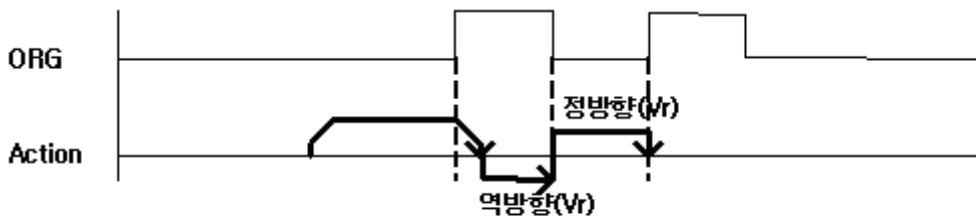
□ MODE 0 : ORG -> Slow down -> Stop

MODE 0 에서는 ORG 신호가 OFF 에서 ON 으로 바뀌는 순간에 모션을 감속 후 정지하고 복귀 작업을 종료합니다.



□ MODE 1 : ORG → Slow down → Go back → Go forward → Stop

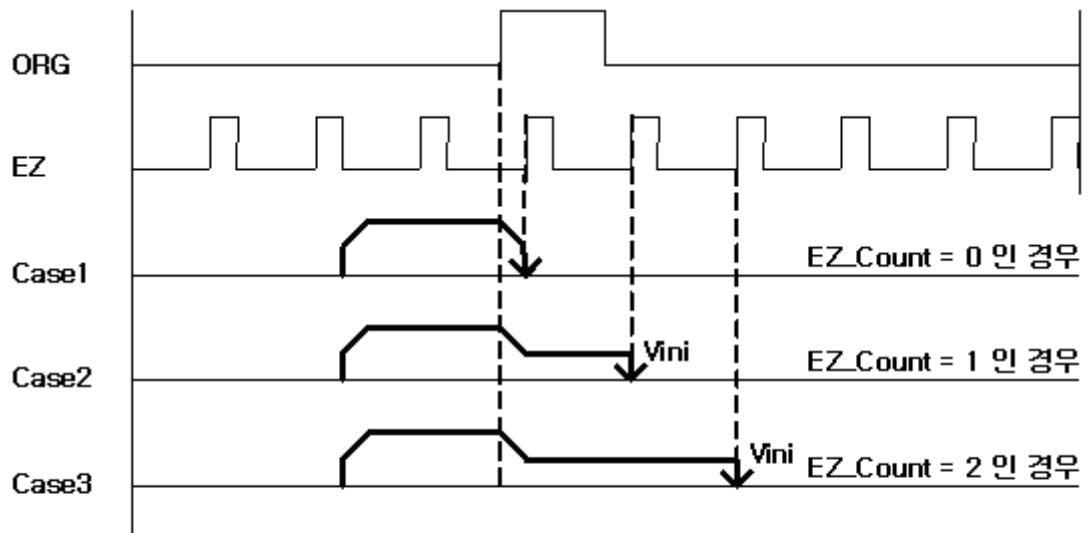
MODE 1에서는 ORG 신호가 OFF에서 ON으로 바뀌는 순간에 모션을 감속 후 정지한 후 ORG 신호가 OFF가 될 때까지 Vr (Reverse Speed)의 속도로 역방향 회전을 수행합니다. ORG 신호가 OFF되는 순간에 다시 Vr의 속도로 정방향 회전을 수행하다가 ORG 신호가 다시 ON되는 순간에 복귀작업을 종료합니다.



Vr : 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

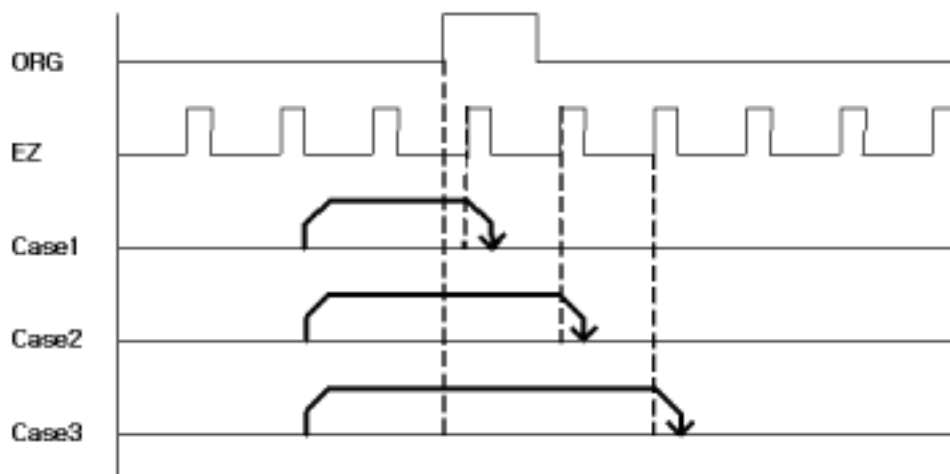
Case2 : EzCount = 1 인 경우, McSetHomeConfig 함수 참조

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 가속 후 정지합니다. McSetHomeConfig 메소드를 통하여 미리 설정된 EzCount 값에 따라 가속을 시작하는 시점은 아래와같이 달라집니다.



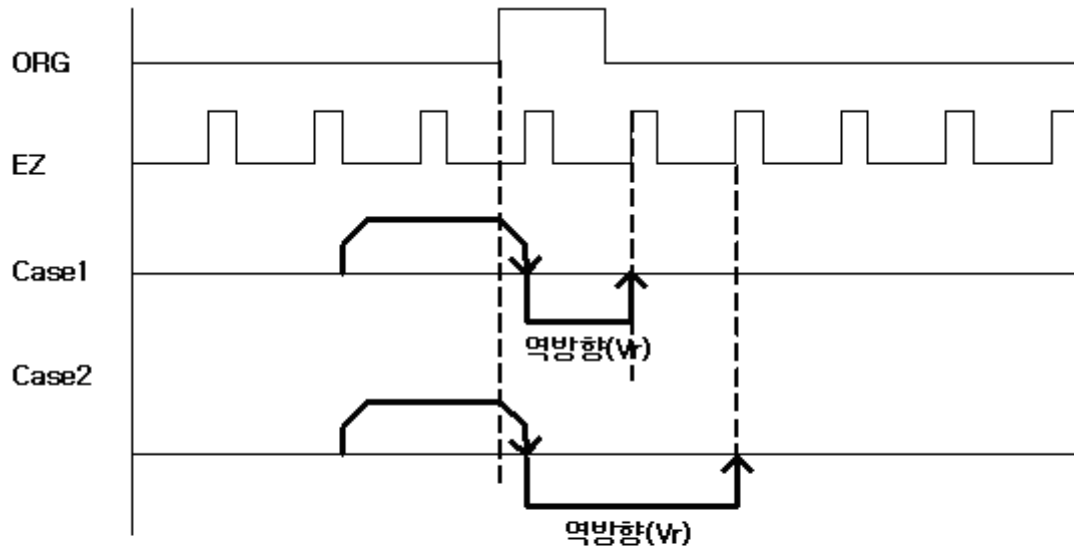
Case1 : EzCount = 0 인 경우, McSetHomeConfig 함수 참조

Case2 : EzCount = 1 인 경우, McSetHomeConfig 함수 참조

Case3 : EzCount = 2 인 경우, McSetHomeConfig 함수 참조

□ 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.



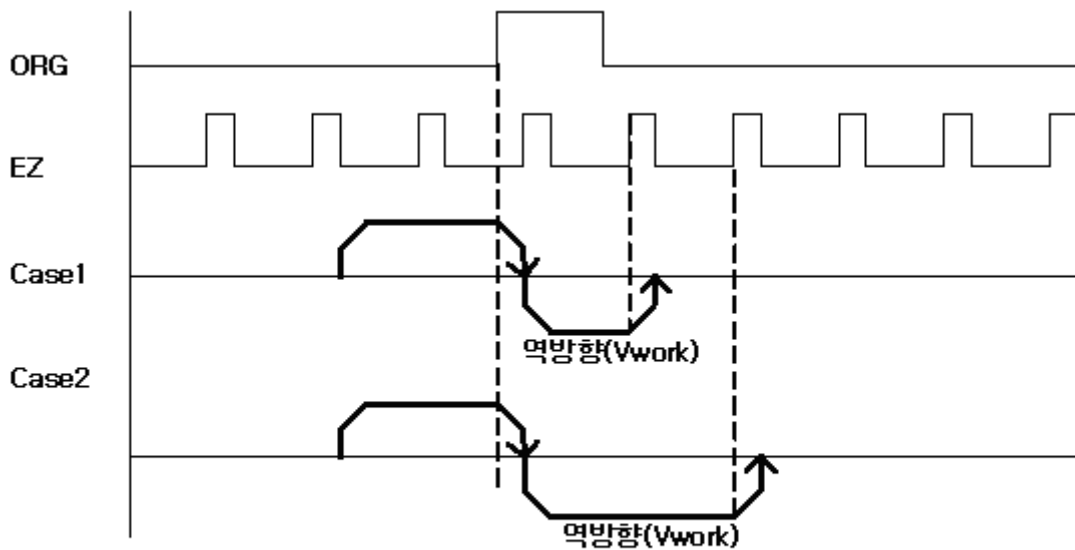
Case1 : EzCount = 1 인 경우, McSetHomeConfig 함수 참조

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 작업 속도까지 가속하여 역방향 회전한 후 EZ 신호에 따라 감속 후 복귀 작업을 종료합니다.

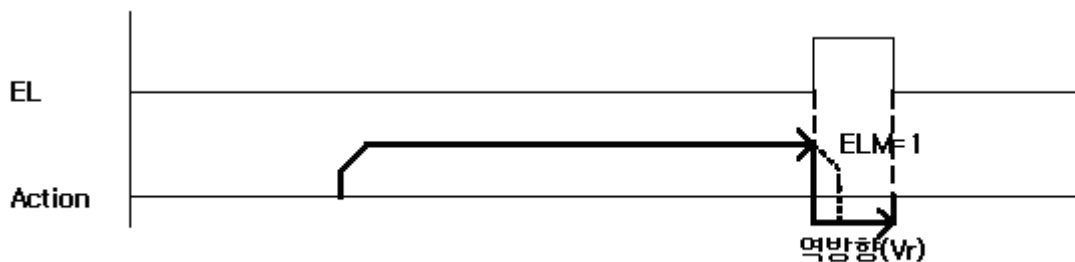


Case1 : EzCount = 1 인 경우, McSetHomeConfig 함수 참조

Case2 : EzCount = 2 인 경우, McSetHomeConfig 함수 참조

□ MODE 6 : EL ON → Stop → Go back at Vr → EL OFF → Stop

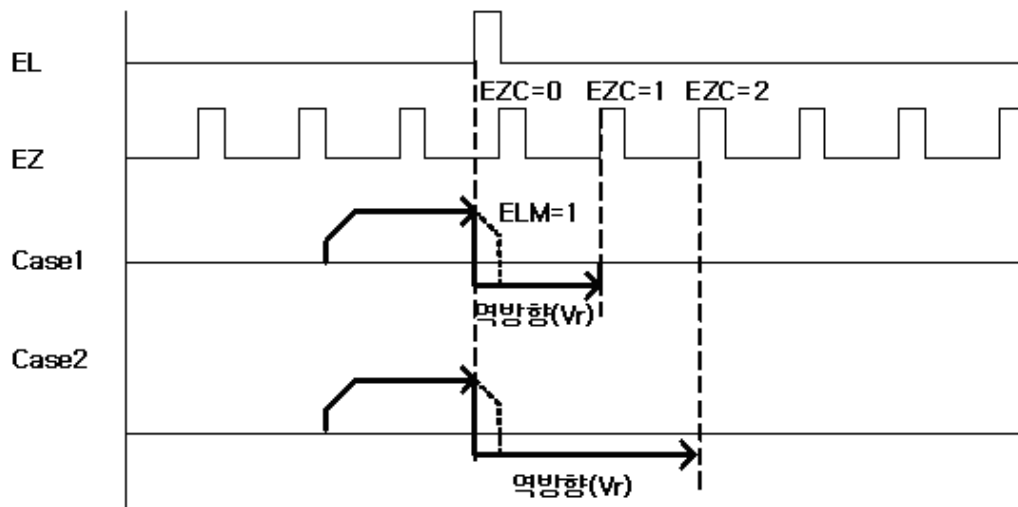
MODE 6에서는 EL 신호가 ON으로 바뀌는 순간 즉시 정지(또는 ELM=1인 경우에 감속후 정지)합니다. 그리고 반대 방향으로 Vr 속도로 회전하다가 EL 신호가 OFF되는 순간에 복귀작업을 종료합니다. 여기서 ELM=1은 McSetMioCfgEL 메소드에서 nElMode를 1로 설정했음을 의미합니다.



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.



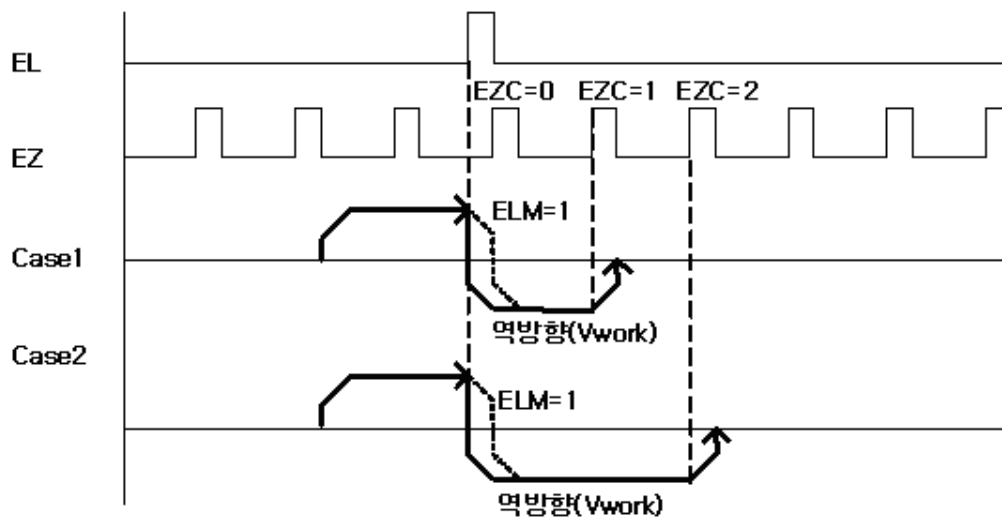
Case1 : EzCount = 1 인 경우, McSetHomeConfig 함수 참조

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 9에서는 MODE 0에서와 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 10에서는 MODE 3에서와 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 11에서는 MODE 5에서와 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 12에서는 MODE 8에서와 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-ST502 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 ComiCxAx1.LoadDevice

Call
ComiCxAx1.McSetHomeConfig(X_AXIS,MODE4,LOW_ACTIVE,0,LOW_ACTIVE,0);

Call ComiCxAx1.McSetSpeedMode(X_AXIS, 1);
Call ComiCxAx1.McSetSpeed(X_AXIS, 200, 5000);
Call ComiCxAx1.McSetAccel(X_AXIS, 5000, 5000);

Call ComiCxAx1.McHomeMove(X_AXIS, 1, 500);
While ( Not ComiCxAx1.McDone (X_AXIS))
Wend

Call ComiCxAx1.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 사용하는 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 ComiCxAx1.LoadDevice

Call
ComiCxAx1.McSetHomeConfig(X_AXIS,MODE4,LOW_ACTIVE,0,LOW_ACTIVE,0);

Call ComiCxAx1.McSetSpeedMode(X_AXIS, 1);
Call ComiCxAx1.McSetSpeed(X_AXIS, 200, 5000);
Call ComiCxAx1.McSetAccel(X_AXIS, 5000, 5000);

Call ComiCxAx1.McHomeMove(X_AXIS, 1, 500);
While Not( ComiCxAx1.McDone(X_AXIS))
Wend

Call ComiCxAx1.COMICX_UnloadDevice
}
```

2.5.7 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-ST501 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 적정한 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 McSetPulserInputMode (ByVal Channel As Long, ByVal InputMode As Long, ByVal Inverse As Integer) Pulser 입력신호 대한 환경을 설정합니다.	
Sub McPulserHomeMove (ByVal Channel As Long, ByVal HomeType As Long) Pulser Input 에 의한 원점 복귀 작업을 수행합니다.	
Sub McStartPulserVMove (ByVal Channel As Long) Stop 메소드가 호출될 때까지 Pulser 신호 입력에 맞추어 계속적인 모션을 수행합니다.	
Sub McStartPulserMove (ByVal Channel As Long, ByVal Distance As Double) Pulser 신호 입력에 맞추어 지정한 거리(상대좌표)만큼 이동을 수행합니다.	
Sub McPulserMove (ByVal Channel As Long, ByVal Distance As Double) Pulser 신호 입력에 맞추어 지정한 거리(상대좌표)만큼 이동을 수행합니다.	
Sub McStartPulserMoveTo (ByVal Channel As Long,By Val Position As Double) Pulser 신호 입력에 맞추어 지정한 위치(절대좌표)로 이동을 수행합니다.	
Sub McPulserMoveTo (ByVal Channel As Long, ByVal Position As Double) Pulser 신호 입력에 맞추어 지정한 위치(절대좌표)로 이동을 수행합니다.	

■ 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 ComiCxAx1.LoadDivice

Const X_AXIS = 0
Const PHASE_1X = 0 ' Pulser input mode => 1X A/B (1 채널 Phase type
입력 모드)

Call ComiCxAx1.McReset
' Pulser 입력신호의 최대 주파수 제한 설정(650000PPS)
Call ComiCxAx1.McSetSpeed(X_AXIS, 0, 650000)
Call ComiCxAx1.McSetPulserInputMode (X_AXIS, PHASE_1X, FALSE)
' Manual Pulser 모드에서 Velocity Move 모션을 시작한다.
Call ComiCxAx1.McStartPulserVMove (X_AXIS)

' 일정시간동안 운행

Call ComiCxAx1.McEmgStop(X_AXIS)
Call ComiCxAx1.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 적절한 값으로 설정하여야 합니다.

□ Manual Pulser 모드 모션을 취소하기 위해서는 McStop() 메소드대신 McEmgStop()메소드를 사용하여야 합니다.

예 제

```
Call ComiCxAx1.LoadDevice

Const X_AXIS = 0
Const PHASE_1X = 0 ' Pulser input mode => 1X A/B (1 채널 Phase type 입력 모드)

Call ComiCxAx1.McReset
' Pulser 입력신호의 최대 주파수 제한 설정(650000PPS)
Call ComiCxAx1.McSetSpeed(X_AXIS, 0, 650000)
Call ComiCxAx1.McSetPulserInputMode(X_AXIS, PHASE_1X, FALSE)
Call ComiCxAx1.McStartPulserMove(X_AXIS, 1000)
While Not ( ComiCxAx1.McDone())
Wend
Call ComiCxAx1.UnloadDevice
```


■ 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 ComiCxAx1.LoadDevice

Const X_AXIS = 0
Const PHASE_1X = 0 ' Pulser input mode => 1X A/B (1 채널 Phase type 입력 모드)

Call ComiCxAx1.McReset
' Pulser 입력신호의 최대 주파수 제한 설정(650000PPS)
Call ComiCxAx1.McSetSpeed(X_AXIS, 0, 650000)
Call ComiCxAx1.McSetPulserInputMode(X_AXIS, PHASE_1X, FALSE)
Call ComiCxAx1.McPulserMove(X_AXIS, 1000)

Call ComiCxAx1.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 ComiCxAx1.LoadDevice
Const X_AXIS = 0
Const PHASE_1X = 0 ` Pulser input mode => 1X A/B (1 채널 Phase type 입력 모드)

Call ComiCxAx1.McReset
` Pulser 입력신호의 최대 주파수 제한 설정(650000PPS)
Call ComiCxAx1.McSetSpeed(X_AXIS, 0, 650000)
Call ComiCxAx1.McSetPulserInputMode(X_AXIS, PHASE_1X, FALSE)
Call ComiCxAx1.McStartPulserMoveTo(X_AXIS, 1000)
While Not ( ComiCxAx1.McDone())
Wend

Call ComiCxAx1.UnloadDevice
```

■ 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 ComiCxAx1.LoadDevice

Const X_AXIS = 0
Const PHASE_1X = 0 ` Pulser input mode => 1X A/B (1 채널 Phase type 입력 모드)

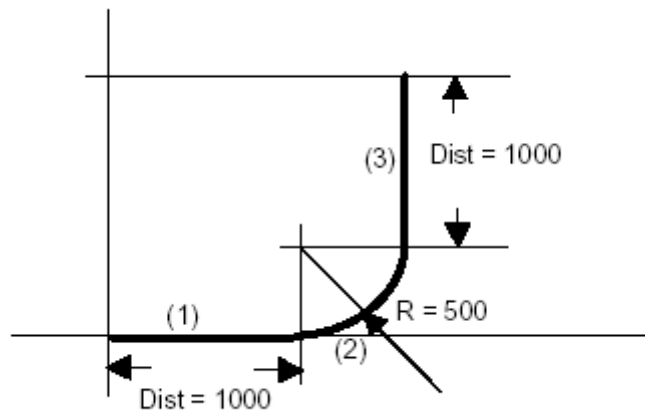
Call ComiCxAx1.McReset
` Pulser 입력신호의 최대 주파수 제한 설정(650000PPS)
Call ComiCxAx1.McSetSpeed(X_AXIS, 0, 650000)
Call ComiCxAx1.McSetPulserInputMode(X_AXIS, PHASE_1X, FALSE)
Call ComiCxAx1.McPulserMoveTo(X_AXIS, 1000)

Call ComiCxAx1.UnloadDevice
```

2.5.8 리스트 모션(Listed Motion) 메소드

이 단원에서는 리스트 모션(Listed Motion)제어에 관련된 메소드들을 소개합니다. 리스트 모션은 수행해야 할 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 메소드와 같은 In-Position 메소드를 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 ComiCxAx1.McBeginList ' 모션 리스트 등록 시작
Call ComiCxAx1.McMapAxes(0, 0x3) ' Map X & Y axis
' Set Trapezoidal Speed Mode
Call ComiCxAx1.McSetSpeedModeMx(0, 1)
' Set work speed=500, Acceleration=1000
Call ComiCxAx1.McSetSpeedMx(0, 500, 1000)
' (1000,0)만큼 직선 보간 이동
DistList(0) = 1000
DistList(1) = 0

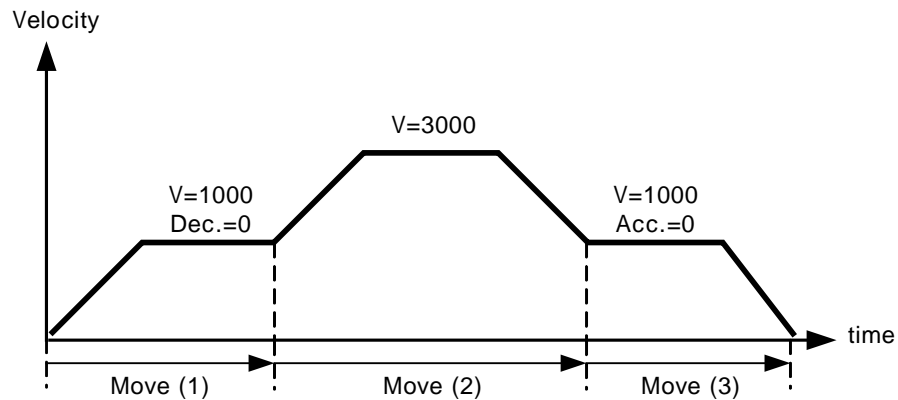
Call ComiCxAx1.McLine(0, DistList(0))
' 중심점 Offset = (0, 500), End Angle = 90 DEG
' 의 조건으로 원호보간 이동
Call ComiCxAx1.McArcA(0, 0, 500, 90)
```

```

` (0,1000)만큼 직선 보간 이동
DistList(0) = 0
DistList(1) = 1000
Call ComiCxAx1.McLine(0, DistList(0))
Call ComiCxAx1.McEndList ` 모션 리스트 등록을 마칩
Call ComiCxAx1.McStartListMotion ` 리스트 모션 수행
` 리스트 모션이 모두 완료될 때까지 기다림
While Not ( ComiCxAx1.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 ComiCxAx1.McBeginList ` 모션 리스트 등록 시작
` Set Trapezoidal Speed Mode
Call ComiCxAx1.McSetSpeedMode(0, 1)
` Move (1)
Call ComiCxAx1.McSetSpeed(0, 0, 1000)
Call ComiCxAx1.McSetAccel(0, 2000, 0)
Call ComiCxAx1.McMoveTo(0, 3000)
` Move (2)
Call ComiCxAx1.McSetSpeed(0, 1000, 3000)
Call ComiCxAx1.McSetAccel(0, 2000, 2000)
Call ComiCxAx1.McMoveTo(0, 8000)
` Move (3)
Call ComiCxAx1.McSetSpeed(0, 0, 1000)
Call ComiCxAx1.McSetAccel(0, 0, 2000)

```

```
Call ComiCxAx1.McMoveTo(0, 11000)
Call ComiCxAx1.McEndList ' 모션 리스트 등록을 마칩

Call ComiCxAx1.McStartListMotion ' 리스트 모션 수행
' 리스트 모션이 모두 완료될 때까지 기다림
While Not ( ComiCxAx1.McCheckListMotionDone)
Wend
```

리스트 모션에 관련된 메소드들은 다음과 같습니다.

메소드 / 설명	페이지
Sub McBeginList List Motion 에서 수행할 작업들의 등록을 시작합니다.	
Sub McEndList List Motion 에서 수행할 작업들의 등록을 종료합니다.	
Sub McStartListMotion List Motion 으로 등록된 작업들에 대하여 실제로 모션을 시작합니다.	
Sub McAbortListMotion 리스트 모션을 수행하고 있는 중에 리스트 모션을 중지합니다.	
Function McCheckListMotionDone As Boolean 리스트 모션 수행이 완료됐는지를 알려줍니다.	

McBeginList

메소드 원형

Sub **McBeginList**

메소드 설명

이 메소드는 List Motion 에서 수행할 작업들의 등록을 시작합니다.

■ 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 수행 되면 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

2.5.9 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.	
Sub McEnableActSpdChk (ByVal Interval As Long) 실제 모션의 속도를 체크하는 기능을 Enable 시킵니다.	
Sub McDisableActSpdChk 실제 모션의 속도(Actual Speed)를 체크하는 기능을 Disable 시킵니다.	
Function McGetActualSpeed (ByVal Channel As Long) As Double EA/EB 단자로 입력되는 Feedback 펄스를 계측하여 실제 모션의 속도를 반환합니다.	
Function McGetPositionA (ByVal Channel As Long) As Double 현재의 실제 위치(Actual Position)를 논리 단위로 반환합니다.	
Sub McSetPositionA (ByVal Channel As Long, ByVal Pos As Double) 현재의 실제 위치(Actual Position)값을 설정하며 단위는 논리단위입니다.	
Function McGetPositionC (ByVal Channel As Long) As Double 현재의 명령 위치(Command Position)를 논리 단위로 반환합니다.	
Sub McSetPositionC (ByVal Channel As Long, ByVal Pos As Double) 명령 위치(Command Position)값을 설정하며 단위는 논리단위입니다.	
Function McGetCountA (ByVal Channel As Long) As Long 현재의 실제 위치(Actual Position) 카운트값을 반환합니다.	
Sub McSetCountA (ByVal Channel As Long, ByVal Count As Long) 현재의 실제 위치(Actual Position) 카운트값을 설정합니다.	
Function McGetCountC (ByVal Channel As Long) As Long 현재의 명령 위치(Command Position) 카운트값을 반환합니다.	
Sub McSetCountC (ByVal Channel As Long, ByVal Count As Long) 현재의 명령 위치(Command Position) 카운트값을 설정합니다.	

Function McGetCountD (ByVal Channel As Long) As Long Deviation Counter 의 값을 읽어서 반환합니다.	
Function McSetCountD (ByVal Channel As Long, ByVal Count As Long) As Long Deviation Counter 의 값을 새로이 설정합니다.	
Function McGetCountG (ByVal Channel As Long) As Long General Counter 의 값을 읽어서 반환합니다.	
Sub McSetCountG (ByVal Channel As Long, ByVal Count As Long) General Counter 의 값을 새로이 설정합니다.	
Function McGetCountR (ByVal Channel As Long) As Long Remained Counter 의 값을 읽어서 반환합니다.	
Sub McSetCountR (ByVal Channel As Long, ByVal Count As Long) Remained Counter 의 값을 새로이 설정합니다.	
Function McGetMotionStatus (ByVal Channel As Long) As Long 현재 모션의 동작 상태를 반환합니다.	
Function McGetMioStatus (ByVal Channel As Long) As Long 현재 모션과 관련된 여러가지 I/O 상태를 반환합니다.	

■ 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 기본적으로는 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 분해
능 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.

■ 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.

■ 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.

■ 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.

■ 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.

■ 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.

■ 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	Reserved
2	Reserved
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	RDY pin input status
BIT1	ALM	Alarm signal status
BIT2	+EL	Positive limit switch status
BIT3	-EL	Negative limit switch status
BIT4	ORG	Orgin switch status
BIT5	DIR	DIR output status
BIT6	Reserved	
BIT7	PCS	PCS signal input status
BIT8	ERC	ERC pin output status
BIT9	EZ	Index signal status
BIT10	Reserved	
BIT11	Latch	Latch signal input status
BIT12	SD	Slow Down signal input status
BIT13	INP	In-Position signal input status
BIT14 ~BIT31	Reserved	

2.5.10 I/O(입출력) 환경설정 메소드

이 단원에서는 I/O(입출력) 환경설정 메소드에 관하여 설명합니다. COMI-ST501 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 McSetMioCfgALM (ByVal Channel As Long, AlarmLogic As Long, AlarmMode As Long) Alarm 신호에 대한 입력모드를 설정합니다.	
Sub McGetMioCfgALM (ByVal Channel As Long, ByRef AlarmLogic As Long, ByRef AlarmMode As Long) 현재 설정된 Alarm 신호에 대한 입력모드를 읽어옵니다.	
Sub McSetMioCfgEL (ByVal Channel As Long, ByVal ElMode As Long) EL 신호에 대한 모드를 설정합니다.	
Sub McGetMioCfgEL (ByVal Channel As Long, ByRef ElMode As Long) EL 신호에 대한 모드 설정값을 읽어옵니다.	
Sub McSetMioCfgINP (ByVal Channel As Long, ByVal InpEnable As Integer, ByVal InpLogic As Long) INP 기능 및 신호에 대한 환경을 설정합니다.	
Sub McGetMioCfgINP (ByVal Channel As Long, ByRef pinpEnable As Long, ByRef InpLogic As Long) INP 기능 및 신호에 대한 환경 설정값을 읽어옵니다.	
Sub McSetMioCfgERC (ByVal Channel As Long, ByVal ErcLogic As Long, ByVal ErcOnTime As Long) ERC 신호의 입력 로직(Logic) 및 ON time 을 설정합니다.	
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.	
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.	
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.	
Sub McSetSoftLimit (ByVal Channel As Long, ByVal LimitP As Double, ByVal LimitN As Double) 소프트웨어적인 Limit 를 설정합니다.	
Sub McEnableSoftLimit (ByVal Channel As Long) 소프트웨어적인 Limit 의 적용을 Enable 시킵니다.	
Sub McDisableSoftLimit (ByVal Channel As Long) 소프트웨어적인 Limit 의 적용을 Disable 시킵니다.	
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.	
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.	

■ 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, ByVal AlarmLogic As Long, ByVal 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.

■ McSetMioCfgEL

메소드 원형

Sub **McSetMioCfgEL** (ByVal Channel As Long, ByVal EIMode As Long)

메소드 설명

이 메소드는 EL 신호에 대한 모드를 설정합니다.

매개 변수

- ▶ **Channel** : 채널(축) 번호, 0 ~ 3
- ▶ **EIMode** : 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

■ 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

■ 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

■ 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.

■ 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.

2.5.11 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.	
Sub McEnableInterrupt (ByVal EventHandle As Long) 인터럽트 이벤트 통지 기능을 Enable 시킵니다.	
Sub McDisableInterrupt 인터럽트 이벤트 통지 기능을 Disable 시킵니다.	
Function McGetAxisIntState (ByVal Channel As Long) As Boolean 각 축에 대하여 인터럽트가 발생하였는지를 알려주는 메소드입니다.	
Sub McGetIntStatus (ByVal Channel As Long, ByRef ErrorStatus As Long, ByRef EventStatus As Long) 각 축의 인터럽트가 발생한 원인을 알려주는 메소드입니다.	

■ 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. 채널별로 각각 다른 인터럽트 조건을 설정할 수 있습니다.
- ▶ **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	Reserved
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 EventHandle As Long)

메소드 설명

이 메소드는 인터럽트 이벤트 통지 기능을 Enable 시킵니다. 즉, 인터럽트가 발생하였을 때 사용자 Application 에게 인터럽트가 발생되었음을 알려주는 기능을 Enable 시킵니다.

매개 변수

▶ **EventHandle** : 인터럽트가 발생하였을 때 사용자 Application 에게 인터럽트가 발생되었음을 알려주기 위하여 사용될 이벤트 핸들. 이 핸들은 CreateEvent() 메소드와 같이 이벤트 핸들을 생성해주는 표준 윈도우 API 메소드를 통하여 먼저 이벤트 핸들을 생성한 후에 생성된 이벤트 핸들을 이 메소드를 통하여 등록하여야 합니다.

■ 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. 채널별로 각각 다른 인터럽트 조건을 설정할 수 있습니다.

Return 값

각 축의 인터럽트 발생 상태를 알려주는 32 비트 정수값

Bit	인터럽트 발생 조건
BIT0	X 축의 인터럽트 상태. 0=>No interrupt, 1=>Interrupt 발생
BIT1	Y 축의 인터럽트 상태. 0=>No interrupt, 1=>Interrupt 발생
BIT2	Z 축의 인터럽트 상태. 0=>No interrupt, 1=>Interrupt 발생
BIT3	U 축의 인터럽트 상태. 0=>No interrupt, 1=>Interrupt 발생
BIT4 ~ BIT31	Reserved

■ 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 interpolation

BIT16 ~ BIT31	Reserved
------------------	----------

▶ **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	Compared triggered for axis 0, 1
BIT13	Reserved
BIT14	Latched for axis2,3
BIT15	ORG input signal ON
BIT16	SD input signal ON

본 부록에서는 COMI-ST 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.

A.1 기능별 메소드 색인	2
A.2 메소드별 지원 가능 디바이스 리스트	6



Appendix A

A.1 기능별 메소드 색인

라이브러리 및 디바이스 시작/종료 Page

Function	LoadDevice As Boolean	11
Sub	UnloadDevice	12

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

아날로그 출력 Page

Sub	DA_Out (ByVal ch As Long, ByVal vmin As Long, ByVal vmax As Long)	19
Function	DA_Out (ByVal ch As Long, ByVal volt As Single) As Boolean	20

디지털 입출력 Page

Function	DiGetOne (ByVal ch As Long) As Long	23
Function	DiGetAll As Long	24
Sub	DoPutOne (ByVal ch As Long, ByVal status As Long)	25
Sub	DoPutAll (ByVal Statuses As Long)	26
Function	SdioInitComm As Boolean	28
Function	SdioCheckModule (ByVal ModuleNo As Long) As Boolean	29
Function	SdioSetDioUsage (ByVal ModuleNo As Long, ByVal Usage As TcmDiDoUsage) As Boolean	30
Function	SdioReadLowByte (ByVal ModuleNo As Long) As Integer	32
Function	SdioReadHighByte (ByVal ModuleNo As Long) As Integer	34
Function	SdioWriteLowByte (ByVal ModuleNo As Long, ByVal LowByte As Integer) As Boolean	36
Function	SdioWriteHighByte (ByVal ModuleNo As Long, ByVal HighByte As Integer) As Boolean	38

Boolean

모션제어 (모션 초기화 및 환경설정)	Page
Sub McReset	42
Sub McSetBlockingMode (ByVal Blocking As Integer)	43
Sub McSetOutputMode (ByVal Channel As Long, ByVal OutputMode As Long)	44
Function McGetOutputMode (ByVal Channel As Long) As Long	45
Sub McSetInputMode (ByVal Channel As Long, ByVal InputMode As Long, ByVal PulseLogic As Long)	46
Sub McSetSpeedRange (ByVal Channel As Long, ByVal MaxSpeed As Long)	47
Sub McGetInputMode (ByVal Channel As Long, ByRef InputMode As Long, ByRef PulseLogic As Long)	49
Sub McSetUnitDistance (ByVal Channel As Long, ByVal UnitDist As Double)	50
Function McGetUnitDistance (ByVal Channel As Long) As Double	52
Sub McSetUnitSpeed (ByVal Channel As Long, ByVal UnitSpeed As Double)	53
Function McGetUnitSpeed (ByVal Channel As Long) As Double	55

모션제어 (Single Axis 모션)	Page
Sub McSetSpeedMode (ByVal Channel As Long, ByVal ModeIndex As Long)	58
Sub McSetSpeed (ByVal Channel, ByVal IniSpeed As Double, ByVal WorkSpeed As Double)	62
Sub McSetAccel (ByVal Channel, ByVal Accel As Double, ByVal Decel As Double)	65
Sub McSetScurve (ByVal Channel, ByVal Svacc As Double, ByVal Svdec Double)	68
Sub McStartVMove (ByVal Channel As Long, ByVal Direction As Long)	71
Sub McStartMove (ByVal Channel As Long, ByVal Distance As Long)	72
Sub McMove (ByVal Channel As Long, ByVal Distance As Double)	74
Sub McStartMoveTo (ByVal Channel As Long, ByVal Position As Double)	76
Sub McMoveTo (ByVal Channel As Long)	78
Sub McStop (ByVal Channel As Long)	80
Sub McEmgStop (ByVal Channel As Long)	81
Function McDone (ByVal Channel As Long) As Boolean	82

모션제어 (Multi-Axis 동시제어)	Page
Sub McStartVMoveAll (ByVal NumAxis As Long, ByRef AxisList[] As Long, ByRef DirList[] As Long)	84

Sub	McStartMoveAll (ByVal NumAxis As Long, ByRef AxisList[] As Long, ByRef DistList[] As Long)	86
Sub	McMoveAll (ByVal NumAxis As Long, ByRef AxisList[] As Long, ByRef DistList[] As Long)	88
Sub	McStartMoveToAll (ByVal NumAxis As Long, ByRef AxisList[] As Long, ByRef PosList[] As Long)	90
Sub	McMoveToAll (ByVal NumAxis As Long, ByRef AxisList[] As Long, ByRef PosList[] As Long)	92
Sub	McStopAll (ByVal NumAxis As Long, ByRef AxisList[] As Long)	94
Sub	McEmgStopAll (ByVal NumAxis As Long, ByRef AxisList[] As Long)	95
Function	McAllDone (ByVal NumAxis As Long, ByRef AxisList[] As Long)As Boolean	96

모션제어 (Coordinated Motion)		Page
Function	McMapAxes (ByVal MapIndex As Long, ByVal MapMask As Integer) As Boolean	99
Sub	McSetSpeedModeMx (ByVal MapIndex As Long, ByVal ModeIndex As Long)	101
Sub	McSetSpeedMx (ByVal MapIndexAs Long, ByVal Speed As Long, ByVal Accel As Long)	103
Sub	McStartLine (ByVal MapIndex As Long, ByRef DistList[] As Double)	107
Sub	McLine (ByVal MapIndex As Long, ByRef DistList[] As Double)	109
Sub	McStartLineTo (ByVal MapIndex As Long, ByRef PosList[] As Double)	111
Sub	McLineTo (ByVal MapIndex As Long, ByRef PosList[] As Double)	114
Sub	McStartArcA (ByVal MapIndex As Long, ByVal XcentOffset As Double, ByVal YcentOffset As Double, ByVal EndAngle As Double)	117
Sub	McArcA (ByVal MapIndex As Long, ByVal XcentOffset As Double, ByVal YcentOffset As Double, ByVal EndAngle As Double)	119
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)	122
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)	124
Sub	McStartArcToA (ByVal MapIndex As Long, ByVal Xcent As Double, ByVal Ycent As Double, ByVal EndAngle As Double)	127
Sub	McArcToA (ByVal MapIndex As Long, ByVal Xcent As Double, ByVal Ycent As Double, ByVal EndAngle As Double)	129
Sub	McStartArcToP (ByVal MapIndex As Long, ByVal Xcent As Double, ByVal Ycent As	133

Appendix A 라이브러리 메소드 리스트

	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)	135
Function	McMxDone (ByVal MapIndex As Long) As Boolean	143

모션제어 (속도 및 위치 오버라이딩(Overriding)) Page

Sub	McOverrideSpeedSet (ByVal Channel As Long)	142
Sub	McOverrideSpeedSetAll (ByVal NumAxis As Long, ByRef AxisList[] As Long)	144
Sub	McOverrideMove (ByVal Channel As Long, ByVal NewDistance As Double)	146
Sub	McOverrideMoveTo (ByVal Channel As Long, ByVal NewPosition As Double)	147

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)	155
Sub	McHomeMove (ByVal Channel As Long, ByVal Direction As Long, ByVal RvsVel As Double)	157

모션제어 (Manual Pulser 모드 모션) Page

Sub	McSetPulserInputMode (ByVal Channel As Long, ByVal InputMode As Long, ByVal Inverse As Integer)	159
Sub	McPulserHomeMove (ByVal Channel As Long, ByVal HomeType As Long)	160
Sub	McStartPulserVMove (ByVal Channel As Long)	161
Sub	McStartPulserMove (ByVal Channel As Long, ByVal Distance As Double)	162
Sub	McPulserMove (ByVal Channel As Long, ByVal Distance As Double)	163
Sub	McStartPulserMoveTo (ByVal Channel As Long, ByVal Position As Double)	164
Sub	McPulserMoveTo (ByVal Channel As Long, ByVal Position As Double)	165

모션제어 (리스트 모션(Listed Motion)) Page

Sub	McBeginList	169
Sub	McBeginList	170
Sub	McStartListMotion	171
Sub	McAbortListMotion	172
Function	McCheckListMotionDone As Boolean	173

모션제어 (상태 감시 및 제어) Page



Function	McGetCurSpeed (ByVal Channel As Long) As Double	176
Sub	McEnableActSpdChk (ByVal Interval As Long)	177
Sub	McDisableActSpdChk	178
Function	McGetActualSpeed (ByVal Channel As Long) As Double	179
Function	McGetPositionA (ByVal Channel As Long) As Double	180
Sub	McSetPositionA (ByVal Channel As Long, ByVal Pos As Double)	181
Function	McGetPositionC (ByVal Channel As Long) As Double	182
Sub	McSetPositionC (ByVal Channel As Long, ByVal Pos As Double)	183
Function	McGetCountA (ByVal Channel As Long) As Long	184
Sub	McSetCountA (ByVal Channel As Long, ByVal Count As Long)	185
Function	McGetCountC (ByVal Channel As Long) As Long	186
Sub	McSetCountC (ByVal Channel As Long, ByVal Count As Long)	187
Function	McGetCountD (ByVal Channel As Long) As Long	188
Function	McSetCountD (ByVal Channel As Long, ByVal Count As Long) As Long	189
Function	McGetCountG (ByVal Channel As Long) As Long	190
Sub	McSetCountG (ByVal Channel As Long, ByVal Count As Long)	191
Function	McGetCountR (ByVal Channel As Long) As Long	192
Sub	McSetCountR (ByVal Channel As Long, ByVal Count As Long)	193
Function	McGetMotionStatus (ByVal Channel As Long) As Long	194
Function	McGetMioStatus (ByVal Channel As Long) As Long	195

모션제어 (I/O(입출력) 환경설정)		Page
Sub	McSetMioCfgALM (ByVal Channel As Long, AlarmLogic As Long, AlarmMode As Long)	198
Sub	McGetMioCfgALM (ByVal Channel As Long, ByRef AlarmLogic As Long, ByRef AlarmMode As Long)	199
Sub	McSetMioCfgEL (ByVal Channel As Long, ByVal EIMode As Long)	200
Sub	McGetMioCfgEL (ByVal Channel As Long, ByRef EIMode As Long)	201
Sub	McSetMioCfgINP (ByVal Channel As Long, ByVal InpEnable As Integer, ByVal InpLogic As Long)	202
Sub	McGetMioCfgINP (ByVal Channel As Long, ByRef pInpEnable As Long, ByRef InpLogic As Long)	203
Sub	McSetMioCfgERC (ByVal Channel As Long, ByVal ErcLogic As Long, ByVal ErcOnTime As Long)	204
Sub	McGetMioCfgERC (ByVal Channel As Long, ByRef ErcLogic As Long, ByRef ErcOnTime As Long)	205

Appendix A 라이브러리 메소드 리스트

Sub	McSetMioCfgSD (ByVal Channel As Long, ByVal SdEnable As Integer, ByVal SdLogic As Long, ByVal SdLatch As Long, ByVal SdMode As Long)	206
Sub	McGetMioCfgSD (ByVal Channel As Long, ByRef SdEnable As Integer, ByRef SdLogic As Long, ByRef SdLatch As Long, ByRef SdMode As Long)	207
Sub	McSetSoftLimit (ByVal Channel As Long, ByVal LimitP As Double, ByVal LimitN As Double)	208
Sub	McEnableSoftLimit (ByVal Channel As Long)	209
Sub	McDisableSoftLimit (ByVal Channel As Long)	210
Sub	McSetErrorCompare (ByVal Channel As Long, ByVal Tol As Double, ByVal Enable As Long)	211
Sub	McSetGeneralCompare (ByVal Channel As Long, ByVal CmpSrc As Long, ByVal CmpMethod As Long, ByVal CmpAction As Long, ByVal Data As Double)	212

모션제어 (인터럽트 관련 메소드)		Page
Sub	McMaskInterrupt (ByVal Channel As Long, ByVal Mask As Long)	214
Sub	McEnableInterrupt (ByVal EventHandle As Long)	215
Sub	McDisableInterrupt	216
Function	McGetAxisIntState (ByVal Channel As Long) As Boolean	217
Sub	McGetIntStatus (ByVal Channel As Long, ByRef ErrorStatus As Long, ByRef EventStatus As Long)	218

A.2 메소드별 지원 가능 디바이스 리스트

메소드명	각 보드별 지원 여부			
	ST201	ST301	ST401	ST502
LoadDevice	V	V	V	V
UnloadDevice	V	V	V	V
AdSetInputType	V			
AdSetRange	V			
AdGetDigit	V			
AdGetVolt	V			
DaSetRange		V		
DaOut		V		
DiGetOne				V
DiGetAll				V
DoPutOne				V
DoPutAll				V
SdioInitComm			V	
SdioCheckModule			V	
SdioSetDioUsage			V	
SdioReadLowByte			V	
SdioReadHighByte			V	
SdioWriteLowByte			V	
SdioWriteHighByte			V	

※ 모션에 관련된 메소드는 COMI-ST501에만 적용 가능하며 본 리스트에는 생략되었습니다.