COMIZOA

LX520 Software Development Kit

TEST & MEASUREMENT & AUTOMATION

COMIZOA LX520 API REFERENCE MANUAL

FEB 2014
P/N 0221-2014-02

API Reference Manual

ComiRTEX Manual

Visit our web page at http://www.comizoa.com
For support requests, contact us at csteam@comizoa.com
For documentation suggestions, corrections, or requests, contact csteam@comizoa.com

고객(顧客) 기술 지원

전자 우편 : csteam@comizoa.com
파일 서버 : ftp.comizoa.com
웹 사이트 : http://www.comizoa.com

본사 안내
대전 광역시 유성구 테크노 2 로 314 번지(탑립동 914 번지)

TEL : 042-936-6500
FAX : 042-936-6507

COMIZOA RTEX Systcmx Integrated Control Library Reference

Table of Contents

Chapter

1

Introduction

본 장에서는 제품 보증안내를 비롯한 제품 보증 규정, 저작권, 상표안내, 주의사항을 설명하고 있습니다. 다양한 모션 제어 환경에서 보다 강력하고 빠른 모션 제어를 위해, 그리고 안정적인 모션을 위해 고객(顧客)님께서 선택하신 저희 ㈜커미조아 제품은 이제 고객(顧客)님에게 큰 감사와 보답으로 이바지 하도록 하겠습니다.

가장 안정적이고, 빠르고 정확한 모션, 그리고 고객(顧客) 중심에서의 제품의 완성을 추구하는 ㈜커미조아는 고객(顧客)님들을 위해 언제나 최선과 정성을 다하는 자세로 지금 이 시간에도 보다 나은 제품 개발을 위해 성실과 열정을 바탕으로 제품에 꿈을 담고 있습니다.

먼저 고객(顧客)님들께, 국내 최고의 모션 기술력을 자랑하는 저희 ㈜커미조아 제품을 선택하여 주신 여러분들께 다시 한번 감사의 말씀을 드리며, 본 장에서는 제품 보증안내, 규정, 저작권, 상표 안내에 대한 내용을 설명하고 있습니다. 본 장에서 설명드리는 내용은 RTEX 운용과는 별개의 내용이오나, 제품의 보증과 규정 그리고 저작권에 대한 내용을 설명 드리고 있으므로, 반드시 습득하여 주시기 바랍니다.

1 ComiRTEX 사전 안내 사항

1.1 Overview

1.1.1 제품 보증 안내

저희 ㈜커미조아는 고객(顧客) 여러분들께 가장 안정된 소프트웨어와 하드웨어를 공급함으로써, 고객(顧客) 여러분들을 만족시켜드리는 것을 최우선의 목표로 하고 있습니다. 만약 구입하신 제품에 외관상의 하자, 동작이상 또는 불량이 발견되는 경우에는 언제든지 저희 ㈜커미조아를 통해 문의(問議)해주시기를 바라며, 가까운 대리점 혹은 총판점을 통해 구입하신 경우에는 해당 구입처(購買處)를 통해 문의하시면, 더욱 빠른 기술 지원을 받으실 수 있습니다.

1.1.2 제품 보증 규정

구입하신 당사의 제품은 소비자의 과실 이외의 자체 결함 및 동작이상에 대해 2 년간 그 전체 혹은 일부에 대해서 보증하고 있습니다. 당사의 제품에 대한 자세한 제품 보증 규정은 별도로 관리되는 각 제품의 '제품 보증 규정' 에 의거하며, 자세한 보증 규정을 알기 원하시는 경우 ㈜커미조아 혹은 총판점(總販店) 및 대리점(代理店) 등 해당 구입처를 통해 문의해 주시기 바랍니다.

1.1.3 저작권

1.1.4 상표안내

1.1.5 주의사항

㈜커미조아의 제품군에는 제품의 특성(特性)에 따라서 하드웨어 및 소프트웨어 기술 지원이 필요한 경우가 있습니다. 필요하신 경우 본사 혹은 총판 및 대리점을 통해 제품 구입 이전에 점검 또는 요청해주시기 바랍니다.

㈜커미조아의 소프트웨어 및 하드웨어 제품군은 제품 성능 향상을 위해 예고 없이 변경될 수 있습니다. 고객(顧客)님께서는 본 매뉴얼을 읽기전에 하드웨어 및 소프트웨어의 최신 변경사항에 대한 정보를 ㈜커미조아를 통해 요청하실 수 있습니다.

본 매뉴얼은 ㈜커미조아 ComiRTEX 에 대한 정보를 포함한, 실제 라이브러리를 통한 모션 제어와 범용 디지털 입출력 제품군의 제어 사항에 대한 기반 설명을 포함하고 있습니다.

최신 내용 및 기술되지 않은 사항 혹은 누락된 사항은 ㈜커미조아 제품군 구입시에 고객(顧客) 등록을 해주신 고객(顧客)님의 정보를 통해 안내해 드릴 예정이며, 기술 지원 요청시에 보다 자세하고 정확한 방법과 내용을 통해 도움 드릴 것을 약속드립니다.

본 매뉴얼에 시작하기 앞서, 본 장(Chapter) 에서는 본 매뉴얼을 보다 정확하게 빠르게 이해(理解)하실 수 있도록 ComiRTEX 의 전체 구성과 형식에 대해서 안내해 드리겠습니다.

1.1.6 매뉴얼 용어 안내

본 매뉴얼에서는 필요에 따라 매뉴얼에 기술된 내용을 설명하기 위해 함축된 의미의 용어나 현장 용어를 사용할 수도 있습니다. 최대한 보충 설명이 필요한 내용에 대해서는 해당 단원(Chapter)에서 설명하도록 하겠습니다.

매뉴얼 전체 범위에서 사용되는 범용적인 용어의 의미는 다음 표 1 매뉴얼 용어 정리 같이 미리 안내하여 드립니다.

명칭	의미
모터 (Motor)	서보모터를 비롯한 스탭 모터
서보팩 (Servo Pack)	서보 앰프
서보 드라이브 (Servo Drive)	서보모터와 서보앰프의 편성
스탭 드라이브 (Step Drive)	스탭모터와 스탭 모터 제어회로의 편성
서보 시스템 (Servo Systcmx)	서보 드라이브와 ㈜커미조아 상위 제어기를 조합한, 일련의 완성된 시스템
지령 위치 출력 신호 (Command Pulse)	㈜커미조아 모션 제어 제품에서 RTEX 네트워크를 통해 서보 팩 혹은 스탭 드라이브 측으로 전송하는 지령 위치 데이터
실제 위치 입력 신호 (Feedback Pulse)	㈜커미조아 모션 제어 제품에서 RTEX 네트워크를 통해 서보 팩 혹은 스탭 드라이브 측의 위치 검출기(Encoder) 로부터 수신한 실제 위치 데이터
RTEX NC (RTEX Network Controller)	RTEX 네트워크 제어를 위한 본 제품을 일컫는 명칭

표 1 매뉴얼 용어 정리

1.1.7 매뉴얼 아이콘의 설명

매뉴얼에서 안내되는 내용을 보다 신속하고 정확하게 알 수 있도록 하며, 그 의미가 바르게 전달 되고 이해(理解)를 돕기 위해 원하는 의미에서 아래와 같은 매뉴얼 아이콘을 사용하고 있습니다.

	이해(理解)하기 어려운 용어나 보충(補充)이 필요한 용어, 해설 및 사전에 설명 없이 새로 나온 용어를 설명합니다.
	고객님의 편의와 기능의 자세한 내용에 대해 부가적으로 안내되는 사항입니다.
	이 동작이나 실행 함수(Function) 에 있어서 그 동작의 주의를 요망하는 내용을 나타냅니다.
	이 동작이나 실행 함수(Function) 에 있어서 그 동작의 이상이나 문제가 발생할 경우 이 사항에 대해서 경고의 의미를 나타냅니다.

표 2 매뉴얼 아이콘의 설명

1.2 Features

㈜커미조아의 RTEX 라이브러리인 ComiRTEX 의 장점은 다음과 같습니다.

1.2.1 독립성

ComiRTEX 는 Microsoft 社의 표준라이브러리인 DLL(Dynamic Link Library) 형태의 독립된 라이브러리 인터페이스를 제공하며, 라이브러리 자체의 독립된 장치 관리의 편의성을 제공하고 있으며, DLL 형태의 라이브러리 장점을 통해 유지 보수와 귀사의 제품 구현에 보다 간편하게 하고 신뢰성 있는 독립형 동적 연결 라이브러리를 제공합니다

1.2.2 호환성

우수한 소프트웨어 개발 도구를 이용하여 전통적인 개발 방법보다 더 적은 시간과 비용으로 더 좋은 품질의 소프트웨어를 개발하는 방법을 이야기하는 최신 RAD(Rapid Application Development) 를 지향하고 있으며, 이에 맞는 최신 소프트웨어 개발 환경을 지원하고 있습니다.

고객(顧客)님께서는 언제나 ComiRTEX 를 통하여 귀사의 제품에 보다 신속하고 정확한 시스템을 구현하실 수 있습니다.

1.2.3 편의성

인터페이스 함수 명명 규칙의 통일화와 의사 코드 주제어를 매뉴얼과 라이브러리 인자(Parameter) 에 부각시켜, 보다 빠른 시간내에 숙련된 라이브러리 사용자로 만들어드립니다. 특히, ComiRTEX 의 모든 함수 명에는 의미적 명명 규칙을 내포하였으며, 이것은 분명 실무 개발 환경에서 많은 부분 이점으로 작용할 것입니다.

1.2.4 신뢰성

㈜커미조아의 라이브러리 제품군은 오랜 시간 산업현장에서의 현장 경험을 바탕으로 형성된 신뢰성있는 제품군입니다. '부드럽고, 정확하고, 빠르고, 쉬운' 모션제어 기능을 통해 모션의 기본에 충실하였으며, 각 동작에 대한 입력과 출력을 ComiRTEX 인터페이스 전역에 걸쳐 사용하기 쉽도록 안내해 놓았습니다.

또한, 응용프로그램에 기반하지 않는 자체 디버그 기능을 바탕으로 응용프로그램에 의한 오류를 최단시간내에 해결할 수 있도록 합니다. 디버그 모드의 지원과 향상된 디버그 정보를 바탕으로 오류발생에 대한 원인을 신속히 분석하실 수 있도록 도움을 드리며, 저희 ㈜커미조아는 고객(顧客)님께 항상 정직과 신뢰를 드릴 수 있도록 최선을 다할 것입니다.

1.2.5 풍부한 예제와 신속한 기술 지원

지금 이시간에도 ㈜커미조아는 타사에 비해 보다 많은 개발 선상의 활용가능한 라이브러리 예제를 지원해드리려고 노력하고 있으며, 예제간 중복 코드와 라이브러리 구성 이외의 부분을 최소화한 예제로 빠른 시간내에 예제의 코드를 바로 적용시켜드릴 수 있도록 노력하고 있습니다.

저희 ㈜커미조아는 최신 .NET Framework 상의 C #(Sharp) 부터, Visual Basic 및 현존하는 RAD(Rapid Application Development) 환경을 지원하는 ComiRTEX 로 보다 향상된 고객(顧客) 지원과 기술 지원을 통해 고객(顧客)여러분들의 성원에 이바지하겠습니다.

Chapter

2

Before working with ComiRTEX

정밀한 모션제어와 고속의 모션제어는 최상의 모션 제어기가 반드시 갖춰야할 필수요건입니다. 안정적인 하드웨어와 더불어 고급 기능의 소프트웨어는 고객(顧客)여러분들의 최상의 응용 프로그램 구현을 돕고 있습니다. 이제 더 이상 고민하지 마십시요. ComiRTEX 는 보다 견고하고 우수한 기능을 바탕으로 여러분들이 원하시는 기능을 보다 빠르고 정확하게 안내하여 드릴 것입니다.

본 장에서는 RTEX 에 대한 정의와 저희 ㈜커미조아의 ComiRTEX 에 대한 안내를 위한 내용으로 구성되어 있습니다. 보다 신속한 기술 지원과 빠른 라이브러리 기능 습득(習得)을 위해 구성(構成)되었으며, 사전 안내사항부터 함수 이름 규칙 가이드까지 모든 내용을 쉽고 빠르게 이해(理解)하실 수 있도록 최선을 다하였습니다.

2 Panasonic Real Time Express® & ComiRTEX

본 장에서는 RTEX 에 대한 사항과 네트워크 제어기를 자세히 안내해 드리고 있습니다.

2.1 Panasonic RTEX 소개

RTEX 는 일본 Panasonic 社에서 개발한 네트워크의 한 종류로서 물리적인 이더넷 계층(P(Physical Layer)을기반하에 상위 계층을 구성하고 있으며, 고속의 하드웨어 처리를 통해 보다 안정적이고 신뢰성있는 산업용네트워크를 실현하였습니다.

Panasonic RTEX 의 상위 제어기 ㈜ 커미조아의 LX520 제품은 범용적인 이더넷 물리계층에서의 연결은 표준이더넷 케이블을 통해 자유롭게 구성할 수 있으며, 타 社의 산업용 네트워크의 데이터 전달과 전송을 위한소프트웨어적인 처리의 취약점에 있어 네트워크 구성간에 응용계층의 전체 구간을 고속의 하드웨어로 처리하게 됩니다

2.2 ComiRTEX 라이브러리 주요 내용

㈜ 커미조아의 LX520 은 전 세계적으로 가장 많은 사용자 층을 확보하고 있는 마이크로소프트 윈도우용 라이브러리를 제공하고 있습니다. 제공되는 라이브러리인 ComiRTEX 의 다양한 기능을 통해 RTEX 네트워크의 상위 네트워크 제어기(Network Controller)인 LX520 의 모든 기능을 경험하실 수 있습니다.

2.2.1 ComiRTEX 개요 사항

ComiRTEX 는 소프트웨어는 윈도우 표준 동적 라이브러리(Dynamic Link Library) 형태로 고객 여러분들께 제공되며, 마이크로소프트의 Visual C++ , Visual Basic .NET Framework 환경의 Visual C#, Visual J# 개발환경, 볼랜드 C++ Builder 전제품, Delphi 전제품 인텔 컴파일러, 기타 Visual Fortan 등의 다양한 개발환경을 지원하고 있습니다.

2.2.2 함수 이름 규칙 가이드

구분 명		ComiRTEX
장치 초기화 (Device Initialize)		cmxGnLoadDevice
		cmxGnUnLoadDevice
MOTION	단축 제어 (Single Axis)	cmxSxMoveStart
		cmxSxMoveToStart
	다축 제어 (Multi Axes)	cmxMxMoveStart
		cmxMxMoveToStart
	보간 제어 (Interpolation)	cmxIxMapAxes
		cmxIxLine
	원점 복귀 (Home Return)	cmxHomeSetConfig
		cmxHomeMove
	리스트 모션 (Listed Motion)	cmxLmxStart
		cmxLmxEnd
	확장보간 제어 (Extended Int.)	cmxIxHelOnceStart
		cmxIxSplineStart

표 3 함수이름 규칙

- 본 지면 상 전체 함수에 대해서 다루지 못한 부분은 각 제품 매뉴얼을 참조해주시기 바랍니다.

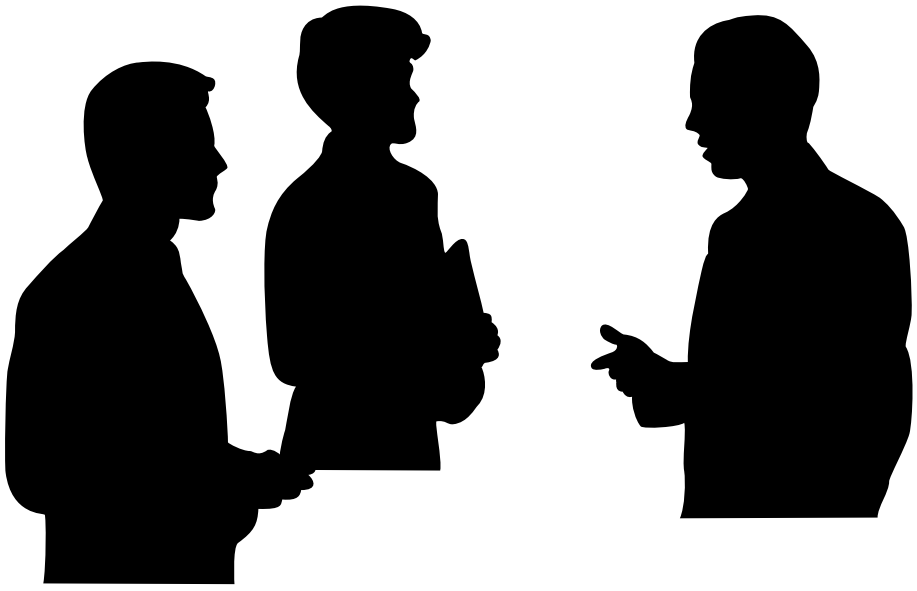
Chapter

3

Development Environment for ComiRTEX

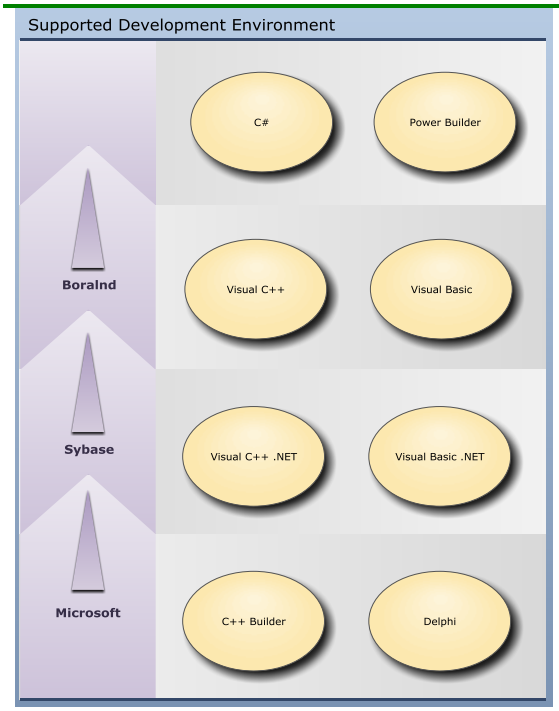
㈜커미조아는 통합 라이브러리 ComiRTEX 를 통해 다양한 최신 개발환경을 지원하기 위해 노력하고 있습니다. 본 장에서 다루지 않는 개발 환경을 이용하시는 고객(顧客)님께서는 저희 ㈜커미조아를 통해 문의해주시면 신속히 대처해 드리도록 하겠으며, 제공되는 ComiRTEX 인터페이스를 통해 보다 편리하고 빠르게 저희 라이브러리에서 사용할 수 있도록 지원하여 드립니다.

커미조아 모션 및 범용 디지털 ComiRTEX 는 다양한 고객(顧客) 여러분들의 요구에 발맞추어 개발되었습니다., 가까운 대리점 혹은 총판점을 통해 구입(購入)하신 경우에는 해당 구입처(購入處)를 통해 문의하시면, 더욱 빠른 기술 지원을 받으실 수 있습니다.



3 개발 환경 별 ComiRTEX 사용 안내

3.1 개발 환경 지원 안내



개발 환경 별	언어별	버전 별	제품 별
Microsoft 社 Visual Studio	Visual C++	6.0 및 이하 버전을 포함한 VS2003, VS2005 환경	Enterprise Edition 을 포함한 전제품
	Visual Basic		
	C# (Sharp)		
Borland International 社 Borland Development Studio	C++ Builder	Builder 5 , Builder 6 및 하위 버전 포함한 2006 버전	Architect, Professional, Enterprise
	Delphi	Delphi 5 , Delphi 6, Delphi 7 및 하위 버전 포함한 2006 버전	
	C# (Sharp)	BDS 2006 버전	
Borland International 社 Borland Turbo Series	C++ Builder	Turbo C++	전 제품
	Delphi	Turbo Delphi / Turbo Delphi for .NET	전 제품
	C# (Sharp)	Turbo C#	전 제품
Sybase PowerBuilder	PowerBuilder	PowerBuilder 8, PowerBuilder 9, PowerBuilder 10.5	전 제품

표 4 지원 개발 환경

ComiRTEX 는 위와 같은 개발 환경을 지원하며, 별도로 명시되지 않은 개발 환경에서도 윈도우의 Dynamic Link Library 형태를 사용가능한 경우 ComiRTEX 를 사용하실 수 있습니다. 이 사항에 대해서 더욱 자세한 내용을 알기 원하실 경우 Appendix(부록) 편을 참고해 주시기 바랍니다.

3.2 ComiRTEX 구성

ComiRTEX 는 아래 그림과 같이 실제 COMIZOA 제품군의 LX520 하드웨어를 추상화하여, 공통(共通) API 구조를 제공하고 있습니다.

ComiRTEX 는 "Integration RTEX Systcmx Control Application Programming Interface" 라 명명하며, 그 의미는 저희 ㈜커미조아 제품군의 기능과 그 기능을 사용하는 방법을 정의한 함수(Function)의 집합이라고 말할 수 있습니다. 고객(顧客)님께서는 이제 저희 ㈜커미조아 API 를 통해 제품군이 가지고 있는 다양한 기능을 사용할 수 있습니다.

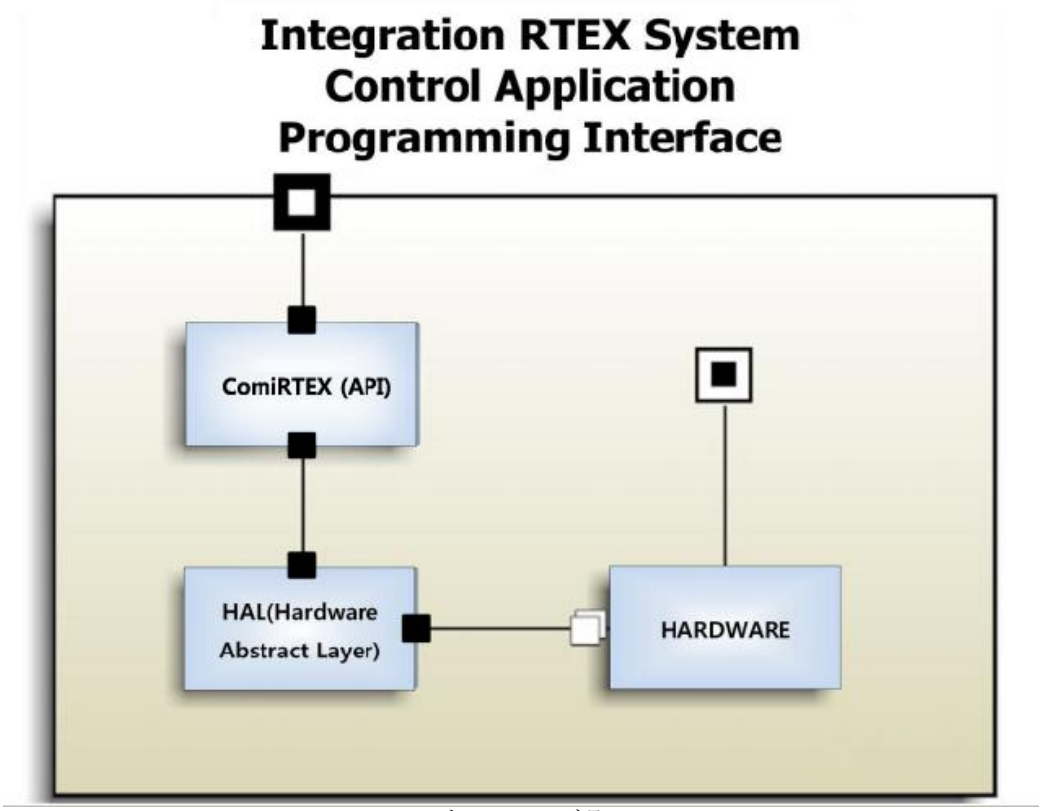


그림 3-1 ComiRTEX 의 구조

3.2.1 HARDWARE Layer

다양한 커미조아의 제품군은 개별적인 인터페이스를 필요로 합니다. 그러나 이제 하드웨어 구조를 추상화하여, 일관적인 API 인터페이스를 제공하게 해주는 ComiRTEX 를 통해 편리하게 모든 모션 장치를 일관적으로 제어하실 수 있습니다.

3.2.2 HAL(Hardware Abstract Layer)

Hardware Device 는 ComiRTEX 가 지원하는 자사의 하드웨어 장치를 총체적으로 이르며, 크게 모션 장치와 범용 디지털 입출력 장치가 있습니다. 특히 모션 장치와 범용 디지털 입출력 장치는 고객(顧客)님께서 개발하는 응용 프로그램 계층과의 인터페이스인 ComiRTEX 에서 총괄하고 있으며, 최상의 속도와 안정적인 동작에 대한 보장을 위한 다양한 기법으로 구조화되어 있습니다. 고객(顧客)님께서는 COMIZOA 의 어떠한 모션 보드나 범용 디지털 입출력 장치에 대해서 단 하나의 API 를 통해 제어하실 수 있습니다. 구조의 바탕은 바로 ComiRTEX 가 고급 HAL(Hardware Abstract Layer) 기능을 바탕화 했기 때문입니다.

3.2.3 ComiRTEX Layer (API Layer)

마이크로소프트社의 Windows 98/ME/2000/XP/Vista 등의 제품군에서 동적 연결 라이브러리 방식을 지원하는 라이브러리를 의미합니다. 사용자는 Integration RTEX Systemx Control API, 즉 ComiRTEX 를 통해 제품 개별적인 하드웨어 인터페이스에 대한 사항들을 사용자는 직접 인지하고 있지 않아도 ComiRTEX 는 고객(顧客)여러분들의 요구에 맞는 인터페이스를 제공하며, 신속하고 안정적인 응용프로그램 개발에 도움을 드립니다.

3.2.4 ComiRTEX 인터페이스 구성 파일

<table>
<tr><th>개발 환경
항목</th><th>MS VC++</th><th>Borland
C++ Builder</th><th>Borland Delphi</th><th>MS Visual
Basic</th><th>MS C# (C
Sharp)</th></tr>
<tr><td rowspan="2">ComiRTEX 인터페이스
파일 명</td><td>ComiRTEX
_SDK.h</td><td>ComiRTEX
_SDK.h</td><td rowspan="3">ComiRTEX
.PAS</td><td rowspan="3">ComiRTEX
.BAS</td><td rowspan="3">ComiRTEX
.CS</td></tr>
<tr><td>ComiRTEX
_SDK.cpp</td><td>ComiRTEX
_SDK.cpp</td></tr>
<tr><td>ComiRTEX
상수(常數)정의 파일</td><td>ComiRTEX
_SDK_Def.h</td><td>ComiRTEX
_SDK_Def.h</td></tr>
</table>

표 5 ComiRTEX 인터페이스 구성 파일 안내

Microsoft 社의 윈도우 운영체제에서 동작하는 DLL(Dynamic Link Library) 형태의 ComiRTEX 는 상기 표에서 나타낸 것 처럼, 라이브러리 인터페이스를 위해 "ComiRTEX 인터페이스" 파일과 라이브러리의 반환값 및 전달인자, 각 데이터 표기등을 위한 "ComiRTEX 상수(常數) 정의 파일"을 제공하고 있습니다.

㈜커미조아 고객(顧客)님께서는 "ComiRTEX 인터페이스 파일" 을 통해 ComiRTEX 의 함수를 명시적으로 응용프로그램에 포함시킬 수 있으며, 이에 따라 상기 표기한 개발 환경을 지원해드리고 있습니다. 만약 이외의 환경에서 저희 ComiRTEX 를 사용하시기 원하신다면, 저희 ㈜커미조아 고객(顧客) 지원 팀으로 연락 주시기 바랍니다.

3.3 각 개발 환경 별 안내

저희 ㈜커미조아에서는 고객(顧客)님들의 개발환경에 대한 고민을 덜어드리기 위하여, 범용적인 객체지향 언어인 C++ 부터 Borland International 社의 Object Pascal 의 Delphi 제품군, 그리고 **최신 .NET 환경까지 고려한**

라이브러리 인터페이스를 제공하고 있습니다. 특히 이번 ComiRTEX 에서는 개발 환경의 새 패러다임인 C Sharp(C#) 언어를 본격 지원하고 있으며, 보다 새로운 개발환경에서 저희 ㈜커미조아 제품 군을 경험하실 수 있도록 만전을 기하고 있습니다.

필요한 인터페이스(Interface) 파일은 실제 개발언어 및 환경에서 Header 파일또는 프로젝트 구성파일로 반드시 사용이 됩니다. 저희 ㈜커미조아 ComiRTEX 에서 제공하는 인터페이스파일의 경로는 다음과 같습니다.

Visual C++ / Borland C++ Builder	
경로 명	C:\Program Files\COMIZOA\AUTOMATION3\NEMO\Libraries\x86\Motion(DLL)\RTEX_LX520\VC++
파일 명	ComiRTEX_SDK.cpp, ComiRTEX_SDK.h, ComiRTEX_SDK_Def.h
Delphi	
경로 명	C:\Program Files\COMIZOA\AUTOMATION3\NEMO\Libraries\x86\Motion(DLL)\RTEX_LX520\Del
파일 명	ComiRTEX.PAS
Visual Basic	
경로 명	C:\Program Files\COMIZOA\AUTOMATION3\NEMO\Libraries\x86\Motion(DLL)\RTEX_LX520\VB
파일 명	ComiRTEX.BAS
C# (CSharp)	
경로 명	C:\Program Files\COMIZOA\AUTOMATION3\NEMO\Libraries\x86\Motion(DLL)\RTEX_LX520\CSharp
파일 명	ComiRTEX.cs

표 6 이 경로는 ㈜커미조아에서 제공하는 COMI-AUTOMATION 설치 프로그램이 기본 경로인'C:' 드라이브에 설치된 것을 기준으로 합니다.

인터페이스 파일에 대해서 자세히 설명해 주십시요.

먼저, 저희 ㈜커미조아에서 제공하는 ComiRTEX 는 DLL 형태의 독립 라이브러리 인터페이스를 제공하는 마이크로소프트웨어의 윈도우 표준 라이브러리를 말합니다. 이것은 당사의 제품의 기능을 담당하는 가장 중요한 함수의 집합 구성입니다.

고객(顧客)님들께서는 이러한 함수, 즉 기능을 통해서 저희 ㈜커미조아의 제품 군을 이용하시게 됩니다.
그러나 마이크로소프트의 DLL 라이브러리 형태는 구조상 제공하는 함수들을 어떻게 써야 하는 지의 정보를 정확하게 알 수 없습니다. 다시 말씀 드려서, 함수의 매개변수의 개수나 매개변수의 형태 즉, 함수의 사용법을 알 수가 없습니다.

ComiRTEX 에 대해서 그 사용방법을 각 개발환경에 맞게 제공해드리기 위해 [인터페이스] 파일을 제공하고 있습니다. 각 개발환경(VC++, Delphi 등)에서 개발을 하시는 고객(顧客) 여러분들께서는 반드시 이 [인터페이스] 파일을 사용하셔야만, ComiRTEX 를 사용할 수 있습니다.

3.3.1 Visual C++ 6.x 개발자를 위한 안내

Microsoft Visual C++ 6.x 에서 ComiRTEX 를 사용하시려면 다음의 절차에 따라 사용하시면 됩니다.

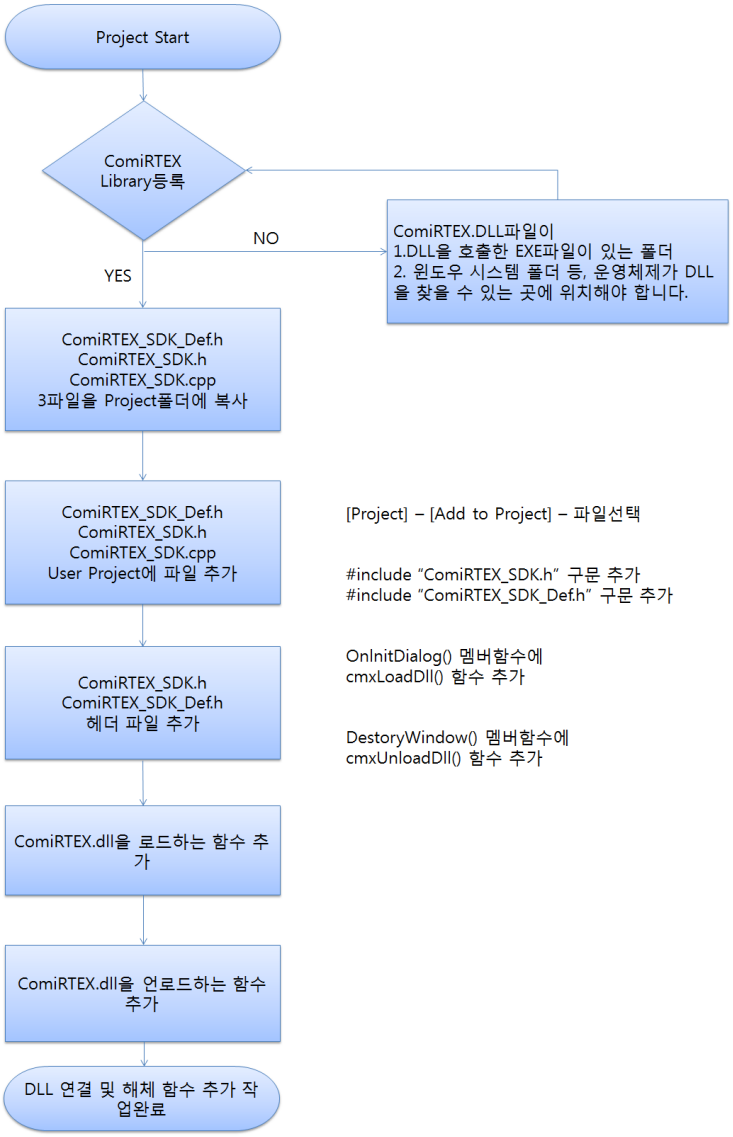


그림 3-2 Visual Studio 6.x 에서의 ComiRTEX 사용 순서도

Visual C++ 6.x 을 실행합니다. 메뉴에서 'File'->'New' 를 선택하여 새로운 프로젝트 생성을 시작합니다.

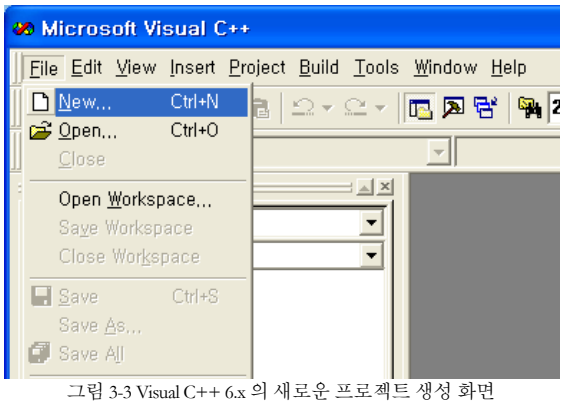


그림 3-3 Visual C++ 6.x 의 새로운 프로젝트 생성 화면

MFC AppWizard(exe)를 선택하고, 프로젝트를 생성할 위치와 프로젝트 이름을 입력한 후 [OK]버튼을 클릭 합니다.

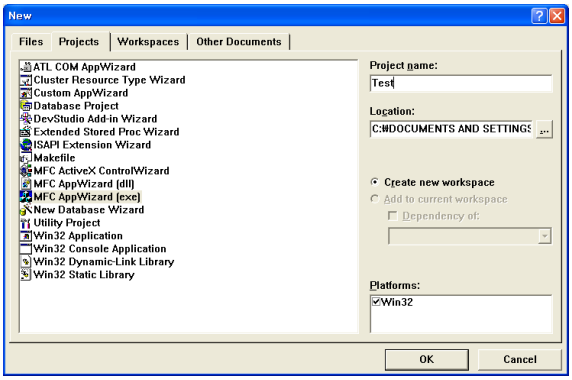


그림 3-4 새로운 프로젝트 생성 화면

MFC AppWizard 창이 나타나면 [Dialog based]를 선택하고 [Finish]버튼을 클릭합니다.

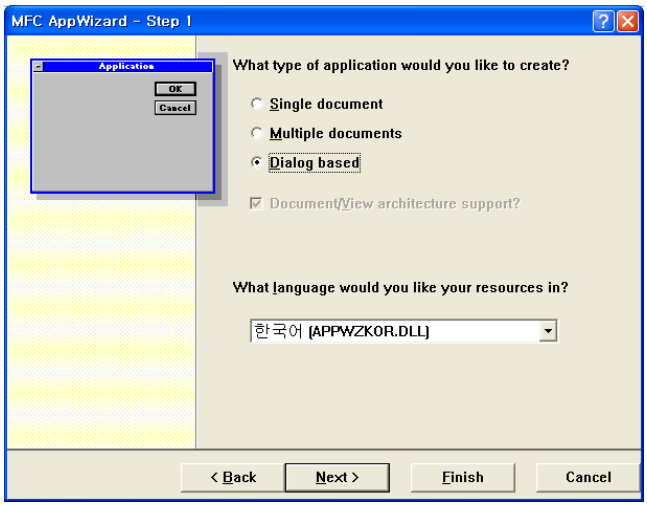


그림 3-5 MFC AppWizard 의 Application Type 선택 화면

VC++ 용 인터페이스 정의 파일인 ComiRTEX_SDK.h, ComiRTEX_SDK.cpp, ComiRTEX_SDK_Def.h 파일을 신규로 생성한 프로젝트 폴더로 복사 합니다.

메뉴에서 [Project]->[Add To Project]->[Files]를 선택합니다.

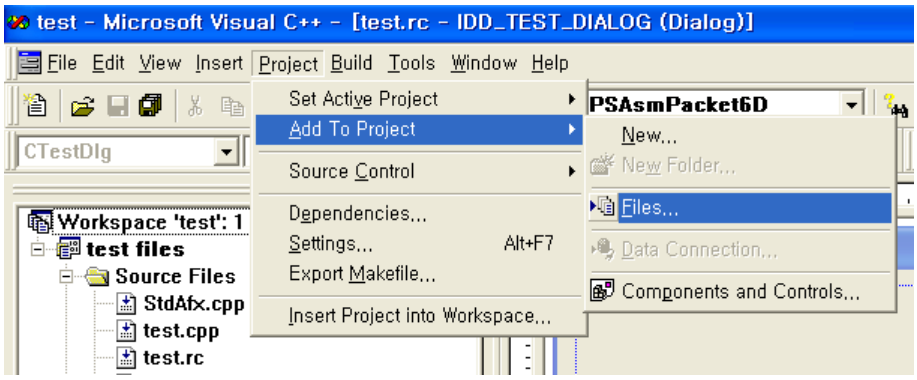


그림 3-6 프로젝트에 새로운 파일 추가 선택화면

추가될 파일을 선택한 후 [OK]버튼을 클릭하여 통합 모션 라이브러리 인터페이스 파일인 세 개의 파일을 프로젝트에 추가 합니다.

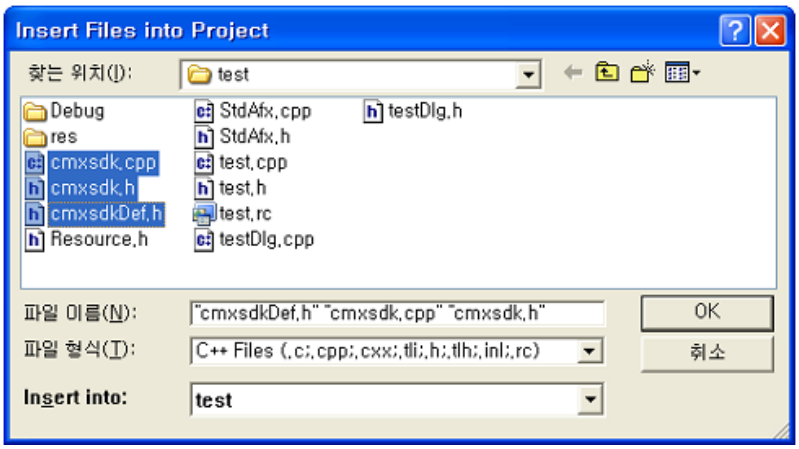


그림 3-7 프로젝트에 추가할 파일 선택 화면

WorkSpace 창의 FileView 탭에서 [생성한 프로젝트 이름]+Dlg.cpp 파일을 선택합니다.

그림 3-8 사용자가 MFC AppWizard 를 통해서 생성한 소스코드에 ComiRTEX 파일을 추가 함

([생성한 프로젝트 이름]+Dlg.cpp) 파일의 OnInitDialog()함수 내부의 "TODO"아래에 "cmxLoadDll();"을 추가 합니다.

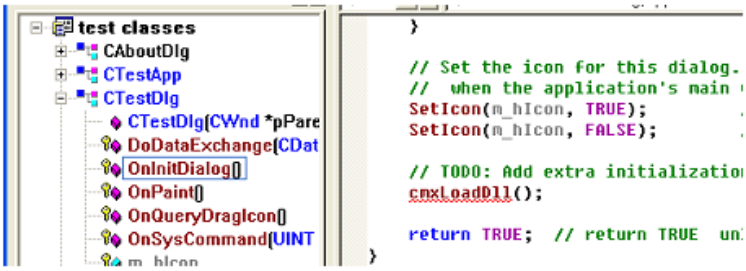


그림 3-9 DLL 로드 함수 호출 화면

사용자 작성 프로그램이 종료되면 DLL 을 Unload 시켜야 합니다. DLL 의 Unload 는 사용자 작성 프로그램의 종료시 이루어져야 하며 cmxUnloadDll()이라는 함수를 통해서 이루어 집니다. cmxUnloadDll()을 추가 하는 방법은 다음과 같습니다.

Class View 창에서 [(생성한 프로젝트 이름)+Dlg] 클래스를 마우스 오른쪽 버튼으로 클릭 합니다. 팝업 메뉴에서 [Add Virtual Function]을 선택합니다.

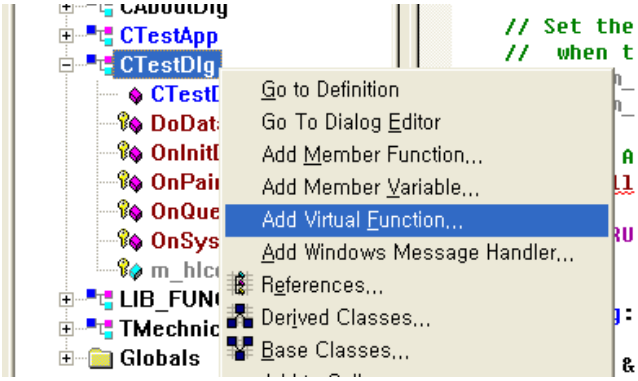


그림 3-10 가상 함수 추가

'New Virtual Functions'항목에서 'DestroyWindow'를 선택한 다음 [Add and Edit]버튼을 클릭합니다.

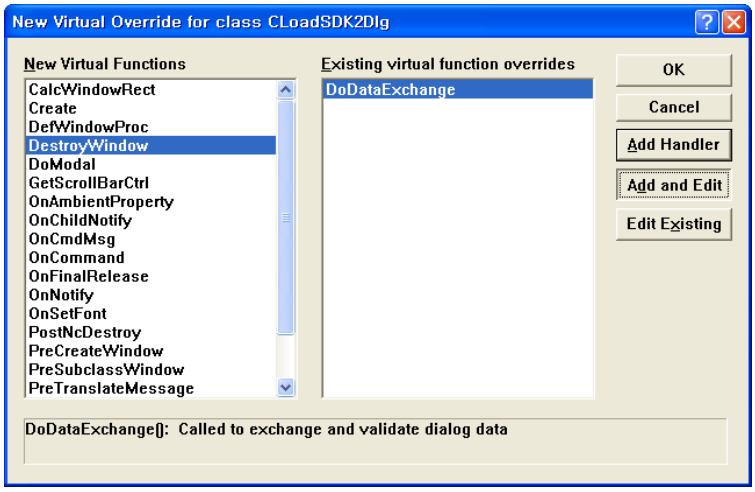


그림 3-11 DestroyWindow 함수 추가

(생성한 프로젝트 이름)+Dlg 클래스의 멤버함수인 'DestroyWindow()'에 'cmxUnloadDll();'을 추가 합니다. 'cmxUnloadDll()'함수를 추가 하면 윈도우가 종료 될 때 자동으로 DLL 도 해제 됩니다.

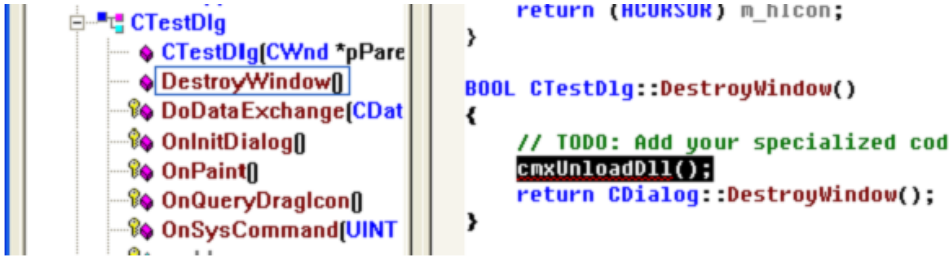


그림 3-12 DLL 로드 및 언로드 코드 추가

3.3.2 Visual C++ 8.x 개발자를 위한 안내

Microsoft 社의 Visual C++ 8.x (Visual Studio 2005)에서 ComiRTEX 를 사용하시려면 다음의 절차에 따라 사용하시면 됩니다.

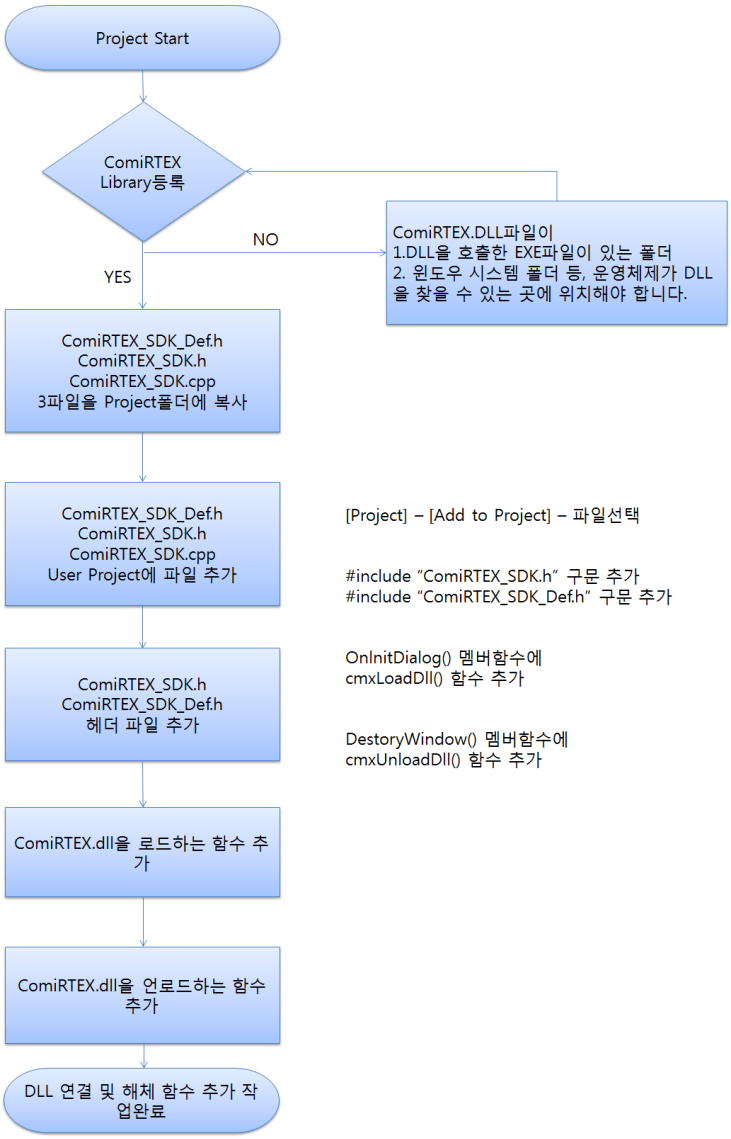


그림 3-13 Visual Studio 8.x 에서의 ComiRTEX 사용 순서도

Microsoft 社의 Visual Studio 2005(이하 VS2005)를 실행 합니다.

메뉴에서 [File]->[New]->[Project]를 선택하여 새로운 프로젝트를 시작 합니다.

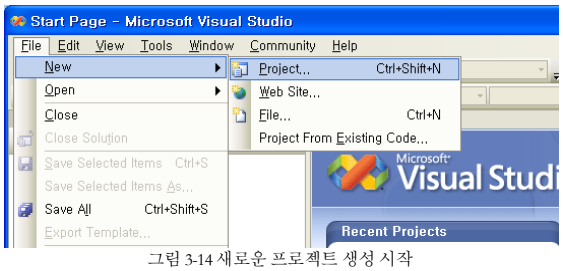


그림 3-14 새로운 프로젝트 생성 시작

[New Project]창이 화면에 나타나면, [Project types]에서는 [Visual C++]을 선택하고, [Tcmxplates]에서 [MFC Application]을 선택합니다. 그리고 프로젝트를 생성할 위치와 프로젝트 이름을 입력한 후 [OK]버튼을 클릭 합니다.

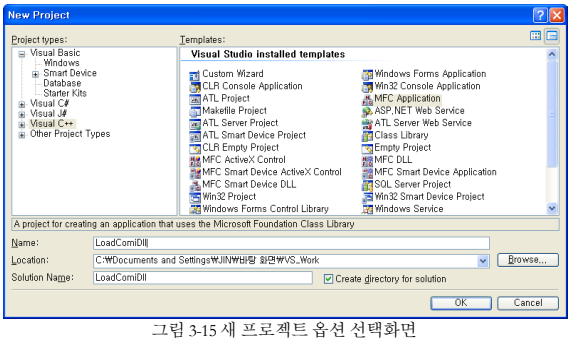


그림 3-15 새 프로젝트 옵션 선택화면

[MFC Application Wizard] 창이 화면에 나타나면, [Next]를 클릭합니다.

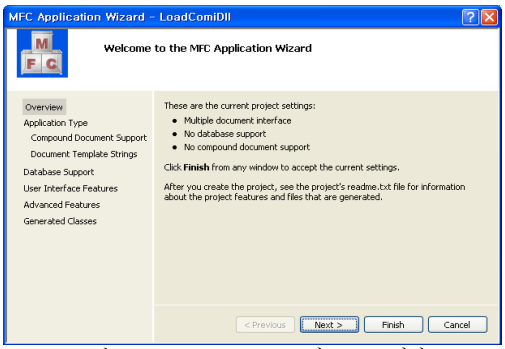


그림 3-16 MFC Application Wizard 의 Overview 화면

[Application Type]에서 [Dialog based]를 선택하고, [Use Unicode Libraries]를 해제(Uncheck) 한 다음 [Finish]를 클릭합니다.

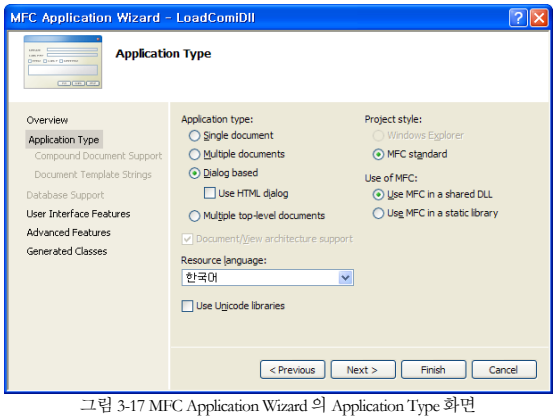


그림 3-17 MFC Application Wizard 의 Application Type 화면

VC++ 용 인터페이스 정의 파일인 ComiRTEX_SDK.h, ComiRTEX_SDK.cpp, ComiRTEX_SDK_Def.h 파일을 신규로 생성한 프로젝트 폴더로 복사 합니다.

메뉴에서 [Project]->[Add Existing Itcmx]을 선택합니다.

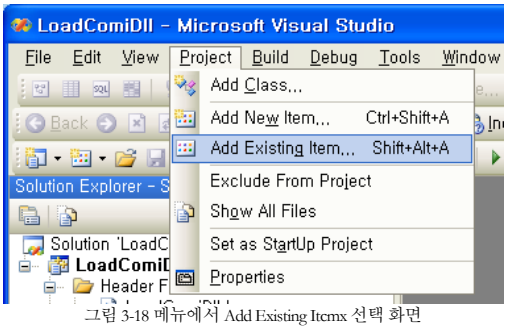


그림 3-18 메뉴에서 Add Existing Itcmx 선택 화면

추가될 파일을 선택한 후 [OK]버튼을 클릭하여 세 개의 파일을 프로젝트에 추가 합니다.

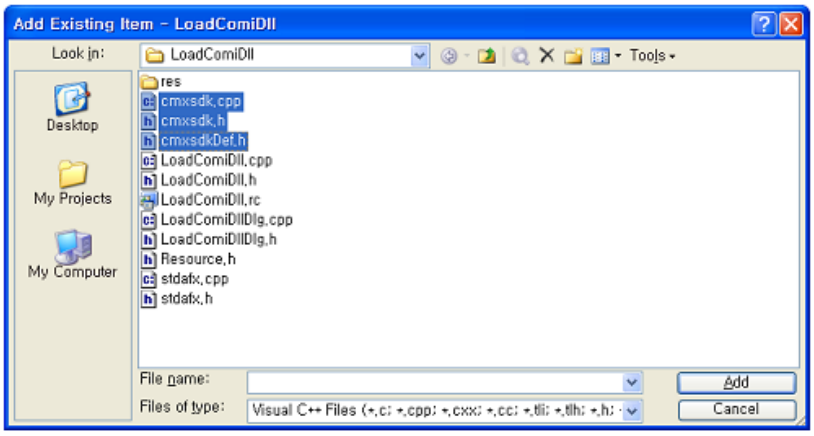


그림 3-19 새로 추가할 파일들 선택 화면

WorkSpace 창의 FileView 탭에서 ([생성한 프로젝트 이름]+Dlg.cpp) 파일을 선택합니다.
파일의 가장 위쪽에 인터페이스 파일을 추가합니다.

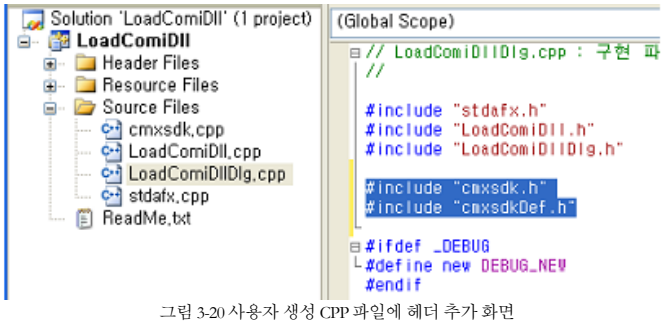


그림 3-20 사용자 생성 CPP 파일에 헤더 추가 화면

(생성한 프로젝트 이름)+Dlg.cpp 파일의 OnInitDialog() 함수 내부의 "TODO"아래에 "cmxLoadDll();"를 추가 합니다.

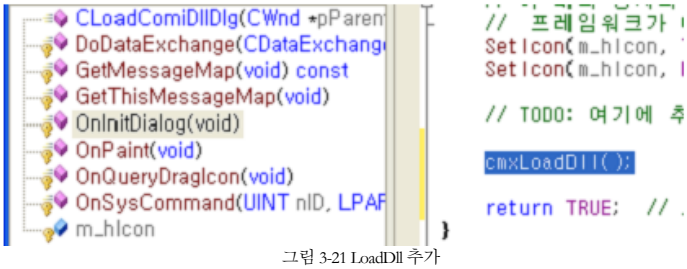


그림 3-21 LoadDll 추가

사용자 작성 프로그램이 종료되면 DLL 을 Unload 시켜야 합니다. DLL 의 Unload 는 사용자 작성 프로그램의 종료시 이루어져야 하며 cmxUnloadDll()이라는 함수를 통해서 이루어 집니다. cmxUnloadDll()을 추가 하는 방법은 다음과 같습니다.

Class View 창에서 ([생성한 프로젝트 이름]+Dlg) 클래스를 선택합니다.

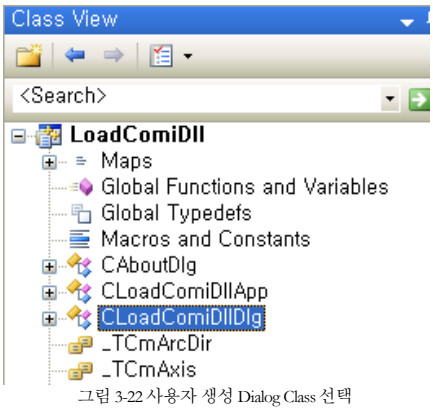


그림 3-22 사용자 생성 Dialog Class 선택

(생성한 프로젝트 이름)+Dlg 클래스가 선택된 상태에서 Properties 창의 'Overrides'를 선택합니다.

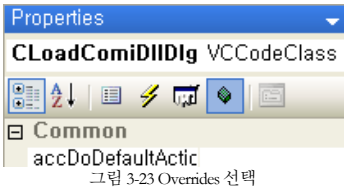


그림 3-23 Overrides 선택

'DestroyWinodw'항목 옆의 콤보 박스를 클릭하여 '<Add>DestroyWindow'를 선택합니다. ([생성한 프로젝트 이름]+Dlg 클래스에 'DestroyWindow'라는 이름의 오버라이드된 함수가 추가 됩니다.

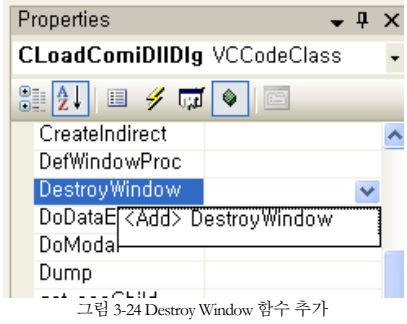


그림 3-24 Destroy Window 함수 추가

([생성한 프로젝트 이름]+Dlg) 클래스의 멤버함수인 DestroyWindow()에 'cmxUnloadDll();'을 추가 합니다. 'cmxUnloadDll()'함수를 추가 하면 윈도우가 종료 될 때 자동으로 DLL 도 해제 됩니다.

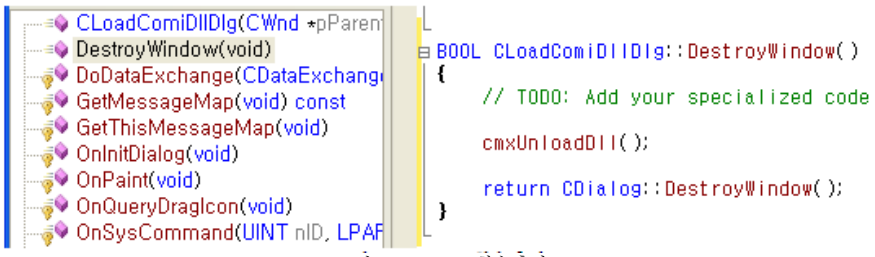


그림 3-25 UnloadDll 함수 추가

3.3.3 C++ Builder 개발자를 위한 안내

Borland/CodeGear C++ Builder 는 해당 개발 환경 버전인 C++ Builder 5, C++ Builder 6 및 Borland Developer Studio 2006, 2007, RAD Studio 2009 버전에서 ComiRTEX 의 인터페이스 연결 방법이 매우 유사하기 때문에 공통적인 부분으로서 안내를 해드립니다.

모든 버전(Version)의 C++ Builder 에서 ComiRTEX 를 사용하시려면 다음의 절차를 통해 안내 받으시기 바랍니다.

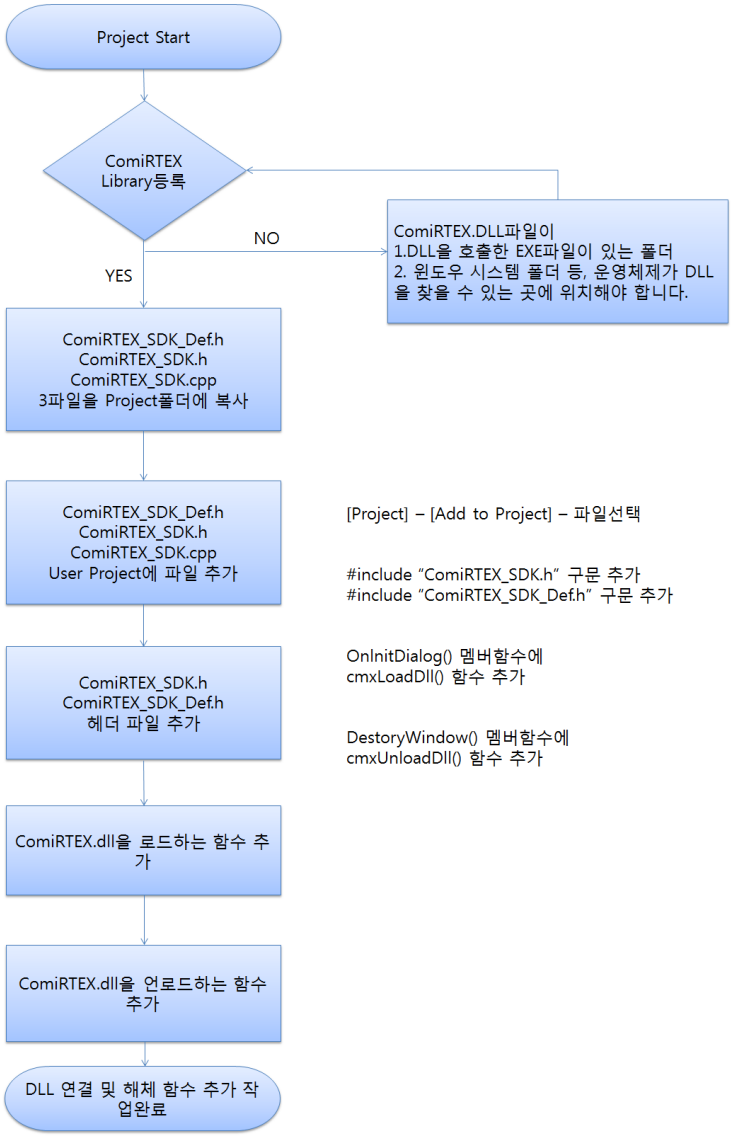


그림 3-26 C++ Builder 에서 ComiRTEX 사용 순서도

본 개발자를 위한 실제 안내에서는 다양한 버전의 C++ Builder 의 화면을 통해 안내 해드리도록 하겠습니다.

C++ Builder 를 실행합니다. 메뉴에서 [File]->[New]->[Application]을 선택하여 새로운 프로젝트를 시작합니다.

그림 3-27 BCB 5 에서 새로운 프로젝트 생성

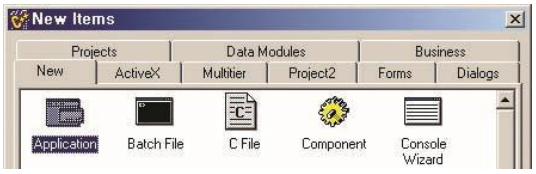


그림 3-28 BCB 5 에서 새로운 프로젝트 생성

그림 3-29 BCB 6 에서 새로운 프로젝트 생성

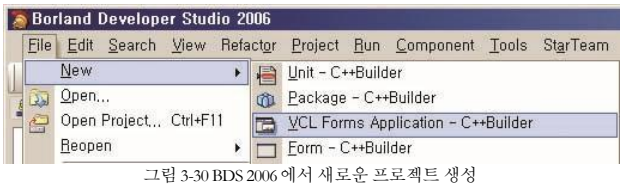


그림 3-30 BDS 2006 에서 새로운 프로젝트 생성

C++ Builder 및 VC++ 용 공용 인터페이스 정의 파일인 ComiRTEX_SDK.h, ComiRTEX_SDK.cpp, ComiRTEX_SDK_Def.h 파일을 신규로 생성한 프로젝트 폴더로 복사 합니다.

그림 3-31 ComiRTEX 사용시 공통으로 사용되는 파일

안내	참고 사항
	Borland 社의 C++ Builder 에서는 [File]-[Save] or [Save All]을 해주어야 프로젝트 폴더 및 프로젝트 관련 파일들을 생성합니다.

그림과 같이 C++ Builder 에서 추가할 인터페이스 파일을 실제 사용자 프로젝트에 추가합니다. Project 의 메뉴의 Add to Project 를 사용하시면 됩니다.

그림 3-32 C++ Builder 에서 프로젝트에 파일 추가 단계 1

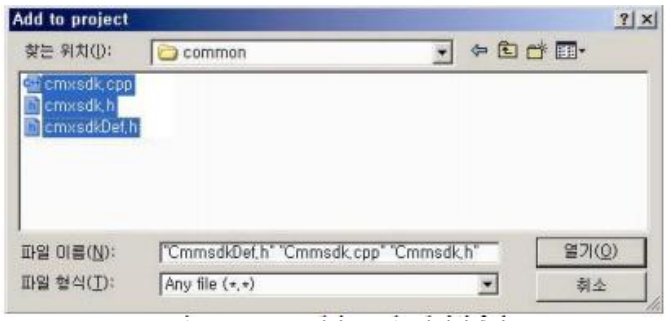


그림 3-33 C++ Builder 에서 프로젝트에 파일 추가

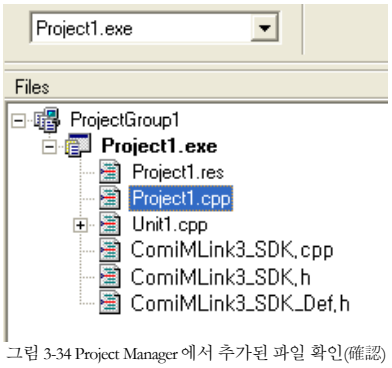


그림 3-34 Project Manager 에서 추가된 파일 확인(確認)

라이브러리 함수를 사용하고자 하는 대상 구현부 응용프로그램 파일에 인터페이스 파일을 선언합니다.

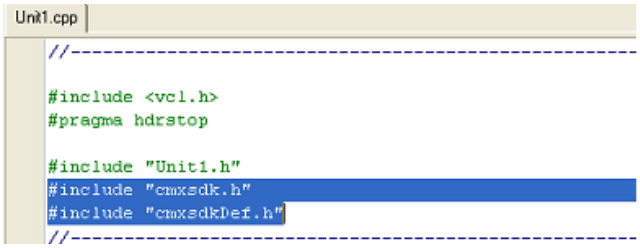


그림 3-35 라이브러리 사용시 필요한 헤더 파일 선언

ComiRTEX.dll 파일을 cmxLoadDll()함수를 이용하여 로드할 수 있도록 FormCreate 함수 또는 응용프로그램 시작 부분에 추가합니다.

ComiRTEX.dll 파일을 cmxLoadDll()함수를 이용하여 로드합니다. cmxLoadDll()을 추가 하는 방법을 FormCreate 를 통해 할 수 있으며, 그 예를 소개해 드립니다.

[Object Inspector] – [Events]탭의 OnCreate 에서 더블클릭합니다.

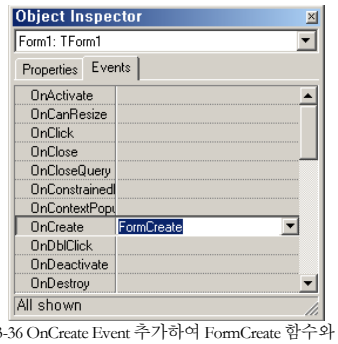


그림 3-36 OnCreate Event 추가하여 FormCreate 함수와 연결

추가된 FormCreate() 또는 응용프로그램 종료 함수 안에 cmxLoadDll()함수를 추가합니다.

그림 3-37 cmxLoadDll 함수 추가

고객(顧客)님이 작성하신 프로그램이 종료되면 DLL 을 명시적으로 Unload 시켜야 합니다.
DLL 의 Unload 시점은 고객(顧客)님께서 작성하신 응용프로그램이 종료되는 시점에 반드시 이루어져야 하며 cmxUnloadDll()이라는 함수를 통해서 이루어 집니다. cmxUnloadDll()을 추가 하는 방법은 다음과 같습니다.

[Object Inspector] – [Events]탭의 OnDestroy 에서 더블클릭합니다.

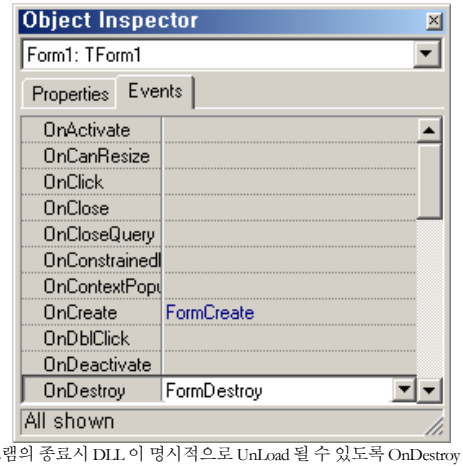


그림 3-38 응용프로그램의 종료시 DLL 이 명시적으로 UnLoad 될 수 있도록 OnDestroy Event 와 함수의 연결

추가된 FormDestroy()또는 응용프로그램 종료 함수에 cmxUnloadDll() 함수를 추가합니다.

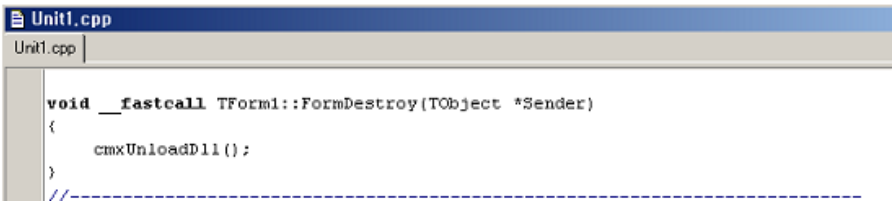


그림 3-39 FormDestroy 함수를 통하여 명시적인 UnLoadDll 함수 구현

3.3.4 Delphi 개발자를 위한 안내

Borland/CodeGear Delphi 는 해당 개발 환경 버전인 Delphi 5, Delphi 6 및 Delphi 7, Borland Developer Studio 2006, 2007, RAD Studio 2009 버전에서 ComiRTEX 의 인터페이스 연결 방법이 매우 유사하기 때문에 공통적인 부분으로서 안내를 해드립니다.

모든 버전(Version)의 Delphi 에서 ComiRTEX 를 사용하시려면 다음의 절차를 통해 안내 받으시기 바랍니다.

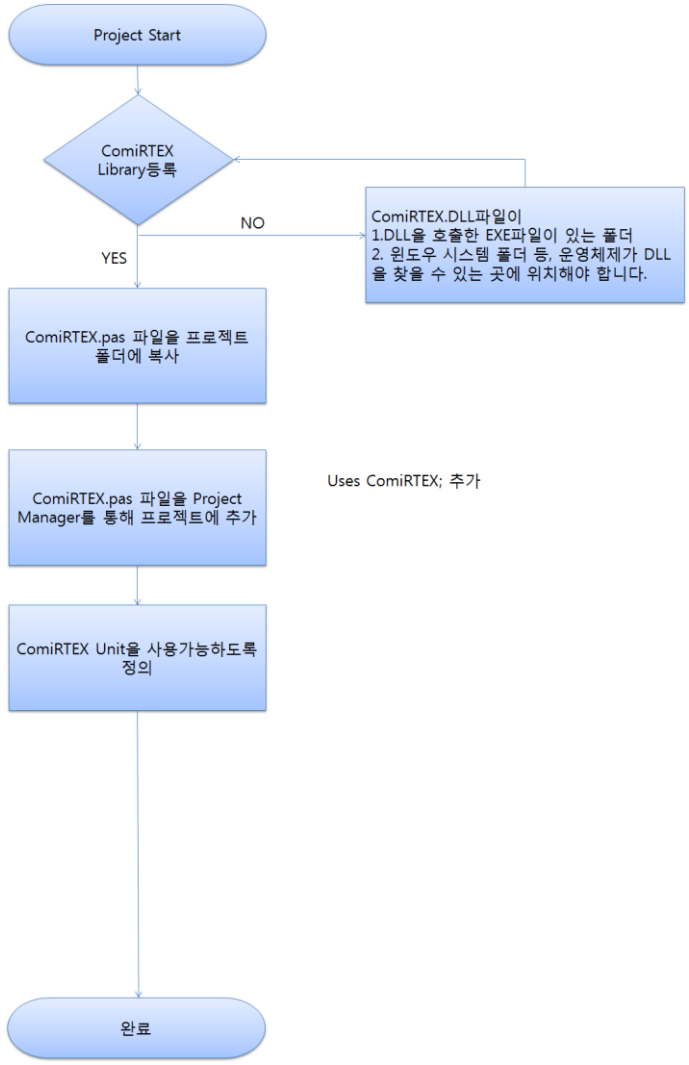


그림 3-40 Borland Delphi 에서 ComiRTEX 사용 순서도

Delphi 7 을 기준으로 설명드리겠습니다. 만약 안내해드리는 도중에 다른 개발 환경과 구분이 되어야 할 내용은 별도로 설명 드리겠습니다. Borland Delphi 를 실행합니다.

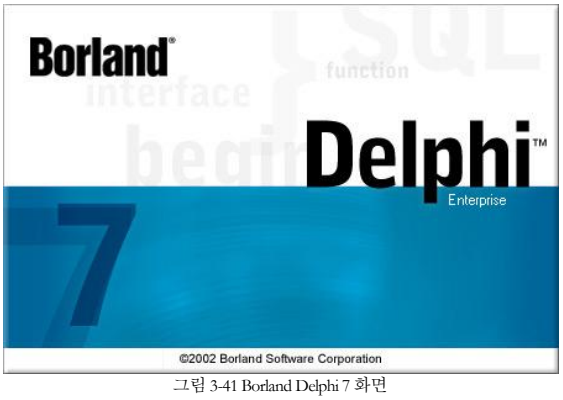


그림 3-41 Borland Delphi 7 화면

프로젝트 시작 전에 ㈜커미조아 ComiRTEX 의 Delphi 용 공용 인터페이스 파일을 프로젝트 폴더에 복사합니다. 이 파일은 ㈜커미조아 ComiRTEX 의 DLL(Dynamic Link Library) 와 고객님의 응용 프로그램과의 인터페 이스를 정의하여 놓은 파일입니다.

델파이(Delphi)는 명시적으로 프로젝트파일 간의 상호 변환이 필요 없습니다. 따라서 Delphi 5, Delphi 6, Dehphi 7 각각의 버전에서 몇 가지 기본적인 컴포넌트에 기반한 내용을 제외하고는 그대로 사용할 수 있습니다. ㈜커미조아 ComiRTEX 에서는 Delphi 에 대한 풍부한 예제를 제공하고 있습니다.

새로운 프로젝트를 시작하기 위해서, 메뉴의 [File]–[New] 명령을 통해, 새로운 응용 프로그램 개발을 시작합니다.

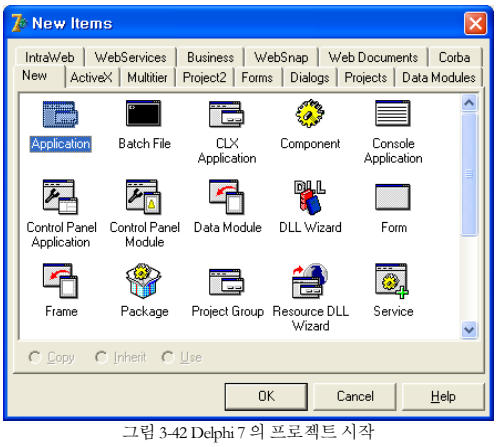


그림 3-42 Delphi 7 의 프로젝트 시작

프로젝트가 시작되면 화면상에 'Form1' 혹은 Delphi IDE 의 Project1 이 나타납니다.

인터페이스 파일을 추가하기 위한 작업으로서 [Project] 메뉴의 [Add to Project] 를 선택합니다.

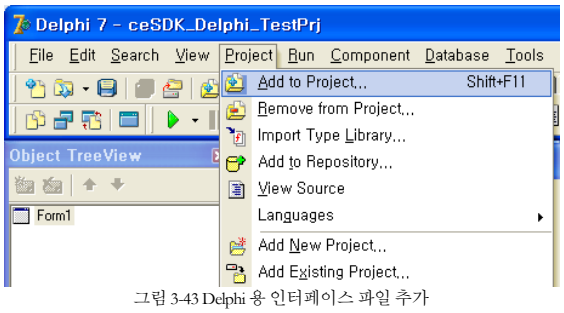


그림 3-43 Delphi 용 인터페이스 파일 추가

(주) 커미조아 ComiRTEX 의 공용 인터페이스 정의 파일인 ComiRTEX.PAS 파일을 프로젝트에 추가합니다.

그림 3-44 Delphi 용 인터페이스 파일

Project Manager 를 통해 확인해 보면 ComiRTEX.PAS 파일이 프로젝트에 등록된 것을 확인할 수 있습니다.

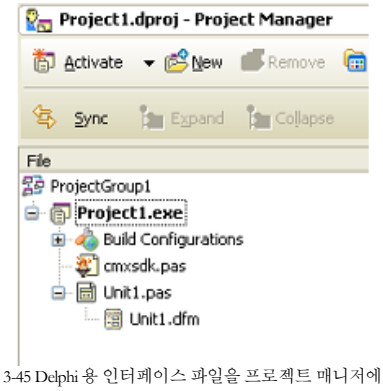


그림 3-45 Delphi 용 인터페이스 파일을 프로젝트 매니저에 추가

보충	Delphi 의 ComiRTEX 인터페이스 파일의 사용에 대한 부가 안내
	저희 ㈜커미조아의 ComiRTEX 의 ComiRTEX.pas 파일은 다른 개발환경(VC++, C++ Builder) 와 달리 DLL 의 묵시적인 로드와 언로드가 자동으로 이루어집니다. 이것은 델파이가 가지고 있는 Initialization 과 Finalization 을 이용한 것으로 프로젝트에 별다른 LoadDll 함수 호출 없이 자동으로 DLL 이 로드 됩니다. 또한 응용프로그램 종료시에 UnloadDll 이 명시적으로 호출되지 않아도, 자동적으로 응용 프로그램이 종료시에 UnloadDll 이 실행됩니다.

실제 구현 부의 코드를 에디터를 통해 확인합니다.

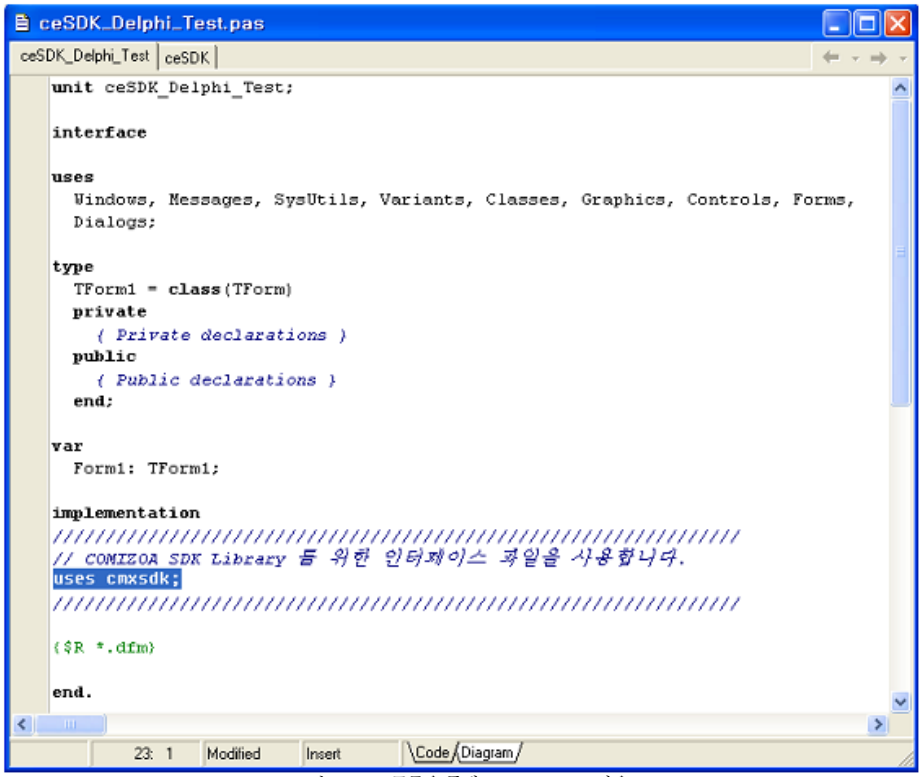


그림 3-46 uses 구문을 통해 ComiRTEX Unit 사용

위와 같이 implcmxentation 부에 **uses 를 통해 라이브러리 Unit 을 사용할 수 있도록 반드시 지정**해 주십시오. (상단의 uses 에 선언하여도 무방합니다) 이후, 델파이 에서는 다른 개발과 동일하게 DLL 라이브러리를 사용하실 수 있습니다.

3.3.5 Visual Basic 개발자를 위한 안내

Visual Basic 6.0 은 마이크로소프트의 컴포넌트 기반 응용프로그램 개발을 위해 태어난 뛰어난 개발 환경입니다. ComiRTEX 는 Visual Basic 6.0 를 완벽히 지원하며, 인터페이스 파일을 제공하고 있습니다. Visual Basic 고객(顧客)님들께서도 응용프로그램 개발에 편의성을 드리기위해 저희 ㈜커미조아는 언제나 노력하고 있습니다.

실제 Visual Basic 6.0 의 프로젝트 시작 전에, 프로젝트 디렉토리에 ㈜커미조아 ComiRTEX 인터페이스 파일인 ComiRTEX.BAS 파일과 ComiRTEX 파일 'ComiRTEX.dll' 파일을 반드시 프로젝트 디렉토리에 복사해주시기 바랍니다.

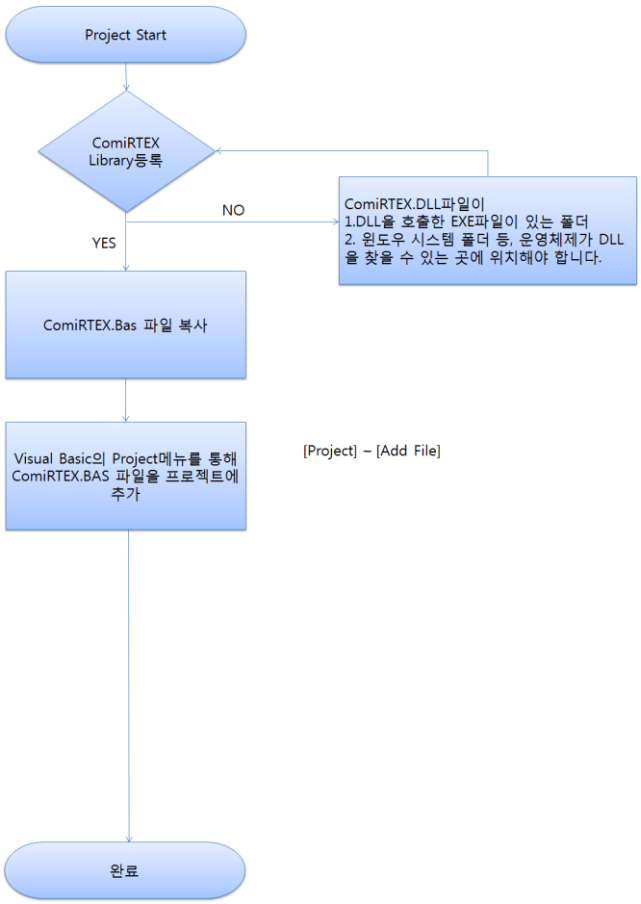


그림 3-47 Visual Basic 에서 ComiRTEX 사용 순서도

Visual Basic 을 실행합니다. Visual Basic 이 시작되면 다음과 같은 신규 프로젝트 화면이 표시됩니다.

그림 3-48 새로운 프로젝트 생성

안내

Visual Basic 에 대한 ComiRTEX 지원 사항은 어떠합니까?

ComiRTEX 는 .NET 환경의 Visual Basic 까지 지원을 하고 있습니다.

특히 미리 구성된 ComiRTEX 인터페이스 파일은 간단히 프로젝트에 추가만 하시면, 바로 ComiRTEX 를 사용하실 수 있도록 도움을 드리고 있습니다.

만약, Visual Basic 사용 고객(顧客)님께서 ComiRTEX 사용에 어려움이 있으시다면, 언제든지 저희 ㈜커미조아 고객(顧客) 지원팀으로 연락주시기 바랍니다. 최선을 다해 지원해 드릴 것을 약속드립니다.

신규프로젝트 창에서 'Standard EXE' 를 통해 표준 응용프로그램 개발을 시작합니다. '열기' 버튼을 누릅니다. 만약 이 화면이 나타나지 않으면, 아래의 화면과 같이 'File' 메뉴의 'New Project' 항목을 통해 신규 프로젝트를 시작합니다.

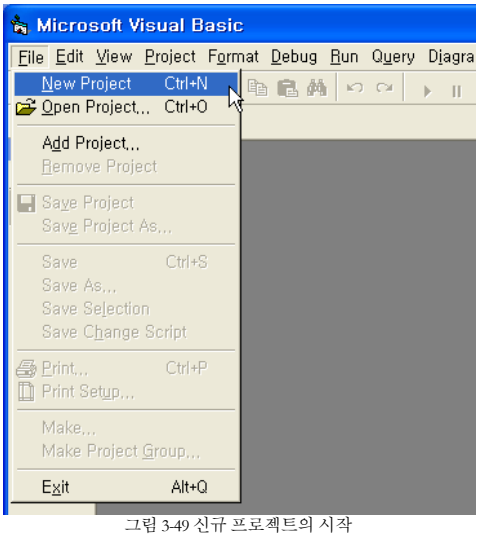


그림 3-49 신규 프로젝트의 시작

시작된 표준 EXE 응용프로그램 개발 메뉴에서 아래 그림과 같이 Project 메뉴를 통해 'Add File…' 을 선택합니다.

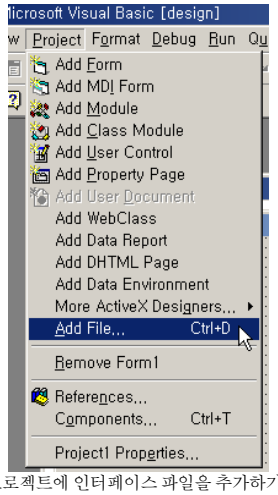


그림 3-50 프로젝트에 인터페이스 파일을 추가하기 위한 과정

그림 3-51 프로젝트에 추가 대상이 되는 ComiRTEX.BAS 파일

ComiRTEX.BAS 파일을 프로젝트에 추가해 주시면, 명시적인 ComiRTEX 로드가 이루어지게 되며, Visual Basic 의 프로젝트에서 함께 사용하실 수 있습니다.

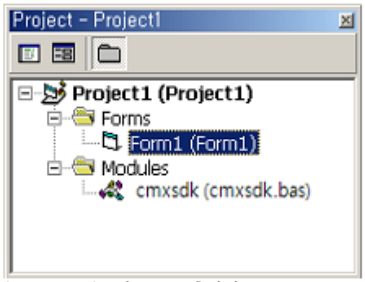


그림 3-52 프로젝트에 모듈로 추가된 ComiRTEX.BAS 파일

추가된 인터페이스 파일을 통해 다음과 같이 실제 응용프로그램 구현이 가능합니다.

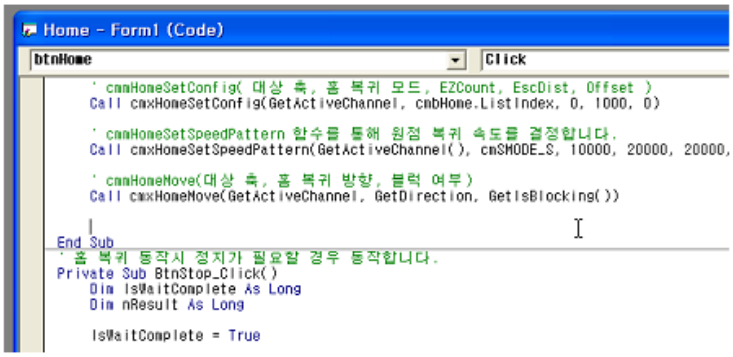


그림 3-53 ComiRTEX 를 통한 응용프로그램 구현

보충	Visual Basic 의 명시적인 ComiRTEX 인터페이스에 대한 부가 안내
	ComiRTEX 에 포함된 Visual Basic 용 인터페이스 파일은 별도의 DLL 라이브러리(ComiRTEX)의 로드(Load) 및 언로드(Unload) 가 필요 없습니다. 따라서, 고객(顧客)님의 프로젝트의 Form1 에 추가된 인터페이스 파일을 통하여, 응용프로그램 구현을 바로 시작하실 수 있습니다.

Chapter

4

ComiRTEX Introduction

다년간의 보다 강력하고 편리한 라이브러리 기술개발을 통해 자신있게 제공하여 드리는 ComiRTEX 는 편리한 함수명의 규칙(Rule)을 통해 사용자 편의성을 극대화 하였습니다. ㈜커미조아 ComiRTEX 의 최신 기능과 숙련된 모션 제어 기술은 결코 흉내낼 수 없는 커미조아의 기술입니다. 지금 확인(確認)하십시요

본 장에서는 ComiRTEX 가 제공하는 라이브러리 인터페이스에 대한 자세한 설명(說明)을 안내합니다. ComiRTEX 는 라이브러리 기능을 보다 강력하고 효율적으로 지원할 수 있는 다양한 런타임(Run-time) 인터페이스와 라이브러리의 다양한 기능을 직관적(直觀的)으로 제공합니다. 본 매뉴얼에서는 ComiRTEX 에서 제공하는 라이브러리 함수에 대한 설명(說明)을 기능에 따라 그룹별로 수록하였습니다.

4 ComiRTEX 소개

4.1 함수의 명명 규칙

ComiRTEX 에서 제공하는 모든 함수는 다른 API 함수와 이름이 중복되는 것을 피하기 위하여 아래의 예와 같이 "cmx"이라는 첨두어가 붙습니다.

cmxLoadDll(), cmxUnloadDll(), cmxGnLoadDevice (), cmxGnUnloadDevice (),...

그리고 "cmx" 첨두어 바로 다음에는 해당 함수가 속하는 기능의 그룹을 대표하는 첨두어가 이어집니다. 이렇게 한 이유는 동일한 기능 그룹에 속한 함수들을 쉽게 구분할 수 있고, 함수가 리스트될 때에 일반적으로 알파벳순으로 정렬되므로 동일 기능 그룹에 속한 함수들이 함께 리스트될 수 있도록 하기 위함입니다. 아래는 몇 가지 기능 그룹을 대표하는 첨두어가 적용된 함수들의 예입니다.

○. General Functions (Gn): cmxGnLoadDevice(), cmxGnSetServoOn (), ...

○. 환경설정 함수들 (Cfg): cmxCfgSetMioProperty(), cmxCfgSetSpeedPattern(), ...

○. 원점복귀 관련 함수들 (Home): cmxHomeMove(), cmxHomeSetConfig(), ...

○. 단축구동 관련 함수들 (Sx): cmxSxMoveStart(), cmxSxStop(), ...

○. 다축구동 관련 함수들 (Mx): cmxMxMoveStart(), cmxMxMoveToStart(), ...

○. 보간구동 관련 함수들 (Ix): cmxIxMapAxes(), cmxIxLine(), ...

4.2 데이터형 표기

당사의 ComiRTEX 인터페이스는 매뉴얼에서 명시한 윈도우 표준 Dynamic Link Library 를 지원하는 어떠한 개발 환경에서도 사용 가능합니다. 하지만 데이터형에 대한 이름은 개발환경에 따라서 서로 다릅니다. 따라서 본 매뉴얼에서는 데이터의 형 표기를 표 7 과 같이 통일하여 표기합니다. 이에 대한 각 컴파일러의 대응되는 데이터형 표기는 표 7 을 참조하여 사용하시기 바랍니다.

그리고 본 매뉴얼에서는 "[in]"과 "[out]" 표기를 사용하여 매개변수가 함수에 전달되는 것인지 아니면 전달받는 것인지를 명시하였습니다. "[in]"은 함수에 값을 전달함을 의미하고, "[out]은 함수로부터 값을 전달받는다는 것을 의미합니다. 단, 이 표기는 본 매뉴얼에서만 사용되는 것이며, 실제 헤더파일에는 표기되어 있지 않습니다.

Data type	Description	C/C++	VB 6.0	Delphi	C#
VT_EMPTY	반환값이 없는 데이터 표현형	void	-	-	void
VT_HANDLE	운영체제가 특정한 정보를 유지하기 위해서, 메모리에 유지하는 정보 블록에 붙은 고유 번호	void *	Long (ByRef)	THandle	IntPtr
VT_I4	4 바이트 부호있는 정수 표현형	long	Long (ByVal)	LongInt	Int
VT_PI4	4 바이트 부호있는 정수 변수의 주소 값 (포인터) 또는 배열주소 표현형	long *	Long (ByRef)	PLongInt	Int[]
VT_R4	4 바이트 부호있는 실수 표현형	float	Double (ByVal)	Double	Float
VT_PR4	4 바이트 부호있는 실수 변수의 주소 값(포인터) 또는 배열주소 표현형	float *	Double (ByRef)	PDouble	float[]
VT_R8	8 바이트 부호있는 실수 표현형	double	Double (ByVal)	Double	double
VT_PR8	8 바이트 부호있는 실수변수의 주소	double *	Double (ByRef)	PDouble	double[]

VT_STR	값(포인터) 또는 배열주소 표현형 선형 메모리 상의 문자열 선두 주소를 지시하는 4 바이트 주소 표현형	char *	String (ByVal)	PChar	String

표 7 언어독립적 데이터 표기 및 각 언어별 대응 데이터 형

Chapter

5

General Functions

㈜커미조아는 ComiRTEX 를 통해 다양한 최신 개발환경을 지원하기 위해 노력하고 있습니다. 본 장에서 다루지 않는 개발 환경을 이용하시는 고객(顧客)님께서는 저희 ㈜커미조아를 통해 문의해주시면 신속히 대처해 드리도록 하겠으며, 제공되는 라이브러리 인터페이스를 통해 보다 편리하고 빠르게 저희 라이브러리를 사용할 수 있도록 지원하여 드립니다. 뛰어난 성능을 기반으로 한 ComiRTEX 라이브러리의 즐거움을 이제 함께 하십시오.

이 단원에서는 장치의 로드/언로드 또는 장치의 초기화와 관련된 가장 일반적인 함수들을 소개합니다. 고객(顧客) 여러분들께서는 라이브러리의 초기화와 사용을 위해서는 반드시 본 장을 읽어주시기 바랍니다. 구성되는 함수에는 ComiRTEX 의 로드 및 언로드, 매개변수(媒介變數) (Parameter)초기화 및 디바이스 리셋(Reset)에 대한 내용들로 구성되어 있습니다.

5 General Functions

5.1 함수 요약

Summary of Functions
❒ BOOL cmxLoadDll([none] VT_EMPTY) 라이브러리를 사용자(使用者) 응용프로그램의 사용을 위해 로드(Load) 합니다.
❒ VT_EMPTY cmxUnloadDll([none] VT_EMPTY) 라이브러리를 사용자(使用者) 응용프로그램의 사용 종료(終了)를 위해 언로드(Unload) 합니다.
❒ VT_HANDLE cmxGnLoadDevice ([in] VT_PI4 NumDevices, [out] VT_PI4 BoardIDList, [out] VT_PI4 NumServos) 라이브러리가 로드된 상태에서 장치를 로드(Load) 하는 역할을 합니다.
❒ VT_I4 cmxGnUnloadDevice ([[none] VT_EMPTY) 라이브러리가 로드된 상태에서 장치를 언로드(Unload) 하는 역할을 합니다.
❒ VT_I4 cmxGnResetDevice ([in] VT_I4 BoardId, [in] VT_I4 ResetMask) 해당 모션의 정보들을 초기화하는 역할을 합니다.

5.2 함수 설명

NAME

cmxLoadDll

- 라이브러리(Library) 로드

INFORMATION

- General Function
- VC++/BCB/.NET
- Level 1
- ☺ 위험 요소 없음

SYNOPSIS

❒ BOOL cmxLoadDll ([none] VT_EMPTY)

DESCRIPTION

ComiRTEX 를 고객(顧客)님의 응용프로그램의 메모리 공간으로 호출합니다. 이 의미는 이 함수가 호출되는 순간 라이브러리는 고객(顧客)님의 프로그램 내부 함수처럼 호출할 수 있게 됩니다. 이 함수는 고객(顧客)님의 전체 프로그램에서 ComiRTEX 를 사용하기 위한 수순으로서는 가장 먼저 호출되어야 합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 파일에서는 Boland 社의 Delphi 나 Microsoft 社의 Visual Basic 에서는 명시적으로 이 동작이 이루어지기 때문에 필요하지 않습니다.

RETURN VALUE

* 이 리턴값은 불 형(Boolean Type) 을 가지고 있습니다.

Value	Meaning
cmxFALSE	DLL 을 로드하는데 실패하였음을 의미합니다.
cmxTRUE	DLL 을 성공적으로 로드하였음을 의미합니다.

SEE ALSO

cmxUnloadDll

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

void StartProgram(void)
{
      // 이 함수의 반환값은 DLL 의 로드의 성공여부를 반환합니다.
       BOOL nIsLoaded = cmxLoadDll();
}


void EndProgram(void)
{
```

```
    // 이 함수의 반환값은 없습니다. 따라서 다른 에러처리는 필요하지 않습니다.
    cmxUnloadDll();
}
```

NAME

cmxUnloadDll

- 라이브러리(Library) 로드 해제(解除)

INFORMATION

- General Function
- VC++/BCB/.NET
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_EMPTY cmxUnloadDll ([none] VT_EMPTY)

DESCRIPTION

ComiRTEX 를 고객(顧客)님의 응용프로그램의 메모리 공간에서 해제합니다. 이 의미는 이 함수가 호출되는 순간 ㈜커미조아의 ComiRTEX 는 고객(顧客)님의 응용프로그램에서 제거됩니다. 이 함수는 고객(顧客)님의 전체 프로그램에서 ComiRTEX 를 사용을 종료 하기 위한 수순으로서는 가장 나중에 호출되어야 합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 파일에서는 Boland 社의 Delphi 나 Microsoft 社의 Visual Basic 에서는 명시적으로 이 동작이 이루어질 수 있도록 구성되어 있기 때문에 필요하지 않습니다.

SEE ALSO

cmxLoadDll

EXAMPLE

C/C++

```
void StartProgram(void)
{
      // 이 함수의 반환값은 DLL 의 로드의 성공여부를 반환합니다.
       BOOL nIsLoaded = cmxLoadDll();
}


void EndProgram(void)
{
      // 이 함수의 반환값은 없습니다. 따라서 다른 에러처리는 필요하지 않습니다.
      cmxUnloadDll();
}
```

NAME

cmxGnLoadDevice

– 장치(裝置) 로드(Load)

INFORMATION

General Function
VC++/VB
BCB/Delphi/.NET
Level 1
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxGnLoadDevice ([out] VT_PI4 NumDevices, [out] VT_PI4 BoardIDList,
[out] VT_PI4 NumServos)

DESCRIPTION

시스템에 설치된 하드웨어 장치를 로드하고 장치를 초기화합니다. 이 함수는 ComiRTEX 의 다른 함수가 호출되기전에 반드시 한번은 수행되어야 합니다. 일반적으로 프로그램의 시작부분에서 수행해주면 됩니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶NumDevices:이매개변수를통하여실제로로드(Load)된네트워크모션보드의개수를반환합니다. 단, 이매개변수에 NULL 을전달하면보드개수를반환하지않습니다.

▶BoardIDList:이매개변수를통하여실제로로드(Load)된네트워크모션보드의 ID 배열을반환합니다. 단, 이매개변수에 NULL 을전달하면보드 ID 배열을반환하지않습니다.

▶NumServos:이매개변수를통하여실제로로드(Load)된서보드라이브의개수를반환합니다. 단, 이매개변수에 NULL 을전달하면제어축수를반환하지않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러 처리' 편을 참고합니다
ERR_NONE	수행 성공

SEE ALSO

cmxGnUnloadDevice

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
```

```
#include "ComiRTEX_SDK_Def.h"

void ProgramInitial(void)
{
	long nNumDevices = 0;
	long DeviceList[16] = {0, };
	long nNumAxes = 0;

	if ( cmxLoadDll() != TRUE ) {

/* OutputDebugString API 는 GUI 프로그램에서 문자열을 디버거에 보낼 수 있습니다. Borland
C++ Builder 에서는 DebugWindows 에 Event Log 를  확인(確認)할 수 있으며, MS VC++ 에서는
Debug 윈도우에서 확인(確認)할 수 있습니다. */

long BoardID = 0;
OutputDebugString("DLL 로드에 실패하였습니다");
// 이후 적절한 에러처리를 해주시기 바랍니다.
	}

	if cmxGnLoadDevice (&nNumDevices, DeviceList, &nNumAxes) != ERR_NONE ){

	// 에러메시지 출력

	}
} /* ProgramInitial(void) 함수의 끝 */
```

Visual Basic

```
' Visual Basic 에서는 명시적인 DLL 로드가 필요 없습니다.

Private Sub Form_Load()

Dim IRetVal As Long

Dim nTotalDevices As Long
Dim DeviceList(16) As Long
Dim nTotalAxes As Long

IRetVal = GnLoadDevice (nTotalDevices, DeviceList(0), nTotalAxes)

If  IRetVal <> ERR_NONE Then

	MsgBox("cmxGnLoadDevice has been failed")

End If

End Sub
```

Delphi

```
//* Delphi 에서는 명시적인 DLL 로드가 필요없습니다.
//* 단, 처음 선언시에 다음과 같은 내용이 포함되어야 합니다.
////////////////////////////////////////////////////////////
// COMZIOA SDK Library 를 위한 인터페이스 파일을 사용합니다.
uses ComiRTEX;
```

```
////////////////////////////////////////////////////////////
procedure TForm1.OnCreate(Sender: TObject);
var
	g_nDevs : LongInt;
	DevList : Array[0..15] of LongInt
	g_nAxes : LongInt;

begin

if ( cmxGnLoadDevice (@g_nDevs, @DevList, @g_nAxes) <> ERR_NONE ) then
begin

	// 에러메시지 출력

	exit;
end;

end;
```

NAME

cmxGnUnloadDevice

– 장치(裝置) 언로드(Unload)

INFORMATION

- General Function
- VC++/VB
- BCB/Delphi/.NET
- Level 1
- ☺ 위험 요소 없음

SYNOPSIS

❒ VT_BOOL cmxGnUnloadDevice ([none] VT_EMPTY)

DESCRIPTION

시스템에 설치된 하드웨어 장치를 언로드하고 장치사용을 종료합니다. 이 함수는 ComiRTEX 의 함수 사용을 종료하기 위해 명시적으로 호출되어야 합니다. 일반적으로 프로그램의 종료부분에서 수행해주면 됩니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

SEE ALSO

cmxGnLoadDevice

NAME

cmxGnResetDevice

– 모션 정보 리셋(Reset)

INFORMATION

- General Function
- VC++/VB
- BCB/Delphi/.NET
- Level 1
- ☺ 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxGnResetDevice ([in] VT_I4 BoardID, [in] VT_I4 ResetMask)

DESCRIPTION

이 함수는 모션 이송시의 정보들을 초기화하는 함수입니다. 단축, 보간, 리스트 모션의 맵 또는 축 정보를 리셋하고, 속도, 원점복귀, 위치, 백리시, 소프트 리미트의 환경설정값들을 초기화합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ ResetMask: 리셋 마스크 값입니다.

Value	Meaning
0x1 또는 RS_SX_ENV	단축 이송에 대한 마스크 값입니다.
0x2 또는 RS_IX_ENV	보간 이송에 대한 마스크 값입니다.
0x4 또는 RS_LM_ENV	리스트 모션에 대한 마스크 값입니다.
0x8 또는 RS_SPEED_ENV	속도에 대한 마스크 값입니다.
0x10 또는 RS_HOME_ENV	원점복귀에 대한 마스크 값입니다.
0x20 또는 RS_POS_ENV	위치에 대한 마스크 값입니다. 모든 축의 위치를 리셋합니다.
0x40 또는 RS_BACKLASH_ENV	백래시에 대한 마스크 값입니다.
0x80 또는 RS_SWLIMIT_ENV	소프트 리미트에 대한 마스크 값입니다.
0x100 또는 RS_UPDATE_INTERVAL_ENV	업데이트 주기를 초기화합니다. (Default : 1 msec)
0xff 또는 ALL_RESET	위 모든 항목을 리셋하는 마스크 값입니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러 처리' 편을 참고합니다

ERR_NONE	수행 성공

Example

단축 이송과 리스트 모션을 리셋할 시 ResetMask 값은 0x5 가 됩니다.
원점복귀와 속도를 리셋할 시 ResetMask 값은 0x18 이 됩니다.

Chapter

6

Etc General Functions

ComiRTEX 는 기본 함수 이외에도 기타 기본 함수들을 지원합니다. 이 함수의 집합에서 가장 두드러진 기능으로서는 GUI 환경에서 모션의 전체 혹은 부분적인 설정(設定)을 완료하여, 그 설정(設定)을 실제 고객(顧客)님의 응용프로그램에서 사용할 수 있는 기능입니다. 이외에도 다양한 편의 기능과 우수한 기능들이 고객(顧客)님들을 기다리고 있습니다.

본 단원에서는 기타 General 함수(函數)들을 소개합니다. 기타 General 함수(函數)들은 그 기능에 있어 모든 내용이 반드시 습득(習得) 될 필요는 없지만, 모션 제어를 위해 꼭 필요한 내용의 기타 General 함수 편으로 구성되어 있습니다. 기본적인 서보 ON 출력 신호제어 기능이 제공됩니다.

6 Etc General Functions

6.1 함수 요약

Summary of Functions
❒ VT_I4 cmxGnSetServoOn ([in] VT_I4 BoardID, [in] VT_I4 Axis [in] VT_I4 dwIsOn) 서보 드라이브의 SERVO ON 신호 출력을 인가(認可) 혹은 차단(遮斷) 합니다.
❒ VT_I4 cmxGnGetServoOn ([in] VT_I4 BoardID, [in] VT_I4 Axis, [out] VT_PI4 pdwIsOn) 서보 드라이브의 SERVO ON 신호 출력(出力) 상태를 반환(返還)합니다.
❒ VT_I4 cmxGnSetAlarmRes ([in] VT_I4 BoardID, [in] VT_I4 Axis) 알람 리셋(Alarm Reset) 신호 출력(出力) 을 제어합니다.
❒ VT_I4 cmxGnGetAlarmRes ([in] VT_I4 BoardID, [in] VT_I4 Axis, [out] VT_PI4 IsOn) 알람 리셋(Alarm Reset) 신호 출력(出力) 을 위한 제어 설정을 반환(返還) 합니다.
❒ VT_I4 cmxGnSetEmergency ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 IsDecStop , [in] VT_I4 IsEnable) 해당 축에 대한 소프트웨어적인 비상 상황을 설정(設定)합니다.
❒ VT_I4 cmxGnGetEmergency ([in] VT_I4 BoardId, [in] VT_I4 Axis, [out] VT_PI4 IsDecStopped , [out] VT_PI4 IsEnabled) 해당 축에 대한 소프트웨어적인 비상 상황 상태를 반환(返還)합니다.
❒ VT_I4 cmxGnSetEmergencyAll ([in] VT_I4 IsDecStop, [in] VT_I4 IsEnable) 소프트웨어적인 비상 상황을 설정(設定)합니다.
❒ VT_I4 cmxGnGetEmergencyAll ([out] VT_PI4 IsDecStopped , [out] VT_PI4 IsEnabled) 소프트웨어적인 비상 상황 상태를 반환(返還)합니다.
❒ VT_I4 cmxGnSetCommPeriod([in] VT_I4 BoardID, [in]VT_I4 nPeriod) 통신 업데이트 주기를 절정(設定) 합니다.
❒ VT_I4 cmxGnGetCommPeriod([in] VT_I4 BoardID, [out]VT_I4 nPeriod) 통신 업데이트 주기를 반환(返還)합니다.
❒ VT_I4 cmxGnSetStatusUpdateInterval ([in] VT_I4 BoardID, [in] VT_I4 dwInterval) 실시간 상태 업데이트 주기를 설정(設定)합니다.
❒ VT_I4 cmxGnGetStatusUpdateInterval ([in] VT_I4 BoardID, [out] VT_PI4 dwInterval) 실시간 상태 업데이트 주기를 반환(返還)합니다.
❒ VT_I4 cmxGnGetAxisMap ([in] VT_I4 BoardID, [out] VT_PI4 AxisMapMask) 연결된 슬레이브 정보를 반환(返還)합니다.
❒ VT_I4 cmxGnResetComm([in] VT_I4 BoardID) 통신을 초기화합니다.
❒ VT_I4 cmxGnSetCommStates([in] VT_I4 BoardID , [in] VT_I4 Axis, [in] VT_I4 cmxtsId, [in]CommStsVal) 통신 상태를 설정(設定)합니다.
❒ VT_I4 cmxGnGetCommStates([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 cmxtsId, [out]CommStsVal) 통신상태를 반환(返還)합니다.
❒ VT_I4 cmxGnSetParam([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 PrmNo1, [in] VT_I4 PrmData1, [in] VT_I4 PrmNo2, VT_I4 PrmData2) 해당 축의 서보 파라미터를 설정(設定)합니다.

❒ VT_I4 cmxGnGetParam([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 PrmNo1 [out] VT_PI4 PrmData1, [in] VT_I4 PrmNo2, [out] VT_PI4 PrmData2) 해당 축의 서보 파라미터를 반환(返還)합니다.
❒ VT_I4 cmxGnSetLogMode ([in] VT_I4 LogMode) 로그를 기록할 방법을 지정합니다.
❒ VT_I4 cmxGnGetLogMode ([out] VT_PI4 LogMode) 로그를 기록하는 방법을 반환합니다.
❒ VT_I4 cmxGnSetLogLevel ([in] VT_I4 LogLevel) 로그를 기록할 기준 레벨을 지정합니다.
❒ VT_I4 cmxGnGetLogLevel ([out] VT_PI4 LogLevel) 로그를 기록할 기준 레벨을 반환합니다.
❒ VT_I4 cmxGnSetFuncLevel ([in] VT_I4 FuncIndex, [in] VT_I4 LogLevel) 개별 특정 함수에 로그 레벨을 설정합니다.
❒ VT_I4 cmxGnGetFuncLevel ([in] VT_I4 FuncIndex, [out] VT_PI4 LogLevel) 개별 특정 함수에 설정된 로그 레벨을 반환합니다.
❒ VT_I4 cmxGnRestoreFuncLevel ([in] VT_I4 FuncIndex) 지정한 함수의 로그 레벨을 원래 레벨로 원상복귀합니다.
❒ VT_I4 cmxGnGetNodeInfo ([in] VT_I4 BoardID, [in] VT_PI4 Axis, [out] TSlaveInfo *SlaveInfo) 해당 축의 슬레이브 정보를 반환합니다.
❒ VT_I4 cmxGnTotalDIOChannel ([in] VT_I4 BoardID, [in] VT_I4 Axis, [out] VT_PI4 Channel) 해당 축에 연결된 총 디지털 I/O(Input/Output) 채널 개수를 반환(返還)합니다.
❒ VT_I4 cmxGnTotalAIChannel([in] VT_I4 BoardID, [in] VT_I4 Axis, [out] VT_PI4 Channel) 해당 축에 연결된 총 아날로그 입력(Analog Input) 채널 개수를 반환(返還)합니다.
❒ VT_I4 cmxGnTotalAOChannel([in] VT_I4 BoardID, [in] VT_I4 Axis, [out] VT_PI4 Channel) 해당 축에 연결된 총 아날로그 출력(Analog Output) 채널 개수를 반환(返還)합니다.

6.2 함수 설명

NAME

cmxGnSetServoOn

cmxGnGetServoOn

- 서보 동작 Turn On/Off 신호 출력(出力) 제어

INFORMATION

Etc General Function

VC++/VB

BCB/Delphi/.NET

Level 1

다소 위험

이 함수는 서보드라이브의 동작을 위한 ON/OFF 상태를 결정합니다. 서보 동작시에 반드시 주의 환경을 점검하고, 안전 사항에 유의해야 합니다.

SYNOPSIS

- ❒ VT_I4 cmxGnSetServoOn ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 dwIsOn)
- ❒ VT_I4 cmxGnGetServoOn ([in] VT_I4 BoardID, [in] VT_I4 Axis, [out] VT_PI4 pdwIsOn)

DESCRIPTION

cmxGnSetServoOn() 함수는 지정한 채널(축)의 SERVO-ON 신호 출력을 제어합니다. 서보 드라이버를 사용하실 때는 외부에서 스위치를 이용하여 서보드라이버의 ON/OFF 를 제어할 수 있도록 하는데, 이를 SERVO-ON 신호라 합니다. 이 함수는 SERVO-ON 신호의 ON/OFF 를 제어하는 함수입니다.
cmxGnGetServoOn() 함수는 현재의 SERVO-ON 신호의 출력 상태를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(축 번호는 3 부터 시작합니다.).

▶ dwIsOn: cmxGnSetServoOn 함수의 인자이며, SERVO-ON 신호의 출력 상태를 설정합니다.

Value	Meaning
0 또는 cmxFALSE	SERVO- OFF
1 또는 cmxTRUE	SERVO- ON

▶ pdwIsOn: cmxGnGetServoOn 함수의 인자이며, SERVO-ON 신호의 출력 상태를 반환합니다.

Value	Meaning
0 또는 cmxFALSE	SERVO- OFF

1 또는 cmxTRUE	SERVO- ON

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

REFERENCE

☐ 서보를 On 시켜야 하는 경우 cmxGnLoadDeivce () 함수 호출 이후에 cmxGnSetServoOn (BoardID,Axis#, cmxTRUE)를 호출해 주셔야 서보 모터가 정상 동작 합니다.

NAME

cmxGnSetAlarmRes

cmxGnGetAlarmRes

- 알람 리셋(Alarm Reset) 신호 출력(出力) 제어(制御)

INFORMATION

- Etc General Function
- VC++/VB
- BCB/Delphi/.NET
- Level 1
- ☺ 다소 주의

SYNOPSIS

- ❒ VT_I4 cmxGnSetAlarmRes ([in] VT_I4 BoardID, [in] VT_I4 Axis)
- ❒ VT_I4 cmxGnGetAlarmRes ([in] VT_I4 BoardID, [in] VT_I4 Axis, [out] VT_PI4 IsOn)

DESCRIPTION

cmxGnSetAlarmRes() 함수는 지정한 채널(축)의 알람 리셋(Reset) 출력을 제어합니다. 서보 드라이버를 사용하실 때는 외부에서 디지털 접점을 이용하여, 서보에서 발생한 알림 상황을 초기화 할 수 있도록 서보 드라이브 상에서 요구하는 입력 신호가 존재하는 데, 이 신호 입력에 대응하기 위해, 모션 보드에서는 기본으로 제공되는 디지털 출력 신호를 통해 디지털 출력 신호를 발생하게 되며, 이 신호를 알람 리셋 신호(Alarm Reset Signal)라 합니다. 이 함수를 통해 서보에 발생한 알람을 해제시킬 수 있습니다. 그리고 통신 리셋 시 피드백 포지션값을 커맨드 포지션 값에 덮어 씌웁니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(축 번호는 3 부터 시작합니다.).

▶ IsOn : cmxGnGetAlarmRes 함수의 인자이며, 출력 상태를 반환합니다. 출력 상태는 다음과 같이 반환 됩니다.

Value	Meaning
0 (cmxFALSE)	출력 상태가 비(非)활성화 상태입니다.
1 (cmxTRUE)	출력 상태가 활성화 상태입니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxGnSetEmergency

cmxGnGetEmergency

- 해당 축에 대한 소프트웨어적인 비상상황 제어

INFORMATION

- Etc General Function
- VC++/VB
- BCB/Delphi/.NET
- Level 1
- 위험요소 없음

SYNOPSIS

❒ VT_I4 cmxGnSetEmergency ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 IsDecStop , [in] VT_I4 IsEnable)

❒ VT_I4 cmxGnGetEmergency ([in] VT_I4 BoardID, [in] VT_I4 Axis, [out] VT_PI4 IsDecStopped , [out] VT_PI4 IsEnabled)

DESCRIPTION

cmxGnSetEmergency() 함수는 소프트웨어적으로 모션컨트롤러의 Axis 축을 Emergency 상태로 설정합니다. 비상정지(停止) 상태가 되면 모션컨트롤러는 현재 진행중인 작업을 모두 정지(停止) 합니다. 비상정지(停止)가 활성화되어 있는 동안에는 이동명령이 호출되어도 작업이 생략됩니다. IsDecStop 매개 변수(媒介變數)에 따라 비상 정지 혹은 감속 후 정지를 수행합니다.
cmxGnGetEmergency() 함수는 모션컨트롤러의 Axis 축의 소프트웨어적인 Emergency 상태를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(축 번호는 3 부터 시작합니다.).

▶ IsDecStop : IsEnable 의 매개변수가 cmxTRUE 로 설정(設定)되면 현재 수행되고 있는 모든 작업은 정지(停止)하게 됩니다. 이때 정지(停止)시에 급정지(停止) 할것인지 감속후 정지(停止)할 것인지를 결정합니다. 단, IsEnable 매개 변수(媒介變數)가 cmxFALSE 이면 전체적인 비상 정지 상태를 비활성화하게 되므로, 본 매개변수의 설정 값은 무시됩니다.

Value	Meaning
0 또는 cmxFALSE	급정지(停止) (감속없음)
1 또는 cmxTRUE	감속후 정지(停止) (감속도는 현재 각 축별로 설정된 감속도 적용)

▶ IsEnable : 소프트웨어적인 비상상황제어의 활성/비활성 상태를 설정합니다.

Value	Meaning

0 또는 cmxFALSE	비상상황제어 비활성 (정상 상태)
1 또는 cmxTRUE	비상상황제어 활성

▶ IsDecStoped : 해당 축의 비상정지 상황시 급정지인지 감속 후 정지인지에 대한 상태값을 반환합니다.

Value	Meaning
0 또는 cmxFALSE	급정지(停止) (감속없음)
1 또는 cmxTRUE	감속후 정지(停止) (감속도는 현재 각 축별로 설정된 감속도 적용)

▶ IsEnabled : 소프트웨어적인 비상상황제어의 활성/비활성 상태를 반환합니다.

Value	Meaning
0 또는 cmxFALSE	비상상황제어 비활성 (정상 상태)
1 또는 cmxTRUE	비상상황제어 활성

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxGnSetEmergencyAll

cmxGnGetEmergencyAll

- 전축에 대한 소프트웨어적인 비상상황 제어

INFORMATION

- Etc General Function
- VC++/VB
- BCB/Delphi/.NET
- Level 1
- 위험요소 없음

SYNOPSIS

- ❒ VT_I4 cmxGnSetEmergencyAll ([in] VT_I4 BoardID, [in] VT_I4 IsDecStop , [in] VT_I4 IsEnable)
- ❒ VT_I4 cmxGnGetEmergencyAll ([in] VT_I4 BoardID, [out] VT_PI4 IsDecStopped , [out] VT_PI4 IsEnabled)

DESCRIPTION

cmxGnSetEmergency() 함수는 소프트웨어적으로 모션컨트롤러 전축을 Emergency 상태로 설정합니다. 비상정지(停止) 상태가 되면 모션컨트롤러는 현재 진행중인 작업을 모두 정지(停止) 합니다. 비상정지(停止)가 활성화되어 있는 동안에는 이동명령이 호출되어도 작업이 생략됩니다. IsDecStop 매개 변수(媒介變數)에 따라 비상 정지 혹은 감속 후 정지를 수행합니다.
cmxGnGetEmergency() 함수는 모션컨트롤러 전축에 대한 소프트웨어적인 Emergency 상태를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ IsDecStop : IsEnable 의 매개변수가 cmxTRUE 로 설정(設定)되면 현재 수행되고 있는 모든 작업은 정지(停止)하게 됩니다. 이때 정지(停止)시에 급정지(停止) 할것인지 감속후 정지(停止)할 것인지를 결정합니다. 단, IsEnable 매개 변수(媒介變數)가 cmxFALSE 이면 전체적인 비상 정지 상태를 비활성화하게 되므로, 본 매개변수의 설정 값은 무시됩니다.

Value	Meaning
0 또는 cmxFALSE	급정지(停止) (감속없음)
1 또는 cmxTRUE	감속후 정지(停止) (감속도는 현재 각 축별로 설정된 감속도 적용)

▶ IsEnable : 비상정지(停止)의 활성/비활성 상태를 설정합니다.

Value	Meaning
0 또는 cmxFALSE	비상정지(停止) 비활성 (정상 상태)
1 또는 cmxTRUE	비상정지(停止) 활성

▶ IsDecStoped : 해당 축의 비상정지 상황시 급정지인지 감속 후 정지인지에 대한 상태값을 반환합니다.

Value	Meaning
0 또는 cmxFALSE	급정지(停止) (감속없음)
1 또는 cmxTRUE	감속후 정지(停止) (감속도는 현재 각 축별로 설정된 감속도 적용)

▶ IsEnabled : 비상정지(停止)의 활성/비활성 상태를 반환합니다.

Value	Meaning
0 또는 cmxFALSE	비상정지(停止) 비활성 (정상 상태)
1 또는 cmxTRUE	비상정지(停止) 활성

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxGnSetCommPeriod

cmxGnGetCommPeriod

- 통신/업데이트 주기 설정 / 반환

INFORMATION

- Etc General Function
- VC++/VB
- BCB/Delphi/.NET
- Level 1
- 위험요소 없음

SYNOPSIS

- ❒ VT_I4 cmxGnSetCommPeriod ([in] VT_I4 BoardID, [in] VT_I4 nPeriod)
- ❒ VT_I4 cmxGnGetCommPeriod ([in] VT_I4 BoardID, [out] VT_PI4 nPeriod)

DESCRIPTION

Master 컨트롤러와 Slave 간 통신 주기 및 Slave 정보 업데이트 주기를 설정합니다. 이 함수를 통해 주기를 변경한 후에는 반드시 Master 컨트롤러를 Reset 하여야 변경한 주기가 적용 됩니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ nPeriod : cmxGnSetCommPeriod 함수의 인자이며, 통신/업데이트 주기를 설정하거나 반환합니다. 설정 가능한 주기는 다음과 같습니다..

Value(10nsec)	Meaning
5000	500usec(0.5msec)
10000	1000usec(1msec)

RETURN VALUE

Value	Meaning
음수	수행실패. 자세한 내용은 '에러처리' 편을 참고합니다.
ERR_NONE	수행 성공

NAME

cmxGnSetStatusUpdateInterval

cmxGnGetStatusUpdateInterval

- 실시간 상태 업데이트 주기 설정/반환

INFORMATION

- Etc General Function
- VC++/VB
- BCB/Delphi/.NET
- Level 1
- 위험요소 없음

SYNOPSIS

- ❒ VT_I4 cmxGnSetStatusUpdateInterval ([in] VT_I4 BoardID, [in] VT_I4 dwInterval)
- ❒ VT_I4 cmxGnGetStatusUpdateInterval ([in] VT_I4 BoardID, [out] VT_PI4 dwInterval)

DESCRIPTION

cmxGnSetStatusUpdateInterval() 함수는 슬레이브로부터 전달되는 실시간 상태 업데이트 주기를 설정합니다. 슬레이브로부터 전달되는 실시간 상태(MIO, Feedback Speed, Feedback Position 등)가 설정된 주기마다 한번씩 업데이트됩니다.
cmxGnGetStatusUpdateInterval() 함수는 슬레이브로부터 전달되는 실시간 상태 업데이트 주기를 반환합니다.
업데이트 주기는 상수 값으로 단위는 μs(10-6s), 초기값은 1μs 입니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ dwInterval: 슬레이브로부터 전달되는 실시간 상태 업데이트 주기입니다. 상수값으로 단위는 μs(10-6s), 초기값은 1μs 입니다.

▶ dwInterval: 슬레이브로부터 전달되는 실시간 상태 업데이트 주기를 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxGnGetAxisMap

- 연결된 슬레이브 정보를 반환합니다.

INFORMATION

- Etc General Function
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- ☺ 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxGnGetAxisMap ([in] VT_I4 BoardID, [out] VT_PI4 AxisMapMask)

DESCRIPTION

연결된 슬레이브 정보를 반환합니다. 이 함수를 통해 현재 마스터 보드에 연결된 슬레이브의 id 를 알 수 있습니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ AxisMapMask: 연결된 슬레이브 id 를 마스크 값(32 비트, BIT0 ~ BIT31)을 반환합니다. 이 값의 BIT0~BIT31 을 이용하여 현재 연결된 축을 확인할 수 있습니다.. 각 비트의 값이 0 이면 해당 축(비트의 순서와 일치하는 축)은 연결되지 않은 것이며 1 이면 해당 축이 연결된 것입니다

예) 8 축을 사용하는 경우

Bit Number	Meaning
BIT0	0 번 축의 연결여부: 0 => 연결안됨, 1 => 연결됨
BIT1	1 번 축의 연결여부: 0 => 연결안됨, 1 => 연결됨
BIT2	2 번 축의 연결여부: 0 => 연결안됨, 1 => 연결됨
BIT3	3 번 축의 연결여부: 0 => 연결안됨, 1 => 연결됨
BIT4	4 번 축의 연결여부: 0 => 연결안됨, 1 => 연결됨
BIT5	5 번 축의 연결여부: 0 => 연결안됨, 1 => 연결됨
BIT6	6 번 축의 연결여부: 0 => 연결안됨, 1 => 연결됨
BIT7	7 번 축의 연결여부: 0 => 연결안됨, 1 => 연결됨

RETURN VALUE

Value	Meaning

음수	수행 실패. 자세한 내용은 '에러 처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#define DEV0 0

#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

// 연결된 슬레이브 정보를 반환합니다.
cmxGnGetAxisMap (DEV0, &AxisMapMask );
```

NAME

cmxGnResetComm

- 슬레이브와의 통신을 초기화합니다.

INFORMATION

- Etc General Function
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxGnResetComm ([in] VT_I4 BoardID)

DESCRIPTION

슬레이브와의 통신을 초기화합니다

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶BoardID: 사용자가 설정한 디바이스(보드) ID.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러 처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#define DEV0 0

#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

// 연결된 슬레이브 정보를 반환합니다.
cmxGnResetComm (DEV0);
```

NAME

cmxGnSetCommStates

cmxGnGetCommStates

- 통신 상태 설정/반환

INFORMATION

- Etc General Function
- VC++/VB
- BCB/Delphi/.NET
- Level 1
- 위험요소 없음

SYNOPSIS

❒ VT_I4 cmxGnSetCommStates ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 cmxtsId, [in] VT_I4 CommStsVal)

❒ VT_I4 cmxGnGetCommStates ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 cmxtsId, [out] VT_PI4 CommStsVal)

DESCRIPTION

cmxGnSetCommStates() 함수는통신 상태를 설정합니다. cmxtsId 에 따라 현재 통신 상태, Real Time State Update Count 또는 Command Count 등을 설정할 수 있습니다.
cmxGnGetCommStates ()함수는통신 상태를 설정합니다 .

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호, 축 번호는 3 부터 시작합니다..

▶ cmxtsId : 이 매개변수를 이용하여 어떤 통신 상태를 설정할 것인지를 선택합니다.

Value	Meaning
0 또는 cmxCSID_COM_STATES	현재 통신 상태
1 또는 cmxCSID_RTS_COUNT	Real Time State Update Count
2 또는 cmxCSID_CMD_COUNT	Command Count
3 또는 cmxCSID_COM_RST	통신 리셋 (설정만 가능)

▶ CommStsVal: 선택한 cmxtsId 에 따라 각각 현재 통신 상태, Real Time State Update Count, Command Count 값을 설정합니다. cmxtsId 의 값으로 cmxCSID_RTS_COUNT, cmxCSID_CMD_COUNT 를 선택한 경우에는 상수값, cmxCSID_COM_STATES 를 선택한 경우에는 -1~4 의 값을 사용할 수 있으며 그 의미는 다음과 같습니다.

Value	Meaning
-1 또는 cmxCOM_STATE_RESET	통신 리셋
0 또는 cmxCOM_STATE_INIT	통신 초기화

1 또는 cmxCOM_STATE_WAITING	통신 대기
2 또는 cmxCOM_STATE_PREPARING	통신 준비
3 또는 cmxCOM_STATE_START	통신 시작
4 또는 cmxCOM_STATE_RUNNING	Running

▶ cmxtsId : 이 매개변수를 이용하여 어떤 통신 상태를 반환할 것인지를 선택합니다.

▶ CommStsVal: 선택한 cmxtsId 에 따라 각각 현재 통신 상태, Real Time State Update Count, Command Count 값을 반환합니다. cmxtsId 의 값으로 cmxCSID_RTS_COUNT, cmxCSID_CMD_COUNT 를 선택한 경우에는 상수값, cmxCSID_COM_STATES 를 선택한 경우에는 -1~4 의 값이 반환될 수 있으며 그 의미는 다음과 같습니다.

Value	Meaning
-1 또는 cmxCOM_STATE_RESET	통신 리셋
0 또는 cmxCOM_STATE_INIT	통신 초기화
1 또는 cmxCOM_STATE_WAITING	통신 대기
2 또는 cmxCOM_STATE_PREPARING	통신 준비
3 또는 cmxCOM_STATE_START	통신 시작
4 또는 cmxCOM_STATE_RUNNING	Running

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxGnSetParam

cmxGnGetParam

- 서보 파라미터 설정 / 반환

INFORMATION

- Etc General Function
- VC++/VB
- BCB/Delphi/.NET
- Level 1
- 다소 주의

SYNOPSIS

❒ VT_I4 cmxGnSetParam ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 ParamNo1, [in] VT_I4 ParamData1, [in] VT_I4 ParamNo2, [in] VT_I4 ParamData2)

❒ VT_I4 cmxGnGetParam ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 ParamNo1, [out] VT_PI4 ParamData1, [in] VT_I4 ParamNo2, [out] VT_PI4 ParamData2)

DESCRIPTION

cmxGnSetParam() 함수는 지정한 축의 서보 파라미터를 설정합니다. 한 번에 최대 두 개의 파라미터를 설정할 수 있습니다.
cmxGnGetParam() 함수는 지정한 축의 서보 파라미터를 반환합니다. 한 번에 최대 두 개의 파라미터를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ Axis : 축(채널) 번호. 축 번호는 3 부터 시작합니다.

▶ ParamNo1/ParamNo2 : cmxGnSetParam 함수의 인자이며, 설정하고자 하는 파라미터의 번호입니다. 파라미터 번호를 0 으로 설정하면 해당 파라미터는 설정되지 않습니다. 이에 대한 자세한 내용은 사용하시는 서보 드라이브의 매뉴얼을 참고하시기 바랍니다.

▶ ParamNo1/ParamNo2 : cmxGnGetParam 함수의 인자이며, 반환받고자 하는 파라미터의 번호입니다. 파라미터 번호를 0 으로 설정하면 해당 파라미터는 반환되지 않습니다. 이에 대한 자세한 내용은 사용하시는 서보 드라이브의 매뉴얼을 참고하시기 바랍니다.

▶ ParmaData1/ParamData2 : cmxGnSetParam 함수의 인자이며, 각 파라미터 번호에 맞는 파라미터 값을 설정합니다. 자세한 내용은 사용하시는 서보 드라이브의 매뉴얼을 참고하시기 바랍니다.

▶ ParmaData1/ParamData2 : cmxGnSetParam 함수의 인자이며, 각 파라미터 번호에 맞는 파라미터 값을 설정합니다. 자세한 내용은 사용하시는 서보 드라이브의 매뉴얼을 참고하시기 바랍니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxGnGetNodeInfo

- 해당 축의 슬레이브 정보를 반환합니다.

INFORMATION

- Etc General Function
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- ☺ 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxGnGetNodeInfo ([in] VT_I4 BoardID, [in] VT_PI4 Axis, [out] TSlaveInfo *SlaveInfo)

DESCRIPTION

해당 축에 연결된 슬레이브 정보를 반환합니다. 이 함수를 통해 해당 슬레이브의 종류(Servo/IO)와 I/O 슬레이브인 경우 슬레이브에 연결된 모듈 개수 및 연결된 모듈의 ID 와 종류를 알 수 있습니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.\

▶ Axis: Axis 번호(사용자가 서보드라이버에 설정한 값입니다).

▶ SlaveInfo: 이 매개변수를 통하여 슬레이브 정보를 반환합니다. 구조체(TSlaveInfo)로 되어 있으며 구성은 다음과 같습니다.

Name	Meaning
nSlaveType	슬레이브 종류(Servo/IO)
nNumModule	연결된 모듈 개수
ModuleType	연결된 모듈 타입(MAX_MODULE_IN_SLAVE(9) 크기의 배열) 배열의 index 는 실제 모듈의 ID 와 같음

▶ nSlaveType

Value	Meaning
0 (cmxST_SERVO)	Servo Slave
1 (cmxST_IO)	I/O(Input/Output) Slave
2 (cmxST_NONE)	None (연결되지 않음)

▶ ModuleType

Value	Meaning
0 (cmxMT_DO)	ceDO32N
1 (cmxMT_DI)	ceDO32N
2 (cmxMT_DIO)	ceD16CM

3 (cmxMT_AI)	ceAI08A
4 (cmxMT_AO)	ceAO02A
5 (cmxMT_PM)	ceMC02P
6 (cmxMT_NONE)	None(연결되지 않음)

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러 처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

Long BoardID =  0;
TSlaveInfo SlaveInfo;

// 0 번 축에 대한 정보를 반환합니다.
cmxGnGetNodeInfo(BoardID, 3, &SlaveInfo );
```

NAME

cmxGnSetLogMode

cmxGnGetLogMode

- 함수 로그 모드 설정 / 반환

INFORMATION

- Etc General Function
- VC++/VB
- BCB/Delphi/.NET
- Level 1
- 다소 주의

SYNOPSIS

- ❒ VT_I4 cmxGnSetLogMode ([in] VT_I4 LogMode)
- ❒ VT_I4 cmxGnGetLogMode ([out] VT_PI4 LogMode)

DESCRIPTION

cmxGnSetLogMode() 함수는 함수 로그를 기록할 방법을 설정합니다.
cmxGnGetLogMode() 함수는 함수 로그를 기록하는 방법을 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ LogMode : 함수 로그를 기록할 방법을 나타냅니다. 이 때 지정하는 값은 서로 OR 연산을 통하여 중복 지정할 수 있습니다.
(예) 로그를 파일과 콘솔 창에 모두 기록하고자 하는 경우 :
cmxGnSetLogMode(6); 혹은
cmxGnSetLogMode(cmxLOG_FILE | cmxLOG_CONSOLE);

Value	Meaning
0 또는 cmxLOG_DISABLE	로그를 기록하지 않습니다.
1 또는 cmxLOG_OUTPUTDEBUG	로그를 디버그 출력 창에 기록합니다.
2 또는 cmxLOG_FILE	로그를 파일로 기록합니다.
4 또는 cmxLOG_CONSOLE	로그를 콘솔 창에 기록합니다.

▶ LogMode : 함수 로그를 기록하는 방법을 나타냅니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxGnSetLogLevel

cmxGnGetLogLevel

- 함수 로그 기준 레벨 설정 / 반환

INFORMATION

- Etc General Function
- VC++/VB
- BCB/Delphi/.NET
- Level 1
- 다소 주의

SYNOPSIS

- ❒ VT_I4 cmxGnSetLogLevel ([in] VT_I4 LogLevel)
- ❒ VT_I4 cmxGnGetLogLevel ([out] VT_PI4 LogLevel)

DESCRIPTION

cmxGnSetLogLevel() 함수는 함수 로그를 기록 시 기준 레벨을 설정합니다.
cmxGnGetLogLevel() 함수는 함수 로그를 기록 시 정해진 기준 레벨을 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ LogLevel : 함수 로그를 기록할 기준 레벨을 나타냅니다. 이 때 주어지는 함수 로그 레벨은 자신보다 낮은 레벨의 기준을 포함합니다.

Value	Meaning
0 또는 cmxLL_NONE	로그를 기록하지 않습니다.
1 또는 cmxLL_ERROR	에러가 발생한 함수만 로그를 기록합니다.
2 또는 cmxLL_CMD	커맨드 함수만 로그를 기록합니다.
3 또는 cmxLL_STATUS	상태 감시 함수를 포함한 로그를 기록합니다.
4 또는 cmxLL_ALL	모든 함수에 대한 로그를 기록합니다.

▶ LogLevel : 함수 로그를 기록하는 기준 레벨을 나타냅니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxGnSetFuncLevel

cmxGnGetFuncLevel

- 특정 함수 로그 레벨 설정 / 반환

INFORMATION

- Etc General Function
- VC++/VB
- BCB/Delphi/.NET
- Level 1
- 다소 주의

SYNOPSIS

- ❒ VT_I4 cmxGnSetFuncLevel ([in] VT_I4 FuncIndex, [in] VT_I4 LogLevel)
- ❒ VT_I4 cmxGnGetFuncLevel ([in] VT_I4 FuncIndex, [out] VT_PI4 LogLevel)

DESCRIPTION

cmxGnSetFuncLevel() 함수는 해당 함수의 새로운 로그 레벨을 설정합니다.
cmxGnGetFuncLevel() 함수는 해당 함수의 로그 레벨을 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ FuncIndex : 함수 로그 레벨을 변경할 함수의 인덱스 번호를 나타냅니다.
▶ LogLevel : 해당 함수의 변경될 함수 로그 레벨을 나타냅니다

Value	Meaning
-1 또는 cmxLL_INCLUDE	해당 함수 호출 시 무조건 로그를 기록합니다.
1 또는 cmxLL_EXCLUDE	해당 함수가 호출되어도 무조건 로그를 기록하지 않습니다.

▶ LogLevel : 해당 함수의 지정된 함수 로그 레벨을 나타냅니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxGnRestoreFuncLevel

- 특정 함수 로그 레벨 원상복귀

INFORMATION

- Etc General Function
- VC++/VB
- BCB/Delphi/.NET
- Level 1
- 다소 주의

SYNOPSIS

❒ VT_I4 cmxGnRestoreFuncLevel ([in] VT_I4 FuncIndex)

DESCRIPTION

해당 함수의 로그 레벨을 원래 지정된 기본 로그 레벨로 재설정합니다.
cmxGnSetLogLevel 함수를 통하여 조정된 로그 레벨이 원래 해당 함수의 기본 로그 레벨로 지정됩니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ FuncIndex : 함수 로그 레벨을 변경할 함수의 인덱스 번호를 나타냅니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxGnTotalDIOChannel

- 해당 축의 전체 디지털 I/O(Input/Output) 채널의 개수 반환

INFORMATION

- Etc General Function
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxGnTotalDIOChannel ([in] VT_I4 BoardID, [in] VT_I4 Axis, [out] VT_PI4 Channel)

DESCRIPTION

해당 축에 연결된 전체 디지털 I/O 채널(Digital Input/Output Channel)의 개수를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(축 번호는 3 부터 시작합니다.).

▶ Channel : 이 매개변수를 통하여 디지털 I/O(Digital Input/Output) 채널의 개수를 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러 처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

SEE ALSO

cmxGnTotalAIChannel, cmxGnTotalAOChannel

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

Long BoardID = 0;
long nDioChannel = 0;

// 0 번축에서 탐색된 전체 디지털 입·출력 채널 수를 반환합니다.
cmxGnTotalDIOChannel (BoardID ,3, &nDioChannel );
```

NAME

cmxGnTotalAIChannel

- 해당 축의 전체 AI(Analog Input) 채널의 개수 반환

INFORMATION

- Etc General Function
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxGnTotalAIChannel ([in] VT_I4 BoardID, [in] VT_I4 Axis, [out] VT_PI4 Channel)

DESCRIPTION

해당 축에 연결된 전체 아날로그 입력 채널(Analog Input Channel)의 개수를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(축 번호는 3 부터 시작합니다.).

▶ Channel : 이 매개변수를 통하여 아날로그 입력(Analog Input) 채널의 개수를 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러 처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

SEE ALSO

cmxGnTotalDIOChannel, cmxGnTotalAOChannel

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

Long BoardID = 0;
long nAiChannel = 0;

// 0 번축에서 탐색된 전체 아날로그 입력 채널의 수를 반환합니다.
cmxGnTotalAIChannel (BoardID,  3, &nAiChannel );
```

cmxGnTotalAOChannel

NAME

cmxGnTotalAOChannel

- 해당 축의 전체 D/A(Analog Output) 채널의 개수 반환

INFORMATION

- Etc General Function
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxGnTotalAOChannel([in] VT_I4 BoardID, [in] VT_I4 Axis, [out] VT_PI4 Channel)

DESCRIPTION

해당 축에 연결된 전체 아날로그 출력 채널(Analog Output Channel)의 개수를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(축 번호는 3 부터 시작합니다.).

▶ Channel : 이 매개변수를 통하여 아날로그 출력(Analog Output) 채널의 개수를 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러 처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

SEE ALSO

cmxGnTotalDIOChannel, cmxGnTotalAIChannel

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

Long BoardID = 0;

long nAoChannel = 0;

// 0 번축에서 탐색된 전체 아날로그 출력 채널의 수를 반환합니다.
cmxGnTotalAOChannel (BoardID,  3, &nAoChannel );
```

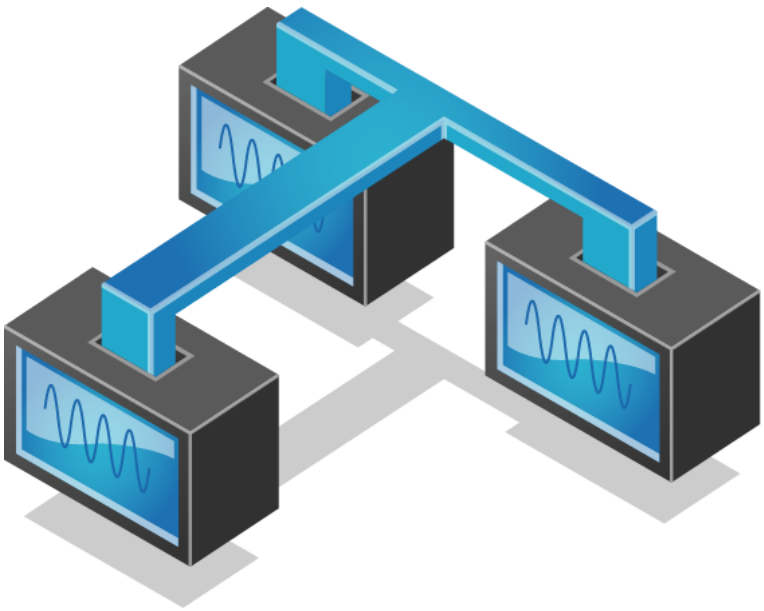
Chapter

7

Environment Configuration Functions

모션 환경설정(環境設定)의 다양한 함수는 결과적으로 ㈜커미조아의 RTEX 보드의 환경 설정이 얼마나 자세하고 명확한지를 판단하게 합니다. 기본적인 모터 드라이브를 위한 입력 펄스 신호 설정부터, 모션 주변신호까지 다양하게 지원합니다.

이 단원에서는 모션컨트롤러의 환경(環境)을 설정(設定)하는 함수(函數)들을 소개합니다. 필요에 따라 런타임시에 동적으로 환경(環境)을 변경해야할 필요가 있을 수 있으며 이러한 때에는 본 단원에서 소개하는 함수(函數)들을 이용하여 동적(動的)으로 환경(環境)을 변경할 수 있습니다.



7 환경 설정 함수 편

7.1 함수 요약

Summary of Functions
❒ VT_I4 cmxCfgSetMioProperty ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Propld, [in] VT_I4 PropVal) 모션 입출력 신호의 환경설정(環境設定)을 구성합니다.
❒ VT_I4 cmxCfgGetMioProperty ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Propld, [out] VT_PI4 PropVal) 모션 입출력 신호의 환경설정(環境設定) 값을 반환(返還)합니다.
❒ VT_I4 cmxCfgSetUnitDist ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_R8 UnitDist) 지정된 모션 축에 대한 논리적(論理的) 거리 단위를 설정(設定)합니다.
❒ VT_I4 cmxCfgGetUnitDist ([in] VT_I4 BoardID, [in] VT_I4 Axis, [out] VT_PR8 UnitDist) 지정된 모션 축에 대한 논리적(論理的) 거리 단위의 설정(設定) 상태를 반환합니다.
❒ VT_I4 cmxCfgSetUnitSpeed ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_R8 UnitSpeed) 지정된 모션 축에 대한 논리적(論理的) 속도 단위를 설정(設定)합니다.
❒ VT_I4 cmxCfgGetUnitSpeed ([in] VT_I4 BoardID, [in] VT_I4 Axis, [out] VT_PR8 UnitSpeed) 지정된 모션 축에 대한 논리적(論理的) 속도 단위의 설정(設定) 상태를 반환(返還)합니다.
❒ VT_I4 cmxCfgSetSpeedPattern ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 SpeedMode, [in] VT_R8 Work, [in] VT_R8 Acc, [in] VT_R8 Dec, [in] VT_R8 Ini, [in] VT_R8 End) 모션 이송의 전역 기준속도를 설정합니다. 이 속도의 비율(比率)을 통해 모션 이송의 실제 속도(實際速度)를 설정할 수 있습니다.
❒ VT_I4 cmxCfgGetSpeedPattern ([in] VT_I4 BoardID, [in] VT_I4 Axis, [out] VT_PI4 SpeedMode, [out] VT_PR8 Work, [out] VT_PR8 Acc, [out] VT_PR8 Dec, [out] VT_PR8 Ini, [out] VT_PR8 End) 모션 이송의 전역 기준 속도를 반환합니다. 반환된 이 속도의 비율(比率)을 통해 모션 이송의 실제 속도(實際速度)가 설정 됩니다.
❒ VT_I4 cmxCfgSetSpeedPattern_T ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 SpeedMode, [in] VT_R8 Work, [in] VT_R8 AccTime, [in] VT_R8 DecTime, [in] VT_R8 Ini, [in] VT_R8 End) 모션 이송의 전역 기준속도를 설정합니다. 이 속도의 비율(比率)을 통해 모션 이송의 실제 속도(實際速度)를 설정할 수 있습니다.
❒ VT_I4 cmxCfgGetSpeedPattern_T ([in] VT_I4 BoardID, [in] VT_I4 Axis, [out] VT_PI4 SpeedMode, [out] VT_PR8 Work, [out] VT_PR8 AccTime, [out] VT_PR8 DecTime, [out] VT_PR8 Ini, [out] VT_PR8 End) 모션 이송의 전역 기준 속도를 반환합니다. 반환된 이 속도의 비율(比率)을 통해 모션 이송의 실제 속도(實際速度)가 설정 됩니다.
❒ VT_I4 cmxCfgSetSoftLimit ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 IsEnable, [in] VT_R8 LimitN, [in] VT_R8 LimitP) 모션의 이송 범위를 소프트웨어적인 이송제한범위(移送制限範圍) 를 설정하여, 제한합니다.
❒ VT_I4 cmxCfgGetSoftLimit ([in] VT_I4 BoardID, [in] VT_I4 Axis, [out] VT_PI4 IsEnable, [out] VT_PR8 LimitN, [out] VT_PR8 LimitP) 모션의 소프트웨어적인 이송제한범위(移送制限範圍)에 대한 해당 설정을 반환합니다.

함수 설명

NAME

cmxCfgSetMioProperty

cmxCfgGetMioProperty

모션 입출력(入出力) 신호

환경설정(環境設定) 및 반환 (返還)

INFORMATION

Environment

Configuration Functions

VC++/VB

BCB/Delphi/.NET

Level 2

다소 주의

본 함수는 모션 입출력 신호의 전역 환경설정(環境設定)을 위한 함수입니다. 반드시 숙지해주시기 바랍니다.

SYNOPSIS

❒ VT_I4 cmxCfgSetMioProperty ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 PropId, [in] VT_I4 PropVal)

❒ VT_I4 cmxCfgGetMioProperty ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 PropId, [out] VT_PI4 ProVal)

DESCRIPTION

cmxCfgSetMioProperty() 각종 모션 입출력 신호에 대한 환경을 설정합니다. 이 함수는 다양한 I/O 신호의 환경을 설정하는데 공통적으로 사용하는 함수입니다. PropId 에 따라 어떠한 환경을 설정할 지를 결정하게 됩니다.

cmxCfgGetMioProperty() 함수는 각종 모션 입출력 신호에 대하여 현재 설정된 환경설정값을 반환합니다. 어떠한 I/O 의 환경설정값을 반환할 지는 PropId 에 따라 결정됩니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(축 번호는 3 부터 시작합니다).

▶ PropId : 어떠한 환경에 대하여 설정할 것인지를 지정하는 매개 변수(媒介變數)입니다. 이 값에 대해서는 아래 표를 참조하십시오.

▶ PropVal : PropId 로 지정된 환경에 대한 설정 및 반환값.

PropId	Meaning & PropVal
0 또는 cmxMIO_PEL_LOGIC	+EL 신호의 입력로직 설정값입니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (cmxLOGIC_A) : A 접점 방식 ▪ 1 (cmxLOGIC_B) : B 접점 방식

PropId	Meaning & PropVal
1 또는 cmxMIO_NEL_LOGIC	-EL 신호의 입력로직 설정값입니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (cmxLOGIC_A) : A 접점 방식 ▪ 1 (cmxLOGIC_B) : B 접점 방식
2 또는 cmxMIO_ORG_LOGIC	ORG(원점센서) 신호의 입력로직 설정값입니다. 설정 및 반환값의 PropVal 은 다음과 같습니다. ▪ 0 (cmxLOGIC_A) : A 접점 방식 ▪ 1 (cmxLOGIC_B) : B 접점 방식
3 또는 cmxMIO_EL_MODE	-/+ EL 신호가 ON 되어 정지(停止)할 때 정지(停止) 방식의 설정값입니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 : 즉시정지(停止) ▪ 1 : 감속후 정지(停止)
4 또는 cmxMIO_INP_EN	INP 신호 입력 활성화의 설정값입니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (cmxFALSE) : INP 비활성 ▪ 1 (cmxTRUE) : INP 활성 => Command 출력이 완료되더라도 INP 신호가 ON 되기 전까지는 작업이 완료되지 않은 것으로 간주.
5 또는 cmxMIO_CFSYNC_EN	서보 ON 시 Command Position 과 Feedback Position 동기화의 설정값입니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (cmxFALSE) : 동기화 비활성 ▪ 1 (cmxTRUE) : 동기화 활성 => 서보 ON 시 Feedback Position 을 Command Position 에 덮어써 Command Position 과 Feedback Position 을 일치시킨다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxCfgSetUnitDist

cmxCfgGetUnitDist

- 논리적(論理的) 거리 단위 설정(設定) 및 반환 (返還)

INFORMATION

Environment

Configuration Functions

VC++/VB

BCB/Delphi/.NET

Level 2

위험 요소 없음

SYNOPSIS

- ❒ VT_I4 cmxCfgSetUnitDist ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_R8 UnitDist)
- ❒ VT_I4 cmxCfgGetUnitDist ([in] VT_I4 BoardID, [in] VT_I4 Axis, [out] VT_PR8 UnitDist)

DESCRIPTION

cmxCfgSetUnitDist() 함수는 논리적 단위 거리에 대한 펄스 수를 설정합니다. 여기서 논리적 단위 거리라 함은 Move 함수에서 사용하는 거리 또는 위치에 대한 단위량을 의미합니다. 이 함수를 사용하여 특별히 지정하지 않는 경우에는 논리적 단위 거리에 대한 펄스 수는 초기 값인 '1' 로 사용됩니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(축 번호는 3 부터 시작합니다).

▶ UnitDist : cmxCfgSetUnitDist 함수의 인자이며, 논리적거리 1 을 이동하기 위해서 출력되어져야 하는 펄스 수를 지정합니다.

▶ UnitDist : cmxCfgGetUnitDist 함수의 인자이며, 설정되어 있는 UnitDistance 값을 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxCfgSetUnitSpeed

cmxCfgGetUnitSpeed

- 논리적(論理的) 속도 단위 설정(設定) 및 반환 (返還)

INFORMATION

- Environment Configuration Functions
- VC++/VB BCB/Delphi/.NET
- Level 2
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxCfgSetUnitSpeed ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_R8 UnitSpeed)

❒ VT_I4 cmxCfgGetUnitSpeed ([in] VT_I4 BoardID, [in] VT_I4 Axis, [out] VT_PR8 UnitSpeed)

DESCRIPTION

논리적 단위 속도에 대한 실제 펄스 출력속도(PPS)를 설정합니다. 여기서 논리적 단위 속도라 함은 속도 지정함수에서 사용하는 속도 또는 가속도에 대한 단위량을 의미합니다. 이 함수를 사용하여 특별히 지정하지 않는 경우에는 단위 속도에 대한 펄스 출력속도는 1 PPS 로 사용됩니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(축 번호는 3 부터 시작합니다).

▶ UnitSpeed : cmxCfgSetUnitSpeed 함수의 인자이며, 단위 속도에 대한 펄스 출력 속도(PPS)를 설정합니다.

▶ UnitSpeed : cmxCfgGetUnitSpeed 함수의 인자이며, 단위 속도에 대한 펄스 출력 속도(PPS)를 반환합니다.

REFERENCE

사용자의 특성(特性)에 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. cmxCfgSetUnitSpeed() 함수는 사용자가 속도의 단위를 결정하도록 하는 함수입니다. 이 함수를 다음의 예를 참고하여 사용하십시오.
Ex 1) 1 회전에 필요한 펄스 수가 3600 펄스인 경우에 속도의 단위를 RPM 으로 하고자 한다면 fUnitSpeed 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 fUnitSpeed 값을 1000 PPS 로 설정합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxCfgSetSpeedPattern

cmxCfgGetSpeedPattern

- 모션 이송 기준 속도 설정 (設定) 및 반환 (返還)

INFORMATION

Environment Configuration Functions

VC++/VB BCB/Delphi/.NET

Level 2

위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxCfgSetSpeedPattern

([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 SpeedMode, [in] VT_R8 Work, [in] VT_R8 Acc, [in] VT_R8 Dec, [in] VT_R8 Ini, [in] VT_R8 End)

❒ VT_I4 cmxCfgGetSpeedPattern

([in] VT_I4 BoardID, [in] VT_I4 Axis, [out] VT_PI4 SpeedMode, [out] VT_PR8 Work, [out] VT_PR8 Acc, [out] VT_PR8 Dec, [out] VT_PR8 Ini, [out] VT_PR8 End)

DESCRIPTION

지정한 축에 대해 속도 모드, 작업속도 및 가속 및 감속도를 설정할 수 있으며, 설정된 값을 읽을 수 있습니다. 이 속도는 각 모션제어의 기준 속도로 설정되며, 해당 기준속도에 비율로 각 모션제어를 수행할 수 있습니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(축 번호는 3 부터 시작합니다).

▶ SpeedMode : cmxCfgSetSpeedPattern 함수의 인자이며, 속도모드의 설정 값입니다. 아래와 같은 설정 값을 가집니다.

Value	Meaning
0 또는 cmxSPEED_CONSTANT	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 또는 cmxSPEED_TRAPEZOIDAL	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 또는 cmxSPEED_SCURVE	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.

가감속 패턴은 S-Curve 형과 선형 가감속 형, 가감속이 없는 형태가 가능합니다.

▶ SpeedMode : cmxCfgGetSpeedPattern 함수의 인자이며, 속도모드의 반환값입니다.

Value	Meaning
0 또는 cmxSPEED_CONSTANT	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.

1 또는 cmxSPEED_TRAPEZOIDAL	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 또는 cmxSPEED_SCURVE	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.

▶ Work : cmxCfgSetSpeedPattern 함수의 인자이며, 작업 속도를 설정합니다.

▶ Work : cmxCfgGetSpeedPattern 함수의 인자이며, 작업 속도를 반환합니다.

▶ Acc : cmxCfgSetSpeedPattern 함수의 인자이며, 가속도를 설정합니다.

▶ Acc : cmxCfgGetSpeedPattern 함수의 인자이며, 가속도를 반환합니다.

▶ Dec : cmxCfgSetSpeedPattern 함수의 인자이며, 감속도를 설정합니다.

▶ Dec : cmxCfgGetSpeedPattern 함수의 인자이며, 감속도를 반환합니다.

▶ Ini : cmxCfgSetSpeedPattern 함수의 인자이며, 초기속도를 설정합니다.

▶ Ini : cmxCfgGetSpeedPattern 함수의 인자이며, 초기속도를 반환합니다.

▶ End : cmxCfgSetSpeedPattern 함수의 인자이며, 이송 완료시 속도(최종속도)를 설정합니다.

▶End : cmxCfgGetSpeedPattern 함수의 인자이며, 이송 완료시 속도(최종속도)를 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

REFERENCE

□ 한번 설정한 속도설정은 변경하기전까지 계속해서 적용됩니다. 따라서 속도를 변경할 필요가 없는 경우에는 이송명령을 수행할때마다 속도설정을 해줄 필요는 없습니다.

□ CONSTANT speed mode
Constant speed mode 에서는 Motion 을 수행할 때 가속/감속을 적용하지 않고 일정속도로 Motion 을 수행합니다. 여기서 적용되는 일정 속도는 WorkSpeed 에서 주어진 값이 적용됩니다.

☐ TRAPEZOIDAL speed mode

Trapezoidal speed mode 에서는 Motion 을 수행하는데 있어서 속도의 패턴을 [그림 1]과 같이 Linear acceleration -> Working speed(constant) -> Linear deceleration 의 형태로 운용하는 모드입니다.

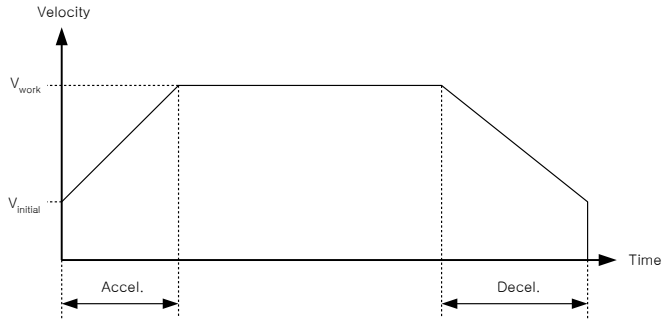


그림 7-1 Trapezoidal speed pattern

여기서 Acceleration time 은 다음과 같습니다.

Tacc = (Vwork – Vinitial)/a

Tacc : Acceleration time
Vinitial : Initial speed
Vwork : Working speed
a : Acceleration setting value

또한, 일반적으로 Vinitial 은 0 이므로, 다음 수식을 만족하며, Deceleration time 또한 위와 같은 계산식이 적용됩니다.
Tacc = Vwork /a

☐ SCURVE SPEED MODE

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 7-2 와 같이 S-curve section 과 Linear acceleration section 으로 구성됩니다.

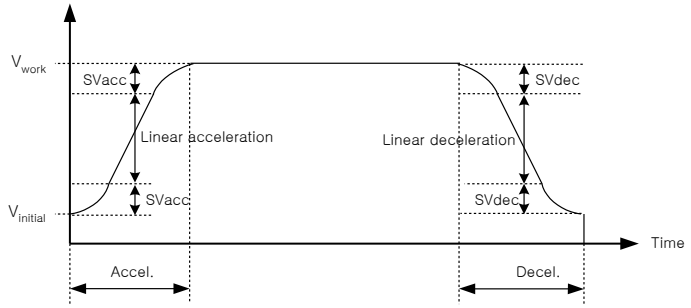


그림 7-2 S-curve speed pattern

※ S-curve speed mode 에서는 설정한 가(감)속 값이 S-curve section 을 포함한 전체 가(감)속 시간을 결정하는 매개 변수(媒介變數)로 사용되며 실제 가(감)속도 또는 Jerk 는 자동으로 계산됩니다. 전체 가속 시간 Tacc 는 다음과 같습니다.

Tacc = (Vwork – Vinitial)/a
Tacc : Acceleration time
Vinitial : Initial speed
Vwork : Working speed

a : Acceleration setting value 과 같으며 Deceleration time 또한 위와 같은 계산식이 적용됩니다.

마지막으로 SVacc 의 의미는 가속구간의 S-curve Section 을 지칭합니다.

NAME

cmxCfgSetSpeedPattern_T

cmxCfgGetSpeedPattern_T

- 모션 이송 기준 속도 설정 (設定) 및 반환 (返還)

INFORMATION

- Environment Configuration Functions
- VC++/VB BCB/Delphi/.NET
- Level 2
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxCfgSetSpeedPattern _T

([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 SpeedMode, [in] VT_R8 Work, [in] VT_R8 AccTime, [in] VT_R8 DecTime, [in] VT_R8 Ini, [in] VT_R8 End)

❒ VT_I4 cmxCfgGetSpeedPattern _T

([in] VT_I4 BoardID, [in] VT_I4 Axis, [out] VT_PI4 SpeedMode, [out] VT_PR8 Work, [out] VT_PR8 AccTime, [out] VT_PR8 DecTime, [out] VT_PR8 Ini, [out] VT_PR8 End)

DESCRIPTION

지정한 축에 대해 속도 모드, 작업속도 및 가속 및 감속 시간을 설정할 수 있으며, 설정된 값을 읽을 수 있습니다. 이 속도구간은 각 모션제어의 기준 속도로 설정되며, 해당 기준속도에 비율로 각 모션제어를 수행할 수 있습니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(축 번호는 3 부터 시작합니다).

▶ SpeedMode : cmxCfgSetSpeedPattern_T 함수의 인자이며, 속도모드의 설정 값입니다. 아래와 같은 설정 값을 가집니다.

Value	Meaning
0 또는 cmxSPEED_CONSTANT	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 또는 cmxSPEED_TRAPEZOIDAL	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 또는 cmxSPEED_SCURVE	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.

가감속 패턴은 S-Curve 형과 선형 가감속 형, 가감속이 없는 형태가 가능합니다.

▶ SpeedMode : cmxCfgGetSpeedPattern_T 함수의 인자이며, 속도모드의 반환값입니다.

Value	Meaning
0 또는 cmxSPEED_CONSTANT	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.

1 또는 cmxSPEED_TRAPEZOIDAL	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 또는 cmxSPEED_SCURVE	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.

▶ Work : cmxCfgSetSpeedPattern_T 함수의 인자이며, 작업 속도를 설정합니다.

▶ Work : cmxCfgGetSpeedPattern_T 함수의 인자이며, 작업 속도를 반환합니다.

▶ Acc : cmxCfgSetSpeedPattern_T 함수의 인자이며, 가속 시간(S)을 설정합니다.

▶ Acc : cmxCfgGetSpeedPattern_T 함수의 인자이며, 가속 시간(S)을 반환합니다.

▶ Dec : cmxCfgSetSpeedPattern_T 함수의 인자이며, 감속 시간(S)을 설정합니다.

▶ Dec : cmxCfgGetSpeedPattern_T 함수의 인자이며, 감속 시간(S)을 반환합니다.

▶ Ini : cmxCfgSetSpeedPattern_T 함수의 인자이며, 초기속도를 설정합니다.

▶ Ini : cmxCfgGetSpeedPattern_T 함수의 인자이며, 초기속도를 반환합니다.

▶ End : cmxCfgSetSpeedPattern_T 함수의 인자이며, 이송 완료시 속도(최종속도)를 설정합니다.

▶End : cmxCfgGetSpeedPattern_T 함수의 인자이며, 이송 완료시 속도(최종속도)를 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

REFERENCE

□ 한번 설정한 속도설정은 변경하기전까지 계속해서 적용됩니다. 따라서 속도를 변경할 필요가 없는 경우에는 이송명령을 수행할때마다 속도설정을 해줄 필요는 없습니다.

□ CONSTANT speed mode
Constant speed mode 에서는 Motion 을 수행할 때 가속/감속을 적용하지 않고 일정속도로 Motion 을 수행합니다. 여기서 적용되는 일정 속도는 WorkSpeed 에서 주어진 값이 적용됩니다.

NAME

cmxCfgSetSoftLimit

cmxCfgGetSoftLimit

- 소프트웨어 이송 범위(Range) 및 한계(Limit) 설정(設定) 및 반환(返還)

INFORMATION

- Environment Configuration Functions
- VC++/VB BCB/Delphi/.NET
- Level 2
- 다소 주의

SYNOPSIS

❒ VT_I4 cmxCfgSetSoftLimit

([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 IsEnable, [in] VT_R8 LimitN, [in] VT_R8 LimitP)

❒ VT_I4 cmxCfgGetSoftLimit

([in] VT_I4 BoardID, [in] VT_I4 Axis, [out] VT_PI4 IsEnable, [out] VT_PR8 LimitN, [out] VT_PR8 LimitP)

DESCRIPTION

이 함수는 소프트웨어 리밋(Limit) 기능을 활성 또는 비활성화 하고 소프트웨어 리밋 범위를 설정합니다. 소프트웨어적인 Limit 은 리밋센서의 설치가 용이하지 않을 때 안전성을 위하여 소프트웨어적인 리밋을 설정하는 것입니다. 소프트웨어적인 Limit 은 Command pulse 카운터의 절대값이 지정한 +/- Limit 값보다 같거나 크게 되면 모션을 자동 정지(停止)하도록 합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(축 번호는 3 부터 시작합니다.).

▶ IsEnable : cmxCfgSetSoftLimit 함수의 인자이며, 소프트웨어 리밋(Limit) 기능의 활성화 여부를 설정합니다.

▶ IsEnable : cmxCfgGetSoftLimit 함수의 인자이며, 소프트웨어 리밋(Limit) 기능의 활성화 여부를 반환합니다.

▶ LimitN : cmxCfgSetSoftLimit 함수의 인자이며, (-) 방향 Limit 값을 설정합니다.

▶ LimitN : cmxCfgGetSoftLimit 함수의 인자이며, (-) 방향 Limit 값을 반환합니다.

▶ LimitP : cmxCfgSetSoftLimit 함수의 인자이며, (+) 방향 Limit 값을 설정합니다.

▶ LimitP : cmxCfgGetSoftLimit 함수의 인자이며, (+) 방향 Limit 값을 반환합니다.

REFERENCE

S/W Limit 의 설정에는 항상 Unit Distance 의 값이 고려되지 않는 상황에서 문제가 발생할 수 있습니다. 만약 설정한 Unit Distance 값이 1000 으로 설정되어 있다면, 이 값에 입력된 LimitN 값과 LimitP 값이 28Bit 로 표현할 수 있는 정수값을 초과해서는 안됩니다.
이 내용을 식으로 표현하면 다음과 같습니다.

Unit Distance * S/W Limit Value < 268,435,456(28bit 정수)

위 의미는 결국 Unit Distance 와 S/W Limit 의 변수값이 28bit 정수보다 작아야 한다는 의미입니다. 본 함수의 인자가 Double 형이라고 할지라도 이 점을 반드시 주의해주시기 바랍니다. 만약 이 값이 28Bit 정수보다 크게 되면, 변수의 값이 Overflow 되어 내부에서 Negative Limit 이 Positive Limit 효과를 가져와, 모터의 축이 +/- 방향으로 움직이지 못하는 현상을 발생시킬 수 있습니다.

소프트웨어 Limit 은 논리적으로 하드웨어적인 Limit 과 동일하게 동작합니다. 또한 Negative Limit 과 Positive Limit 값은 Unit Distance 를 고려하여, 입력값의 Overflow 를 주의해야 합니다.

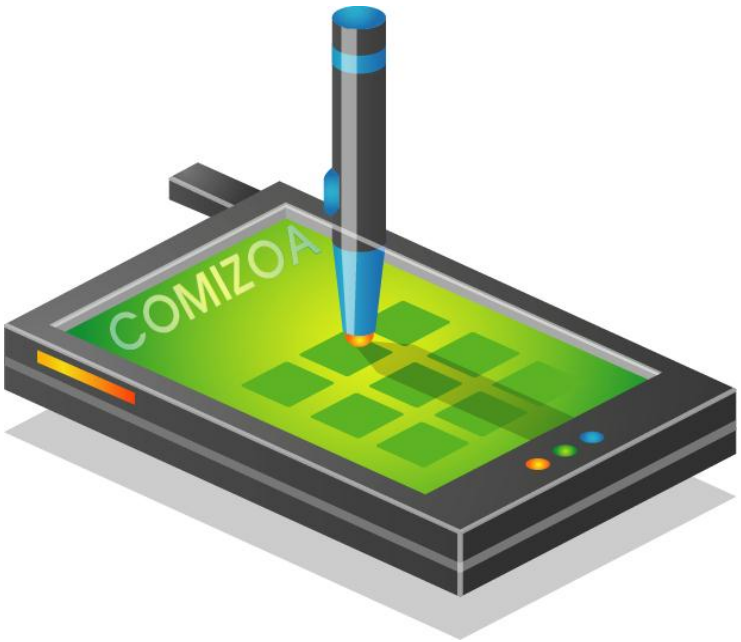
Chapter

8

Basic Motion Control

기본(基本) 단축(單軸)과 다축(多軸) 모션 제어는 모션 제어에 있어 모터 구동의 첫걸음 이자, 가장 중요한 부분입니다. 고객(顧客) 여러분들께서는 단축(單軸) 모션을 활용하여, 다축 모션과 각종 보간 제어, 특수 조건 모션 제어를 구현하실 수 있습니다. 모션제어에 필요한 속도 설정과 기본적인 모션 제어를 위한 첫단계인 본 장을 잘 활용하시기 바랍니다.

기본 모션제어에 관련된 함수들을 소개(紹介)합니다. 이 장에서는 단축 모션 제어부터 다축(多軸) 모션제어, 기본 보간 제어, 원점복귀(原點復歸)등의 내용으로 구성되어 있습니다. 단축 제어는 단일 축을 독립적으로 제어하는 작업을 의미합니다. 다축(多軸) 제어는 다수의 복수(複數) 축을 제어하는 것을 의미하며, 보간제어는 직선 보간과 원호 보간(補間) 기능(機能)으로 구성되어 있습니다. 원점복귀(原點復歸) 기능을 통해 다양한 초기 위치를 결정할 수 있습니다.



8 기본 모션 제어 편

8.1 단축(Single-Axis) 모션제어

각 속도를 설정하고 이동 함수를 사용하여 이동 작업을 수행합니다. 그리고 필요에 따라 정지(停止) 함수를 사용하여 모션을 정지(停止)합니다.

8.1.1 함수 요약

Summary of Functions
❒ VT_I4 cmxSxMove ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_R8 Distance, [in] VT_I4 IsBlocking) 단축(單軸) 상대좌표이송(相對座標移送) 을 시작합니다. 이 구동 함수는 구동 시작 후 바로 반환되지 않습니다.
❒ VT_I4 cmxSxMoveStart ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_R8 Distance) 단축(單軸) 상대좌표이송(相對座標移送) 을 시작합니다. 이 구동 함수는 구동 시작 후 바로 반환됩니다.
❒ VT_I4 cmxSxMoveTo ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_R8 Position, [in] VT_I4 IsBlocking) 단축(單軸) 절대좌표이송(絶對座標移送) 을 시작합니다. 이 구동 함수는 구동 시작 후 바로 반환되지 않습니다.
❒ VT_I4 cmxSxMoveToStart ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_R8 Position) 단축(單軸) 절대좌표이송(絶對座標移送) 을 시작합니다. 이 구동 함수는 구동 시작 후 바로 반환됩니다.
❒ VT_I4 cmxSxVMoveStart ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Dir) 단축(單軸) 연속속도이송(連續速度移送) 을 시작합니다. 이 구동 함수는 구동 시작 후 바로 반환됩니다.
❒ VT_I4 cmxSxStop ([in] VT_I4 BoardID, [in] VT_I4 Axis) 단축(單軸) 이송을 감속 후 정지(停止) 합니다. 이 정지(停止) 함수는 이송완료 (移送完了)시 까지 대기(待機) 할 수 있습니다.
❒ VT_I4 cmxSxStopEmg ([in] VT_I4 BoardID, [in] VT_I4 Axis) 단축(單軸) 이송을 비상정지(非常停止) 합니다. 이 정지(停止) 함수는 감속(減速)을 무시(無視) 합니다.
❒ VT_I4 cmxSxIsDone ([in] VT_I4 BoardID, [in] VT_I4 Axis, [out] VT_PI4 pdwIsDone) 단축(單軸) 이송의 완료(完了)를 확인(確認)합니다.
❒ VT_I4 cmxSxWaitDone ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 IsBlocking) 단축(單軸) 이송의 완료(完了) 시점까지 대기(待機)합니다.
❒ VT_I4 cmxSxSetCorrection ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 CorrMode, [in] VT_R8 CorrAmount, [in] VT_R8 CorrVel, [in] VT_I4 CntrMask) 단축(單軸) 모션의 백래쉬 혹은 슬립 보정(補正)을 위해 설정(設定)하는 함수입니다.
❒ VT_I4 cmxSxGetCorrection ([in] VT_I4 BoardID, [in] VT_I4 Axis, [out] VT_PI4 CorrMode, [out] VT_PR8 CorrAmount, [out] VT_PR8 CorrVel, [out] VT_PI4 CntrMask) 단축(單軸) 모션의 백래쉬 혹은 슬립 보정(補正)의 설정(設定)을 반환(返還)하는 함수입니다.

8.1.2 함수 설명

NAME

cmxSxMove

cmxSxMoveStart

- 단축(單軸)상대 좌표 이송(相對座標移送)

INFORMATION

Single Axis Control

VC++/VB

BCB/Delphi/.NET

Level 3

이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인(確認)합니다.

SYNOPSIS

- VT_I4 cmxSxMove ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_R8 Distance, [in] VT_I4 IsBlocking)
- VT_I4 cmxSxMoveStart ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_R8 Distance)

DESCRIPTION

하나의 축에 대하여 현재의 위치에서 지정한 거리(상대 위치)만큼 이동을 수행합니다. cmxSxMove 함수는 모션이 완료되기 전까지 반환되지 않으며, cmxSxMoveStart 함수는 모션을 시작시킨 후에 바로 반환됩니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(축 번호는 3 부터 시작합니다.).

▶ Distance : 이동할 거리를 지정합니다. 이 값은 현재의 위치에 대한 상대 좌표이며, 거리의 단위는 논리적 거리(Logic distance) 단위를 사용합니다. "Unit distance"를 1 로 한 경우에 거리의 단위는 Pulse 수가 됩니다. 즉, Distance 값 1 은 1 Pulse 출력을 의미합니다.

▶ IsBlocking : 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다.

Value	Meaning
cmxFALSE	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
cmxTRUE	블록(Blocking)을 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

SEE ALSO

cmxSxMoveTo, cmxSxMoveToStart

REFERENCE

□cmxSxMoveStart 함수를 사용하는 경우에는 cmxSxIsDone() 함수나 cmxSxWaitDone() 함수를 사용하여 모션의 완료를 확인(確認)할 수 있습니다.

□ cmxSxMove 함수를 사용하는 경우에는 내부적으로 루프를 수행하면서 모션이 완료되기를 기다리는데, 이때 "Blocking Mode"설정에 따라 윈도우 이벤트를 처리하는 방식이 달라집니다. 그러나 일반적으로 윈도우의 작업 쓰레드(Work Thread) 에서는 블록모드를 사용하여, 함수내부에서 지연없이 쓰레드 내부의 작업에 집중할 수 있도록 설정하는 것이 바람직합니다.

□ INP 입력신호가 Enable 로 설정되었으면 Command 펄스 출력이 완료되어도 INP 입력이 ON 이 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환되지 않습니다.

그러나, 모션 소프트웨어에서 INP 설정이 되어 있는 경우 EL 신호가 검출 된 후에 일부 서보 드라이브에서는 진행 방향에서 정지(停止) 한후 더 이상 움직이지 않는 상황이 발생하며, 이 상황에서 INP 신호가 출력되지 않아, 모션 이송이 완료되지 못하고, 명시적으로 STOP 을 해줘야만 하는 강제적으로 모션 종료가 되는 현상이 발생할 수 있습니다. 이 현상은 EL 모드를 통해 원점 복귀를 하는 상황에서도 발생할 수 있습니다.

따라서, 이러한 경우에는 반드시 원점 복귀나 EL 검출시에 인터럽트 이벤트나 타이머를 통해 INP 를 무기한 대기하는 현상에 대해서 적절히 대처하시거나 INP 신호 사용을 배제 해야 합니다.

윈도우 이벤트라는 것은 무엇입니까?

윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

RETURN VALUE

Value	Meaning
음수	수행 실패 또는 모션에러 발생.
ERR_NONE	수행 성공

EXAMPLE

```
C/C++
//본 예제는 cmxSxMoveStart 를 사용하여 X 축을 (+)5000 이동한 후 다시 (-)5000 만큼
이동하는 예입니다

#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

Long BoardID = 0;
/*************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.
*************************************************************/
```

```
void OnProgramInitial()
{
	long m_nNumDevices;
	long m_nNumAxes;
	long m_DeviceLis[16];
	cmxLoadDll();
	if(cmxGnLoadDevice ((&m_nNumDevices, m_DeviceList, &m_nNumAxes) != ERR_NONE)
	{
		//Handle은 사용자가 생성한 폼의 핸들 값입니다.
		// 에러메시지 출력
		return;
	}

/*************************************************************
* OnSetSpeed : 이 함수는 정격속도설정의 변경이 필요할 때
* 호출되는 가상의 함수 입니다. 이때 m_fVwork, m_fAcc, m_fDec 변수를
* 통하여 속도, 가속도 , 감속도 값이 적절하게 전달된다고 가정합니다.
*************************************************************/
void OnSetSpeed()
{
	//첫 번째 축(Axis)의 기본 속도를 설정 합니다.
	cmxCfgSetSpeedPattern(BoardID ,3, cmxMODE_S, m_fVwork, m_fAcc, m_fDec,0,0 );
}

/*************************************************************
* DoMotion : 작업명령시에 호출되는 가상의 함수 입니다.
*************************************************************/
void DoMotion()
{

	cmxSxMoveStart(BoardID ,0, 5000);//Move 5000
	if(cmxSxWaitDone(BoardID ,0, cmxFALSE) != ERR_NONE){
		// 에러메시지 출력
		return;
	}

	if(!m_bAbortMotion) //Stop 버튼이 눌리지 않았는지 확인(確認)
		cmxSxMoveStart(BoardID ,0, -5000);//Move -5000
	if(cmxSxWaitDone(BoardID ,0, cmxFALSE) != ERR_NONE){
		// 에러메시지 출력
		return;
	}

}
```

Visual Basic

```
'============================================================
'GnLoadDevice 함수로 장치를 초기화 합니다.
'============================================================

Private Sub Form_Load()
   Dim nTotalDevices As Long
   Dim DeviceList(16) As Long
   Dim nTotalAxis As Long
   Dim IRetVal As Long
```

```
   Dim BoardID As Long

BoardID = 0;

'=============================================================
 ' GnLoadDevice 함수로 장치를 초기화합니다.
  IRetVal = GnLoadDevice (nTotalDevices, DeviceList(0), nTotalAxis)

  If IRetVal <> ERR_NONE Then
     MsgBox ("GnLoadDevice has been failed")
  End If

'=============================================================
End Sub

Private Sub CfgSpeed(nTotalAxis As Long)

Dim i As Integer
 '=====================================================
   ' 이 함수에서 CfgSetSpeedPattern 함수로 속도를 설정하는 것은
        ' 모든 모션의 기준속도(Standard Speed) 가 됩니다.
   ' 단축 구동을 비롯한 대부분의 모션 동작은 이 기준 속도의 비율로
        ' 동작되게 됩니다.
   ' 아래 함수는 전체 축에 대해서 임의의 기준 속도를 설정하고 있습니다.
'=====================================================
   For i =0 To nTotalAxis -1
       Call CfgSetSpeedPattern(BoardID ,i, cmxMODE_S, 1000, 2000, 2000,0,0)
   Next
End Sub

Private Sub btnLeft_Click()
   Dim nResult As Long

   ' 왼쪽 버튼을 누르게 되면, 입력된 거리의 방향으로 이송합니다.
   nResult = SxMoveStart(BoardID , 0, 5000)
End Sub

Private Sub btnRight_Click()
 Dim nResult As Long

   ' 오른쪽 버튼을 누르게 되면, 입력된 거리의 반대 방향으로 이송합니다.
   if(SxWaitDone(BoardID , 0, cmxFALSE) != ERR_NONE) {
                    // 에러메시지 출력
                    return;
   }
   nResult = SxMoveStart(BoardID , 0, -5000)
End Sub
```

Delphi

```
// * Description :
// * CME 빌더를 통한 모션 환경설정이 되었다는 가정하에 진행합니다.
// *
// * 이 함수는 폼이 생성될때 이벤트에 의해 불려지며 , 장치를 로드하는 함수입
// * 니다.
```

```
procedure OnCreate();
var
	g_nAxis : LongInt;
	g_nDevs : LongInt;
	BoardID :  LongInt;
	DevList : Array[0..15] of LongInt

Begin
	BoardID := 0;

	// Load ComiRTEX(DLL) Library
	if (cmxGnLoadDevice (@g_nDevs, @DevList,@g_nAxis) <> ERR_NONE ) then
	begin
		// 마지막에 발생한 에러를 화면에 표시합니다.
		// 함수 인자로는 Form 의 Handle 이 전달됩니다.
		// 에러메시지 출력
		exit;
	end
end;

// * Description : 속도를 설정 하는 함수
procedure btnSetSpeedClick();
var
   fAccelSpeed : Double;
   fDecelSpeed : Double;
   fWorkSpeed : Double;
   nSMODE : LongInt;

begin
	fWorkSpeed := 50000; //각 변수들의 값을 설정 합니다.
	fAccelSpeed := 100000;
	fDecelSpeed := 100000;
	nSMODE := cmxMODE_S;
	// 설정된 기준 속도를 실제 SDK 함수에 전달합니다.

	cmxCfgSetSpeedPattern(
		0,		// 현재 Board 의 ID 를 입력합니다.
		0,		// 현재 활성화 되어 있는 채널을 선택합니다.
		nSMODE,		// 가감속이 없는 모드와 선형 가감속,
				// S-CURVE 가감속을 설정합니다.
		fWorkSpeed,	// 작업 속도를 설정합니다.
		fAccelSpeed,	// 가속도를 설정합니다.
		fDecelSpeed,	// 감속도를 설정합니다.
		0,		//초기속도를 설정합니다.
		0);		//최종속도를 설정합니다.
    end;
end;

// * Description : 이 함수는 버튼 이벤트에 의해 + 방향으로 설정된 거리만큼
// * 이동하는 함수입니다.

procedure btnPositiveClick();
	////////////////////////////////////////////////////////////
	// 저희 COMIZOA 에서는 다음과 같은 형태의 함수를 제공합니다.
	// 지령 펄스 출력 후 바로 종료를 위한 함수
	// 설명 : 위 함수는 지정된 지령 위치의 펄스 출력을 내보내고,
```

```
	//		함수가 반환됩니다.
	//		본 함수를 사용 했을 경우 사용자가 직접 위치 검출기를 통한 모션
	//		종료를 판단할 수 있는 함수는
	//		cmxSxIsDone 혹은 cmxSxWaitDone 함수가 있습니다.
	// cmxSxMoveStart [상대좌표]
	// cmxSxMoveToStart [절대좌표]
	////////////////////////////////////////////////////////////

cmxSxMoveStart
	(
	0,
	0,
	5000, // (-)일 경우 반대 방향으로 이동합니다.
	);

end;
```

NAME

cmxSxMoveTo

cmxSxMoveToStart

- 단축(單軸) 절대 좌표 이송(絶對座標移送)

INFORMATION

Single Axis Control

VC++/VB

BCB/Delphi/.NET

Level 3

이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인(確認)합니다.

SYNOPSIS

❒ VT_I4 cmxSxMoveTo ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_R8 Position, [in] VT_I4 IsBlocking)

❒ VT_I4 cmxSxMoveToStart ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_R8 Position)

DESCRIPTION

하나의 축에 대하여 지정한 절대좌표로의 이동을 수행합니다. cmxSxMoveTo() 함수는 모션이 완료되기 전까지 반환되지 않으며, cmxSxMoveToStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(축 번호는 3 부터 시작합니다).

▶ Position : 이동할 절대 좌표 값을 지정합니다. 거리의 단위는 논리적 거리(Logic distance) 단위를 사용합니다. "Unit distance"를 1 로 한 경우에 거리의 단위는 Pulse 수가 됩니다. 즉, Distance 값 1 은 1 Pulse 출력을 의미합니다.

▶ IsBlocking : 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다. 단, 쓰레드내에서 실행할 때는 이 값을 1(cmxTRUE)로 설정해주어야 합니다.

Value	Meaning
0 (cmxFALSE)	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
1 (cmxTRUE)	블록(Blocking)을 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

SEE ALSO

cmxSxMove, cmxSxMoveStart

REFERENCE

□ cmxSxMoveToStart() 함수를 사용하는 경우에는 cmxSxIsDone() 함수를 사용하여 모션의 완료를 확인(確認)할 수 있습니다.

□ cmxSxMoveTo() 함수를 사용하는 경우에는 내부적으로 루프를 수행하면서 모션이 완료되기를 기다리는데, 이때 "Blocking Mode"설정에 따라 윈도우 이벤트를 처리하는 방식이 달라집니다. 그러나 일반적으로 윈도우의 작업 쓰레드(Work Thread) 에서는 블록모드를 사용하여, 함수내부에서 지연없이 쓰레드 내부의 작업에 집중할 수 있도록 설정하는 것이 바람직합니다.

□ INP 입력신호가 Enable 로 설정되었으면 Command 펄스 출력이 완료되어도 INP 입력이 ON 이 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환되지 않습니다.

□ 스탭 드라이브를 사용 중인 고객(顧客)님들께서는 다음을 참조해 주십시요.
스탭 드라이브는 INP 출력이 없는 경우가 일반적인데, 고객(顧客)님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스탭 드라이브를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다.
고객(顧客) 여러분들께서는 스탭 드라이브 사용시에 이점을 주의해주시기를 부탁드립니다.

□ 서보 드라이브의 LSP , LSN 신호를 사용 중인 고객(顧客)님께서는 다음을 참조해 주십시요.
서보드라이브의 입력 신호 중 하나인 EL(End of Limit) 신호는 저희 ㈜커미조아 RTEX 보드 뿐만 아니라 서보드라이브에도 전달 될 수 있도록 설정할 수 있습니다. 통상적으로 LSP 신호와 LSN 신호로 불리어 지는 이 신호는 실제 기구물에서 양의 방향(Positive Direction) 혹은 음의 방향(Negative Direction) 에 장착되어 있는 EL(End of Limit) 신호를 서보 드라이브 측에 전달하기 위한 용도로 사용됩니다.

그러나, 모션 소프트웨어에서 INP 설정이 되어 있는 경우 EL 신호가 검출 된 후에 일부 서보 드라이브에서는 진행 방향에서 정지(停止) 한후 더 이상 움직이지 않는 상황이 발생하며, 이 상황에서 INP 신호가 출력되지 않아, 모션 이송이 완료되지 못하고, 명시적으로 STOP 을 해줘야만 하는 강제적으로 모션 종료가 되는 현상이 발생할 수 있습니다. 이 현상은 EL 모드를 통해 원점 복귀를 하는 상황에서도 발생할 수 있습니다.

따라서, 이러한 경우에는 반드시 원점 복귀나 EL 검출시에 인터럽트 이벤트나 타이머를 통해 INP 를 무기한 대기하는 현상에 대해서 적절히 대처하시거나 INP 신호 사용을 배제 해야 합니다.

윈도우 이벤트라는 것은 무엇입니까?
윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

EXAMPLE

본 예제는 cmxSxMoveToStart 를 사용하여 X 축을 절대좌표 1000 지점으로 이동한 후 다시 절대좌표 0 지점으로 이동하는 예입니다.

```
C/C++
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

/*************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.
*************************************************************/
Long BoardID = 0;

void OnProgramInitial()
{
	long m_nNumDevices;
	long m_DeviceList[16];
	long m_nNumAxes;
	cmxLoadDll();
	if(cmxGnLoadDevice (&m_nNumDevices, m_DeviceList, &m_nNumAxes) != ERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		// 에러메시지 출력
		return;
	}
}

/*************************************************************
* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
* 호출되는 가상의 함수 입니다. 이때 m_fVwork, m_fAcc, m_fDec 변수를
* 통하여 속도, 가속도 , 감속도 값이 적절하게 전달된다고 가정합니다.
*************************************************************/
void OnSetSpeed()
{
	//첫 번째 축(Axis)의 기본 속도를 설정 합니다.
	cmxCfgSetSpeedPattern(BoardID, 3, cmxMODE_S, m_fVwork, m_fAcc,  m_fDec,0,0 );
}

/*************************************************************
* DoMotion : 작업명령시에 호출되는 가상의 함수 입니다.
*************************************************************/
void DoMotion()
{
	if(cmxSxMoveToStart(BoardID, 3, 1000.0) != ERR_NONE){
			// 에러메시지 출력
			return ;
	}
	if(cmxSxWaitDone(BoardID, 3, cmxFALSE) != ERR_NONE){
			// 에러메시지 출력
			return ;
	}

	if(cmxSxMoveToStart(BoardID, 3, 0.0) != ERR_NONE){
			// 에러메시지 출력
			return ;
```

```
        }
        if(cmxSxWaitDone(BoardID, 3, cmxFALSE) != ERR_NONE){
                        // 에러메시지 출력
                        return ;
        }

}
```

Visual Basic

```
‘=============================================================
‘GnLoadDevice 함수로 장치를 초기화 합니다.
‘=============================================================
Private Sub Form_Load()

    Dim BoardID As Long
    Dim nTotalDevices As Long
    Dim DeviceList(16) As Long
    Dim nTotalAxis As Long
    Dim IRetVal As Long
    Dim Hwnd As Long

'=============================================================
 ' GnLoadDevice 함수로 장치를 초기화합니다.
  BoardID = 0;
  IRetVal = GnLoadDevice (nTotalDevices, DeviceList(0),  nTotalAxis)

  If IRetVal <> ERR_NONE Then
     MsgBox ("GnLoadDevicehas been failed")
  End If

'=============================================================

End Sub

Private Sub CfgSpeed(nTotalAxis As Long)
    Dim i As Integer
    '=========================================================
    ' 이 함수에서 cmxCfgSetSpeedPattern 함수로 속도를 설정하는 것은
     ‘ 모든 모션의 기준속도(Standard Speed) 가 됩니다.
    ' 단축 구동을 비롯한 대부분의 모션 동작은 이 기준 속도의 비율로
     ‘ 동작되게 됩니다.
    ' 아래 함수는 전체 축에 대해서 임의의 기준 속도를 설정하고 있습니다.
    '=========================================================
     Call CfgSetSpeedPattern(BoardID, 3, cmxMODE_S, 1000, 2000,  2000,0,0)
    Next
End Sub

Private Sub btnLeft_Click()
    Dim nResult As Long

    ' 왼쪽 버튼을 누르게 되면, 입력된 거리의 반대 방향으로 이송합니다.
    nResult = SxMoveToStart(BoardID, 3, 1000)
End Sub

Private Sub btnRight_Click()
    Dim nResult As Long
```

```
   ' 오른쪽 버튼을 누르게 되면, 입력된 거리의 정 방향으로 이송합니다.
   IRetVal = SxWaitDone(BoardID, 3, cmxFALSE)
   If IRetVal == ERR_NONE Then
        nResult = SxMoveToStart(BoardID, 3, 0)
   End If
End Sub
```

Delphi

```
// * Description :
// * 이 함수는 폼이 생성될때 이벤트에 의해 불려지며 , 장치를 로드하는 함수입
// * 니다.

procedure OnCreate();
var
        g_nDevs : LongInt;
        BoardID :  LongInt;
        DevList : Array[0..15] of LongInt
        g_nAxis : LongInt;
begin
        BoardID := 0;
        // Load ComiRTEX(DLL) Library
        if ( cmxGnLoadDevice (@g_nDevs, @DevList,@g_nAxis) <> ERR_NONE ) then
        begin
                // 마지막에 발생한 에러를 화면에 표시합니다.
                // 함수 인자로는 Form 의 Handle 이 전달됩니다.
                // 에러메시지 출력
                exit;
        end
end;

// * Description : 속도를 설정 하는 함수
procedure btnSetSpeedClick();
var
   fInitialSpeed : Double;
   fAccelSpeed : Double;
   fDecelSpeed : Double;
   fWorkSpeed : Double;
   nSMODE : LongInt;

begin
         //각 변수들의 값을 설정 합니다.
         fWorkSpeed := 50000;
         fAccelSpeed := 100000;
         fDecelSpeed := 100000;
         nSMODE := cmxMODE_S;
         // 설정된 기준 속도를 실제 SDK 함수에 전달합니다.

        cmxCfgSetSpeedPattern(
        0,                  // 현재 Board 의 ID 를 입력합니다.
        0,                  // 현재 활성화 되어 있는 채널을 선택합니다.
        nSMODE,             // 가감속이 없는 모드와 선형 가감속,
                            // S-CURVE 가감속 모드를 설정합니다.
        fWorkSpeed,         // 작업 속도를 설정합니다.
        fAccelSpeed,        // 가속도를 설정합니다.
        fDecelSpeed,        // 감속도를 설정합니다.
```

```
        0,                  //초기속도를 설정합니다.
        0);                 //최종속도를 설정합니다.

    end;
end;

// * Description :
// * 이 함수는 버튼 이벤트에 의해 + 방향으로 설정된 거리만큼 [절대좌표]로
// * 이동하는 함수입니다.

procedure btnPositiveClick();
var
    fWorkSpeedRatio : Double;
    fAccelSpeedRatio : Double;
    fDecelSpeedRatio : Double;

begin

        ////////////////////////////////////////////////////////////
        // 저희 COMIZOA 에서는 다음과 같은 형태의 함수를 제공합니다.
        // 지령 펄스 출력 후 바로 종료를 위한 함수
        // 설명 : 위 함수는 지정된 지령 위치의 펄스 출력을 내보내고,
        // 함수가 반환됩니다.
        // 본 함수를 사용 했을 경우 사용자가 직접 위치 검출기를 통한
        // 모션 종료를 판단할 수 있는 함수는
        // cmxSxIsDone 혹은 cmxSxWaitDone 함수가 있습니다.

        // cmxSxMoveStart [상대좌표]
        // cmxSxMoveToStart [절대좌표]
        ////////////////////////////////////////////////////////////


   cmxSxMoveToStart(BoardID, 3, 1000 );
   if ( cmxSxWaitDone (BoardID, 3, cmxFALSE) == ERR_NONE ) then
        begin
                cmxSxMoveToStart (BoardID, 3, 0);
        end
end;
```

NAME

cmxSxVMoveStart

- 단축(單軸) 연속속도이송(連續速度移送)

INFORMATION

Single Axis Control

VC++/VB

BCB/Delphi/.NET

Level 3

☺ 이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인(確認)합니다.

SYNOPSIS

❒ VT_I4 cmxSxVMoveStart ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Dir)

DESCRIPTION

작업속도까지 가속한 후에 작업속도를 유지하며 정지(停止)함수가 호출될 때까지 지정한 방향으로의 모션을 계속 수행합니다. 이 함수는 모션을 시작시킨 후에 바로 반환됩니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(축 번호가 3 부터 시작됩니다.).

▶ Dir : 모션의 방향을 설정합니다.

Value	Meaning
0 또는 cmDIR_N	(-) 방향 => Negative direction
1 또는 cmDIR_P	(+) 방향 => Positive direction

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

EXAMPLE

C/C++ :

```
/*************************************************************
다음의 예제는 "Jog 이동"을 하는 예입니다. 본 예제에서의
"Jog 이동"은 버튼이 눌려진 상태에서는 Axis0 축의 이동을
```

```
수행하다가, 버튼이 풀리면 이동을 멈추는 예입니다.
*************************************************************/

#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

/*************************************************************
* onprograminitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.
*************************************************************/
Long BoardID = 0;

void OnProgramInitial()
{
	long m_nNumDevices;
	long m_nNumAxes;
	long m_DeviceList[16];
	cmxLoadDll();
	if(cmxGnLoadDevice (&m_nNumDevices, m_DeviceList, &m_nNumAxes) != ERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		// 에러메시지 출력
		return;
	}
}
/*************************************************************
* OnSetSpeed(): 이 함수는 속도설정의 변경이 필요할 때 호출되는 가상의 함수입니다.
* 이때 m_fVwork, m_fAcc, m_fDec 변수를 통하여 속도, 가속도, 감속도 값이
* 적절하게 전달된다고 가정합니다.
*************************************************************/
void OnSetSpeed()
{
	//첫 번째 축(Axis)의 기본 속도를 설정 합니다.
	cmxCfgSetSpeedPattern(BoardID, 3, cmxMODE_S, m_fVwork, m_fAcc, m_fDec, 0,0);
}

/*************************************************************
* OnPlusButtonDown() : (+)Move 버튼이 눌렸을 때 호출되는 가상의 함수
* 이 함수에서 (+)방향으로 V-Move 를 시작합니다.
*************************************************************/
void OnPlusButtonDown ()
{
	cmxSxVMoveStart(BoardID, 3, cmDIR_P);//Positive dir V-MOVE
}

/*************************************************************
* OnPlusButtonUp() : (-)Move 버튼이 올라올 때 호출되는 가상의 함수
* 이 함수에서는 V-Move 를 종료합니다.
*************************************************************/
void OnPlusButtonUp ()
{
	cmxSxStop(BoardID, 3, cmxFALSE, cmxFALSE);
}

/*************************************************************
* OnMinusButtonDown(): (-)Move 버튼이 눌렸을 때 호출되는 가상의 함수
* 이 함수에서 (+)방향으로 V-Move 를 시작합니다.
```

```
************************************************************/
void OnMinusButtonDown()
{
        cmxSxVMoveStart(BoardID, 3, cmDIR_N); // Negative dir V-MOVE
}
/************************************************************
* OnMinusButtonUp() : (-)Move 버튼이 올라올 때 호출되는 가상의 함수
* 이 함수에서는 V-Move 를 종료합니다.
************************************************************/
void OnMinusButtonUp()
{
        cmxSxStop(BoardID, 3, cmxFALSE, cmxFALSE);
}
```

```
Visual Basic

‘=============================================================
‘GnLoadDevice 함수로 장치를 초기화 합니다.
‘=============================================================
Private Sub Form_Load()
    Dim BoardID As Long
    Dim nTotalDevices As Long
    Dim DeviceList(16) As Long
    Dim nTotalAxis As Long
    Dim IRetVal As Long
    Dim Hwnd As Long
'=============================================================
 ' GnLoadDevice 함수로 장치를 초기화합니다.
  BoardID = 0;
  IRetVal = GnLoadDevice(nTotalDevices, DeviceList(0), nTotalAxis)

  If IRetVal <> ERR_NONE Then
      MsgBox ("GnLoadDevice  has been failed")
  End If

'=============================================================
End Sub

Private Sub CfgSpeed(nTotalAxis As Long)
   ‘=========================================================
   ‘ 이 함수에서 CfgSetSpeedPattern 함수로 속도를 설정하는 것은
   ‘ 모든 모션의 기준속도(Standard Speed) 가 됩니다.
   ‘ 단축 구동을 비롯한 대부분의 모션 동작은 이 기준 속도의 비율로 동작되게
   ‘ 됩니다.
   ‘ 아래 함수는 전체 축에 대해서 임의의 기준 속도를 설정하고 있습니다.
   ‘=========================================================

        Call CfgSetSpeedPattern(BoardID, 3, cmxMODE_S, 1000, 2000, 2000,0,0)

End Sub

Private Sub btnStart_Click()
   ' 지정된 방향으로 연속적 속도 이동을 시작합니다. 이 이동은
    ‘ 속도 이동이기 때문에, 정지(停止) 함수가 호출될때까지 계속 이송합니다.
   Call SxVMoveStart(BoardID, 3, cmDIR_N)
```

```
End Sub
```

Delphi

```
// * 이 함수는 폼이 생성될때 이벤트에 의해 불려지며 , 장치를 로드하는 함수입
// * 니다.
procedure OnCreate();
var
        g_nDevs : LongInt;
        BoardID : LongInt;
        DevList : Array[0..15] of LongInt
        g_nAxis : LongInt;
begin
        BoardID := 0;
        // Load ComiRTEX(DLL) Library
        if ( cmxGnLoadDevice(@g_nDevs, @DevList, @g_nAxis) <> ERR_NONE ) then
        begin
                // 마지막에 발생한 에러를 화면에 표시합니다.
                // 함수 인자로는 Form 의 Handle 이 전달됩니다.
                // 에러메시지 출력
                exit;
        end
end;

// * Description : 속도를 설정 하는 함수
procedure btnSetSpeedClick();
var
   fAccelSpeed : Double;
   fDecelSpeed : Double;
   fWorkSpeed : Double;
   nSMODE : LongInt;

begin
          //각 변수들의 값을 설정 합니다.
            fWorkSpeed := 50000;
            fAccelSpeed := 100000;
            fDecelSpeed := 100000;
            nSMODE := cmxMODE_S;
           // 설정된 기준 속도를 실제 SDK 함수에 전달합니다.

        cmxCfgSetSpeedPattern(
                0,                  // 현재 Board 의 ID 를 입력합니다.
                0,                  // 현재 활성화 되어 있는 채널을 선택합니다.
                nSMODE,             // 가감속이 없는 모드와 선형 가감속,
                                    //S-CURVE 가감속을 설정합니다.
                 fWorkSpeed,        // 작업 속도를 설정합니다.
                fAccelSpeed,        // 가속도를 설정합니다.
                fDecelSpeed,        // 감속도를 설정합니다.
                0,                   //초기속도를 설정합니다.
                0) ;                //최종속도를 설정합니다.
end;

// * Description :
// * 이 함수는 버튼 이벤트에 의해 + 방향으로 설정된 거리만큼 [절대좌표]로
// * 이동하는 함수입니다.
```

```
Procedure btnPositiveClick();
var
    fWorkSpeedRatio : Double;
    fAccelSpeedRatio : Double;
    fDecelSpeedRatio : Double;

begin

      ////////////////////////////////////////////////////////////
      // 저희 COMIZOA 에서는 다음과 같은 형태의 함수를 제공합니다.
      // 지령 펄스 출력 후 바로 종료를 위한 함수
      // 설명 : 위 함수는 지정된 지령 위치의 펄스 출력을 내보내고,
      //        함수가 반환됩니다.
      //        본 함수를 사용 했을 경우 사용자가 직접 위치 검출기를 통한
      //        모션 종료를 판단할 수 있는 함수는
      //        cmxSxIsDone 혹은 cmxSxWaitDone 함수가 있습니다.
      // cmxSxMoveStart [상대좌표]
      // cmxSxMoveToStart [절대좌표]
      ////////////////////////////////////////////////////////////


   cmxSxVMoveStart
   (BoardID,
      0,
      cmDIR_N
   );

end;
```

NAME

cmxSxStop

cmxSxStopEmg

- 단축(單軸) 이송 정지(停止)

비상 정지(非常停止)

INFORMATION

Single Axis Control

VC++/VB

BCB/Delphi/.NET

Level 3

정지(停止) 함수

고속 이송시에 급 정지(停止)(비상 정지(停止)를 주의하십시요. 기구물의 손상이나 안전사고에 원인이 될 수 있습니다.

SYNOPSIS

- VT_I4 cmxSxStop ([in] VT_I4 BoardID, [in] VT_I4 Axis)
- VT_I4 cmxSxStopEmg ([in] VT_I4 BoardID, [in] VT_I4 Axis)

DESCRIPTION

지정한 축에 대한 모션을 정지(停止)합니다. cmxSxStop() 함수는 정지(停止)시에 감속 후 정지(停止)를 수행하며, cmxSxStopEmg() 함수는 감속없이 즉시정지(停止)를 수행합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(축 번호가 3 부터 시작됩니다.).

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxSxIsDone

- 단축(單軸) 모션 완료 확인(確認)

INFORMATION

- Single Axis Control
- VC++/VB
- BCB/Delphi/.NET
- Level 3
- ☺ 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxSxIsDone ([in] VT_I4 BoardID, [in] VT_I4 Axis, [out] VT_PI4 pdwIsDone)

DESCRIPTION

단일 축에 대하여 모션 완료를 확인(確認)합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(축 번호가 3 부터 시작됩니다.).

▶ pdwIsDone: 이 매개 변수로 인해 모션 작업이 완료되었는지를 판단할 수 있습니다.

Value	Meaning
0	모션작업이 완료되지 않음
1	모션작업이 완료됨

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

SEE ALSO

cmxSxWaitDone

REFERENCE

□ INP 입력신호가 Enable 로 설정되었으면 Command 펄스 출력이 완료되어도 INP 입력이 ON 이 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환되지 않습니다.

□ 스탭 드라이브를 사용 중인 고객(顧客)님들께서는 다음을 참조해 주십시요.

스탭 드라이브는 INP 출력이 없는 경우가 일반적인데, 고객(顧客)님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스탭 드라이브를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다. 고객(顧客) 여러분들께서는 스탭 드라이브 사용시에 이점을 주의해주시기를 부탁드립니다.

□ 서보 드라이브의 LSP , LSN 신호를 사용 중인 고객(顧客)님께서는 다음을 참조해 주십시요. 서보드라이브의 입력 신호 중 하나인 EL(End of Limit) 신호는 저희 ㈜커미조아 RTEX 보드 뿐만 아니라 서보드라이브에도 전달 될 수 있도록 설정할 수 있습니다. 통상적으로 LSP 신호와 LSN 신호로 불리어 지는 이 신호는 실제 기구물에서 양의 방향(Positive Direction) 혹은 음의 방향(Negative Direction) 에 장착되어 있는 EL(End of Limit) 신호를 서보 드라이브 측에 전달하기 위한 용도로 사용됩니다.

그러나, 모션 소프트웨어에서 INP 설정이 되어 있는 경우 EL 신호가 검출 된 후에 일부 서보 드라이브에서는 진행 방향에서 정지(停止) 한후 더 이상 움직이지 않는 상황이 발생하며, 이 상황에서 INP 신호가 출력되지 않아, 모션 이송이 완료되지 못하고, 명시적으로 STOP 을 해줘야만 하는 강제적으로 모션 종료가 되는 현상이 발생할 수 있습니다. 이 현상은 EL 모드를 통해 원점 복귀를 하는 상황에서도 발생할 수 있습니다.

따라서, 이러한 경우에는 반드시 원점 복귀나 EL 검출시에 인터럽트 이벤트나 타이머를 통해 INP 를 무기한 대기하는 현상에 대해서 적절히 대처하시거나 INP 신호 사용을 배제 해야 합니다.

윈도우 이벤트라는 것은 무엇입니까?

윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

EXAMPLE

```
C/C++ :

long nIsDone=0;
long BoardID = 0;

cmxSxMoveStart(BoardID, 3, 1000);

while (1){
        cmxSxIsDone(BoardID, 3, &nIsDone);
        if(nIsDone == cmxTRUE) break;
        else{
                ...
        }
}
```

```
Visual Basic
Dim Board As Long
Board = 0;
SxMoveStart(BoardID, 3, 1000)

Do Until IsDone
   ' 지정된 2 축에 대한 모션 완료 여부를 판단합니다.
```

```
   Call SxIsDone(BoardID, 3, IsDone)
…
```

Delphi

```
  // IsDone 은 모션 완료를 검사하기 위한 가상의 변수 입니다.
cmxSxMoveStart(0, 3, 1000);

While (cmxTRUE) do
Begin
        cmxSxIsDone(BoardID,  0, @IsDone);
        if ( IsDone = cmxTRUE ) then break;
…
end;
```

NAME

cmxSxWaitDone

- 단축(單軸) 모션 완료 대기(完了待機)

INFORMATION

- Single Axis Control
- VC++/VB
- BCB/Delphi/.NET
- Level 3
- ☺ 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxSxWaitDone ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 IsBlocking)

DESCRIPTION

단일 축에 대하여 모션 완료될 때까지 기다립니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(축 번호가 3 부터 시작됩니다.).

▶ IsBlocking: 완료될 때까지 기다리는 동안 윈도우 메시지를 블록할 것인지를 결정합니다.

Value	Meaning
0	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
1	블록(Blocking)을 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

SEE ALSO

cmxSxIsDone

REFERENCE

□ INP 입력신호가 Enable 로 설정되었으면 Command 펄스 출력이 완료되어도 INP 입력이 ON 이 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환되지 않습니다.

□ 스탭 드라이브를 사용 중인 고객(顧客)님들께서는 다음을 참조해 주십시요.

스탭 드라이브는 INP 출력이 없는 경우가 일반적인데, 고객(顧客)님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스탭 드라이브를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다. 고객(顧客) 여러분들께서는 스탭 드라이브 사용시에 이점을 주의해주시기를 부탁드립니다.

□ 서보 드라이브의 LSP , LSN 신호를 사용 중인 고객(顧客)님께서는 다음을 참조해 주십시요. 서보드라이브의 입력 신호 중 하나인 EL(End of Limit) 신호는 저희 ㈜커미조아 RTEX 보드 뿐만 아니라 서보드라이브에도 전달 될 수 있도록 설정할 수 있습니다. 통상적으로 LSP 신호와 LSN 신호로 불리어 지는 이 신호는 실제 기구물에서 양의 방향(Positive Direction) 혹은 음의 방향(Negative Direction) 에 장착되어 있는 EL(End of Limit) 신호를 서보 드라이브 측에 전달하기 위한 용도로 사용됩니다.

그러나, 모션 소프트웨어에서 INP 설정이 되어 있는 경우 EL 신호가 검출 된 후에 일부 서보 드라이브에서는 진행 방향에서 정지(停止) 한후 더 이상 움직이지 않는 상황이 발생하며, 이 상황에서 INP 신호가 출력되지 않아, 모션 이송이 완료되지 못하고, 명시적으로 STOP 을 해줘야만 하는 강제적으로 모션 종료가 되는 현상이 발생할 수 있습니다. 이 현상은 EL 모드를 통해 원점 복귀를 하는 상황에서도 발생할 수 있습니다.

따라서, 이러한 경우에는 반드시 원점 복귀나 EL 검출시에 인터럽트 이벤트나 타이머를 통해 INP 를 무기한 대기하는 현상에 대해서 적절히 대처하시거나 INP 신호 사용을 배제 해야 합니다.

윈도우 이벤트라는 것은 무엇입니까?

윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

EXAMPLE

C/C++

```
Long BoardID = 0;

if(cmxSxMoveStart(BoardID, 3, 5000.0) != ERR_NONE){
        // 에러메시지 출력
        return ;
}
//모션이 완료 될 때 까지 기다립니다.
if(cmxSxWaitDone(BoardID, 3, cmxFALSE) != ERR_NONE){
        // 에러메시지 출력
        return ;
}
```

Visual Basic

```
Dim BoardID As Long;
BoardID = 0
If(SxMoveStart(BoardID, 3, 1000) <> ERR_NONE) Then
        // 에러메시지 출력
        Exit Sub
```

```
End If
'Wait till motion done
If(SxWaitDone(BoardID, 3, cmxFALSE) <> ERR_NONE) Then
        // 에러메시지 출력
        Exit Sub
End If
```

Delphi

```
if(cmxSxMoveStart(0, 3, 1000) <> ERR_NONE) then begin
        // 에러메시지 출력
        exit
end;
//Wait till motion done //
if(cmxSxWaitDone(0, 3, cmxFALSE) <> ERR_NONE) then begin
        // 에러메시지 출력
        exit;
end;
```

NAME

cmxSxSetCorrection

cmxSxGetCorrection

- 백래쉬 / 슬립보정설정

INFORMATION

- Single Axis Control
- VC++/VB
- BCB/Delphi/.NET
- Level 3
- ☺ 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxSxSetCorrection

([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 CorrMode, [in] VT_R8 CorrAmount, [in] VT_R8 CorrVel, [in] VT_I4 CntrMask)

❒ VT_I4 cmxSxGetCorrection

([in] VT_I4 BoardID, [in] VT_I4 Axis, [out] VT_PI4 CorrMode, [out] VT_PR8 CorrAmount, [out] VT_PR8 CorrVel, [out] VT_PI4 CntrMask)

DESCRIPTION

cmxSxSetCorrection() 함수는 백래쉬(Backlash) 또는 슬립(Slip)에 대한 보정을 설정하는 함수입니다. 구조적으로 백래쉬나 슬립 현상이 심하게 일어나는 경우에는 이에 대한 보정이 필요할 수 있습니다.
백래쉬는 일반적으로 모터의 구동 방향이 바뀔 때 발생됩니다. 따라서 ㈜커미조아 모션컨트롤러의 백래쉬 보정은 모터의 제어 방향이 바뀔때에만 적용됩니다. 백래쉬 보정을 활성화하면 모션컨트롤러에서 지령되는 이동 명령의 이동 방향이 이전의 이동 방향과 다른 경우에 자동적으로 백래쉬 보정 설정에 따라 보정 펄스가 출력된 후에 지정된 이동을 수행합니다. 이 설정은 단축구동뿐 아니라, 다축구동, 보간구동에서도 적용됩니다.
슬립은 일반적으로 정지(停止) 후 재기동시에 발생합니다. 따라서 슬립보정은 이동방향에 상관없이 기동시에 보정 펄스가 출력됩니다. 슬립보정을 활성화한 후에 이동명령이 하달되면 모션컨트롤러는 슬립 보정 설정에 따라 보정 펄스가 출력된 후에 지정된 이동을 수행합니다.
cmxSxGetCorrection() 함수는 백래쉬/슬립 보정에 대한 현재 설정값을 읽어들이는 함수입니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(축 번호가 3 부터 시작됩니다.).

▶ CorrMode : cmxSxSetCorrection 함수의 인자이며, 보정 모드의 설정값은 다음과 같습니다.

BIT No.	Meaning
0 (cmxCORR_DISABLE)	보정기능을 비활성화합니다.
1 (cmxCORR_BACKLASH)	보정모드를 백래쉬 보정모드로 설정합니다.
2 (cmxCORR_SLIP)	보정모드를 슬립 보정모드로 설정합니다.

▶ CorrMode : cmxSxGetCorrection 함수의 인자이며, 반환값은 다음과 같습니다.

BIT No.	Meaning
0 (cmxCORR_DISABLE)	보정기능이 비활성화 상태입니다.
1 (cmxCORR_BACKLASH)	보정모드가 백래쉬 보정모드로 동작중입니다.
2 (cmxCORR_SLIP)	보정모드가 슬립 보정모드로 동작중입니다.

▶ CorrAmount : cmxSxSetCorrection 함수의 인자이며, 보정 펄스의 수를 결정합니다. 단, 이 값은 논리적 거리 단위로 설정해야 합니다. 따라서 "Unit distance"(Du)를 1 이 아닌 값으로 설정한 경우에 실제 출력되는 보정 펄스수(Nc)는 다음과 같습니다.

$$Nc = CorrAmount * Du$$

그리고 보정펄스의 수(Nc)는 0 ~ 4095 의 값이어야 합니다.

▶ CorrAmount : cmxSxGetCorrection 함수의 인자이며, 보정 펄스의 수를 반환합니다.

▶ CorrVel : cmxSxSetCorrection 함수의 인자이며, 보정펄스의 출력 주파수를 결정합니다. 단, 이 값은 논리적 속도 단위로 설정해야 합니다. 따라서 "Unit speed"(Vu)를 1 이 아닌 값으로 설정한 경우에 실제 출력 주파수(Fc)는 다음과 같습니다.

$$Fc\ (PPS) = CorrVel * Vu$$

그리고, 하드웨어적으로 보정펄스 출력 주파수를 설정하는 레지스터는 원점복귀시의 Reverse Velocity (Vr)과 같은 레지스터를 사용합니다. 따라서 원점복귀시의 Vr 이 보정펄스 출력시의 속도와 다른 경우에는 원점복귀를 수행한 후에 이 함수를 다시 수행해주어야 합니다.

▶ CorrVel : cmxSxGetCorrection 함수의 인자이며, 보정펄스의 주파수를 반환합니다.

▶ CntrMask : : cmxSxSetCorrection 함수의 인자이며, 보정펄스가 출력되는 동안에 각 카운터의 동작여부를 아래의 표와 같이 각 비트별로 설정하여 결정합니다. 예를 들어 이 값의 BIT0 을 1 로 하면 보정펄스가 출력되는 동안에도 Command Counter 의 값은 증가 또는 감소합니다. 그리고 BIT0 을 0 으로 하면 보정펄스가 출력되는 동안에는 Command Counter 의 값이 변화하지 않습니다.

Value	Meaning
BIT0	1 : 보정펄스 출력시에 Command Counter 가 동작합니다.
BIT1	1 : 보정펄스 출력시에 Feedback Counter 가 동작합니다.
BIT2	1 : 보정펄스 출력시에 Deviation Counter 가 동작합니다.
BIT3	1 : 보정펄스 출력시에 General Counter 가 동작합니다.

▶ CntrMask : cmxSxGetCorrection 함수의 인자이며, 보정펄스가 출력되는 동안에 각 카운터의 동작여부를 비트별로 반환합니다. 각 비트가 나타내는 값은 아래 표를 참고 하십시오.

Value	Meaning
BIT0	1 : 보정펄스 출력시에 Command Counter 가 동작하는 모드입니다.
BIT1	1 : 보정펄스 출력시에 Feedback Counter 가 동작하는 모드입니다.
BIT2	1 : 보정펄스 출력시에 Deviation Counter 가 동작하는 모드입니다.
BIT3	1 : 보정펄스 출력시에 General Counter 가 동작하는 모드입니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

EXAMPLE 1

C/C++

```
long BoardID = 0;
다음의 예제는 백래쉬 보정을 적용할 때의 동작에 대한 설명입니다. 본 예에서는 "Unit
distance"와 "Unit speed"가 각각 1 로 설정되었음을 가정합니다.

    // 백래쉬보정 모드로 설정 (보정펄스수:1000, 보정펄스출력속도:1000 PPS) //
    cmxSxSetCorrection (BoardID, 3, cmxORR_BACK, 1000, 1000, 0x0);

    ////////////////////////////////////////////////////////////////
    // 이전에 (-)방향 이동을 수행하였다면 아래에서 백래쉬 보정을 수행합니다.
    // 1000 PPS 의 속도로 (+)1000 펄스를 출력한 후에 지정한 SxMove()가 수행됩니다.
    cmxSxMove(BoardID, 3, 10000);

    // 이동방향이 이전과 동일하므로 아래에서는 백래쉬보정을 하지 않습니다.
    cmxSxMove(BoardID, 3, 10000);

    // 이동방향이 이전과 동일하므로 아래에서는 백래쉬보정을 하지 않습니다.
    cmxSxMove(BoardID, 3, 10000);

    ////////////////////////////////////////////////////////////////
      // 이동방향이 전환되므로 아래에서 백래쉬 보정을 수행합니다. 1000 PPS 의 속도로 (-
)1000 펄스를 출력한
      // 후에 지정한 SxMove()가 수행됩니다.
    cmxSxMove(BoardID, 3, -10000);

    // 이동방향이 이전과 동일하므로 아래에서는 백래쉬보정을 하지 않습니다.
    cmxSxMove(BoardID, 3, -10000);
```

Visual Basic

```
'다음의 예제는 백래쉬 보정을 적용할 때의 동작에 대한 설명입니다.
'본 예에서는 "Unit distance"와 "Unit speed"가 각각 1 로 설정되었음을 가정합니다.
'BoardID 는 0 으로 가정합니다.

    '// 백래쉬보정 모드로 설정 (보정펄스수:1000, 보정펄스출력속도:1000 PPS)
    Call SxSetCorrection(BoardID, 3, cmxCORR_BACK, 1000, 1000, 0)

    '////////////////////////////////////////////////////////////////
    '// 이전에 (-)방향 이동을 수행하였다면 아래에서 백래쉬 보정을 수행합니다.
    '// 1000 PPS 의 속도로 (+)1000 펄스를 출력한 후에 지정한 SxMove()가
    '// 수행됩니다.
    Call SxMove(BoardID, 3, 10000)
```

```
'// 이동방향이 이전과 동일하므로 아래에서는 백래쉬보정을 하지 않습니다.
Call SxMove(BoardID, 3, 10000)

'// 이동방향이 이전과 동일하므로 아래에서는 백래쉬보정을 하지 않습니다.
Call SxMove(BoardID, 3, 10000)

'////////////////////////////////////////////////////////////
'// 이동방향이 전환되므로 아래에서 백래쉬 보정을 수행합니다. 1000 PPS 의
'속도로(-)1000 펄스를 출력한 후에 지정한 SxMove()가 수행됩니다.
Call SxMove(BoardID, 3, -10000)
'// 이동방향이 이전과 동일하므로 아래에서는 백래쉬보정을 하지 않습니다.
Call SxMove(BoardID, 3, -10000)
```

EXAMPLE 2

다음의 예제는 슬립보정을 적용할 때의 동작에 대한 설명입니다. 본 예에서는 "Unit distance"와 "Unit speed"가 각각 1 로 설정되었음을 가정합니다.

C/C++

```
//BoardID 는 0 으로 가정합니다.

// 슬립보정 모드로 설정 (보정펄스수:1000, 보정펄스출력속도:1000 PPS) //
cmxSxSetCorrection (BoardID, 3, cmxCORR_SLIP, 1000, 1000, 0);

// (+)1000 펄스의 보정펄스가 출력된 후에 SxMove()가 수행됩니다
cmxSxMove(BoardID, 3, 10000);

// (+)1000 펄스의 보정펄스가 출력된 후에 SxMove()가 수행됩니다
cmxSxMove(BoardID, 3, 10000);

// (-)1000 펄스의 보정펄스가 출력된 후에 SxMove()가 수행됩니다
cmxSxMove(BoardID, 3, -10000);

// (-)1000 펄스의 보정펄스가 출력된 후에 SxMove()가 수행됩니다
cmxSxMove(BoardID, 3, -10000);
```

Visual Basic

```
'BoardID 는 0 으로 가정합니다.

'// 슬립보정 모드로 설정 (보정펄스수:1000, 보정펄스출력속도:1000 PPS)
Call cmxSxSetCorrection(BoardID, 3, cmxORR_SLIP, 1000, 1000, 0)

'// (+)1000 펄스의 보정펄스가 출력된 후에 SxMove()가 수행됩니다
Call cmxSxMove(BoardID, 3, 10000)

'// (+)1000 펄스의 보정펄스가 출력된 후에 SxMove()가 수행됩니다
Call cmxSxMove(BoardID, 3, 10000)

'// (-)1000 펄스의 보정펄스가 출력된 후에 SxMove()가 수행됩니다
Call cmxSxMove(BoardID, 3, -10000)
```

```
'// (-)1000 펄스의 보정펄스가 출력된 후에 SxMove()가 수행됩니다
Call cmxSxMove(BoardID, 3, -10000)
```

8.2 다축(Multi-Axes) 동시제어

이 단원에서는 다축 동시제어에 관련된 함수들을 소개합니다. 다축 동시제어는 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.

다축 동시제어 기능은 Velocity Move 와 In-position Move 모두에 적용 가능합니다. 이와 관련된 함수들은 다음과 같습니다.

※ 다축 동시제어 함수들은 단축(Single Axis) 모션제어 관련 함수들과 혼용이 가능합니다. 특히, 다축 동시제어시에도 각 축에 대한 기준 속도에 대한 설정은 cmxCfgSetSpeedPattern() 함수를 사용하여야 합니다.

8.2.1 함수 요약

Summary of Functions
❒ VT_I4 cmxMxMove ([in] VT_I4 BoardID, [in] VT_I4 NumAxes, [in] VT_PI4 NumChannel, [in] VT_PR8 DistList, [in] VT_I4 IsBlocking) 다축(多軸) 상대좌표이송(相對座標移送) 을 시작합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)되지 않습니다.
❒ VT_I4 cmxMxMoveStart ([in] VT_I4 BoardID, [in] VT_I4 NumAxes, [in] VT_PI4 NumChannel, [in] VT_PR8 DistList) 다축(多軸) 상대좌표이송(相對座標移送) 을 시작합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)됩니다.
❒ VT_I4 cmxMxMoveTo ([in] VT_I4 BoardID, [in] VT_I4 NumAxes, [in] VT_PI4 NumChannel, [in] VT_PR8 PosList, [in] VT_I4 IsBlocking) 다축(多軸) 절대좌표이송(絶對座標移送) 을 시작합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)되지 않습니다.
❒ VT_I4 cmxMxMoveToStart ([in] VT_I4 BoardID, [in] VT_I4 NumAxes, [in] VT_PI4 NumChannel, [in] VT_PR8 PosList) 다축(多軸) 절대좌표이송(絶對座標移送) 을 시작합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)됩니다.
❒ VT_I4 cmxMxVMoveStart ([in] VT_I4 BoardID, [in] VT_I4 NumAxes, [in] VT_PI4 NumChannel, [in] VT_PI4 DirList) 다축(多軸) 연속속도이송(連續速度移送) 을 시작합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)됩니다.
❒ VT_I4 cmxMxStop ([in] VT_I4 BoardID, [in] VT_I4 NumAxes, [in] VT_PI4 NumChannel) 다축(多軸) 이송을 감속 후 정지(停止) 합니다. 이 정지(停止) 함수는 이송완료 (移送完了)시 까지 대기(待機) 할 수 있습니다.
❒ VT_I4 cmxMxStopEmg ([in] VT_I4 BoardID, [in] VT_I4 NumAxes, [in] VT_PI4 NumChannel) 다축(多軸) 이송을 비상정지(非常停止) 합니다. 이 정지(停止) 함수는 감속(減速)을 무시(無視) 합니다.
❒ VT_I4 cmxMxIsDone ([in] VT_I4 BoardID, [in] VT_I4 NumAxes, [in] VT_PI4 NumChannel, [out] VT_PI4 IsDone) 다축(多軸) 이송의 완료(完了) 를 확인(確認)합니다.
❒ VT_I4 cmxMxWaitDone ([in] VT_I4 BoardID, [in] VT_I4 NumAxes, [in] VT_PI4 NumChannel, [in] VT_I4 IsBlocking) 다축(多軸) 이송의 완료(完了) 시점까지 대기(待機)합니다.

8.2.2 함수 설명

NAME

cmxMxMove

cmxMxMoveStart

- 다축(多軸) 상대 좌표 이송(相對座標移送)

INFORMATION

Multi Axes Control

VC++/VB

BCB/Delphi/.NET

Level 3

이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인(確認)합니다.

SYNOPSIS

❒ VT_I4 cmxMxMove

([in] VT_I4 BoardID, [in] VT_I4 NumAxes, [in] VT_PI4 NumChannel, [in] VT_PR8 DistList, [in] VT_I4 IsBlocking)

❒ VT_I4 cmxMxMoveStart

([in] VT_I4 BoardID, [in] VT_I4 NumAxes, [in] VT_PI4 NumChannel, [in] VT_PR8 DistList)

DESCRIPTION

여러 개의 축에 대하여 현재의 위치에서 지정한 거리만큼 이동을 동시에 시작합니다. 이 함수를 사용하면 여러 개의 축이 동시에 작업을 시작합니다. 따라서 이 함수는 여러축이 동기를 맞추어 작업을 시작해야하는 경우에 유용하게 사용될 수 있습니다. cmxMxMove() 함수는 지정한 모든 축의 모션이 완료되기 전까지 반환되지 않으며, cmxMxMoveStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ NumAxes : 동시에 작업을 수행할 대상 축의 수

▶ NumChannel : 동시에 작업을 수행할 대상 축의 배열 주소값. 이 배열의 크기는 NumAxes 값과 일치하거나 커야 합니다.

▶ DistList : 이동할 거리값의 배열 주소값. 이 배열의 크기는 NumAxes 값과 일치하거나 커야 합니다. 이동 거리값은 현재의 위치에 대한 상대 좌표이며 거리의 단위는 논리적 거리(Logic distance) 단위를 사용합니다. "Unit distance"를 1 로 한 경우에 거리의 단위는 Pulse 수가 됩니다. 즉, Distance 값 1 은 1 Pulse 출력을 의미합니다.

▶ IsBlocking : 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다.

Value	Meaning
cmxFALSE	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
cmxTRUE	블록(Blocking)을 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

REFERENCE

□ cmxMxMoveStart 함수를 사용하는 경우에는 cmxMxIsDone() 함수나 cmxMxWaitDone() 함수를 사용하여 모션의 완료를 확인(確認)할 수 있습니다.

□ cmxMxMove 함수를 사용하는 경우에는 내부적으로 루프를 수행하면서 모션이 완료되기를 기다리는데, 이때 "Blocking Mode"설정에 따라 윈도우 이벤트를 처리하는 방식이 달라집니다. 그러나 일반적으로 윈도우의 작업 쓰레드(Work Thread) 에서는 블록모드를 사용하여, 함수내부에서 지연없이 쓰레드 내부의 작업에 집중할 수 있도록 설정하는 것이 바람직합니다.

□ INP 입력신호가 Enable 로 설정되었으면 Command 펄스 출력이 완료되어도 INP 입력이 ON 이 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환되지 않습니다.

□ 스탭 드라이브를 사용 중인 고객(顧客)님들께서는 다음을 참조해 주십시요.
스탭 드라이브는 INP 출력이 없는 경우가 일반적인데, 고객(顧客)님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스탭 드라이브를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다.
고객(顧客) 여러분들께서는 스탭 드라이브 사용시에 이점을 주의해주시기를 부탁드립니다.

□ 서보 드라이브의 LSP , LSN 신호를 사용 중인 고객(顧客)님께서는 다음을 참조해 주십시요.
서보드라이브의 입력 신호 중 하나인 EL(End of Limit) 신호는 저희 ㈜커미조아 RTEX 보드 뿐만 아니라 서보드라이브에도 전달 될 수 있도록 설정할 수 있습니다. 통상적으로 LSP 신호와 LSN 신호로 불리어지는 이 신호는 실제 기구물에서 양의 방향(Positive Direction) 혹은 음의 방향(Negative Direction) 에 장착되어 있는 EL(End of Limit) 신호를 서보 드라이브 측에 전달하기 위한 용도로 사용됩니다.

그러나, 모션 소프트웨어에서 INP 설정이 되어 있는 경우 EL 신호가 검출 된 후에 일부 서보 드라이브에서는 진행 방향에서 정지(停止) 한후 더 이상 움직이지 않는 상황이 발생하며, 이 상황에서 INP 신호가 출력되지 않아, 모션 이송이 완료되지 못하고, 명시적으로 STOP 을 해줘야만 하는 강제적으로 모션 종료가 되는 현상이 발생할 수 있습니다. 이 현상은 EL 모드를 통해 원점 복귀를 하는 상황에서도 발생할 수 있습니다.

따라서, 이러한 경우에는 반드시 원점 복귀나 EL 검출시에 인터럽트 이벤트나 타이머를 통해 INP 를 무기한 대기하는 현상에 대해서 적절히 대처하시거나 INP 신호 사용을 배제 해야 합니다.

윈도우 이벤트라는 것은 무엇입니까?
윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

/************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.
************************************************************/
Long BoardID = 0;

void OnProgramInitial()
{
	long m_nNumDevices;
	long m_DeviceList[16];;
	long m_nNumAxes;
	cmxLoadDll();
	if(cmxGnLoadDevice (&m_nNumDevices, m_DeviceList, &m_nNumAxes) != ERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		// 에러메시지 출력
		return;
	}
}

/************************************************************
* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
* 호출되는 가상의 함수 입니다. 이때 m_fVwork, m_fAcc, m_fDec 변수를
* 통하여 속도, 가속도 , 감속도 값이 적절하게 전달된다고 가정합니다.
************************************************************/
void OnSetSpeed()
{
	//각 축(Axis)의 기본 속도를 설정 합니다.
	cmxCfgSetSpeedPattern(BoardID, 3, cmxMODE_S, m_fVwork, m_fAcc, m_fDec,0,0 );
	cmxCfgSetSpeedPattern(BoardID, cmY1, cmxMODE_S, m_fVwork, m_fAcc, m_fDec ,0,0);
	cmxCfgSetSpeedPattern(BoardID, cmZ1, cmxMODE_S, m_fVwork, m_fAcc, m_fDec,0,0 );

}

/************************************************************
* DoMotion : 작업명령시에 호출되는 가상의 함수 입니다.
************************************************************/
void DoMotion()
{
	long nNumChannel[3] = {0, cmY1, cmZ1};//움직일 축의 목록입니다.
	double fDistList[3] = {5000, 5000, 5000};//각축의 이동 거리입니다.
```

```
	//세개의 축을 상대거리 5000 만큼 이동 시킵니다.
	cmxMxMove(BoardID, 3, nNumChannel, fDistList, cmxFALSE);

	//반대방향으로 5000 만큼 움직이기 위해서 거리 값들을 -5000 으로 변경합니다.
	fDistList[0] = -5000; fDistList[1] = -5000; fDistList[2] = -5000;

	//세개의 축을 상대러기 -5000 만큼 이동 시킵니다.
	cmxMxMove(BoardID, 3, nNumChannel, fDistList, cmxFALSE);

	//위의 cmxMxMove()함수 대신에 cmxMxMoveStart() 함수를 사용하려면
	//아래코드를 사용합니다.
	//cmxMxMoveStart(BoardID, 3, nNumChannel, fDistList);
	//cmxMxWaitDone(BoardID, 3, NumChannel, cmxFALSE);
	//fDistList[0] = -5000; fDistList[1] = -5000; fDistList[2] = -5000;
	//cmxMxMoveStart(BoardID, 3, nNumChannel, fDistList);
	//cmxMxWaitDone(BoardID, 3, NumChannel, cmxFALSE);
}
```

Visual Basic

```
‘=============================================================
‘GnLoadDevice 함수로 장치를 초기화 합니다.
‘=============================================================

Private Sub Form_Load()
   Dim nTotalDevices As Long
   Dim BoardID As Long
   Dim DeviceList(16) As Long
   Dim nTotalAxis As Long
   Dim IRetVal As Long

'=============================================================
 ' GnDeviceLoad 함수로 장치를 초기화합니다.
  BoardID  = 0;
  IRetVal = GnLoadDevice (nTotalDevices, DeviceList(0), nTotalAxis)

  If IRetVal <> ERR_NONE Then
     MsgBox ("GnLoadDevice has been failed")
  End If

'=============================================================
End Sub

Private Sub CfgSpeed(nTotalAxis As Long)

       Dim i As Integer
       '=========================================================
       ' 이 함수에서 CfgSetSpeedPattern 함수로 속도를 설정하는 것은
       ‘ 모든 모션의 기준속도(Standard Speed) 가 됩니다.
       ' 단축 구동을 비롯한 대부분의 모션 동작은 이 기준 속도의 비율로 동작되게
       ‘ 됩니다.
        ' 아래 함수는 전체 축에 대해서 임의의 기준 속도를 설정하고 있습니다.
       '=========================================================

   For i = 0 To nTotalAxis-1
      Call CfgSetSpeedPattern(BoardID, i, cmxMODE_S, 10000, 20000, 20000,0,0)
```

```
   Next

End Sub

Private Sub btnMove_Click()

   Dim DistanceList(2) As Double
   Dim NumChannel(2) As Long

   NumChannel(0) = 0
   NumChannel(1) = 1

   DistanceList(0) = 1000
   DistanceList(1) = 1000

   ' 각 인자는 순서대로
   ' 대상축의 갯수, 대상 축의 배열, 대상 거리의 배열, 블록 여부입니다.
    Call MxMove(BoardID, 2, NumChannel(0), DistanceList(0), cmxFALSE)

End Sub
```

```
Delphi

// * 이 함수는 폼이 생성될때 이벤트에 의해 불려지며 , 장치를 로드하는 함수입
// * 니다.
procedure OnCreate();
var
        BoardID : LongInt;
        g_nDevs : LongInt;
        DevList : Array[0..15] of LongInt
        g_nAxis : LongInt;
begin
        BoardID := 0;
        // Load ComiRTEX(DLL) Library
        if (cmxGnLoadDevice (@g_nDevs, @DevList,@g_nAxis) <> ERR_NONE ) then
        begin
                // 마지막에 발생한 에러를 화면에 표시합니다.
                // 함수 인자로는 Form 의 Handle 이 전달됩니다.
                // 에러메시지 출력
                exit;
        end
end;

// * Description : 속도를 설정 하는 함수
procedure btnSetSpeedClick();
var
   fAccelSpeed : Double;
   fDecelSpeed : Double;
   fWorkSpeed : Double;
   nSMODE : LongInt;

begin
            //각 변수들의 값을 설정 합니다.
            fWorkSpeed := 50000;
            fAccelSpeed := 100000;
            fDecelSpeed := 100000;
```

```
        nSMODE := cmxMODE_S;
        // 설정된 기준 속도를 실제 SDK 함수에 전달합니다.


cmxCfgSetSpeedPattern(
      0,                // 현재 Board 의 ID 를 입력합니다.
      0,                // 현재 활성화 되어 있는 채널을 선택합니다.
      nSMODE,           // 가감속이 없는 모드와 선형 가감속, S-CURVE 가감속을 설정합니다.
      fWorkSpeed,       // 작업 속도를 설정합니다.
      fAccelSpeed,      // 가속도를 설정합니다.
      fDecelSpeed,      // 감속도를 설정합니다.
      0,                //초기속도를 설정합니다.
      0) ;              //최종속도를 설정합니다.

     cmxCfgSetSpeedPattern(
      0,                // 현재 Board 의 ID 를 입력합니다.
      cmY1,             // 현재 활성화 되어 있는 채널을 선택합니다.
      nSMODE,           // 가감속이 없는 모드와 선형 가감속,
                        //S-CURVE 가감속 모드를 설정합니다.
      fWorkSpeed,       // 작업 속도를 설정합니다.
      fAccelSpeed,      // 가속도를 설정합니다.
      fDecelSpeed,      // 감속도를 설정합니다.
      0,                //초기속도를 설정합니다.
      0) ;              //최종속도를 설정합니다.

   end;
end;

// * Description :
// * 이 함수는 버튼 이벤트에 설정된 거리만큼 이동하는 함수입니다.

Procedure btnPositiveClick();
var

      NumChannel : Array[0..1] of LongInt;
      DistanceList : Array[0..1] of Double;

begin
      NumChannel[0] := 0;
      NumChannel[1] := cmY1;
      DistanceList[0] := 1000;
      DistanceList[1] := 2000;

      cmxMxMove(BoardID, g_nTargetAxis, @NumChannel, @DistanceList, cmxFALSE);

end;
```

NAME

cmxMxMoveTo

cmxMxMoveToStart

- 다축(多軸) 절대 좌표 이송(絶對座標移送)

INFORMATION

Multi Axes Control

VC++/VB

BCB/Delphi/.NET

Level 3

이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인(確認)합니다.

SYNOPSIS

❒ VT_I4 cmxMxMoveTo

([in] VT_I4 BoardID, [in] VT_I4 NumAxes, [in] VT_PI4 NumChannel, [in] VT_PR8 PosList, [in] VT_I4 IsBlocking)

❒ VT_I4 cmxMxMoveToStart

([in] VT_I4 BoardID, [in] VT_I4 NumAxes, [in] VT_PI4 NumChannel, [in] VT_PR8 PosList)

DESCRIPTION

여러 개의 축에 대하여 지정한 절대좌표로의 이동을 시작합니다. 이 함수를 사용하면 여러 개의 축이 동시에 작업을 시작합니다. 따라서 이 함수는 여러축이 동기를 맞추어 작업을 시작해야하는 경우에 유용하게 사용될 수 있습니다.
cmxMxMoveTo() 함수는 지정한 모든 축의 모션이 완료되기 전까지 반환되지 않으며, cmxMxMoveToStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ NumAxes : 동시에 작업을 수행할 대상 축의 수

▶ NumChannel : 동시에 작업을 수행할 대상 축의 배열 주소값. 이 배열의 크기는 NumAxes 값과 일치하거나 커야 합니다.

▶ PosList : 절대좌표값의 배열 주소값. 이 배열의 크기는 NumAxes 값과 일치하거나 커야 합니다. 절대좌표의 단위는 논리적 거리(Logic distance) 단위를 사용합니다. “Unit distance”를 1 로 한 경우에 거리의 단위는 Pulse 수가 됩니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

REFERENCE

□ cmxMxMoveToStart() 함수를 사용하는 경우에는 cmxMxIsDone() 함수나 cmxMxWaitDone() 함수를 사용하여 모션의 완료를 확인(確認)할 수 있습니다.

□ cmxMxMoveTo() 함수를 사용하는 경우에는 내부적으로 루프를 수행하면서 모션이 완료되기를 기다리는데, 이때 "Blocking Mode"설정에 따라 윈도우 이벤트를 처리하는 방식이 달라집니다. 그러나 일반적으로 윈도우의 작업 쓰레드(Work Thread) 에서는 블록모드를 사용하여, 함수내부에서 지연없이 쓰레드 내부의 작업에 집중할 수 있도록 설정하는 것이 바람직합니다.

□ INP 입력신호가 Enable 로 설정되었으면 Command 펄스 출력이 완료되어도 INP 입력이 ON 이 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환되지 않습니다.

□ 스탭 드라이브를 사용 중인 고객(顧客)님들께서는 다음을 참조해 주십시요.
스탭 드라이브는 INP 출력이 없는 경우가 일반적인데, 고객(顧客)님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스탭 드라이브를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다.
고객(顧客) 여러분들께서는 스탭 드라이브 사용시에 이점을 주의해주시기를 부탁드립니다.

□ 서보 드라이브의 LSP , LSN 신호를 사용 중인 고객(顧客)님께서는 다음을 참조해 주십시요.
서보드라이브의 입력 신호 중 하나인 EL(End of Limit) 신호는 저희 ㈜커미조아 모션 보드 뿐만 아니라 서보드라이브에도 전달 될 수 있도록 설정할 수 있습니다. 통상적으로 LSP 신호와 LSN 신호로 불리어 지는 이 신호는 실제 기구물에서 양의 방향(Positive Direction) 혹은 음의 방향(Negative Direction) 에 장착되어 있는 EL(End of Limit) 신호를 서보 드라이브 측에 전달하기 위한 용도로 사용됩니다.

그러나, 모션 소프트웨어에서 INP 설정이 되어 있는 경우 EL 신호가 검출 된 후에 일부 서보 드라이브에서는 진행 방향에서 정지(停止) 한후 더 이상 움직이지 않는 상황이 발생하며, 이 상황에서 INP 신호가 출력되지 않아, 모션 이송이 완료되지 못하고, 명시적으로 STOP 을 해줘야만 하는 강제적으로 모션 종료가 되는 현상이 발생할 수 있습니다. 이 현상은 EL 모드를 통해 원점 복귀를 하는 상황에서도 발생할 수 있습니다.

따라서, 이러한 경우에는 반드시 원점 복귀나 EL 검출시에 인터럽트 이벤트나 타이머를 통해 INP 를 무기한 대기하는 현상에 대해서 적절히 대처하시거나 INP 신호 사용을 배제 해야 합니다.

윈도우 이벤트라는 것은 무엇입니까?

윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

EXAMPLE

본 예제는 cmxMxMoveTo 를 사용하여 X1,Y1,Z1 축을 절대좌표 (1000,1000,1000) 지점으로 이동한 후 다시 절대좌표 (BoardID, 3,0,0) 지점으로 이동하는 예입니다. 이때 각축의 속도와 이동거리가 같으므로 동시에 종료될 것입니다. 하지만 속도 설정이 서로 다르거나 이동거리가 서로 다른 경우에는 각축이 동시에 시작해도 종료시점은 다를 수 있습니다.

```
C/C++
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

/************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.
************************************************************/
Long BoardID = 0;
void OnProgramInitial()
{
	long m_nNumDevices;
	long m_nNumAxes;
	long m_DeviceList[16];
	cmxLoadDll();
	if(cmxGnLoadDevice (&m_nNumDevices, m_DeviceList, &m_nNumAxes) != ERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		// 에러메시지 출력
		return;
	}
}

/************************************************************
* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
* 호출되는 가상의 함수 입니다. 이때 m_fVwork, m_fAcc, m_fDec 변수를
* 통하여 속도, 가속도 , 감속도 값이 적절하게 전달된다고 가정합니다.
************************************************************/
void OnSetSpeed()
{
	//각 축(Axis)의 기본 속도를 설정 합니다.
	cmxCfgSetSpeedPattern(BoardID, 3, cmxMODE_S, m_fVwork, m_fAcc, m_fDec ,0,0);
	cmxCfgSetSpeedPattern(BoardID, cmY1, cmxMODE_S, m_fVwork, m_fAcc, m_fDec ,0, 0);

	cmxCfgSetSpeedPattern(BoardID, cmZ1, cmxMODE_S, m_fVwork, m_fAcc, m_fDec ,0,0);

}

/************************************************************
* DoMotion : 작업명령시에 호출되는 가상의 함수 입니다.
************************************************************/
void DoMotion()
{
	long nNumChannel[3] = {0, cmY1, cmZ1};//움직일 축의 목록입니다.
	double fPosList[3] = {1000, 1000, 1000};//각축의 이동할 거리입니다.

	//세개의 축을 절대좌표 5000 으로 이동 시킵니다.
	cmxMxMoveTo(BoardID, 3, NumChannel, fPosList, cmxFALSE);

	//각 축들을 절대 좌표 0 으로 움직이기 위해서 위치 값들을 0 으로 바꿉니다.
```

```
    fPosList[0] = 0; fPosList[1] = 0; fPosList[2] = 0;

    //세개의 축을 절대좌표 0 으로 이동 시킵니다.
    cmxMxMoveTo(BoardID, 3, NumChannel, fPosList, cmxFALSE);

}
```

Visual Basic

```
‘=============================================================
‘GnLoadDevice 함수로 장치를 초기화 합니다.
‘=============================================================

Private Sub Form_Load()

    Dim nTotalDevices As Long
    Dim BoardID As Long
    Dim DeviceList(16) As Long
    Dim nTotalAxis As Long
    Dim IRetVal As Long

'=============================================================
     ' GnLoadDevice 함수로 장치를 초기화합니다.
     BoardID = 0;
     IRetVal = GnLoadDevice (nTotalDevices, DeviceList(0), nTotalAxis)

     If IRetVal <> ERR_NONE Then
        MsgBox ("GnLoadDevice has been failed")
     End If

'=============================================================

End Sub

Private Sub CfgSpeed(nTotalAxis As Long)
      Dim i As Integer

    '=============================================================
     ' 이 함수에서 CfgSetSpeedPattern 함수로 속도를 설정하는 것은
     ‘ 모든 모션의 기준속도(Standard Speed) 가 됩니다.
     ' 단축 구동을 비롯한 대부분의 모션 동작은 이 기준 속도의 비율로 동작되게
     ‘ 됩니다.
     ' 아래 함수는 전체 축에 대해서 임의의 기준 속도를 설정하고 있습니다.
     '=============================================================

    For i = 0 To nTotalAxis-1
         Call CfgSetSpeedPattern(BoardID, i, cmxMODE_S, 10000, 20000, 20000,0,0)
    Next

End Sub

‘ IsBlocking 은 함수 실행 시간에 윈도우 메시지를 처리할 것인지에 대한 플래그를 말합니다.
‘ 아래는 IsBlocking 을 반환하는 가상의 함수입니다. 실제의 코드에서는 사용자나 환경
‘ 설정에서 이 설정을 반환받는 것이 옳은 방법입니다.
Dim IsBlocking As Long
```

```
Private Function GetIsBlocking() As Long

   GetIsBlocking = IsBlocking

End Function

Private Sub btnMove_Click()

      Dim DistanceList(2) As Double
      Dim NumChannel(2) As Long
      Dim nRetVal As Long

      NumChannel(0) = 0
      NumChannel(1) = 1

      DistanceList(0) = 1000
      DistanceList(1) = 1000


      ' 각 인자는 순서대로
      ' 대상축의 갯수, 대상 축의 배열, 대상 거리의 배열, 블록 여부입니다.

      nRetVal = MxMoveTo(BoardID, 2, NumChannel(0), DistanceList(0), GetIsBlocking())


End Sub
```

```
Delphi

// * Description :
// * CME 빌더를 통한 모션 환경설정이 되었다는 가정하에 진행합니다.
// *
// * 이 함수는 폼이 생성될때 이벤트에 의해 불려지며 , 장치를 로드하는 함수입
// * 니다.

procedure OnCreate();
var
      BoardID : LongInt;
      g_nDevs : LongInt;
      DevList : Array[0..15] of LongInt
      g_nAxis : LongInt;
begin

      BoardID := 0;
      // Load ComiRTEX(DLL) Library
      if (cmxGnLoadDevice (@g_nDevs, @DevList,@g_nAxis) <> ERR_NONE ) then
      begin
            // 마지막에 발생한 에러를 화면에 표시합니다.
            // 함수 인자로는 Form 의 Handle 이 전달됩니다.
            // 에러메시지 출력
            exit;
      end
end;

// * Description : 속도를 설정 하는 함수
procedure btnSetSpeedClick();
```

```
var
   fAccelSpeed : Double;
   fDecelSpeed : Double;
   fWorkSpeed : Double;
   nSMODE : LongInt;

begin
                //각 변수들의 값을 설정 합니다.
                fWorkSpeed := 50000;
                fAccelSpeed := 100000;
                fDecelSpeed := 100000;
                nSMODE := cmxMODE_S;
                // 설정된 기준 속도를 실제 SDK 함수에 전달합니다.

                cmxCfgSetSpeedPattern(
                        0,              // 현재 Board 의 ID 를 입력합니다.
                        0,              // 현재 활성화 되어 있는 채널을 선택합니다.
                        nSMODE,         // 가감속이 없는 모드와 선형 가감속,
                                        //S-CURVE 가감속을 설정합니다.
                        fWorkSpeed,     // 작업 속도를 설정합니다.
                        fAccelSpeed,    // 가속도를 설정합니다.
                        fDecelSpeed);   // 감속도를 설정합니다.
                        0,              //초기속도를 설정합니다.
                        0) ;            //최종속도를 설정합니다.

               cmxCfgSetSpeedPattern(
                        0,              // 현재 Board 의 ID 를 입력합니다.
                        cmY1,           // 현재 활성화 되어 있는 채널을 선택합니다.
                        nSMODE,         // 가감속이 없는 모드와 선형 가감속,
                                        //S-CURVE 가감속을 설정합니다.
                        fWorkSpeed,     // 작업 속도를 설정합니다.
                        fAccelSpeed,    // 가속도를 설정합니다.
                        fDecelSpeed);   // 감속도를 설정합니다.
                        0,              //초기속도를 설정합니다.
                        0) ;            //최종속도를 설정합니다.
end;
// * Description :
// * 이 함수는 버튼 이벤트에 설정된 거리만큼 이동하는 함수입니다.
// *

Procedure btnPositiveClick();
var

        NumChannel : Array[0..1] of LongInt;
        DistanceList : Array[0..1] of Double;

begin
        NumChannel[0] := 0;
        NumChannel[1] := cmY1;
        DistanceList[0] := 1000;
        DistanceList[1] := 2000;

         cmxMxMoveTo(BoardID, 2, @NumChannel, @DistanceList, cmxFALSE);

end;
```

NAME

cmxMxVMoveStart

- 다축(多軸) 연속속도이송(連續速度移送)

INFORMATION

Multi Axes Control

VC++/VB

BCB/Delphi/.NET

Level 3

이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인(確認)합니다.

SYNOPSIS

❒ VT_I4 cmxMxVMoveStart

([in] VT_I4 BoardID, [in] VT_I4 NumAxes, [in] VT_PI4 NumChannel, [in] VT_PI4 DirList)

DESCRIPTION

여러 개의 축에 대하여 Velocity Move 작업을 동시에 시작합니다. Velocity Move 는 작업속도까지 가속한 후에 작업속도를 유지하며 정지(停止)함수가 호출될 때까지 지정한 방향으로의 모션을 계속 수행합니다. 이 함수를 사용하면 여러 개의 축이 동시에 작업을 시작합니다. 따라서 이 함수는 여러축이 동기를 맞추어 작업을 시작해야하는 경우에 유용하게 사용될 수 있습니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ NumAxes : 동시에 작업을 수행할 대상 축의 수

▶ NumChannel : 동시에 작업을 수행할 대상 축의 배열 주소값. 이 배열의 크기는 NumAxes 값과 일치하거나 커야 합니다.

▶ DirList : 방향을 지시하는 값의 배열 주소값. 이 배열의 크기는 NumAxes 값과 일치하거나 커야 합니다. 모션의 방향을 지시하는 값은 다음과 같습니다.

Value	Meaning
0 또는 cmxDIR_N	(-) 방향
1 또는 cmxDIR_P	(+) 방향

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다

ERR_NONE	수행 성공

EXAMPLE

```
C/C++
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

/************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.
************************************************************/
void OnProgramInitial()
{
	Long BoardID = 0;
	long m_nNumDevices;
	long m_DeviceList[16];
	long m_nNumAxes;
	cmxLoadDll();
	if(cmxGnLoadDevice (&m_nNumDevices, m_DeviceList, &m_nNumAxes) != ERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		// 에러메시지 출력
		return;
	}
}

/************************************************************
* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
* 호출되는 가상의 함수 입니다. 이때 m_fVwork, m_fAcc, m_fDec 변수를
* 통하여 속도, 가속도 , 감속도 값이 적절하게 전달된다고 가정합니다.
************************************************************/
void OnSetSpeed()
{
	//각 축(Axis)의 기본 속도를 설정 합니다.
	cmxCfgSetSpeedPattern(BoardID, 3, cmxMODE_S, m_fVwork, m_fAcc, m_fDec ,0,0);
	cmxCfgSetSpeedPattern(BoardID, cmY1, cmxMODE_S, m_fVwork, m_fAcc, m_fDec ,0,0);

	cmxCfgSetSpeedPattern(BoardID, cmZ1, cmxMODE_S, m_fVwork, m_fAcc, m_fDec,0,0 );

}

/************************************************************
* OnDoMotion() : 작업명령시에 호출되는 가상의 함수
* 이 함수에서는 X1, Y1, Z1 축에 대하여 Velocity Move 를 시작합니다.
************************************************************/
void OnDoMotion()
{
	long nNumChannel[3] = {0, cmY1, cmZ1};
	long nDirList[3] = {cmDIR_P, cmDIR_P, cmDIR_P}; //Positive dir

	// Start V-Move of X1&Y1&Z1 //
	if(cmxMxVMoveStart(BoardID, 3, nNumChannel, nDirList) != ERR_NONE){
		// 에러메시지 출력
		return;
	}
}
```

Visual Basic

```
Private Sub CfgSpeed(nTotalAxis As Long)

   Dim i As Integer
   Dim BoardID As Long

   BoardID = 0;
   '=========================================================
   ' 이 함수에서 CfgSetSpeedPattern 함수로 속도를 설정하는 것은
   ‘ 모든 모션의 기준속도(Standard Speed) 가 됩니다.
   ' 단축 구동을 비롯한 대부분의 모션 동작은 이 기준 속도의 비율로 동작되게
   ‘ 됩니다.
   ' 아래 함수는 전체 축에 대해서 임의의 기준 속도를 설정하고 있습니다.
   '=========================================================

   For i = 0 To nTotalAxis-1
      Call CfgSetSpeedPattern(BoardID, i, cmxMODE_S, 10000, 20000, 20000,0,0)
   Next

End Sub

Private Sub btnMove_Click()

   Dim Direction(2) As Long
   Dim NumChannel(2) As Long
   Dim nRetVal As Long

   NumChannel(0) = 0
   NumChannel(1) = 1

   Direction(0) = cmDIR_P
   Direction(1) = cmDIR_P

   ' 이 함수는 지정된 속도로 정지(停止) 함수가 호출 될때까지 계속 이동합니다.
   nRetVal = MxVMoveStart(BoardID, 2, NumChannel(0), Direction(0))


End Sub
```

Delphi

```
Procedure btnSetSpeedClick();
Var
   BoardID : LongInt;
   fAccelSpeed : Double;
   fDecelSpeed : Double;
   fWorkSpeed : Double;
   nSMODE : LongInt;

begin
        BoardID := 0;
        fAccelSpeed := 30000;
        fDecelSpeed := 30000;
        fWorkSpeed := 10000;
```

```
        nSMODE := cmxMODE_S;

        // 0 을 위해 설정된 기준 속도를 실제 SDK 함수에 전달합니다.
        cmxCfgSetSpeedPattern(
        0,                      // 현재 Board 의 ID 를 입력합니다.
        0,                  // 현재 활성화 되어 있는 채널을 선택합니다.
        nSMODE,             // 가감속이 없는 모드와 선형 가감속, S-CURVE 가감속을 설정합니다.
        fWorkSpeed,         // 작업 속도를 설정합니다.
        fAccelSpeed,        // 가속도를 설정합니다.
        fDecelSpeed,       // 감속도를 설정합니다.
        0,                  //초기속도를 설정합니다.
        0) ;                //최종속도를 설정합니다.


        // cmY1 을 위해 설정된 기준 속도를 실제 SDK 함수에 전달합니다.
        cmxCfgSetSpeedPattern(
        0,                      // 현재 Board 의 ID 를 입력합니다.
        cmY1,                   // 현재 활성화 되어 있는 채널을 선택합니다.
        nSMODE,             // 가감속이 없는 모드와 선형 가감속, S-CURVE 가감속을 설정합니다.
        fWorkSpeed,         // 작업 속도를 설정합니다.
        fAccelSpeed,        // 가속도를 설정합니다.
        fDecelSpeed,       // 감속도를 설정합니다.
        0,                  //초기속도를 설정합니다.
        0) ;                //최종속도를 설정합니다.
end;

procedure FormCreate();
var
        g_nDevs : LongInt;
        DevList : Array[0..15] of LongInt
        g_nAxis : LongInt;
begin

        // Load ComiRTEX(DLL) Library
        if (cmxGnLoadDevice (@g_nDevs, @ DevList,@g_nAxis) <> ERR_NONE ) then
        begin
                // 마지막에 발생한 에러를 화면에 표시합니다.
                // 함수 인자로는 Form 의 Handle 이 전달됩니다.
                // 에러메시지 출력
                exit;
        end
end;

// * Description :
// *
// *   속도 모드로 반대 방향으로 다축(Multi Axes) 이동을 시작합니다.
procedure TForm1.btnNegativeClick(Sender: TObject);
var

  NumChannel : Array[0..1] of LongInt;
  DirList : Array[0..1] of LongInt;

  i : LongInt;
begin

      // 이송 버튼을 비활성화합니다.
```

```
    btnPositive.Enabled := FALSE;
    btnNegative.Enabled := FALSE;

    For i:=0 to 1 do begin
            DirList[i] := cmDIR_N;   // 역방향
    end;

    NumChannel[0] := 0;
    NumChannel[1] := cmY1;

    //다축을 대상으로 속도제어(지정한 속도로 정지(停止)명령이 있을 때 까지 이동)
    // 에는 다음과 같이 cmxMxVMoveStart(...) 함수를 사용합니다.
    cmxMxVMoveStart(BoardID, 2,@NumChannel,@DirList);

end;
```

NAME

cmxMxStop

cmxMxStopEmg

- 다축(多軸) 이송 정지(停止)
- 다축비상 정지(非常停止)

INFORMATION

Multi Axes Control

VC++/VB

BCB/Delphi/.NET

Level 3

정지(停止) 함수

고속 이송시에 급 정지(停止)(비상정지(停止)를 주의하십시요. 기구물의 손상이나 안전사고에 원인이 될 수 있습니다.

SYNOPSIS

- ❒ VT_I4 cmxMxStop ([in] VT_I4 BoardID, [in] VT_I4 NumAxes, [in] VT_PI4 NumChannel)
- ❒ VT_I4 cmxMxStopEmg ([in] VT_I4 BoardID, [in] VT_I4 NumAxes, [in] VT_PI4 NumChannel)

DESCRIPTION

지정한 모든 축에 대한 모션을 정지(停止)합니다. cmxMxStop() 함수는 정지(停止)시에 감속 후 정지(停止)를 수행하며, cmxMxStopEmg() 함수는 감속없이 즉시정지(停止)를 수행합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ NumAxes : 동시에 작업을 수행할 대상 축의 수

▶ NumChannel : 동시에 작업을 수행할 대상 축의 배열 주소값. 이 배열의 크기는 NumAxes 값과 일치하거나 커야 합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

REFERENCE

윈도우 이벤트라는 것은 무엇입니까?

윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

EXAMPLE

C/C++

```
Long BoardID = 0;

void CmxMotionDlg::OnStop()
{
	long nAxes[4]={0, 1, 2, 3};
	GetDlgItcmx(IDC_btnStop)->EnableWindow(FALSE);
	cmxMxStop(BoardID, 4, nAxes);
	GetDlgItcmx(IDC_btnStop)->EnableWindow(TRUE);
}
```

Visual Basic

```
Private Sub btnStop_Click()
	Dim nRetVal As Long
	Dim BoardID As Long
	Dim NumChannel(2) As Long

	BoardID = 0

	NumChannel(0) = 0
	NumChannel(1) = 1

	' 각 인자에 대한 설명을 드립니다.
	' MxStop( 축 갯수, 배열, 완료대기여부, 블록여부)

	' 1. 축 갯수
	' 다축제어에서 배열 요소에 해당하는 대상 축의 갯수입니다.

	' 2. 배열
	' 축 배열을 전달합니다. 이 배열 내부의 축은 X, Y, Z, U 축을 기본으로
	하지만 사용자가 원하는 축의 조합 (예 : X1, Y2, U1, Z1) 등의 조합의
	배열로도 전달 할 수 있습니다.
	' 단 '1. 축 갯수'는 대상 축의 총 갯수입니다

	' 3. 완료 대기 여부
	' MxStop 함수의 이 인자의 값이 True 일 경우 함수는 정지(停止) 명령을
	' 송달한 후  반환하지 않습니다.
	' 만약 False 일 경우 정지(停止) 완료까지 기다리지 않습니다.

	' 4. 블록 여부
	' 이 매뉴얼에서 설명한 내용이므로 생략합니다.
	nRetVal = MxStop(BoardID, 3, NumChannel(0))
```

```
End Sub
```

Delphi

```
// * Description :
// * 이 함수는 버튼 이벤트에 의해 모션 동작을 정지(停止)하는 함수입니다.
// *
procedure btnStopClick();
var
	BoardID : LongInt;
	NumChannel : Array[0..1] of LongInt;
	gnTargetAxis : LongInt;
begin
	BoardID := 0;
	NumChannel[0] := 0;
	NumChannel[1] := cmY1;
	gnTargetAxis := 2;

	// 정지(停止) 함수의 원형은 cmxSxStop(BoardID, [TargetAxis], [IsWaitComplete],
	// [IsBlocking]) 입니다.
	// TargetAxis : 정지(停止) 할 대상 축을 설정합니다.

	cmxMxStop(BoardID, gnTargetAxis, @NumChannel);
end;
```

NAME

cmxMxIsDone

- 다축(多軸) 모션 완료 확인(確認)

INFORMATION

Multi Axes Control
VC++/VB
BCB/Delphi/.NET
Level 3
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxMxIsDone ([in] VT_I4 BoardID, [in] VT_I4 NumAxes, [in] VT_PI4 NumChannel, [out] VT_PI4 IsDone)

DESCRIPTION

여러 개의 축에 대하여 지정한 모든 축의 모션이 완료됐는지를 확인(確認)합니다. 이 함수는 다축제어뿐 아니라 원점복귀나 단축모션제어 작업시에도 활용할 수 있습니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ NumAxes : 동시에 작업을 수행할 대상 축의 수

▶ NumChannel : 작업완료를 확인(確認)할 대상 축의 배열 주소값. 이 배열의 크기는 NumAxes 값과 일치하거나 커야 합니다.

▶ IsDone : 다축구동 완료 여부를 판단할 수 있는 매개변수 입니다.

Value	Meaning
cmxFALSE	모션작업이 완료되지 않음
cmxTRUE	모션작업이 완료됨

RETURN VLAUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

SEE ALSO

cmxMxWaitDone

EXAMPLE

C/C++

```
long nIsDone;
long BoardID = 0;
long nNumChannel[2] = {0, cmY1};
double fDistList[2] = {1000, 1000};

if(cmxMxMove(BoardID, 2, nNumChannel, fDistList, cmxFALSE) != ERR_NONE){
        //Handle은 사용자가 생성한 폼의 핸들 값입니다.
        // 에러메시지 출력
        return;
}

while (1){
    cmxMxIsDone(BoardID, 2, nNumChannel, &nIsDone);
    if(nIsDone == cmxTRUE) break;
    else{
        // 다축 모션이 종료되지 않은 경우입니다. 적절한 처리를 합니다.
    }
}
```

Visual Basic

```
Dim BoardID As Long
Dim nNumChannel(2) As Long
Dim fDistList(2) As Double

BoardID = 0
‘ 대상 축 설정
nNumChannel(0) = 0
nNumChannel(1) = cmY1

‘ 대상 축에 대한 이송 거리 설정
fDistList(0) = 1000
fDistList(1) = 1000

If(MxMove(BoardID, 2, nNumChannel(0), fDistList(0), cmxFALSE) <> ERR_NONE) Then
    // 에러메시지 출력
    Exit Sub
End If

While(MxIsDone(BoardID, 2, nNumChannel(0), cmxTRUE) <> ERR_NONE) Then
    // 에러메시지 출력
    Exit Sub;
End If
```

Delphi

```
BoardID := 0;
// 대상 축 설정
nNumChannel[0] := 0;
nNumChannel[1] := cmY1;

// 대상 축에 대한 이송 거리 설정
fDistList[0] := 1000;
```

```
fDistList[1] := 1000;

if(cmxMxMove(BoardID, 2, @nNumChannel, @fDistList) <> ERR_NONE) then begin
	// 에러메시지 출력
end;

while(cmxMxIsDone(BoardID, 2, @nNumChannel, @IsDone) <> ERR_NONE) do begin

	// 여기서 IsDone 이 cmxTRUE 이면 Loop 를 탈출하면 됩니다.
	…
end;

if(cmxErrGetLastCode() <> ERR_NONE) then begin
	// 에러메시지 출력
end;
```

NAME

cmxMxWaitDone

- 다축(多軸) 모션 완료대기(完了待機)

INFORMATION

- Multi Axes Control
- VC++/VB
- BCB/Delphi/.NET
- Level 3
- ☺ 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxMxWaitDone ([in] VT_I4 BoardID, [in] VT_I4 NumAxes, [in] VT_PI4 NumChannel, [in] VT_I4 IsBlocking)

DESCRIPTION

여러 축에 대하여 모션이 완료(完了)될 때까지 기다립니다. 이 함수는 다축제어뿐 아니라 원점복귀(原點復歸)나 단축(單軸)모션제어 작업시에도 활용할 수 있습니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ NumAxes : 동시에 작업을 수행할 대상 축의 수

▶ NumChannel : 동시에 작업을 수행할 대상 축의 배열 주소값. 이 배열의 크기는 NumAxes 값과 일치하거나 커야 합니다.

▶ IsBlocking : 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다.

Value	Meaning
cmxFALSE	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
cmxTRUE	블록(Blocking)을 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

SEE ALSO

cmxMxIsDone

REFERENCE

□ INP 입력신호가 Enable 로 설정되었으면 Command 펄스 출력이 완료되어도 INP 입력이 ON 이 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환되지 않습니다.

□ 스탭 드라이브를 사용 중인 고객(顧客)님들께서는 다음을 참조해 주십시요.
스탭 드라이브는 INP 출력이 없는 경우가 일반적인데, 고객(顧客)님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스탭 드라이브를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다.
고객(顧客) 여러분들께서는 스탭 드라이브 사용시에 이점을 주의해주시기를 부탁드립니다.

□ 서보 드라이브의 LSP , LSN 신호를 사용 중인 고객(顧客)님께서는 다음을 참조해 주십시요.
서보드라이브의 입력 신호 중 하나인 EL(End of Limit) 신호는 저희 ㈜커미조아 모션 보드 뿐만 아니라 서보드라이브에도 전달 될 수 있도록 설정할 수 있습니다. 통상적으로 LSP 신호와 LSN 신호로 불리어 지는 이 신호는 실제 기구물에서 양의 방향(Positive Direction) 혹은 음의 방향(Negative Direction) 에 장착되어 있는 EL(End of Limit) 신호를 서보 드라이브 측에 전달하기 위한 용도로 사용됩니다.

그러나, 모션 소프트웨어에서 INP 설정이 되어 있는 경우 EL 신호가 검출 된 후에 일부 서보 드라이브에서는 진행 방향에서 정지(停止) 한후 더 이상 움직이지 않는 상황이 발생하며, 이 상황에서 INP 신호가 출력되지 않아, 모션 이송이 완료되지 못하고, 명시적으로 STOP 을 해줘야만 하는 강제적으로 모션 종료가 되는 현상이 발생할 수 있습니다. 이 현상은 EL 모드를 통해 원점 복귀를 하는 상황에서도 발생할 수 있습니다.

따라서, 이러한 경우에는 반드시 원점 복귀나 EL 검출시에 인터럽트 이벤트나 타이머를 통해 INP 를 무기한 대기하는 현상에 대해서 적절히 대처하시거나 INP 신호 사용을 배제 해야 합니다.

윈도우 이벤트라는 것은 무엇입니까?
윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

EXAMPLE

C/C++

```
Long BoardID = 0;
long nNumChannel[2] = {0, cmY1};
double fDistList[2] = {1000, 1000};

if(cmxMxMoveStart(BoardID, 2, nNumChannel, fDistList) != ERR_NONE){
        //Handle 은 사용자가 생성한 폼의 핸들 값입니다.
        // 에러메시지 출력
        return ;
}

//모션이 완료 될 때 까지 기다립니다.
if(cmxMxWaitDone(BoardID, 2, nNumChannel, cmxFALSE) != ERR_NONE){
        // 에러메시지 출력
        return ;
}
```

Visual Basic

```
Dim BoardID As Long
Dim nNumChannel(1)
Dim fDistList(1)

BoardID = 0;
‘ 대상 축 설정
nNumChannel(0) = 0
nNumChannel(1) = cmY1

‘ 대상 축에 대한 이송 거리 설정
fDistList(0) = 1000
fDistList(1) = 1000

if(MxMove(BoardID, 2, nNumChannel(0), fDistList(0), cmxFALSE) <> ERR_NONE) then
        // 에러메시지 출력
        exit sub
end if

‘Wait till motion done
if(MxWaitDone(BoardID, 2, nNumChannel(0), cmxFALSE) <> ERR_NONE) then
        // 에러메시지 출력
        exit sub
end if
```

Delphi

```
BoardID := 0;
// 대상 축 설정
nNumChannel[0] := 0;
nNumChannel[1] := cmY1;

// 대상 축에 대한 이송 거리 설정
fDistList[0] := 1000;
fDistList[1] := 1000;
if(cmxMxMove(BoardID, 2, @nNumChannel, @fDistList, cmxFALSE) <> ERR_NONE) then begin
        // 에러메시지 출력
        // 적절한 에러처리를 수행합니다.
end;
// Wait till motion done //
if(cmxMxWaitDone(BoardID, 2, @nNumChannel, cmxFALSE) <> ERR_NONE) then begin
        // 에러메시지 출력
        exit;
end;
```

8.3 기본 보간제어 **(Interpolation Motion)**

이 단원에서는 보간(Interpolation) 모션제어에 관련된 함수들을 소개합니다. 보간(Interpolation) 모션제어란 두 축 이상의 축이 연동되어 직선 보간(Linear Interpolation), 원호 보간(Circular Interpolation) 등의 모션을 수행하는 것을 의미합니다.

여러축을 동시에 제어한다는 면에서는 다축동시제어와 비슷한 기능이지만 보간작업은 사용자가 원하는 경로를 추종하기 위해서 보간이동에 관련된 각 축의 속도가 자동으로 조절되면서 이동을 수행한다는 점이 다축동시제어와 가장 큰 차이점입니다.

㈜커미조아의 RTEX 보드는 모션 축을 대상으로 총 32 개의 축(Axes) 을 대상으로 보간제어 기능을 제공합니다. 당사의 RTEX 보드가 제공하는 보간제어의 사양표는 다음과 같습니다.

<table>
<tr><th>보간 제어 형태</th><th colspan="2">축 구성 범위</th><th>축 구성 제한 사항
(해당 축 범위 이내)</th></tr>
<tr><td>직선 보간</td><td colspan="2">32 축(Axes) 이내</td><td>제한 없음</td></tr>
<tr><td>원호 보간</td><td colspan="2">2 축</td><td>제한 없음</td></tr>
<tr><td rowspan="2">확장 보간제어</td><td>헬리컬</td><td>3 축</td><td rowspan="2">제한 없음</td></tr>
<tr><td>스플라인</td><td>2 축</td></tr>
</table>

㈜커미조아의 모션보드는 축 구성에 제약이 없는 직선보간 이동, 2 축 원호보간 이동, 3 축 또는 4 축의 헬리컬보간 이동작업을 자동으로 수행하는 기능을 제공합니다. 또한 직선보간과 리스트모션(Listed Motion) 기능을 응용하여 스플라인 보간이동을 수행할 수 있도록하는 기능도 제공합니다. 직선보간과 원호보간은 “기본보간제어” 기능으로 분류되며, 헬리컬보간과 스플라인보간은 “확장보간제어” 기능으로 분류됩니다. “기본보간제어” 기능과 “확장보간제어” 기능은 사용법이 약간차이가 있으며, 본 단원에는 “기본보간제어” 기능 관련 함수들에 대해서만 설명합니다. “확장보간제어” 기능에 대한 설명은 고급 모션제어의 확장 보간제어에 대한 설명을 참조하시기 바랍니다.

8.3.1 함수 요약

"기본보간제어"에 관련된 함수들은 모두 "Ix..."의 형식으로 이름이 구성되었으며 그 리스트는 다음과 같습니다.

Summary of Functions
❒ VT_I4 cmxIxMapAxes ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_I4 MapMask, [in] VT_I4 IxMode) 보간(補間) 대상 축 그룹을(Group) 설정합니다.
❒ VT_I4 cmxIxUnMapAxes ([in] VT_I4 BoardID, [in] VT_I4 MapIndex) 보간(補間) 대상 축 그룹을(Group) 해제합니다.
❒ VT_I4 cmxIxGetMapIndex ([in] VT_I4 BoardID, [in] VT_I4 Axis, [out] VT_PI4 MapIndex) 대상 축의 그룹 번호를 반환합니다.
❒ VT_I4 cmxIxSetSpeedPattern ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_I4 IsVectorSpeed, [in] VT_I4 SpeedMode, [in] VT_R8 IniRatio, [in] VT_R8 EndRatio, [in] VT_R8 VelRatio, [in] VT_R8 AccRatio, [in] VT_R8 DecRatio) 보간(補間) 이송 속도(移送速度)를 설정합니다. 보간 이송 속도는 마스터 속도 모드와 백터 속도모드를 설정(設定)할 수 있습니다.
❒ VT_I4 cmxIxGetSpeedPattern ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [out] VT_PI4 IsVectorSpeed, [out] VT_PI4 SpeedMode, [out] VT_PR8 IniRatio, [out] VT_PR8 EndiRatio, [out] VT_PR8 VelRatio, [out] VT_PR8 AccRatio, [out] VT_PR8 DecRatio) 설정된 보간(補間) 이송 속도(移送速度)를 반환합니다. 보간 이송 속도는 마스터 속도 혹은 백터 모드에 해당하는 속도모드를 반환(返還)합니다.
❒ VT_I4 cmxIxSetSpeedPattern_T ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_I4 IsVectorSpeed, [in] VT_I4 SpeedMode, [in] VT_R8 IniRatio, [in] VT_R8 EndRatio, [in] VT_R8 VelRatio, [in] VT_R8 AccTime, [in] VT_R8 DecTime) 보간(補間) 이송 속도(移送速度)를 설정합니다. 보간 이송 속도는 마스터 속도 모드와 백터 속도모드를 설정(設定)할 수 있습니다.
❒ VT_I4 cmxIxGetSpeedPattern_T ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [out] VT_PI4 IsVectorSpeed, [out] VT_PI4 SpeedMode, [out] VT_PR8 IniRatio, [out] VT_PR8 EndiRatio, [out] VT_PR8 VelRatio, [out] VT_PR8 AccTime, [out] VT_PR8 DecTime) 설정된 보간(補間) 이송 속도(移送速度)를 반환합니다. 보간 이송 속도는 마스터 속도 혹은 백터 모드에 해당하는 속도모드를 반환(返還)합니다.
❒ VT_I4 cmxIxLine ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_PR8 DistList, [in] VT_I4 IsBlocking) 보간(補間) 제어에 있어 직선 보간을 수행하며, 상대(相對的) 좌표 이송(相對座標移送)을 수행합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)되지 않습니다.
❒ VT_I4 cmxIxLineStart ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_PR8 DistList) 보간(補間) 제어에 있어 직선 보간을 수행하며, 상대(相對的) 좌표 이송(相對座標移送)을 수행합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)됩니다.
❒ VT_I4 cmxIxLineTo ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_PR8 PosList, [in] VT_I4 IsBlocking) 보간(補間) 제어에 있어 직선 보간을 수행하며, 절대(絶對) 좌표 이송(絶對座標移送)을 수행합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)되지 않습니다.
❒ VT_I4 cmxIxLineToStart ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_PR8 PosList) 보간(補間) 제어에 있어 직선 보간을 수행하며, 절대(絶對) 좌표 이송(絶對座標移送)을 수행합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)됩니다.

❒ VT_I4 cmxIxArcA ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_R8 XCentOffset, [in] VT_R8 YCentOffset, [in] VT_R8 EndAngle, [in] VT_I4 IsBlocking)
보간(補間) 제어에 있어 원호 보간을 수행하며, 상대적(相對的) 중심 좌표와 각도를 통해 원호 보간을 수행합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)되지 않습니다.

❒ VT_I4 cmxIxArcAStart ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_R8 XCentOffset, [in] VT_R8 YCentOffset, [in] VT_R8 EndAngle)
보간(補間) 제어에 있어 원호 보간을 수행하며, 상대(相對) 적중심(中心) 좌표와 각도를 통해 원호 보간을 수행합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)됩니다.

❒ VT_I4 cmxIxArcATo ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_R8 XCent, [in] VT_R8 YCent, [in] VT_R8 EndAngle, [in] VT_I4 IsBlocking)
보간(補間) 제어에 있어 원호 보간을 수행하며, 절대(絶對)적 중심 좌표와 각도를 통해 원호 보간을 수행합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)되지 않습니다.

❒ VT_I4 cmxIxArcAToStart ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_R8 XCent, [in] VT_R8 YCent, [in] VT_R8 EndAngle)
보간(補間) 제어에 있어 원호 보간을 수행하며, 절대(絶對)적 중심(中心) 좌표와 각도를 통해 원호 보간을 수행합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)됩니다.

❒ VT_I4 cmxIxArcP ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_R8 XCentOffset, [in] VT_R8 YCentOffset, [in] VT_R8 XEndPointDist, [in] VT_R8 YEndPointDist, [in] VT_I4 Direction, [in] VT_I4 IsBlocking)
보간(補間) 제어에 있어 원호 보간을 수행하며, 상대적 중심 좌표와 종점(終點) 좌표를 통해 원호 보간을 수행합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)되지 않습니다.

❒ VT_I4 cmxIxArcPStart ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_R8 XCentOffset, [in] VT_R8 YCentOffset, [in] VT_R8 XEndPointDist, [in] VT_R8 YEndPointDist, [in] VT_I4 Direction)
보간(補間) 제어에 있어 원호 보간을 수행하며, 상대적 중심 좌표와 종점(終點) 좌표를 통해 원호 보간을 수행합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)됩니다.

❒ VT_I4 cmxIxArcPTo ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_R8 XCent, [in] VT_R8 YCent, [in] VT_R8 XEndPos, [in] VT_R8 YEndPos, [in] VT_I4 Direction, [in] VT_I4 IsBlocking)
보간(補間) 제어에 있어 원호 보간을 수행하며, 절대(絶對)적 중심 좌표와 종점(終點) 좌표를 통해 원호 보간을 수행합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)되지 않습니다.

❒ VT_I4 cmxIxArcPToStart ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_R8 XCent, [in] VT_R8 YCent, [in] VT_R8 XEndPos, [in] VT_R8 YEndPos, [in] VT_I4 Direction)
보간(補間) 제어에 있어 원호 보간을 수행하며, 절대(絶對)적 중심(中心) 좌표와 종점(終點) 좌표를 통해 원호 보간을 수행합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)됩니다.

❒ VT_I4 cmxIxArc3P ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_R8 P2[], [in] VT_R8 P3[], [in] VT_R8 EndAngle, [in] VT_I4 IsBlocking)
보간 (補間) 제어에 있어 원호 보간을 수행하며, 현재 위치와 두개의 좌표를 통해 원호 보간을 수행합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)하지 않습니다.

❒ VT_I4 cmxIxArc3PStart ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_R8 P2[], [in] VT_R8 P3[], [in] VT_R8 EndAngle)
보간 (補間) 제어에 있어 원호 보간을 수행하며, 현재 위치와 두개의 좌표를 통해 원호 보간을 수행합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)됩니다.

❒ VT_I4 cmxIxArc3PTo ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_R8 P2[], [in] VT_R8 P3[], [in] VT_R8 EndAngle, [in] VT_I4 IsBlocking) 보간 (補間) 제어에 있어 원호 보간을 수행하며, 현재 절대(絶對)적 위치와 두개의 좌표를 통해 원호 보간을 수행합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)하지 않습니다
❒ VT_I4 cmxIxArc3PStart ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_R8 P2[], [in] VT_R8 P3[], [in] VT_R8 EndAngle) 보간 (補間) 제어에 있어 원호 보간을 수행하며, 현재 절대(絶對)적 위치와 두개의 좌표를 통해 원호 보간을 수행합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)됩니다.
❒ VT_I4 cmxIxIsDone ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [out] VT_PR4 IsDone) 보간(補間) 제어 구동 이송의 완료(完了) 를 확인(確認)합니다.
❒ VT_I4 cmxIxWaitDone ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_I4 IsBlocking) 보간(補間) 제어 구동 이송이 완료(完了)될 때까지 대기(待機)합니다.
❒ VT_I4 cmxIxStop ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_I4 IsWaitComplete, [in] VT_I4 IsBlocking) 보간(補間) 제어 구동 이송을 감속 후 정지(停止)합니다.
❒ VT_I4 cmxIxStopEmg ([in] VT_I4 BoardID, [in] VT_I4 MapIndex) 보간(補間) 제어 구동 이송을 비상 정지(非常停止) 합니다.

8.3.2 함수 설명

NAME

cmxIxMapAxes

- 보간(補間) 대상 축 그룹(Group) 설정

INFORMATION

Interpolation Motion

VC++/VB

BCB/Delphi/.NET

Level 3

다소 주의

보간 대상 축 그룹은 모든 보간 이송 작업전에 선행되어야 합니다.

SYNOPSIS

❒ VT_I4 cmxIxMapAxes ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_I4 MapMask, [in] VT_I4 IxMode)

DESCRIPTION

이 함수는 보간작업을 수행할 축들을 맵번호(Map index)로 맵핑(Mapping)합니다. 맵번호는 다른 "기본보간제어"에 관련된 함수들의 첫번째 매개 변수(媒介變數)로 전달되므로써 각 함수들이 제어해야할 축들에 대한 정보가 간편하게 전달됩니다. 따라서 다른 "기본보간제어"에 관련된 함수들을 사용하기전에 가장 먼저 이 함수를 사용하여 "기본보간제어"에 사용할 축들을 맵핑하여야 합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ MapIndex : 맵번호(Map index), 이 번호 범위는 0 ~ 15 입니다.

▶ MapMask: 축맵에 포함할 축들을 지정할 마스크 값(32 비트, BIT0 ~ BIT31). 이 값의 BIT0~BIT31 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. 이 매개 변수(媒介變數)의 각 비트별 정보는 아래 표와 같습니다.

▶ IxMode: 보간 모드값 입니다.

Value	Meaning
0 또는 cmxIX_MODE_LINEAR	직선 보간
1 또는 cmxIX_MODE_CIRCULAR	원호 보간
2 또는 cmxIX_MODE_HELICARL	헬리컬 보간
3 또는 cmxIX_MODE_SPLINE	스플라인 보간

예) 8 축을 사용하는 경우

Bit Number	Meaning
BIT0	0 번 축의 포함여부 : 0 => 포함안함, 1 => 포함
BIT1	1 번 축의 포함여부 : 0 => 포함안함, 1 => 포함
BIT2	2 번 축의 포함여부 : 0 => 포함안함, 1 => 포함
BIT3	3 번 축의 포함여부 : 0 => 포함안함, 1 => 포함
BIT4	4 번 축의 포함여부 : 0 => 포함안함, 1 => 포함
BIT5	5 번 축의 포함여부 : 0 => 포함안함, 1 => 포함
BIT6	6 번 축의 포함여부 : 0 => 포함안함, 1 => 포함
BIT7	7 번 축의 포함여부 : 0 => 포함안함, 1 => 포함

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

EXAMPLE 1

X1 축과 Y1 축의 직선보간

C/C++ :

```
// 맵번호 설정
#define MAP0            0

Long BoardID = 0;

cmxIxMapAxes(BoardID, MAP0, 0x3, 0);
// 또는 cmxIxMapAxes(BoardID, 3, 0_MASK | cmY1_MASK, 0); //

// 보간 제어 속도 설정, 두번째 인자는 백터 모드 혹은 마스터 속도 모드를 의미함.
cmxIxSetSpeedPattern(BoardID, MAP0, cmxFALSE cmxMODE_S,  0, 0, 1000, 10000, 10000);

// 보간 이송 거리 리스트 설정
double fDistList[2] = {1000, 1000};
cmxIxLine(BoardID, MAP0, fDistList);
```

Visual Basic

```
' BoardID = 0, 맵 번호 MAP0 은 이미 선언되어 있다고 가정함

Call IxMapAxes(BoardID, MAP0, &H3, 0)
' 또는 IxMapAxes(BoardID, MAP0, 0_MASK Or cmY1_MASK, 0)

'보간 제어 속도 설정, 두번째 인자는 백터 모드 혹은 마스터 속도 모드를 의미함.
Call IxSetSpeedPattern(BoardID, MAP0, cmxFALSE, cmxMODE_S ,0, 0,  1000, 10000, 10000)

' 보간 이송 거리 리스트 설정
fDistList(0) = 1000
fDistList(1) = 1000
```

```
IxLine(BoardID, MAP0, fDistList(0))
```

Delphi

```
// BoardID = 0,  맵 번호 MAP0 은 이미 선언되어 있다고 가정함

cmxIxMapAxes(BoardID, MAP0, $3, $0);
// 또는 cmxIxMapAxes(BoardID, 3, 0_MASK or cmY1_MASK, 0);

//보간 제어 속도 설정, 두번째 인자는 백터 모드 혹은 마스터 속도 모드를 의미함.
cmxIxSetSpeedPattern(BoardID, MAP0, cmxFALSE, cmxMODE_S, 0, 0, 1000, 10000, 10000);

// 보간 이송 거리 리스트 설정
fDistList[0] := 1000;
fDistList[1] := 1000;
cmxIxLine(BoardID, MAP0, @fDistList);
```

NAME

cmxIxUnMapAxes

- 보간(補間) 대상 축 그룹(Group) 해제

INFORMATION

Interpolation Motion

VC++/VB

BCB/Delphi/.NET

Level 3

다소 주의

보간 대상 축 그룹은 모든 보간 이송 작업전에 선행되어야 합니다.

SYNOPSIS

❒ VT_I4 cmxIxUnMapAxes ([in] VT_I4 BoardID, [in] VT_I4 MapIndex)

DESCRIPTION

이 함수는 보간작업을 수행하는 맵을 해제합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ MapIndex : 맵번호(Map index), 이 번호 범위는 0 ~ 15 입니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxIxGetMapIndex

- 대상 축의 그룹 번호 반환

INFORMATION

Interpolation Motion
VC++/VB
BCB/Delphi/.NET
Level 3
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxIxGetMapIndex ([in] VT_I4 BoardID, [in] VT_I4 Axis, [out] VT_PI4 MapIndex)

DESCRIPTION

cmxIxGetMapIndex() 함수를 통해 해당 축의 맵핑된 맵 번호를 확인할 수 있습니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(사용자가 서보드라이버에 설정한 값입니다).

▶ MapIndex : 맵번호(Map index), 이 번호 범위는 0 ~ 15 입니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxIxSetSpeedPattern

cmxIxGetSpeedPattern

- 보간(補間) 이송 속도 설정

INFORMATION

- Interpolation Motion
- VC++/VB
- BCB/Delphi/.NET
- Level 3
- 다소 주의

SYNOPSIS

❒ VT_I4 cmxIxSetSpeedPattern

([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_I4 IsVectorSpeed, [in] VT_I4 SpeedMode, [in] VT_R8 IniRatio, [in] VT_R8 EndiRatio, [in] VT_R8 VelRatio, [in] VT_R8 AccRatio, [in] VT_R8 DecRatio)

❒ VT_I4 cmxIxGetSpeedPattern

([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [out] VT_PI4 IsVectorSpeed, [out] VT_PI4 SpeedMode, [out] VT_PR8 IniRatio, [out] VT_PR8 EndiRatio, [out] VT_PR8 VelRatio, [out] VT_PR8 AccRatio, [out] VT_PR8 DecRatio)

DESCRIPTION

cmxIxSetSpeedPattern 은 "기본보간제어"의 이송 속도에 대한 환경설정을 정의합니다.
사용자가 지정한 작업 속도는 "IsVectorSpeed"의 설정값이 'TRUE'이면 벡터 스피드,
'FALSE'이면 마스터 스피드가 적용됩니다. "벡터속도"에 대한 자세한 내용은 아래의
"REFERENCE" 항목을 참조하십시오.
보간 작업 속도를 벡터속도로 설정해야만 하는 특별한 경우를 제외하고는 보간 작업 속도를
마스터속도로 설정하는 것이 모터의 최대속도를 활용하는데 있어서 편리합니다.
cmxIxGetSpeedPattern()은 "기본보간제어"의 이송속도에 대한 설정된 값을 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ MapIndex: 맵번호(Map index), 이 맵번호를 사용하기전에 먼저 cmxIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑되어 있어야 합니다.

▶ IsVertorSpeed : cmxIxSetSpeedPattern 함수의 인자이며, TRUE 로 설정했을 경우에는 벡터스피드 모드로, FALSE 로 설정했을 경우에는 마스터스피드 모드로 설정됩니다.

▶ IsVertorSpeed : cmxIxGetSpeedPattern 함수의 인자이며, 스피드 모드를 반환합니다. TRUE 일 경우엔 벡터스피드 모드, FALSE 일 경우엔 마스터스피드 모드입니다.

▶ SpeedMode : cmxIxSetSpeedPattern 함수의 인자이며, 속도모드를 설정합니다. 설정값은 다음과 같습니다.

Value	Meaning
0 또는 cmxSMODE_C	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 또는 cmxSMODE_T	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 또는 cmxSMODE_S	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.

▶ SpeedMode : cmxIxGetSpeedPattern 함수의 인자이며, 속도모드를 반환합니다. 반환값은 다음과 같습니다.

Value	Meaning
0 또는 cmxSMODE_C	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 또는 cmxSMODE_T	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 또는 cmxSMODE_S	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.

▶ IniRatio: cmxIxSetSpeedPattern 함수의 인자이며, 마스터스피드모드 일 때는 초기속도 비율을 설정합니다. 벡터스피드모드 일 때는 PPS 단위를 사용하여 설정 합니다.

▶ IniRatio: cmxIxGetSpeedPattern 함수의 인자이며, 마스터스피드모드 일 때는 초기속도 비율을 반환합니다. 벡터스피드모드 일 때의 반환값은 PPS 단위입니다.

▶ EndiRatio: cmxIxSetSpeedPattern 함수의 인자이며, 마스터스피드모드 일 때는 최종속도 비율을 설정합니다. 벡터스피드모드 일 때는 PPS 단위를 사용하여 가속도를 설정합니다.

▶ EndiRatio: cmxIxGetSpeedPattern 함수의 인자이며, 마스터스피드모드 일 때는 최종속도 비율을 반환합니다. 벡터스피드모드 일 때의 반환값은 PPS 단위입니다.

▶ VelRatio : cmxIxSetSpeedPattern 함수의 인자이며, 마스터스피드모드 일 때는 작업속도 비율을 설정합니다. 벡터스피드모드 일 때는 PPS 단위를 사용하여 설정 합니다.

▶ VelRatio : cmxIxGetSpeedPattern 함수의 인자이며, 마스터스피드모드 일 때는 작업속도 비율을 반환합니다. 벡터스피드모드 일 때의 반환값은 PPS 단위입니다.

▶ AccRatio : cmxIxSetSpeedPattern 함수의 인자이며, 마스터스피드모드 일 때는 가속도 비율을 설정합니다. 벡터스피드모드 일 때는 PPS 단위를 사용하여 가속도를 설정합니다.

▶ AccRatio : cmxIxGetSpeedPattern 함수의 인자이며, 마스터스피드모드 일 때는 가속도 비율을 반환합니다. 벡터스피드모드 일 때의 반환값은 PPS 단위입니다.

▶ DecRatio : cmxIxSetSpeedPattern 함수의 인자이며, 마스터스피드모드 일 때는 감속도 비율을 설정합니다. 벡터스피드모드 일 때는 PPS 단위를 사용하여 감속도를 설정합니다.

▶ DecRatio : cmxIxGetSpeedPattern 함수의 인자이며, 마스터스피드모드 일 때는 감속도 비율을 반환합니다. 벡터스피드모드 일 때의 반환값은 PPS 단위입니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

REFERENCE

주의	보간 제어의 속도에는 마스터 속도 모드와 백터 속도 모드가 존재합니다. 본 함수의 설명을 잘 읽고, 보간 속도 설정에 주의를 기울여 주시기 바랍니다. 특히 서로 다른 속성을 가지고 있는 서보드라이브나 스탭 드라이버에서는 보간 속도 설정에 반드시 주의를 요합니다.

□ 직선 보간 이동시에 작업속도의 적용

마스터 속도 모드(Master Speed Mode)로 보간 작업시에는 각 축의 속도가 각 축의 이동거리에 비례하여 자동으로 설정됩니다. 이때 cmxIxSetSpeedPattern() 함수의 WorkSpeed 매개변수(媒介變數)를 통하여 지정되는 보간 작업속도는 마스터속도로 적용됩니다.
각 보간 이동시에 이동거리가 가장 큰 축을 "마스터축"이라고 하며 마스터축의 속도를 "마스터속도"라 합니다. 각 보간 이동시에 마스터축의 속도는 사용자가 지정한 보간 작업속도로 설정되며, 마스터축 이외의 다른 축의 속도는 마스터축과 해당 축의 이동 거리 비에 따라서 자동으로 설정됩니다.

보간 작업속도의 적용 예
cmxIxSetSpeedPattern() 함수의 WorkSpeed 를 10000 으로 설정하고 X,Y,Z 축의 보간 작업을 수행하는 경우에 이동 거리에 따른 각 축의 속도 관계는 아래와 같습니다(표에서 배경이 회색으로 되어 있는 것은 마스터축임을 의미하는 것입니다).

	이동 거리			각 축의 이동 속도		
	X	Y	Z	Vx	Vy	Vz
보간이동 1	1000	2000	5000	2000	4000	10000
보간이동 2	5000	1000	2000	10000	2000	4000
보간이동 3	2000	20000	10000	1000	10000	5000
보간이동 4	10000	0	0	10000	0	0

표 8 마스터 속도 모드에 대한 실제 속도 적용 예시 표

□ 직선 보간 이동시의 벡터 속도
그림 8-1 X, Y 축간의 직선 보간 이송은 2 축(편의상 X, Y 축으로 가정) 직선 보간 이동을 그래프로 나타낸 것입니다.

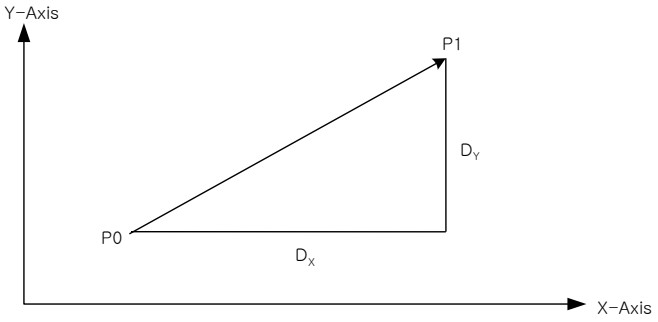


그림 8-1 X, Y 축간의 직선 보간 이송

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 속도에 대한 관계식은 다음과 같습니다.

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

4 축의 경우에는 각축의 속도에 대한 관계식은 다음과 같습니다.

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

예제 코드를 예를 들어 설명하면 다음과 같습니다.

```
#define DEV0                0 // 디바이스 번호 => 0
#define MAP_IDX             0 // 맵번호 => 0

// 코드의 간결성을 위하여 앞에서 행해져야할 초기화 루틴은 모두 생략 //
……

// X1 축과 Y1 축을 0 번 맵번호로 맵핑 //
cmxIxMapAxes (DEV0, MAP_IDX, 0_MASK | cmY1_MASK, 0);

// 속도패턴 설정 : 벡터속도 1000 PPS, 벡터가속도 10000 PPS/sec (가속시간 0.1 초) //
// 여기서 2 번째 인자가 cmxTRUE 이면, Vector 속도 모드를 의미한다.
cmxIxSetSpeedPattern(DEV0, MAP_IDX, ,cmxTRUE, cmxMODE_S, 1000, 10000, 10000);

// 직선보간이동 수행 : (3000, 4000) 만큼 이동 //
double fDistList[2]={3000, 4000};
cmxIxLine (DEV0, MAP_IDX, fDistList, cmxFALSE);
```

위의 코드는 현재 위치가 (BoardID, 3,0) 이라고 가정할 때 (3000, 4000)의 좌표로 직선 보간 이동을 수행합니다. 벡터 속도를 1000 으로 지정하였으므로 각 축의 속도 계산식은 다음과 같습니다.

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

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

NAME

cmxIxSetSpeedPattern_T

cmxIxGetSpeedPattern_T

- 보간(補間) 이송 속도 설정

INFORMATION

Interpolation Motion

VC++/VB

BCB/Delphi/.NET

Level 3

다소 주의

SYNOPSIS

❒ VT_I4 cmxIxSetSpeedPattern

([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_I4 IsVectorSpeed, [in] VT_I4 SpeedMode, [in] VT_R8 IniRatio, [in] VT_R8 EndiRatio, [in] VT_R8 VelRatio, [in] VT_R8 AccTime, [in] VT_R8 DecTime)

❒ VT_I4 cmxIxGetSpeedPattern

([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [out] VT_PI4 IsVectorSpeed, [out] VT_PI4 SpeedMode, [out] VT_PR8 IniRatio, [out] VT_PR8 EndiRatio, [out] VT_PR8 VelRatio, [out] VT_PR8 AccTime, [out] VT_PR8 DecTime)

DESCRIPTION

cmxIxSetSpeedPattern_T()은 "기본보간제어"의 이송 속도에 대한 환경을 정의합니다.
사용자가 지정한 작업 속도는 "IsVectorSpeed"의 설정값이 'TRUE'이면 벡터 스피드, 'FALSE'이면 마스터 스피드가 적용됩니다. "벡터속도"에 대한 자세한 내용은 아래의 "REFERENCE" 항목을 참조하십시오.
보간 작업 속도를 벡터속도로 설정해야만 하는 특별한 경우를 제외하고는 보간 작업 속도를 마스터속도로 설정하는 것이 모터의 최대속도를 활용하는데 있어서 편리합니다.
cmxIxGetSpeedPattern_T()은 "기본보간제어"의 이송속도에 대한 설정된 값을 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ MapIndex: 맵번호(Map index), 이 맵번호를 사용하기전에 먼저 cmxIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑되어 있어야 합니다.

▶ IsVertorSpeed : cmxIxSetSpeedPattern_T 함수의 인자이며, TRUE 로 설정했을 경우에는 벡터스피드 모드로, FALSE 로 설정했을 경우에는 마스터스피드 모드로 설정됩니다.

▶ IsVertorSpeed : cmxIxGetSpeedPattern_T 함수의 인자이며, 스피드 모드를 반환합니다. TRUE 일 경우엔 벡터스피드 모드, FALSE 일 경우엔 마스터스피드 모드입니다.

▶ SpeedMode : cmxIxSetSpeedPattern_T 함수의 인자이며, 속도모드를 설정합니다. 설정값은 다음과 같습니다.

Value	Meaning
0 또는 cmxSMODE_C	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 또는 cmxSMODE_T	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 또는 cmxSMODE_S	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.

▶ SpeedMode : cmxIxGetSpeedPattern_T 함수의 인자이며, 속도모드를 반환합니다. 반환값은 다음과 같습니다.

Value	Meaning
0 또는 cmxSMODE_C	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 또는 cmxSMODE_T	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 또는 cmxSMODE_S	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.

▶ IniRatio: cmxIxSetSpeedPattern_T 함수의 인자이며, 마스터스피드모드 일 때는 초기속도 비율을 설정합니다. 벡터스피드모드 일 때는 PPS 단위를 사용하여 설정 합니다.

▶ IniRatio: cmxIxGetSpeedPattern_T 함수의 인자이며, 마스터스피드모드 일 때는 초기속도 비율을 반환합니다. 벡터스피드모드 일 때의 반환값은 PPS 단위입니다.

▶ EndiRatio: cmxIxSetSpeedPattern_T 함수의 인자이며, 마스터스피드모드 일 때는 최종속도 비율을 설정합니다. 벡터스피드모드 일 때는 PPS 단위를 사용하여 가속도를 설정합니다.

▶ EndiRatio: cmxIxGetSpeedPattern_T 함수의 인자이며, 마스터스피드모드 일 때는 최종속도 비율을 반환합니다. 벡터스피드모드 일 때의 반환값은 PPS 단위입니다.

▶ VelRatio : cmxIxSetSpeedPattern_T 함수의 인자이며, 마스터스피드모드 일 때는 작업속도 비율을 설정합니다. 벡터스피드모드 일 때는 PPS 단위를 사용하여 설정 합니다.

▶ VelRatio : cmxIxGetSpeedPattern_T 함수의 인자이며, 마스터스피드모드 일 때는 작업속도 비율을 반환합니다. 벡터스피드모드 일 때의 반환값은 PPS 단위입니다.

▶ AccTime : cmxIxSetSpeedPattern_T 함수의 인자이며, 가속 시간(S)을 설정합니다.

▶ AccTime : cmxIxGetSpeedPattern_T 함수의 인자이며, 가속 시간(S)을 반환합니다.

▶ DecTime : cmxIxSetSpeedPattern_T 함수의 인자이며, 감속 시간(S)을 설정합니다.

▶ DecTime : cmxIxGetSpeedPattern_T 함수의 인자이며, 감속 시간(S)을 설정합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

REFERENCE

주의	보간 제어의 속도에는 마스터 속도 모드와 백터 속도 모드가 존재합니다. 본 함수의 설명을 잘 읽고, 보간 속도 설정에 주의를 기울여 주시기 바랍니다. 특히 서로 다른 속성을 가지고 있는 서보드라이브나 스텝 드라이버에서는 보간 속도 설정에 반드시 주의를 요합니다.

□ 직선 보간 이동시에 작업속도의 적용

마스터 속도 모드(Master Speed Mode)로 보간 작업시에는 각 축의 속도가 각 축의 이동거리에 비례하여 자동으로 설정됩니다. 이때 cmxIxSetSpeedPattern_T() 함수의 WorkSpeed 매개변수(媒介變數)를 통하여 지정되는 보간 작업속도는 마스터속도로 적용됩니다.
각 보간 이동시에 이동거리가 가장 큰 축을 "마스터축"이라고 하며 마스터축의 속도를 "마스터속도"라 합니다. 각 보간 이동시에 마스터축의 속도는 사용자가 지정한 보간 작업속도로 설정되며, 마스터축 이외의 다른 축의 속도는 마스터축과 해당 축의 이동 거리 비에 따라서 자동으로 설정됩니다.

보간 작업속도의 적용 예
cmxIxSetSpeedPattern_T() 함수의 WorkSpeed 를 10000 으로 설정하고 X,Y,Z 축의 보간 작업을 수행하는 경우에 이동 거리에 따른 각 축의 속도 관계는 아래와 같습니다(표에서 배경이 회색으로 되어 있는 것은 마스터축임을 의미하는 것입니다).

	이동 거리			각 축의 이동 속도		
	X	Y	Z	Vx	Vy	Vz
보간이동 1	1000	2000	5000	2000	4000	10000
보간이동 2	5000	1000	2000	10000	2000	4000
보간이동 3	2000	20000	10000	1000	10000	5000
보간이동 4	10000	0	0	10000	0	0

표 9 마스터 속도 모드에 대한 실제 속도 적용 예시 표

□ 직선 보간 이동시의 벡터 속도
그림 8-1 X, Y 축간의 직선 보간 이송은 2 축(편의상 X, Y 축으로 가정) 직선 보간 이동을 그래프로 나타낸 것입니다.

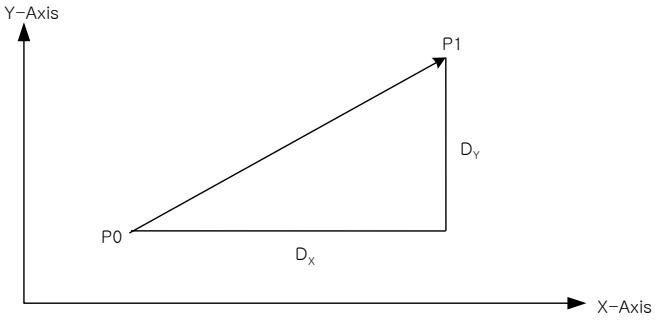


그림 8-2 X, Y 축간의 직선 보간 이송

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 속도에 대한 관계식은 다음과 같습니다.

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

4 축의 경우에는 각축의 속도에 대한 관계식은 다음과 같습니다.

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

예제 코드를 예를 들어 설명하면 다음과 같습니다.

```
#define DEV0                0 // 디바이스 번호 => 0
#define MAP_IDX             0 // 맵번호 => 0

// 코드의 간결성을 위하여 앞에서 행해져야할 초기화 루틴은 모두 생략 //
……

// X1 축과 Y1 축을 0 번 맵번호로 맵핑 //
cmxIxMapAxes (DEV0, MAP_IDX, 0_MASK | cmY1_MASK, 0);

// 속도패턴 설정 : 벡터속도 1000 PPS, 벡터가속도 10000 PPS/sec (가속시간 0.1 초) //
// 여기서 2 번째 인자가 cmxTRUE 이면, Vector 속도 모드를 의미한다.
cmxIxSetSpeedPattern(DEV0, MAP_IDX, ,cmxTRUE, cmxMODE_S, 1000, 10000, 10000);

// 직선보간이동 수행 : (3000, 4000) 만큼 이동 //
double fDistList[2]={3000, 4000};
cmxIxLine (DEV0, MAP_IDX, fDistList, cmxFALSE);
```

위의 코드는 현재 위치가 (BoardID, 3,0) 이라고 가정할 때 (3000, 4000)의 좌표로 직선 보간 이동을 수행합니다. 벡터 속도를 1000 으로 지정하였으므로 각 축의 속도 계산식은 다음과 같습니다.

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

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

NAME

cmxIxLine

cmxIxLineStart

- 직선 보간(補間) 상대(相對) 좌표 이송(移送)

INFORMATION

Interpolation Motion

VC++/VB

BCB/Delphi/.NET

Level 3

이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인(確認)합니다.

SYNOPSIS

❒ VT_I4 cmxIxLine ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_PR8 DistList, [in] VT_I4 IsBlocking)

❒ VT_I4 cmxIxLineStart ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_PR8 DistList)

DESCRIPTION

이 함수는 현재 위치로부터의 상대 좌표로의 직선 보간 이동을 수행합니다. cmxIxLine() 함수는 모션이 완료되기 전까지 반환되지 않으며, cmxIxLineStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ MapIndex : 맵번호(Map index), 이 맵번호를 사용하기전에 먼저 cmxIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑되어 있어야 합니다.

▶ DistList : 현재 위치로부터의 상대적인 이동 좌표값(각 축의 이동 거리값)의 배열 주소. 이 배열의 크기는 cmxIxMapAxes() 함수를 통하여 맵핑된 축의 수와 일치하여야 합니다. 거리의 단위는 "Unit distance"에 의해 정의되는 논리적 거리를 적용합니다.

▶ IsBlocking : 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다.

Value	Meaning
0 또는 cmxFALSE	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
1 또는 cmxTRUE	블록(Blocking)을 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

SEE ALSO

cmxIxLineTo, cmxIxLineToStart

REFERENCE

☐ cmxIxLineStart() 함수를 사용하는 경우에는 cmxIxIsDone() 함수나 cmxIxWaitDone() 함수를 사용하여 모션의 완료를 확인(確認)할 수 있습니다.

☐ cmxIxLine() 함수를 사용하는 경우에는 INP 입력신호가 Enable 로 설정되었으면 Command 펄스 출력이 완료되어도 INP 입력이 ON 이 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환되지 않습니다.
스탭 드라이브는 INP 출력이 없는 경우가 일반적인데, 고객(顧客)님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스탭 드라이브를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다.
고객(顧客) 여러분들께서는 스탭 드라이브 사용시에 이점을 주의해주시기를 부탁드립니다.

윈도우 이벤트라는 것은 무엇입니까?

윈도우 운영체제는 Event Driven 혹은 Message Driven 방식[1]의 구조로 되어 있습니다. 각 응용프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

// BoardID 는 0 으로 선언되었다고 가정함

/*************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.
*************************************************************/
void OnProgramInitial()
{
	long m_nNumDevices;
	long m_nNumAxes;
	long m_DeviceList[16];
	cmxLoadDll();
	if(cmxGnLoadDevice (&m_nNumDevices, m_DeviceList, &m_nNumAxes) != ERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		// 에러메시지 출력
```

1 이것을 순수 우리말로 옮기면 이벤트 반응형 운영체제라고 할 수 있습니다.

```
                return;
        }
}

/*************************************************************
* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
* 호출되는 가상의 함수 입니다.
*************************************************************/
#define MAP0 0 //맵번호 (0)
void OnSetSpeed()
{
        cmxIxMapAxes(BoardID, MAP0, 0_MASK | cmY1_MASK, 0 );
        //또는 cmxIxMapAxes(BoardID, MAP0, 0x3, 0x0);
        //보간 이동할 축들의 기본속도를 설정합니다.
        cmxCfgSetSetSpeedPattern(BoardID, 3, cmxMODE_T, 1000, 5000, 5000,0,0);
        cmxCfgSetSetSpeedPattern(BoardID, cmY1, cmxMODE_T, 1000, 5000, 5000,0,0);
}

/*************************************************************
* DoMotion : 작업명령시에 호출되는 가상의 함수 입니다.
* 이 함수는 X1, Y1 축에 대하여 (1000, 2000)만큼 이동한 후 다시
* (-1000, -2000)만큼 이동을 수행합니다.
*************************************************************/
void DoMotion()
{
        double fDistList[2] = {1000, 2000};//각축의 이동할 거리입니다.

        //MAP0 를 마스터 속도 모드, Trapezoidal 속도 패턴으로 작업속도의 100%,
        //가속도의 70%, 감속도의 70%로 설정 합니다.
        cmxIxSetSpeedPattern(BoardID, MAP0, cmxFALSE, cmxMODE_T,0,0, 100, 70, 70 );
        cmxIxLine(BoardID, MAP0, fDistList, cmxFALSE);
        fDistList[0] = -1000; fDistList[1] = -2000;
        cmxIxLine(BoardID, MAP0, fDistList, cmxFALSE);

        //cmxIxLineStart() 함수를 사용하는 경우에는 다음과 같이 코드를 작성합니다.
        //double fDistList[2] = {1000, 2000};
        //cmxIxSetSpeedPattern(BoardID, MAP0, cmxFALSE, cmxMODE_T,0,0, 100, 70,
        //70 );
        //cmxIxLineStart(BoardID, MAP0, fDistList);
        //cmxIxWaitDone(BoardID, MAP0, cmxFALSE);
        //fDistList[0] = -1000; fDistList[1] = -2000;
        //cmxIxLineStart(BoardID, MAP0, fDistList);
        //cmxIxWaitDone(BoardID, MAP0, cmxFALSE);
}
```

```
Visual Basic
'BoardID 는 0 으로 선언되었다고 가정함
'맵 번호 MAP0 은 이미 선언되어 있다고 가정함.
'=============================================================
'GnLoadDevice 함수로 장치를 초기화 합니다.
'=============================================================

Private Sub Form_Load()
   Dim nTotalDevices As Long
   Dim DeviceList(16) As Long
```

```
    Dim nTotalAxis As Long
    Dim IRetVal As Long
'============================================================
 ' GnLoadDevice 함수로 장치를 초기화합니다.
  IRetVal = GnLoadDevice (nTotalDevices, DeviceList(0), nTotalAxis)

  If IRetVal <> ERR_NONE Then
     MsgBox ("GnLoadDevice has been failed")
  End If
 '============================================================
End Sub

' 직선 보간제어 이송을 시작합니다.
Private Sub btnMove_Click()

    Dim DistanceList(2) As Double
    Dim nRetVal As Long


    DistanceList(0) = 1000
    DistanceList(1) = 2000

    ' ============================================================
    ' IxMapAxes 함수는 다음과 같은 인자를 필요로 합니다.   '
    ' IxMapAxes(맵번호, 비트(Bit)를 통한 맵구성 #1, 비트(Bit)를  통한 맵구성 #2
    ' ============================================================

    nRetVal = IxMapAxes(BoardID, MAP0, &H3, &H0)
    ' //////////////////////////////////////////////////////////
    If IxSetSpeedPattern(BoardID, MAP0, cmxFALSE, cmxMODE_S,0,0, 100, 100,
                100) <> ERR_NONE Then

       ' 아래와 같은 커미조아 ComiRTEX 전용 에러 표시 함수를
       ‘ 사용할 수 있습니다.
       ' // 에러메시지 출력
       ' ------------------------------------------------------
       MsgBox ("IxSetSpeedPattern has been failed")
    End If

    ' IxLine 함수를 통해 선형 보간을 수행합니다. 인자는 순서대로, 맵번호,
         ‘ 거리정보를 가지고 있는 배열, 블록 여부 입니다.
    nRetVal = IxLine(BoardID, MAP0, DistanceList(0), cmxFALSE)
End Sub


Private Sub CfgSpeed(nTotalAxis As Long)

  Dim i As Integer

'============================================================
    ' 이 함수에서 CfgSetSpeedPattern 함수로 속도를 설정하는 것은 모든 모션의
기준속도(Standard
    ' Speed) 가 됩니다.
    ' 단축 구동을 비롯한 대부분의 모션 동작은 이 기준 속도의 비율로 동작되게 됩니다.
    ' 아래 함수는 전체 축에 대해서 임의의 기준 속도를 설정하고 있습니다.

'============================================================
```

```
   For i = 0 To nTotalAxis-1
      Call CfgSetSpeedPattern(BoardID, 3, cmxMODE_S, 1000, 2000, 2000,0,0)
   Next
End Sub
```

```
Delphi
//* BoardID 는 0 으로 선언되었다고 가정함

Const
MAPINDEX = 0;
// * 이 함수는 폼이 생성될때 이벤트에 의해 불려지며 , 장치를 로드하는 함수입
// * 니다.

procedure OnCreate();
var
        g_nDevs : LongInt;
        DevList : Array[0..15] of LongInt
        g_nAxis : LongInt;
begin
        // Load ComiRTEX(DLL) Library
        if ( cmxGnDeviceLoad(@g_nDevs, @DevList, @g_nAxis) <> ERR_NONE ) then
        begin
                // 마지막에 발생한 에러를 화면에 표시합니다.
                // 함수 인자로는 Form 의 Handle 이 전달됩니다.
                // 에러메시지 출력
                exit;
        end
end;

// * Description : 구동 속도를 설정합니다.
procedure btnSetSpeedClick();
var
   fAccelSpeed : Double;
   fDecelSpeed : Double;
   fWorkSpeed : Double;
   nSMODE : LongInt;

begin

        fAccelSpeed := 50000;
        fDecelSpeed := 50000;
        fWorkSpeed := 10000;
        nSMODE := cmxMODE_S;

        // 0 을 위해 설정된 기준 속도를 실제 SDK 함수에 전달합니다.
        // cmY1 을 위해 설정된 기준 속도를 실제 SDK 함수에 전달합니다.

        // 이 예제에서는 보간제어의 속도가 [Master 속도 모드]일때
        // 거리에 따라서, Slave 축이 최대속도를 초과하는 경우
        // Master 축의 속도는 자동으로 조절되어,
        // Slave 축의 속도 초과 문제를 해결합니다.
        cmxCfgSetSpeedPattern(
        0,                      // 현재 Board 의 ID 를 입력합니다.
        0,                      // Master 축의 축 번호입니다.
        nSMODE,         // 가감속이 없는 모드와 선형 가감속, S-CURVE 가감속을 설정합니다.
```

```
        fWorkSpeed,       // 작업 속도를 설정합니다.
        fAccelSpeed,      // 가속도를 설정합니다.
        fDecelSpeed);     // 감속도를 설정합니다.


        // 이때 아래 설정되는 Slave 축의 기준 속도(Standard Speed) 는
        // Slave 축의 최대 속도가 됩니다.
        // 이 최대 속도에 의해서 Master 속도모드에서 계산된 Master 축의
        // 속도는 자동으로 조절이됩니다.
        cmxCfgSetSpeedPattern(
        0,                  // 현재 Board 의 ID 를 입력합니다.
        cmY1,             // Slave 축의 축 번호입니다.
        nSMODE,       // 가감속이 없는 모드와 선형 가감속, S-CURVE 가감속을 설정합니다.
        1000,             // 작업 속도를 설정합니다.
        2000,             // 가속도를 설정합니다.
        2000);            // 감속도를 설정합니다.

    end;

// * Description :
// *
// *   상대 좌표를 목표 위치로 하여 직선 보간을 수행합니다.
// *

procedure btnMoveClick();
var

      NumChannel : Array[0..1] of LongInt;
      fDistanceList : Array[0..1] of Double;

begin
      // cmxIxMapAxes 함수로 보간제어에 해당하는 축을
      // 그룹(Group)화 합니다.
      // $3 의 의미는 Delphi 에서 0x3 을 의미하며, 해당 축의 구성은
      // 개별 비트를 의미합니다. 즉 1 번째 비트와 2 번째 비트를 의미하며,
      // 해당 비트를 16 진수로 보았을 때에는 0x3 이 됩니다.
      cmxIxMapAxes(BoardID, MAPINDEX,$3, cmxIX_MODE_LINEAR);

      // ---------------------------------------------------------
      // 보간제어의 속도 모드에 대해서 다음과 같이 설정할 수 있습니다.

      // 아래는 Vecter Speed 모드로 동작하는 예제입니다.
      //cmxIxSetSpeedPattern(BoardID, MAPINDEX, cmxTRUE, cmxMODE_S,0,0, 1000,  2000,
2000);

      // 아래는 Master Speed 모드로 동작하는 예제입니다.
      // Master Speed 설정
      // 가속도 : 100%
      // 감속도 : 100%
      // 작업속도 : 100%

      cmxIxSetSpeedPattern(BoardID, MAPINDEX, cmxFALSE, cmxMODE_S,
                  0, 0, 100,100,100);

      // ---------------------------------------------------------
```

```
    NumChannel[0] := 0;
    NumChannel[1] := cmY1;

    fDistanceList[0] := 1000;
    fDistanceList[1] := 1000;

    // 보간 대상 그룹을 통해 실제 보간 작업을 수행합니다.
    // 이때 함수의 이름을 통해
    // cmxIxLine 은 상대 좌표 보간을 의마힙니다.
    // 직선 보간이 완료된 후 반환됩니다.
    // cmxIxLineTo 는 절대 좌표 보간을 의미합니다.
    // 직선 보간이 완료된 후 반환됩니다.
    // cmxIxLineStart / cmxIxLineToStart 는 각각 상대좌표와 절대 좌표를
    // 목적 위치로 하여,
    // 직선 보간이 시작되자 마자 바로 반환합니다.
    cmxIxLine(BoardID, MAPINDEX, @fDistanceList, cmxFALSE);

end;
```

NAME

cmxIxLineTo

cmxIxLineToStart

- 직선 보간(補間) 절대(絶對) 좌표 이송(移送)

INFORMATION

Interpolation Motion

VC++/VB

BCB/Delphi/.NET

Level 3

이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인(確認)합니다.

SYNOPSIS

❒ VT_I4 cmxIxLineTo ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_PR8 PosList, [in] VT_I4 IsBlocking)

❒ VT_I4 cmxIxLineToStart ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_PR8 PosList)

DESCRIPTION

이 함수는 절대 좌표로의 직선 보간 이동을 수행합니다. cmxIxLineTo() 함수는 모션이 완료되기 전까지 반환되지 않으며, cmxIxLineToStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ MapIndex : 맵번호(Map index), 이 맵번호를 사용하기전에 먼저 cmxIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑되어 있어야 합니다.

▶ PosList : 이동할 목표 절대좌표값(각 축의 절대좌표값)의 배열 주소. 이 배열의 크기는 cmxIxMapAxes() 함수를 통하여 맵핑된 축의 수와 일치하여야 합니다. 거리의 단위는 "Unit distance"에 의해 정의되는 논리적 거리를 적용합니다.

▶ IsBlocking : 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다.

Value	Meaning
0 또는 cmxFALSE	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
1 또는 cmxTRUE	블록(Blocking)을 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

SEE ALSO

□ cmxIxLineToStart() 함수를 사용하는 경우에는 cmxIxIsDone() 함수나 cmxIxWaitDone() 함수를 사용하여 모션의 완료를 확인(確認)할 수 있습니다.

□ cmxIxLineTo() 함수를 사용하는 경우에는 INP 입력신호가 Enable 로 설정되었으면 Command 펄스 출력이 완료되어도 INP 입력이 ON 이 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환되지 않습니다.
스탭 드라이브는 INP 출력이 없는 경우가 일반적인데, 고객(顧客)님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스탭 드라이브를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다.
고객(顧客) 여러분들께서는 스탭 드라이브 사용시에 이점을 주의해주시기를 부탁드립니다.

윈도우 이벤트라는 것은 무엇입니까?

윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

// BoardID 는 0 으로 선언되었다고 가정함

/*************************************************************
 * OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
 * 적용되는 부분을 의미합니다.
*************************************************************/
void OnProgramInitial()
{

	long m_nNumDevices;
	long m_DeviceList[16];
	long m_nNumAxes;

	cmxLoadDll();
	if(cmxGnLoadDevice (&m_nNumDevices, m_DeviceList, &m_nNumAxes)!= ERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		// 에러메시지 출력
		return;
	}
}
```

```
/*************************************************************
* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
* 호출되는 가상의 함수 입니다.
*************************************************************/
#define MAP0 0 //맵번호 (0)
void OnSetSpeed()
{
	cmxIxMapAxes(BoardID, MAP0, 0_MASK | cmY1_MASK, 0 );
	//또는 cmxIxMapAxes(BoardID, MAP0, 0x3, 0x0);
	//보간 이동할 축들의 기본속도를 설정합니다.
	cmxCfgSetSetSpeedPattern(BoardID, 3, cmxMODE_T, 1000, 5000, 5000,0,0);
	cmxCfgSetSetSpeedPattern(BoardID, cmY1, cmxMODE_T, 1000, 5000, 5000,0,0);
}

/*************************************************************
* DoMotion : 작업명령시에 호출되는 가상의 함수 입니다.
* 이 함수는 X1, Y1 축에 대하여 절대좌표 (1000, 2000)으로 이동한 후
* 다시 (BoardID, 3, 0)으로 이동을 수행합니다.
*************************************************************/
void DoMotion()
{
	double fPosList[2] = {1000, 2000};//각축의 이동할 좌표입니다.

	//MAP0 를 마스터 속도 모드, Trapezoidal 속도 패턴으로 작업속도의 100%,
	//가속도의 70%, 감속도의 70%로 설정 합니다.
	cmxIxSetSpeedPattern(BoardID, MAP0, cmxFALSE, cmxMODE_T,0,0, 100, 70, 70 );
	cmxIxLineTo(BoardID, MAP0, fPosList, cmxFALSE);
	fPosList[0] = 0; fPosList[1] = 0;
	cmxIxLineTo(BoardID, MAP0, fPosList, cmxFALSE);

	//cmxIxLineToStart() 함수를 사용하는 경우에는 다음과 같이 코드를
	//작성합니다.
	//double fPosList[2] = {1000, 2000};
	//cmxIxSetSpeedPattern(BoardID, MAP0, cmxFALSE, cmxMODE_T,0,0, 100, 70,
	//70 );
	//cmxIxLineToStart(BoardID, MAP0, fPosList);
	//cmxIxWaitDone(BoardID, MAP0, cmxFALSE);
	//fPosList[0] = -1000; fPosList[1] = -2000;
	//cmxIxLineToStart(BoardID, MAP0, fPosList);
	//cmxIxWaitDone(BoardID, MAP0, cmxFALSE);
}
```

Visual Basic

```
'BoardID 는 0 으로 선언되었다고 가정함

'맵 번호 MAP0 은 이미 선언되어 있다고 가정함.
'=============================================================
'GnLoadDevice 함수로 장치를 초기화 합니다.
'=============================================================

Private Sub Form_Load()

   Dim nTotalDevices As Long
```

```
    Dim DeviceList(16) As Long
    Dim nTotalAxis As Long
    Dim IRetVal As Long
'============================================================
 ' GnLoadDevice 함수로 장치를 초기화합니다.
  IRetVal = GnLoadDevice (nTotalDevices, DeviceList(0), nTotalAxis)

  If IRetVal <> ERR_NONE Then
      MsgBox ("cmxGnLoadDevice has been failed")
  End If
'============================================================

End Sub

' 실제 직선 보간 제어 모션 이송을 시작합니다.
Private Sub btnMove_Click()

    Dim DistanceList(2) As Double
    Dim nRetVal As Long

    DistanceList(0) = 1000
    DistanceList(1) = 1000

    ' ============================================================
    ' IxMapAxes 함수는 다음과 같은 인자를 필요로 합니다.   '
    ' IxMapAxes(맵번호, 비트(Bit)를 통한 맵구성 #1, 비트(Bit)를  통한 맵구성 #2

        nRetVal = IxMapAxes(BoardID, MAP0, &H3, cmxIX_MODE_LINEAR)
        If IxSetSpeedPattern(BoardID, MAP0, cmxFALSE, cmxMODE_S,0,0, 100, 100,
                100) <> ERR_NONE Then
        MsgBox ("IxSetSpeedPattern has been failed")
    End If

       nRetVal = IxLineTo(BoardID, MAP0, DistanceList(0), cmxFALSE)
End Sub

Private Sub btnStop_Click()

    Dim nRetVal As Long

    ' IxStop 을 통해 보간제어를 정지(停止)합니다.
    ' 각 자세한 함수인자는 매뉴얼을 참조해주시기 바랍니다.
    nRetVal = IxStop(BoardID, MAP0, cmxTRUE, cmxFALSE)

    If nRetVal <> ERR_NONE Then
       //Handle 은 사용자가 생성한 폼의 핸들 값입니다.
       // 에러메시지 출력
    End If

End Sub

Private Sub CfgSpeed(nTotalAxis As Long)

    Dim i As Integer
    Dim nTotalAxis As Integer
    '============================================================
    ' 이 함수에서 CfgSetSpeedPattern 함수로 속도를 설정하는 것은 모든
    ‘ 모션의 기준속도(Standard  Speed) 가 됩니다.
```

```
    ' 단축 구동을 비롯한 대부분의 모션 동작은 이 기준 속도의 비율로 동작되게 됩니다.
    ' 아래 함수는 전체 축에 대해서 임의의 기준 속도를 설정하고 있습니다.
    '============================================================

    For i = 0 To nTotalAxis-1
        Call CfgSetSpeedPattern(BoardID, i, cmxMODE_S, 1000, 2000, 2000,0,0)
    Next

End Sub
```

```
Delphi

//* BoardID 는 0 으로 선언되었다고 가정함

const
  g_nTargetAxis = 2;
  MAPINDEX = 0;
var
    g_nAxis : LongInt;
    g_nSMODE : LongInt;

// * Description :
// * CME 빌더를 통한 모션 환경설정이 되었다는 가정하에 진행합니다.
// *
// * 이 함수는 폼이 생성될때 이벤트에 의해 불려지며 , 장치를 로드하는 함수입
// * 니다.

procedure OnCreate();
var
        g_nDevs : LongInt;
        DevList : Array[0..15] of LongInt
        g_nAxis : LongInt;
begin

        // Load ComiRTEX(DLL) Library
        if ( cmxGnLoadDevice (@g_nDevs, @DevList, @g_nAxis) <> ERR_NONE ) then
        begin
                // 마지막에 발생한 에러를 화면에 표시합니다.
                // 함수 인자로는 Form 의 Handle 이 전달됩니다.
                // 에러메시지 출력
                exit;
        end
end;

// * Description : 구동 속도를 설정합니다.
procedure btnSetSpeedClick();
var
    fAccelSpeed : Double;
    fDecelSpeed : Double;
    fWorkSpeed : Double;
    nSMODE : LongInt;


begin
     if cmxExampleHelper.cmxShowSpeedSetupDlg() = cmxTRUE then
     begin
         fAccelSpeed := 50000;
```

```
        fDecelSpeed := 50000;
        fWorkSpeed := 10000;
        nSMODE := cmxMODE_S;

        // 0 을 위해 설정된 기준 속도를 실제 SDK 함수에 전달합니다.
        // cmY1 을 위해 설정된 기준 속도를 실제 SDK 함수에 전달합니다.

        // 이 예제에서는 보간제어의 속도가 [Master 속도 모드]일때
        // 거리에 따라서, Slave 축이 최대속도를 초과하는 경우
        // Master 축의 속도는 자동으로 조절되어,
        // Slave 축의 속도 초과 문제를 해결합니다.
        cmxCfgSetSpeedPattern(
        0,                      // 현재 Board 의 ID 를 입력합니다.
        0,            // Master 축의 축 번호입니다.
        nSMODE,          // 가감속이 없는 모드와 선형 가감속, S-CURVE 가감속을 설정합니다.
        fWorkSpeed,      // 작업 속도를 설정합니다.
        fAccelSpeed,     // 가속도를 설정합니다.
        fDecelSpeed);    // 감속도를 설정합니다.


        // 이때 아래 설정되는 Slave 축의 기준 속도(Standard Speed) 는
        // Slave 축의 최대 속도가 됩니다.
        // 이 최대 속도에 의해서 Master 속도모드에서 계산된 Master 축의
        // 속도는 자동으로 조절이됩니다.
        cmxCfgSetSpeedPattern(
        0,                      // 현재 Board 의 ID 를 입력합니다.
        cmY1,           // Slave 축의 축 번호입니다.
        nSMODE,    // 가감속이 없는 모드와 선형 가감속, S-CURVE 가감속을 설정합니다.
        1000,           // 작업 속도를 설정합니다.
        2000,          // 가속도를 설정합니다.
        2000);         // 감속도를 설정합니다.

    end;
end;

// * Description :
// *
// *   상대 좌표를 목표 위치로 하여 직선 보간을 수행합니다.
// *

procedure btnMoveClick();
var

      NumChannel : Array[0..1] of LongInt;
      fDistanceList : Array[0..1] of Double;
begin

      // cmxIxMapAxes 함수로 보간제어에 해당하는 축을
      // 그룹(Group)화 합니다.
      // $3 의 의미는 Delphi 에서 0x3 을 의미하며, 해당 축의 구성은
      // 개별 비트를 의미합니다. 즉 1 번째 비트와 2 번째 비트를 의미하며,
      // 해당 비트를 16 진수로 보았을 때에는 0x3 이 됩니다.
      cmxIxMapAxes(BoardID, MAPINDEX,$3, cmxIX_MODE_LINEAR);

      // ------------------------------------------------------
```

```
    // 보간제어의 속도 모드에 대해서 다음과 같이 설정할 수 있습니다.

    // 아래는 Vecter Speed 모드로 동작하는 예제입니다.
    //cmxIxSetSpeedPattern(BoardID, MAPINDEX, cmxTRUE, cmxMODE_S, 0,0,1000, 2000,
2000);

    // 아래는 Master Speed 모드로 동작하는 예제입니다.
    cmxIxSetSpeedPattern(BoardID, MAPINDEX, cmxFALSE, cmxMODE_S,
            0, 0, 100,100,100);
    // Master Speed 설정
    // 가속도 : 100%
    // 감속도 : 100%
    // 작업속도 : 100%

    // ------------------------------------------------------

    NumChannel[0] := 0;
    NumChannel[1] := cmY1;

    fDistanceList[0] := 1000;
    fDistanceList[1] := 1000;

    // 보간 대상 그룹을 통해 실제 보간 작업을 수행합니다.
    // 이때 함수의 이름을 통해
    // cmxIxLine 은 상대 좌표 보간을 의마힙니다.
    // 직선 보간이 완료된 후 반환됩니다.
    // cmxIxLineTo 는 절대 좌표 보간을 의미합니다.
    // 직선 보간이 완료된 후 반환됩니다.
    // cmxIxLineStart / cmxIxLineToStart 는 각각 상대좌표와 절대 좌표를
    // 목적 위치로 하여,
    // 직선 보간이 시작되자 마자 바로 반환합니다.

            If cmxIxLineTo(BoardID, MAPINDEX, @DistanceList, cmxFALSE) <>
ERR_NONE then begin
                    //Handle 은 사용자가 생성한 폼의 핸들 값입니다.
                    // 에러메시지 출력
            end;
 end;

// * Description :
// *
// *  현재 수행되고 있는 모션 동작에 대해서 감속 후 정지(停止) 합니다.

procedure btnStopClick();
begin
    cmxIxStop(BoardID, MAPINDEX,cmxTRUE, cmxFALSE);
end;
```

NAME

cmxIxArcA

cmxIxArcAStart

- 원호 보간(補間) 상대(相對) 좌표 이송(移送)
(상대적 중심 좌표와 각도)

INFORMATION

Interpolation Motion

VC++/VB

BCB/Delphi/.NET

Level 3

이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인(確認)합니다.

SYNOPSIS

❒ VT_I4 cmxIxArcA

([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_R8 XCentOffset, [in] VT_R8 YCentOffset, [in] VT_R8 EndAngle, [in] VT_I4 IsBlocking)

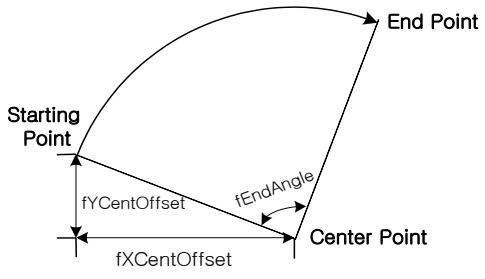
❒ VT_I4 cmxIxArcAStart

([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_R8 XcentOffset, [in] VT_R8 YCentOffset, [in] VT_R8 EndAngle)

DESCRIPTION

중심좌표와 원호의 각도를 매개 변수(媒介變數)로 하여 원호보간이동을 수행합니다. 이때 중심좌표는 상대좌표로 표현됩니다. cmxIxArcA() 함수는 모션이 완료되기 전까지 반환되지 않으며, cmxIxArcAStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.
원호보간은 임의의 두 축에 대해서 적용됩니다. 아래 설명에서는 맵핑된 두 축을 X, Y 축으로 간주하여 설명합니다. 여기서 X 축이라 함은 맵핑된 두 축 중에서 축번호가 낮은 축을 의미하며 Y 축은 축번호가 높은 축을 의미합니다. 예를 들어 Z 축과 U 축이 맵핑된 두 축이라면 Z 축이 X 축에 해당하며 U 축이 Y 축에 해당합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.



PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ MapIndex : 맵번호(Map index), 이 맵번호를 사용하기전에 먼저 cmxIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑되어 있어야 합니다.

▶ XCentOffset : 현재 위치(시작 위치)로부터 원의 중심까지 X 축 상대좌표값. 거리의 단위는 "Unit distance"에 의해 정의되는 논리적 거리를 적용합니다.

▶ YCentOffset : 현재 위치(시작 위치)로부터 원의 중심까지 Y 축 상대좌표값. 거리의 단위는 "Unit distance"에 의해 정의되는 논리적 거리를 적용합니다.

▶ EndAngle : 원호보간 이동을 완료할 목표지점의 현재 위치에 대한 각도값을 Degree(°)값으로 지정합니다. 각도의 부호가 (+)이면 반시계방향, (-)이면 시계방향으로의 이동을 의미합니다.

▶ IsBlocking : 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다.

Value	Meaning
0 또는 cmxFALSE	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
1 또는 cmxTRUE	블록(Blocking)을 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

SEE ALSO

☐ cmxIxArcAStart() 함수를 사용하는 경우에는 cmxIxIsDone() 함수나 cmxIxWaitDone() 함수를 사용하여 모션의 완료를 확인(確認)할 수 있습니다.

☐ cmxIxArcA() 함수를 사용하는 경우에는 내부적으로 루프를 수행하면서 모션이 완료되기를 기다리는데, 이때 "Blocking Mode"설정에 따라 윈도우 이벤트를 처리하는 방식이 달라집니다. 그러나 일반적으로 윈도우의 작업 쓰레드(Work Thread) 에서는 블록모드를 사용하여, 함수내부에서 지연없이 쓰레드 내부의 작업에 집중할 수 있도록 설정하는 것이 바람직합니다.

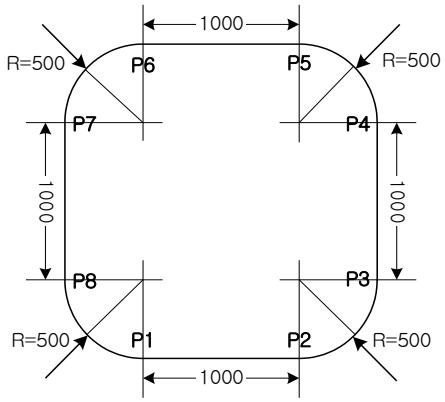
☐ cmxIxArcA() 함수를 사용하는 경우에는 INP 입력신호가 Enable 로 설정되었으면 Command 펄스 출력이 완료되어도 INP 입력이 ON 이 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환되지 않습니다.
스탭 드라이브는 INP 출력이 없는 경우가 일반적인데, 고객(顧客)님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스탭 드라이브를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다.
고객(顧客) 여러분들께서는 스탭 드라이브 사용시에 이점을 주의해주시기를 부탁드립니다.

윈도우 이벤트라는 것은 무엇입니까?

	윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

EXAMPLE

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.



C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

// BoardID 는 0 으로 선언되었다고 가정함

/************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.
************************************************************/
void OnProgramInitial()
{

	long m_nNumDevices;
	long m_DeviceList[16];
	long m_nNumAxes;
	cmxLoadDll();

	if(cmxGnLoadDevice(&m_nNumDevices, m_DeviceList, &m_nNumAxes)!= ERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		// 에러메시지 출력
		return;
	}
}
```

```
/*************************************************************
* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
* 호출되는 가상의 함수 입니다.
*************************************************************/
#define MAP0 0 //맵번호 (0)
#define MAP1 1 //맵번호 (1)

void OnSetSpeed()
{
	cmxIxMapAxes(DEV0, MAP0, cmxX1_MASK | cmxY1_MASK, cmxIX_MODE_LINEAR);
	cmxIxMapAxes(DEV0, MAP1, cmxX1_MASK | cmxY1_MASK, cmxIX_MODE_CIRCULAR);
	//또는 cmxIxMapAxes(DEV0, MAP0, 0x3, cmxIX_MODE_LINEAR);
	// cmxIxMapAxes(DEV0, MAP1, 0x3, cmxIX_MODE_CIRCULAR);
	//보간 이동할 축들의 기본속도를 설정합니다.
	cmxCfgSetSpeedPattern(DEV0, 0, cmxSMODE_T, 1000, 5000, 5000,0,0);
	cmxCfgSetSpeedPattern(DEV0, cmxY1, cmxSMODE_T, 1000, 5000, 5000,0,0);
}

/*************************************************************
* OnDoMotion() : 작업명령시에 호출되는 가상의 함수
*************************************************************/
void OnDoMotion()
{
	double fDistList[2];

	//MAP0 를 마스터 속도 모드, Trapezoidal 속도 패턴으로 작업속도의 100%,
	//가속도의 70%, 감속도의 70%로 설정 합니다.
	cmxIxSetSpeedPattern(BoardID, MAP0, cmxFALSE, cmxMODE_T, 0,0,100, 70, 70 );
	cmxIxSetSpeedPattern(BoardID, MAP1, cmxFALSE, cmxMODE_T, 0,0,100, 70, 70 );

	// Move from P1 to P2 //
	fDistList[0]=1000; fDistList[1]=0;
	cmxIxLine(BoardID, MAP0, fDistList, cmxFALSE);

	// Move from P2 to P3 //
	cmxIxArcA(BoardID, MAP1, 0, 500, 90, cmxFALSE);

	// Move from P3 to P4 //
	fDistList[0]=0; fDistList[1]=1000;
	cmxIxLine(BoardID, MAP0, fDistList, cmxFALSE);

	// Move from P4 to P5 //
	cmxIxArcA(BoardID, MAP1, -500, 0, 90, cmxFALSE);

	// Move from P5 to P6 //
	fDistList[0]=-1000; fDistList[1]=0;
	cmxIxLine(BoardID, MAP0, fDistList, cmxFALSE);

	// Move from P6 to P7 //
	cmxIxArcA(BoardID, MAP1, 0, -500, 90, cmxFALSE);

	// Move from P7 to P8 //
	fDistList[0]=0; fDistList[1]=-1000;
	cmxIxLine(BoardID, MAP0, fDistList, cmxFALSE);

	// Move from P8 to P1 //
	cmxIxArcA(BoardID, MAP1, 500, 0, 90, cmxFALSE);
```

```
}
```

Visual Basic

```
'BoardID 는 0 으로 선언되었다고 가정함

'맵 번호 MAP0 은 이미 선언되어 있다고 가정함.
'=============================================================
'GnLoadDevice 함수로 장치를 초기화 합니다.
'=============================================================

Private Sub Form_Load()
    Dim nTotalDevices As Long
    Dim DeviceList(16) As Long
    Dim nTotalAxis As Long
    Dim IRetVal As Long
'=============================================================
 ' GnLoadDevice 함수로 장치를 초기화합니다.
    IRetVal = GnLoadDevice(nTotalDevices, DeviceList(0), nTotalAxis)

  If IRetVal <> ERR_NONE Then
      MsgBox ("cmxGnLoadDevice has been failed")
  End If
'=============================================================
End Sub

Private Sub CfgSpeed(nTotalAxis As Long)

    Dim i As Integer
   '=============================================================
    ' 이 함수에서 CfgSetSpeedPattern 함수로 속도를 설정하는 것은
    ' 모든 모션의 기준속도(Standard Speed) 가 됩니다.
    ' 단축 구동을 비롯한 대부분의 모션 동작은 이 기준 속도의 비율로 동작되게
    ' 됩니다.
    ' 아래 함수는 전체 축에 대해서 임의의 기준 속도를 설정하고 있습니다.
    '=============================================================

    For i = 0 To nTotalAxis-1
       Call CfgSetSpeedPattern(BoardID, i, cmxMODE_S, 1000, 2000, 2000,0,0)
    Next

End Sub

Private Sub btnMove_Click()

    Dim nRetVal As Long
    Dim dXCentOfs As Double
    Dim dYCentOfs As Double
    Dim dAngle As Double

nRetVal = IxMapAxes(BoardID, MAP0, &H3, cmxIX_MODE_CIRCULAR)

If IxSetSpeedPattern(BoardID, MAP0, False, cmxMODE_S,0,0, 100, 100, 100) <> ERR_NONE Then

        MsgBox ("IxSetSpeedPattern has been failed")
 End If
```

```
    dXCentOfs = 5000
    dYCentOfs = 5000
    dAngle = 90

    nRetVal = IxArcA(BoardID, MAP0, dXCentOfs, dYCentOfs, dAngle, cmxFALSE)

End Sub
```

Delphi

```
//* BoardID 는 0 으로 선언되었다고 가정함

const
  g_nTargetAxis = 2;
  MAPINDEX = 0;

// * 이 함수는 폼이 생성될때 이벤트에 의해 불려지며 , 장치를 로드하는 함수입
// * 니다.

procedure OnCreate();
var
        g_nDevs : LongInt;
        DevList : Array[0..15] of LongInt
        g_nAxis : LongInt;
begin

        // Load ComiRTEX(DLL) Library
        if ( cmxGnLoadDevice (@g_nDevs, @DevList, @g_nAxis) <> ERR_NONE ) then
        begin
                // 마지막에 발생한 에러를 화면에 표시합니다.
                // 함수 인자로는 Form 의 Handle 이 전달됩니다.
                // 에러메시지 출력
                exit;
        end
end;

// * Description :
// *
// *   상대 좌표를 목표 위치로 하여 ArcA 원호 보간을 수행합니다.
// *

procedure TForm1.btnMoveClick(Sender: TObject);
var
        fWorkSpeedRatio : Double;
        fAccelSpeedRatio : Double;
        fDecelSpeedRatio : Double;

        dXCentOfs : Double;
        dYCentOfs : Double;
        dAngle : Double;
begin

        btnMove.Enabled := Boolean(FALSE);
        // cmxIxMapAxes 함수로 보간제어에 해당하는 축을
        // 그룹(Group) 화 합니다.
```

```
    // $3 의 의미는 Delphi 에서 0x3 을 의미하며, 해당 축의 구성은
    // 개별 비트를 의미합니다. 즉 1 번째 비트와 2 번째 비트를 의미하며,
    // 해당 비트를 16 진수로 보았을 때에는 0x3 이 됩니다.
    cmxIxMapAxes(BoardID, MAPINDEX,$3, cmxIX_MODE_CIRCULAR);

    dXCentOfs := 1000;
    dYCentOfs := 1000;
    dAngle    := 90;

    // 기준 속도란 cmxCfgSetSpeedPattern 함수를 통해 설정된 속도를 의미하며,
    // 아래의 cmxIxSetSpeedPattern 함수는 보간 축을 대상으로 축의 속도를
    // 기준 속도 대비 Percent(%) 단위로 설정하고 있습니다.
    fAccelSpeedRatio := 100;
    fDecelSpeedRatio := 100;
    fWorkSpeedRatio := 100;

    // ------------------------------------------------------
    // 보간제어의 속도 모드에 대해서 다음과 같이 설정할 수 있습니다.

    // 아래는 Vecter Speed 모드로 동작하는 예제입니다.
    //cmxIxSetSpeedPattern(BoardID, MAPINDEX, cmxTRUE, cmxMODE_S,0,0, 1000,  2000,
2000);

    // 아래는 Master Speed 모드로 동작하는 예제입니다.
    cmxIxSetSpeedPattern(BoardID, MAPINDEX, cmxFALSE, cmxMODE_S, 0,0,
fWorkSpeedRatio, fAccelSpeedRatio, fDecelSpeedRatio);

    // ------------------------------------------------------

    // 원호 보간을 수행합니다.

    // cmxIxArcA 는 중심좌표와 원호의 각도를 매개 변수(媒介變數)로 하여
    //원호보간을 수행하는 함수입니다.

    // 원호 보간시에는 다음 4 가지 유형의 함수를 사용할 수 있습니다
    // 1. cmxIxArcA : 상대 거리를 목표로 하여, 원호 보간이 완료된 상태에서 함수가
    //    반환됩니다.
    // 2. cmxIxArcAStart : 상대 거리를 목표로 하여,  원호 보간이 시작된 후 바로 반환됩니다.
    // 3. cmxIxArcATo : 절대 거리를 목표로 하여, 원호 보간이 완료된 상태에서 함수가
    //    반환됩니다.
    // 4. cmxIxArcAToStart : 절대 거리를 목표로 하여,  원호 보간이  시작된 후 바로
    //   반환됩니다.

    cmxIxArcA(BoardID, MAPINDEX, dXCentOfs, dYCentOfs, dAngle, cmxFALSE);
end;
```

NAME

cmxIxArcATo

cmxIxArcAToStart

- 원호 보간(補間) 절대(絶對) 좌표 이송(移送)
(절대적 중심 좌표와 각도)

INFORMATION

Interpolation Motion

VC++/VB

BCB/Delphi/.NET

Level 3

이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인(確認)합니다.

SYNOPSIS

❒ VT_I4 cmxIxArcATo

([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_R8 XCent, [in] VT_R8 YCent, [in] VT_R8 EndAngle, [in] VT_I4 IsBlocking)

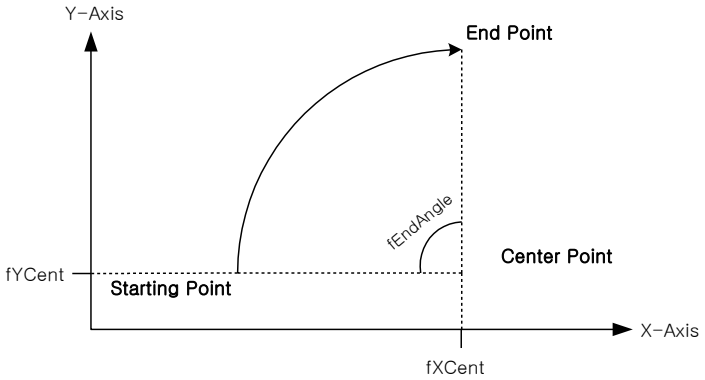
❒ VT_I4 cmxIxArcAToStart

([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_R8 XCent, [in] VT_R8 YCent, [in] VT_R8 EndAngle)

DESCRIPTION

중심좌표와 원호의 각도를 매개 변수(媒介變數)로 하여 원호보간이동을 수행합니다. 이때 중심좌표는 절대좌표로 표현됩니다. cmxIxArcATo() 함수는 모션이 완료되기 전까지 반환되지 않으며, cmxIxArcAToStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다. 원호보간은 임의의 두 축에 대해서 적용됩니다. 아래 설명에서는 맵핑된 두 축을 X, Y 축으로 간주하여 설명합니다. 여기서 X 축이라 함은 맵핑된 두 축 중에서 축번호가 낮은 축을 의미하며 Y 축은 축번호가 높은 축을 의미합니다. 예를 들어 Z 축과 U 축이 맵핑된 두 축이라면 Z 축이 X 축에 해당하며 U 축이 Y 축에 해당합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.



PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ MapIndex : 맵번호(Map index), 이 맵번호를 사용하기전에 먼저 cmxIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑되어 있어야 합니다.

▶ XCent : 중심점의 X 축 절대좌표. 좌표의 단위는 "Unit distance"에 의해 정의되는 논리적 거리를 적용합니다.

▶ YCent : 중심점의 Y 축 절대좌표. 좌표의 단위는 "Unit distance"에 의해 정의되는 논리적 거리를 적용합니다.

▶ EndAngle : 원호보간 이동을 완료할 목표지점의 현재 위치에 대한 각도값을 Degree(°)값으로 지정합니다. 각도의 부호가 (+)이면 반시계방향, (-)이면 시계방향으로의 이동을 의미합니다.

▶ IsBlocking : 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다.

Value	Meaning
0 또는 cmxFALSE	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
1 또는 cmxTRUE	블록(Blocking)을 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

SEE ALSO

☐ cmxIxArcAToStart() 함수를 사용하는 경우에는 cmxIxIsDone() 함수나 cmxIxWaitDone() 함수를 사용하여 모션의 완료를 확인(確認)할 수 있습니다.

☐ cmxIxArcATo() 함수를 사용하는 경우에는 내부적으로 루프를 수행하면서 모션이 완료되기를 기다리는데, 이때 "Blocking Mode"설정에 따라 윈도우 이벤트를 처리하는 방식이 달라집니다. 그러나 일반적으로 윈도우의 작업 쓰레드(Work Thread) 에서는 블록모드를 사용하여, 함수내부에서 지연없이 쓰레드 내부의 작업에 집중할 수 있도록 설정하는 것이 바람직합니다.

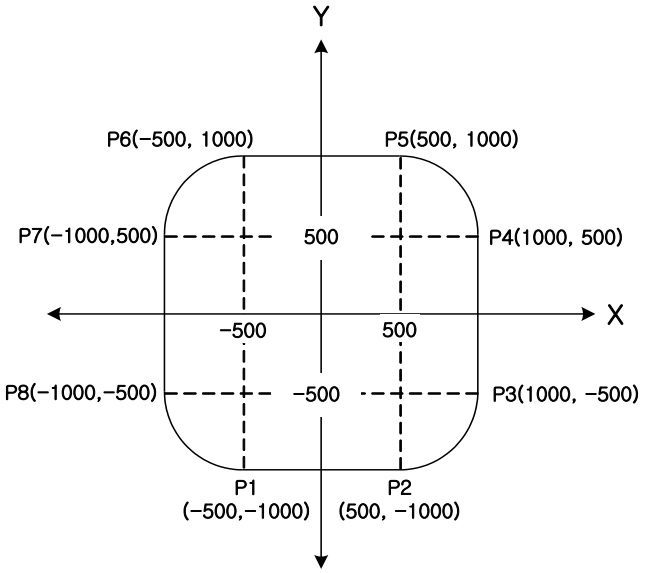
☐ cmxIxArcATo() 함수를 사용하는 경우에는 INP 입력신호가 Enable 로 설정되었으면 Command 펄스 출력이 완료되어도 INP 입력이 ON 이 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환되지 않습니다
스탭 드라이브는 INP 출력이 없는 경우가 일반적인데, 고객(顧客)님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스탭 드라이브를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다.
고객(顧客) 여러분들께서는 스탭 드라이브 사용시에 이점을 주의해주시기를 부탁드립니다.

보충	윈도우 이벤트라는 것은 무엇입니까?

	윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

EXAMPLE 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.



C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

// BoardID 는 0 으로 선언되었다고 가정함

/*************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.
*************************************************************/
void OnProgramInitial()
{

	long m_nNumDevices;
	long m_DeviceList[16];
	long m_nNumAxes;
```

```
	cmxLoadDll();

	if(cmxGnLoadDevice (&m_nNumDevices, m_DeviceList, &m_nNumAxes)!= ERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		// 에러메시지 출력
		return;
	}
}

/************************************************************
* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
* 호출되는 가상의 함수 입니다.
************************************************************/
#define MAP0 0 //맵번호 (0)
#define MAP1 1 //맵번호 (1)

void OnSetSpeed()
{
	cmxIxMapAxes(BoardID, MAP0, 0_MASK | cmY1_MASK, cmxIX_MODE_LINEAR );
	cmxIxMapAxes(BoardID, MAP1, 0_MASK | cmY1_MASK, cmxIX_MODE_CIRCULAR );

	//또는 cmxIxMapAxes(BoardID, MAP0, 0x3, 0x0);
	//cmxIxMapAxes(BoardID, MAP1, 0x3, cmxIX_MODE_CIRCULAR);

	//보간 이동할 축들의 기본속도를 설정합니다.
	cmxCfgSetSpeedPattern(BoardID, 3, cmxMODE_T, 1000, 5000, 5000,0,0);
	cmxCfgSetSpeedPattern(BoardID, cmY1, cmxMODE_T, 1000, 5000, 5000,0,0);
}

/************************************************************
* OnDoMotion() : 작업명령시에 호출되는 가상의 함수
************************************************************/
void OnDoMotion()
{
	double fPosList[2];

	//MAP0 를 마스터 속도 모드, 사다리꼴(Trapezoidal) 속도 패턴으로 작업속도의 100%,
	//가속도의 70%, 감속도의 70%로 설정 합니다.
	cmxIxSetSpeedPattern(BoardID, MAP0, cmxFALSE, cmxMODE_T,0,0, 100, 70, 70 );
	cmxIxSetSpeedPattern(BoardID, MAP1, cmxFALSE, cmxMODE_T,0,0, 100, 70, 70 );

	// Move from P1 to P2 //
	fPosList[0]=500; fPosList[1]=-1000;
	cmxIxLineTo(BoardID, MAP0, fPosList, cmxFALSE);

	// Move from P2 to P3 //
	cmxIxArcATo(BoardID, MAP1, 500, -500, 90, cmxFALSE);

	// Move from P3 to P4 //
	fPosList[0]=1000; fPosList[1]=500;
	cmxIxLineTo(BoardID, MAP0, fPosList, cmxFALSE);

	// Move from P4 to P5 //
	cmxIxArcATo(BoardID, MAP1, 500, 500, 90, cmxFALSE);

	// Move from P5 to P6 //
```

```
    fPosList[0]=-500; fPosList[1]=1000;
    cmxIxLineTo(BoardID, MAP0, fPosList, cmxFALSE);

    // Move from P6 to P7 //
    cmxIxArcATo(BoardID, MAP1, -500, 500, 90, cmxFALSE);

    // Move from P7 to P8 //
    fPosList[0]=-1000; fPosList[1]=-500;
    cmxIxLineTo(BoardID, MAP0, fPosList, cmxFALSE);

    // Move from P8 to P1 //
    cmxIxArcATo(BoardID, MAP1, -500, -500, 90, cmxFALSE);
}
```

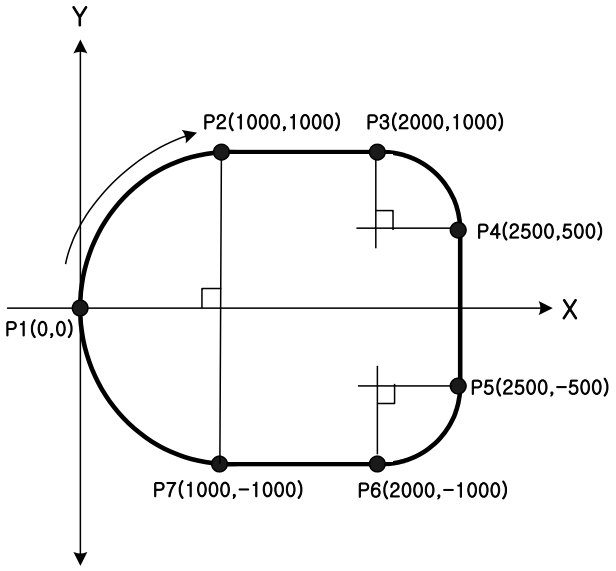
Visual Basic

```
'BoardID 는 0 으로 선언되었다고 가정함

'맵 번호 MAP0 은 이미 선언되어 있다고 가정함.
'/*****************************************************************
'* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
'* 호출되는 가상의 함수 입니다.
'*****************************************************************/
Private Sub OnSetSpeed()

    Call IxMapAxes(BoardID, MAP0, &H3, cmxIX_MODE_LINEAR) '//&H3 is 0 | cmxY1
    Call IxMapAxes(BoardID, MAP1, &H3, cmxIX_MODE_CIRCULAR) '//&H3 is 0 | cmxY1

    '보간 이동할 축들의 기본속도를 설정합니다.
    Call CfgSetSpeedPattern(BoardID, 3, cmxMODE_T, 1000, 5000, 5000,0,0)
    Call CfgSetSpeedPattern(BoardID, cmY1, cmxMODE_T, 1000, 5000, 5000,0,0)

End Sub


'/*****************************************************************
'* OnDoMotion() : 작업명령시에 호출되는 가상의 함수
'*****************************************************************/
Private Sub OnDoMotion()

    Dim fPosList(2) As Double

    '//MAP0 를 마스터 속도 모드, Trapezoidal 속도 패턴으로 작업속도의 100%,
    '//가속도의 70%, 감속도의 70%로 설정 합니다.
    Call IxSetSpeedPattern(BoardID, MAP0, cmxFALSE, cmxMODE_T,0,0, 100, 70, 70)
    Call IxSetSpeedPattern(BoardID, MAP1, cmxFALSE, cmxSMODE_T,0,0, 100, 70, 70)


    '// Move from P1 to P2 //
    fPosList(0) = 500
    fPosList(1) = -1000

    Call IxLineTo(BoardID, MAP0, fPosList(0), cmxFALSE)

    '// Move from P2 to P3 //
    Call IxArcATo(BoardID, MAP1, 500, -500, 90, cmxFALSE)
```

```
'// Move from P3 to P4 //
fPosList(0) = 1000
fPosList(1) = 500
Call IxLineTo(BoardID, MAP0, fPosList(0), cmxFALSE)

'// Move from P4 to P5 //
Call IxArcATo(BoardID, MAP1, 500, 500, 90, cmxFALSE)

'// Move from P5 to P6 //
fPosList(0) = -500
fPosList(1) = 1000
Call IxLineTo(BoardID, MAP0, fPosList(0), cmxFALSE)

'// Move from P6 to P7 //
Call IxArcATo(BoardID, MAP1, -500, 500, 90, cmxFALSE)

'// Move from P7 to P8 //
fPosList(0) = -1000
fPosList(1) = -500
Call IxLineTo(BoardID, MAP0, fPosList(0), cmxFALSE)

'// Move from P8 to P1 //
Call IxArcATo(BoardID, MAP1, -500, -500, 90, cmxFALSE)

End Sub
```

EXAMPLE 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.



C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

// BoardID 는 0 으로 선언되었다고 가정함

/*************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.
*************************************************************/
void OnProgramInitial()
{

	long m_nNumDevices;
	long m_DeviceList[16];
	long m_nNumAxes;
	cmxLoadDll();
	if(cmxGnLoadDevice (&m_nNumDevices, m_DeviceList, &m_nNumAxes)!= ERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		// 에러메시지 출력
		return;
	}
}

/*************************************************************
* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
* 호출되는 가상의 함수 입니다.
*************************************************************/
#define MAP0 0 //맵번호 (0)
#define MAP1 1 //맵번호 (1)

void OnSetSpeed()
{
	cmxIxMapAxes(BoardID, MAP0, cmxX1_MASK | cmxY1_MASK, cmxIX_MODE_LINEAR);
	cmxIxMapAxes(BoardID, MAP1, cmxX1_MASK | cmxY1_MASK,
cmxIX_MODE_CIRCULAR );

	//또는 cmxIxMapAxes(BoardID, MAP0, 0x3, 0x0);
	//cmxIxMapAxes(BoardID, MAP1, 0x3, cmxIX_MODE_CIRCULAR);
	//보간 이동할 축들의 기본속도를 설정합니다.
	cmxCfgSetSpeedPattern(BoardID, 3, cmxMODE_T, 1000, 5000, 5000,0,0);
	cmxCfgSetSpeedPattern(BoardID, cmY1, cmxMODE_T, 1000, 5000, 5000,0,0);
}

/*************************************************************
* OnDoMotion() : 작업명령시에 호출되는 가상의 함수
*************************************************************/
void OnDoMotion()
{
	double fPosList[2];

	//MAP0 를 마스터 속도 모드, Trapezoidal 속도 패턴으로 작업속도의 100%,
	//가속도의 70%, 감속도의 70%로 설정 합니다.
	cmxIxSetSpeedPattern(BoardID, MAP0, cmxFALSE, cmxMODE_T,0,0, 100, 70, 70 );
	cmxIxSetSpeedPattern(BoardID, MAP1, cmxFALSE, cmxSMODE_T,0,0, 100, 70, 70 );
```

```
	// Move from P1 to P2 //
	cmxIxArcATo(BoardID, MAP1, 1000, 0, -90, cmxFALSE);

	// Move from P2 to P3 //
	fPosList[0]=2000; fPosList[1]=1000;
	cmxIxLineTo(BoardID, MAP0, fPosList, cmxFALSE);

	// Move from P3 to P4 //
	cmxIxArcATo(BoardID, MAP1, 2000, 500, -90, cmxFALSE);

	// Move from P4 to P5 //
	fPosList[0]=2500; fPosList[1]=-500;
	cmxIxLineTo(BoardID, MAP0, fPosList, cmxFALSE);

	// Move from P5 to P6 //
	cmxIxArcATo(BoardID, MAP1, 2000, -500, -90, cmxFALSE);

	// Move from P6 to P7 //
	fPosList[0]=1000; fPosList[1]=-1000;
	cmxIxLineTo(BoardID, MAP0, fPosList, cmxFALSE);

	// Move from P7 to P1 //
	cmxIxArcATo(BoardID, MAP1, 1000, 0, -90, cmxFALSE);
}
```

Visual Basic

```
'BoardID 는 0 으로 선언되었다고 가정함

'맵 번호 MAP0 은 이미 선언되어 있다고 가정함.
'/****************************************************************
'* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
'* 호출되는 가상의 함수 입니다.
'****************************************************************/
Private Sub OnSetSpeed()

    Call IxMapAxes(BoardID, MAP0, &H3, cmxIX_MODE_LINEAR) '//&H3 = 0 | cmxY1
   Call IxMapAxes(BoardID, MAP1, &H3, cmxIX_MODE_CIRCULAR) '//&H3 = 0 | cmxY1

   '//보간 이동할 축들의 기본속도를 설정합니다.
   Call CfgSetSpeedPattern(BoardID, 3, cmxMODE_T, 1000, 5000, 5000,0,0)
   Call CfgSetSpeedPattern(BoardID, cmY1, cmxMODE_T, 1000, 5000, 5000,0,0)

End Sub

'/****************************************************************
'* OnDoMotion() : 작업명령시에 호출되는 가상의 함수
'****************************************************************/
Private Sub OnDoMotion()

	Dim fPosList(2) As Double

	'//MAP0 를 마스터 속도 모드, Trapezoidal 속도 패턴으로 작업속도의 100%,
	'//가속도의 70%, 감속도의 70%로 설정 합니다.
	Call IxSetSpeedPattern(BoardID, MAP0, cmxFALSE, cmxSMODE_T,0,0, 100, 70, 70)
	Call IxSetSpeedPattern(BoardID, MAP1, cmxFALSE, cmxSMODE_T,0,0, 100, 70, 70)
```

```
    '// Move from P1 to P2 //
    Call IxArcATo(BoardID, MAP1, 1000, 0, -90, cmxFALSE)

    '// Move from P2 to P3 //
    fPosList(0) = 2000
    fPosList(1) = 1000
    Call IxLineTo(BoardID, MAP0, fPosList(0), cmxFALSE)

    '// Move from P3 to P4 //
    Call IxArcATo(BoardID, MAP1, 2000, 500, -90, cmxFALSE)

    '// Move from P4 to P5 //
    fPosList(0) = 2500
    fPosList(1) = -500
    Call IxLineTo(BoardID, MAP0, fPosList(0), cmxFALSE)

    '// Move from P5 to P6 //
    Call IxArcATo(BoardID, MAP1, 2000, -500, -90, cmxFALSE)

    '// Move from P6 to P7 //
    fPosList(0) = 1000
    fPosList(1) = -1000
    Call IxLineTo(BoardID, MAP0, fPosList(0), cmxFALSE)

    '// Move from P7 to P1 //
    Call IxArcATo(BoardID, MAP1, 1000, 0, -90, cmxFALSE)

End Sub
```

NAME

cmxIxArcP

cmxIxArcPStart

- 원호 보간(補間) 상대(相對) 좌표 이송(移送)

(상대적 중심좌표와 종점 좌표)

INFORMATION

Interpolation Motion

VC++/VB

BCB/Delphi/.NET

Level 3

이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인(確認)합니다.

SYNOPSIS

❒ VT_I4 cmxIxArcP

([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_R8 XCentOffset, [in] VT_R8 YCentOffset, [in] VT_R8 XEndPointDist, [in] VT_R8 YEndPointDist, [in] VT_I4 Direction, [in] VT_I4 IsBlocking)

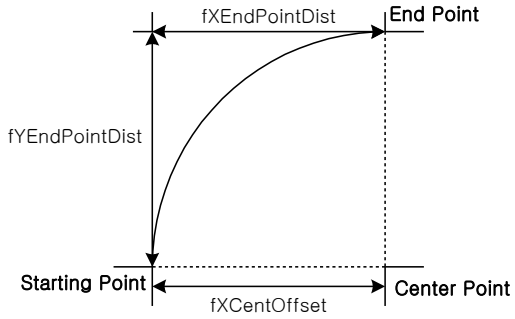
❒ VT_I4 cmxIxArcPStart

([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_R8 XCentOffset, [in] VT_R8 YCentOffset, [in] VT_R8 XEndPointDist, [in] VT_R8 YEndPointDist, [in] VT_I4 Direction)

DESCRIPTION

중심좌표와 종점좌표를 매개 변수(媒介變數)로 하여 원호보간이동을 수행합니다. 이때 각 좌표는 상대좌표로 표현됩니다. cmxIxArcP() 함수는 모션이 완료되기 전까지 반환되지 않으며, cmxIxArcPStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.
원호보간은 임의의 두 축에 대해서 적용됩니다. 아래 설명에서는 맵핑된 두 축을 X, Y 축으로 간주하여 설명합니다. 여기서 X 축이라 함은 맵핑된 두 축 중에서 축번호가 낮은 축을 의미하며 Y 축은 축번호가 높은 축을 의미합니다. 예를 들어 Z 축과 U 축이 맵핑된 두 축이라면 Z 축이 X 축에 해당하며 U 축이 Y 축에 해당합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.



PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ MapIndex : 맵번호(Map index), 이 맵번호를 사용하기전에 먼저 cmxIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 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.

▶ Direction : 회전 방향을 지정합니다.

Value	Meaning
0 또는 cmxARC_CW	시계 방향(CW)으로 회전
1 또는 cmxARC_CCW	반시계 방향(CCW)으로 회전

▶ IsBlocking : 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다.

Value	Meaning
0 또는 cmxFALSE	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
1 또는 cmxTRUE	블록(Blocking)을 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

SEE ALSO

□ cmxIxArcPStart() 함수를 사용하는 경우에는 cmxIxIsDone() 함수나 cmxIxWaitDone() 함수를 사용하여 모션의 완료를 확인(確認)할 수 있습니다.

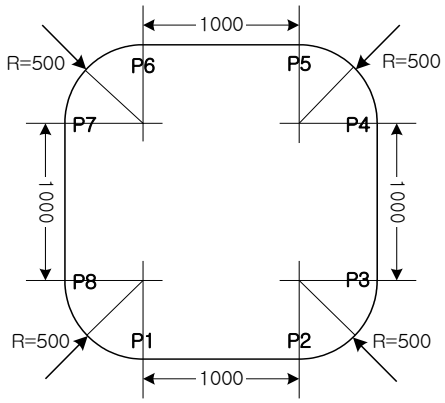
□ cmxIxArcP() 함수를 사용하는 경우에는 내부적으로 루프를 수행하면서 모션이 완료되기를 기다리는데, 이때 "Blocking Mode"설정에 따라 윈도우 이벤트를 처리하는 방식이 달라집니다.

□ cmxIxArcP() 함수를 사용하는 경우에는 INP 입력신호가 Enable 로 설정되었으면 Command 펄스 출력이 완료되어도 INP 입력이 ON 이 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환되지 않습니다.
스탭 드라이브는 INP 출력이 없는 경우가 일반적인데, 고객(顧客)님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스탭 드라이브를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다.
고객(顧客) 여러분들께서는 스탭 드라이브 사용시에 이점을 주의해주시기를 부탁드립니다.

보충	윈도우 이벤트라는 것은 무엇입니까?
	윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

EXAMPLE 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.



C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

// BoardID 는 0 으로 선언되었다고 가정함

/*************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.

*************************************************************/
void OnProgramInitial()
{

	long m_nNumDevices;
	long m_DeviceList[16];
	long m_nNumAxes;
	cmxLoadDll();
	if(cmxGnLoadDevice (&m_nNumDevices, m_DeviceList, &m_nNumAxes!= ERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		// 에러메시지 출력
		return;
	}
}
```

```
/*************************************************************
* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
* 호출되는 가상의 함수 입니다.
*************************************************************/
#define MAP0 0 //맵번호 (0)
#define MAP1 1 //맵번호 (1)

void OnSetSpeed()
{
	cmxIxMapAxes(BoardID, MAP0, cmxX1_MASK | cmxY1_MASK, cmxIX_MODE_LINEAR);
	cmxIxMapAxes(BoardID, MAP1, cmxX1_MASK | cmxY1_MASK,
cmxIX_MODE_CIRCULAR );

	//또는 cmxIxMapAxes(BoardID, MAP0, 0x3, cmxIX_MODE_LINEAR);
	//cmxIxMapAxes(BoardID, MAP1, 0x3, cmxIX_MODE_CIRCULAR);

	//보간 이동할 축들의 기본속도를 설정합니다.
	cmxCfgSetSpeedPattern(BoardID, 3, cmxMODE_T, 1000, 5000, 5000,0,0);
	cmxCfgSetSpeedPattern(BoardID, cmY1, cmxMODE_T, 1000, 5000, 5000,0,0);
}

/*************************************************************
* OnDoMotion() : 작업명령시에 호출되는 가상의 함수
*************************************************************/
void OnDoMotion()
{
	double fDistList[2];

	//MAP0 를 마스터 속도 모드, Trapezoidal 속도 패턴으로 작업속도의 100%,
	//가속도의 70%, 감속도의 70%로 설정 합니다.
	cmxIxSetSpeedPattern(BoardID, MAP0, cmxFALSE, cmxSMODE_T,0,0, 100, 70, 70 );
	cmxIxSetSpeedPattern(BoardID, MAP1, cmxFALSE, cmxSMODE_T,0,0, 100, 70, 70 );

	// Move from P1 to P2 //
	fDistList[0]=1000; fDistList[1]=0;
	cmxIxLine(BoardID, MAP0, fDistList, cmxFALSE);

	// Move from P2 to P3 //
	cmxIxArcP(BoardID, MAP1, 0, 500, 500, 500, cmARC_CCW, cmxFALSE);

	// Move from P3 to P4 //
	fDistList[0]=0; fDistList[1]=1000;
	cmxIxLine(BoardID, MAP0, fDistList, cmxFALSE);

	// Move from P4 to P5 //
	cmxIxArcP(BoardID, MAP1, -500, 0, -500, 500, cmARC_CCW, cmxFALSE);

	// Move from P5 to P6 //
	fDistList[0]=-1000; fDistList[1]=0;
	cmxIxLine(BoardID, MAP0, fDistList, cmxFALSE);

	// Move from P6 to P7 //
	cmxIxArcP(BoardID, MAP1, 0, -500, -500, -500, cmARC_CCW, cmxFALSE);

	// Move from P7 to P8 //
	fDistList[0]=0; fDistList[1]=-1000;
	cmxIxLine(BoardID, MAP0, fDistList, cmxFALSE);
```

```
    // Move from P8 to P1 //
    cmxIxArcP(BoardID, MAP1, 500, 0, 500, -500, cmARC_CCW, cmxFALSE);
}
```

NAME

cmxIxArcPTo

cmxIxArcPToStart

- 원호 보간(補間) 절대(絶對) 좌표 이송(移送)

(절대적 중심좌표와 종점 좌표)

INFORMATION

Interpolation Motion

VC++/VB

BCB/Delphi/.NET

Level 3

이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인(確認)합니다.

SYNOPSIS

❒ VT_I4 cmxIxArcPTo

([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_R8 XCent, [in] VT_R8 YCent, [in] VT_R8 XEndPos, [in] VT_R8 YEndPos, [in] VT_I4 Direction, [in] VT_I4 IsBlocking)

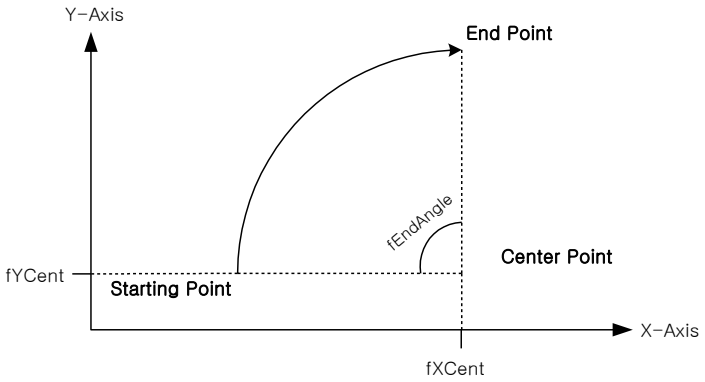
❒ VT_I4 cmxIxArcPToStart

([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_R8 XCent, [in] VT_R8 YCent, [in] VT_R8 XEndPos, [in] VT_R8 YEndPos, [in] VT_I4 Direction)

DESCRIPTION

중심좌표와 종점좌표를 매개 변수(媒介變數)로 하여 원호보간이동을 수행합니다. 이때 각 좌표는 절대좌표로 표현됩니다. cmxIxArcPTo() 함수는 모션이 완료되기 전까지 반환되지 않으며, cmxIxArcPToStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.
원호보간은 임의의 두 축에 대해서 적용됩니다. 아래 설명에서는 맵핑된 두 축을 X, Y 축으로 간주하여 설명합니다. 여기서 X 축이라 함은 맵핑된 두 축 중에서 축번호가 낮은 축을 의미하며 Y 축은 축번호가 높은 축을 의미합니다. 예를 들어 Z 축과 U 축이 맵핑된 두 축이라면 Z 축이 X 축에 해당하며 U 축이 Y 축에 해당합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.



PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ MapIndex : 맵번호(Map index), 이 맵번호를 사용하기전에 먼저 cmxIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 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

▶ Direction : 회전 방향을 지정합니다.

Value	Meaning
0 또는 cmxARC_CW	시계 방향(CW)으로 회전
1 또는 cmxARC_CCW	반시계 방향(CCW)으로 회전

▶ IsBlocking : 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다.

Value	Meaning
0 (cmxFALSE)	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
1 (cmxTRUE)	블록(Blocking)을 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

SEE ALSO

□ cmxIxArcPToStart() 함수를 사용하는 경우에는 cmxIxIsDone() 함수나 cmxIxWaitDone() 함수를 사용하여 모션의 완료를 확인(確認)할 수 있습니다.

□ cmxIxArcPTo() 함수를 사용하는 경우에는 내부적으로 루프를 수행하면서 모션이 완료되기를 기다리는데, 이때 "Blocking Mode"설정에 따라 윈도우 이벤트를 처리하는 방식이 달라집니다. 그러나 일반적으로 윈도우의 작업 쓰레드(Work Thread) 에서는 블록모드를 사용하여, 함수내부에서 지연없이 쓰레드 내부의 작업에 집중할 수 있도록 설정하는 것이 바람직합니다.

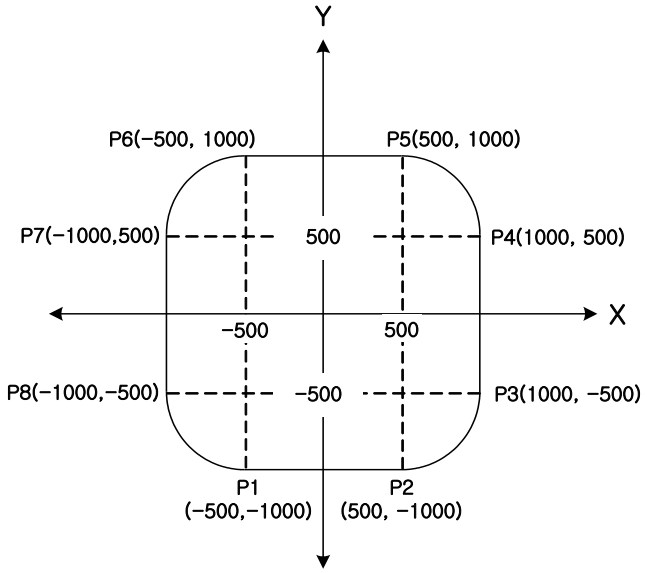
□ cmxIxArcPTo() 함수를 사용하는 경우에는 INP 입력신호가 Enable 로 설정되었으면 Command 펄스 출력이 완료되어도 INP 입력이 ON 이 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환되지 않습니다.
스탭 드라이브는 INP 출력이 없는 경우가 일반적인데, 고객(顧客)님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스탭 드라이브를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다.

고객(顧客) 여러분들께서는 스탭 드라이브 사용시에 이점을 주의해주시기를 부탁드립니다.

보충	윈도우 이벤트라는 것은 무엇입니까?
	윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

EXAMPLE 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.



C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

// BoardID 는 0 으로 선언되었다고 가정함

/*************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.
*************************************************************/
void OnProgramInitial()
{
```

```
	long m_nNumDevices;
	long m_DeviceList[16];
	long m_nNumAxes;
	cmxLoadDll();
	if(cmxGnLoadDevice (&m_nNumDevices, m_DeviceList, &m_nNumAxes)!= ERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		// 에러메시지 출력
		return;
	}
}

/************************************************************
* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
* 호출되는 가상의 함수 입니다.
* ***********************************************************/
#define MAP0 0 //맵번호 (0)
#define MAP1 1 //맵번호 (1)

void OnSetSpeed()
{
	cmxIxMapAxes(BoardID, MAP0, cmxX1_MASK | cmxY1_MASK, cmxIX_MODE_LINEAR);
	cmxIxMapAxes(BoardID, MAP1, cmxX1_MASK | cmxY1_MASK,
cmxIX_MODE_CIRCULAR );

	//또는 cmxIxMapAxes(BoardID, MAP0, 0x3, cmxIX_MODE_LINEAR);
	//cmxIxMapAxes(BoardID, MAP1, 0x3, cmxIX_MODE_CIRCULAR);

	//보간 이동할 축들의 기본속도를 설정합니다.
	cmxCfg000SetSpeedPattern(BoardID, 3, cmxMODE_T, 1000, 5000, 5000,0,0);
	cmxCfgSetSpeedPattern(BoardID, cmY1, cmxMODE_T, 1000, 5000, 5000,0,0);
}

/************************************************************
* OnDoMotion() : 작업명령시에 호출되는 가상의 함수
************************************************************/
void OnDoMotion()
{
	double fPosList[2];

	//MAP0 를 마스터 속도 모드, Trapezoidal 속도 패턴으로 작업속도의 100%,
	//가속도의 70%, 감속도의 70%로 설정 합니다.
	cmxIxSetSpeedPattern(BoardID, MAP0, cmxFALSE, cmxMODE_T,0,0, 100, 70, 70 );
	cmxIxSetSpeedPattern(BoardID, MAP1, cmxFALSE, cmxSMODE_T,0,0, 100, 70, 70 );

	// Move from P1 to P2 //
	fPosList[0]=500; fPosList[1]=-1000;
	cmxIxLineTo(BoardID, MAP0, fPosList, cmxFALSE);

	// Move from P2 to P3 //
	cmxIxArcPTo(BoardID, MAP1, 500, -500, 1000, -500, cmARC_CCW, cmxFALSE);

	// Move from P3 to P4 //
	fPosList[0]=1000; fPosList[1]=500;
	cmxIxLineTo(BoardID, MAP0, fPosList, cmxFALSE);

	// Move from P4 to P5 //
	cmxIxArcPTo(BoardID, MAP1, 500, 500, 500, 1000, cmARC_CCW, cmxFALSE);
```

```
    // Move from P5 to P6 //
    fPosList[0]=-500; fPosList[1]=1000;
    cmxIxLineTo(BoardID, MAP0, fPosList, cmxFALSE);

    // Move from P6 to P7 //
    cmxIxArcPTo(BoardID, MAP1, -500, 500, -1000, 500, cmARC_CCW, cmxFALSE);

    // Move from P7 to P8 //
    fPosList[0]=-1000; fPosList[1]=-500;
    cmxIxLineTo(BoardID, MAP0, fPosList, cmxFALSE);

    // Move from P8 to P1 //
    cmxIxArcPTo(BoardID, MAP1,-500, -500, -500, -1000, cmARC_CCW, cmxFALSE);

}
```

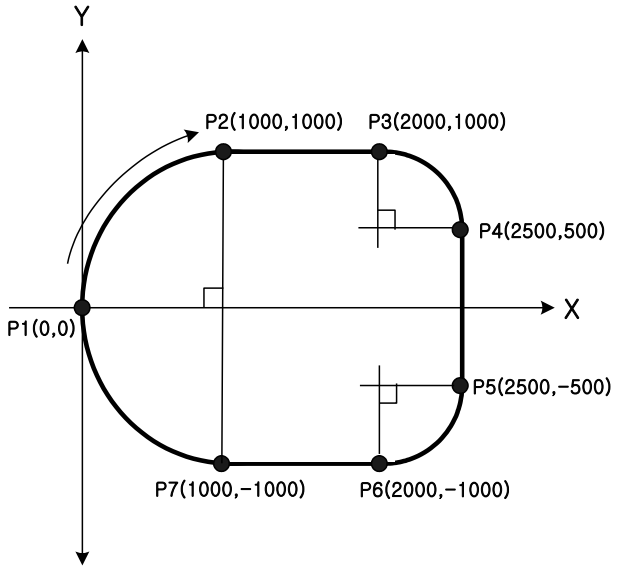
Visual Basic

```
'BoardID 는 0 으로 선언되었다고 가정함

'맵번호 MAP0 은 이미 선언되어 있다고 가정함.
'/*****************************************************************
'* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
'* 호출되는 가상의 함수 입니다.
'*****************************************************************/
Private Sub OnSetSpeed()

    Call IxMapAxes(BoardID, MAP0, &H3, cmxIX_MODE_LINEAR) '//&H3 = 0 | cmxY1
    Call IxMapAxes(BoardID, MAP1, &H3, cmxIX_MODE_CIRCULAR) '//&H3 = 0 | cmxY1

    '//보간 이동할 축들의 기본속도를 설정합니다.
    Call CfgSetSpeedPattern(BoardID, 3, cmxMODE_T, 1000, 5000, 5000,0,0)
    Call CfgSetSpeedPattern(BoardID, cmY1, cmxMODE_T, 1000, 5000, 5000,0,0)

End Sub

'/*****************************************************************
'* OnDoMotion() : 작업명령시에 호출되는 가상의 함수
'*****************************************************************/
Private Sub OnDoMotion()

    Dim fPosList(2) As Double
    '//MAP0 를 마스터 속도 모드, Trapezoidal 속도 패턴으로 작업속도의 100%,
    '//가속도의 70%, 감속도의 70%로 설정 합니다.
    Call IxSetSpeedPattern(BoardID, MAP0, cmxFALSE, cmxSMODE_T,0,0, 100, 70, 70)
    Call IxSetSpeedPattern(BoardID, MAP1, cmxFALSE, cmxSMODE_T,0,0, 100, 70, 70)

    '// Move from P1 to P2 //
    fPosList(0) = 500
    fPosList(1) = -1000
    Call IxLineTo(BoardID, MAP0, fPosList(0), cmxFALSE)

    '// Move from P2 to P3 //
    Call IxArcPTo(BoardID, MAP1, 500, -500, 1000, -500, cmARC_CCW, cmxFALSE)

    '// Move from P3 to P4 //
```

```
    fPosList(0) = 1000
    fPosList(1) = 500
    Call IxLineTo(BoardID, MAP0, fPosList(0), cmxFALSE)

    '// Move from P4 to P5 //
    Call IxArcPTo(BoardID, MAP1, 500, 500, 500, 1000, cmARC_CCW, cmxFALSE)

    '// Move from P5 to P6 //
    fPosList(0) = -500
    fPosList(1) = 1000
    Call IxLineTo(BoardID, MAP0, fPosList(0), cmxFALSE)

    '// Move from P6 to P7 //
    Call IxArcPTo(BoardID, MAP1, -500, 500, -1000, 500, cmARC_CCW, cmxFALSE)

    '// Move from P7 to P8 //
    fPosList(0) = -1000
    fPosList(1) = -500
    Call IxLineTo(BoardID, MAP0, fPosList(0), cmxFALSE)

    '// Move from P8 to P1 //
    Call IxArcPTo(BoardID, MAP1, -500, -500, -500, -1000, cmARC_CCW, cmxFALSE)

End Sub
```

EXAMPLE 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.



C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

// BoardID 는 0 으로 선언되었다고 가정함

/*************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.

*************************************************************/
void OnProgramInitial()
{

	long m_nNumDevices;
	long m_DeviceList[16];
	long m_nNumAxes;
	cmxLoadDll();
	if(cmxGnLoadDevice (&m_nNumDevices, m_DeviceList, &m_nNumAxes)!= ERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		// 에러메시지 출력
		return;
	}
}

/*************************************************************
* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
* 호출되는 가상의 함수 입니다.
*************************************************************/
#define MAP0 0 //맵번호 (0)
void OnSetSpeed()
{
	cmxIxMapAxes(BoardID, MAP0, cmxX1_MASK | cmxY1_MASK, cmxIX_MODE_LINEAR);
	cmxIxMapAxes(BoardID, MAP1, cmxX1_MASK | cmxY1_MASK,
cmxIX_MODE_CIRCULAR );

	//또는 cmxIxMapAxes(BoardID, MAP0, 0x3, cmxIX_MODE_LINEAR);
	//cmxIxMapAxes(BoardID, MAP1, 0x3, cmxIX_MODE_CIRCULAR);

	//보간 이동할 축들의 기본속도를 설정합니다.
	cmxCfgSetSpeedPattern(BoardID, 3, cmxSMODE_T, 1000, 5000, 5000,0,0);
	cmxCfgSetSpeedPattern(BoardID, 3, cmxSMODE_T, 1000, 5000, 5000,0,0);
}

/*************************************************************
* OnDoMotion() : 작업명령시에 호출되는 가상의 함수
*************************************************************/
void OnDoMotion()
{
	double fPosList[2];

	//MAP0 를 마스터 속도 모드, Trapezoidal 속도 패턴으로 작업속도의 100%,
	//가속도의 70%, 감속도의 70%로 설정 합니다.
	cmxIxSetSpeedPattern(BoardID, MAP0, cmxFALSE, cmxSMODE_T,0,0, 100, 70, 70 );
	cmxIxSetSpeedPattern(BoardID, MAP1, cmxFALSE, cmxSMODE_T,0,0, 100, 70, 70 );

	// Move from P1 to P2 //
	cmxIxArcPTo(BoardID, MAP1, 1000, 0, 1000, 1000, cmARC_CW, cmxFALSE);
```

```
	// Move from P2 to P3 //
	fPosList[0]=2000; fPosList[1]=1000;
	cmxIxLineTo(BoardID, MAP0, fPosList, cmxFALSE);

	// Move from P3 to P4 //
	cmxIxArcPTo(BoardID, MAP1, 2000, 500, 2500, 500, cmARC_CW, cmxFALSE);

	// Move from P4 to P5 //
	fPosList[0]=2500; fPosList[1]=-500;
	cmxIxLineTo(BoardID, MAP0, fPosList, cmxFALSE);

	// Move from P5 to P6 //
	cmxIxArcPTo(BoardID, MAP1, 2000, -500, 2000, -1000, cmARC_CW, cmxFALSE);

	// Move from P6 to P7 //
	fPosList[0]=1000; fPosList[1]=-1000;
	cmxIxLineTo(BoardID, MAP0, fPosList, cmxFALSE);

	// Move from P7 to P1 //
	cmxIxArcPTo(BoardID, MAP1, 1000, 0, 0, 0, cmARC_CW, cmxFALSE);
}
```

Visual Basic

```
'BoardID 는 0 으로 선언되었다고 가정함

'맵번호 MAP0 은 이미 선언되어 있다고 가정함.
'/*************************************************************
'* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
'* 호출되는 가상의 함수 입니다.
'*************************************************************/
Private Sub OnSetSpeed()

	Call IxMapAxes(BoardID, MAP0, &H3, cmxIX_MODE_LINEAR)
	Call IxMapAxes(BoardID, MAP1, &H3, cmxIX_MODE_CIRCULAR)

	'//또는 IxMapAxes(BoardID, MAP0, 0x3, cmxIX_MODE_LINEAR)
	'//IxMapAxes(BoardID, MAP1, 0x3, cmxIX_MODE_CIRCULAR)


	'//보간 이동할 축들의 기본속도를 설정합니다.
	Call CfgSetSpeedPattern(BoardID, 3, cmxMODE_T, 1000, 5000, 5000,0,0)
	Call CfgSetSpeedPattern(BoardID, cmY1, cmxMODE_T, 1000, 5000, 5000,0,0)

End Sub

'/*************************************************************
'* OnDoMotion() : 작업명령시에 호출되는 가상의 함수
'*************************************************************/
Private Sub OnDoMotion()

	Dim fPosList(2) As Double
	'//MAP0 를 마스터 속도 모드, Trapezoidal 속도 패턴으로 작업속도의 100%,
	'//가속도의 70%, 감속도의 70%로 설정 합니다.
	Call IxSetSpeedPattern(BoardID, MAP0, cmxFALSE, cmxSMODE_T,0,0, 100, 70, 70)
	Call IxSetSpeedPattern(BoardID, MAP1, cmxFALSE, cmxSMODE_T,0,0, 100, 70, 70)
```

```
    '// Move from P1 to P2 //
    Call IxArcPTo(BoardID, MAP1, 1000, 0, 1000, 1000, cmARC_CW, cmxFALSE)
    '// Move from P2 to P3 //
    fPosList(0) = 2000
    fPosList(1) = 1000
    Call IxLineTo(BoardID, MAP0, fPosList(0), cmxFALSE)

    '// Move from P3 to P4 //
    Call IxArcPTo(BoardID, MAP1, 2000, 500, 2500, 500, cmARC_CW, cmxFALSE)

    '// Move from P4 to P5 //
    fPosList(0) = 2500
    fPosList(1) = -500
    Call IxLineTo(BoardID, MAP0, fPosList(0), cmxFALSE)

    '// Move from P5 to P6 //
    Call IxArcPTo(BoardID, MAP1, 2000, -500, 2000, -1000, cmARC_CW, cmxFALSE)

    '// Move from P6 to P7 //
    fPosList(0) = 1000
    fPosList(1) = -1000
    Call IxLineTo(BoardID, MAP0, fPosList(0), cmxFALSE)

    '// Move from P7 to P1 //
    Call IxArcPTo(BoardID, MAP1, 1000, 0, 0, 0, cmARC_CW, cmxFALSE)

End Sub
```

NAME

cmxIxArc3P

cmxIxArc3PStart

- 3 점을 통한 원호보간

INFORMATION

Interpolation Motion

VC++/VB

BCB/Delphi/.NET

Level 3

이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인(確認)합니다.

SYNOPSIS

❒ VT_I4 cmxIxArc3P

([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_R8 P2[], [in] VT_R8 P3[], [in] VT_R8 EndAngle, [in] VT_I4 IsBlocking)

❒ VT_I4 cmxIxArc3PStart

([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_R8 P2[], [in] VT_R8 P3[], [in] VT_R8 EndAngle)

DESCRIPTION

현재 좌표와 두 개의 매개변수로 지원되는 좌표를 통하여 원호보간이동을 수행합니다. cmxIxArc3P() 함수는 모션이 완료되기 전까지 반환되지 않으며, cmxIxArc3PStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.
원호보간은 현재 좌표와 임의의 두 좌표에 대해서 적용됩니다. 세 점을 통해 만들어지는 원에서 매개변수로 주어지는 각의 값만큼 보간 이송을 합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ MapIndex : 맵번호(Map index), 이 맵번호를 사용하기 전에 먼저 cmxIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑되어 있어야 합니다.

▶ P2[] : 두번째 좌표배열입니다.

▶ P3[] : 세번째 좌표배열입니다.

▶ EndAngle: 현재 좌표에서 원하는 위치까지의 각도를 나타냅니다. 현재 좌표에서 세 점을 통해 만들어지는 원위를 이 파라미터 값만큼 보간이송합니다.

▶ IsBlocking : 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다.

Value	Meaning
0 (cmxFALSE)	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
1 (enTRUE)	블록(Blocking)을 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

SEE ALSO

☐ cmxIxArc3PStart() 함수를 사용하는 경우에는 cmxIxIsDone() 함수나 cmxIxWaitDone() 함수를 사용하여 모션의 완료를 확인(確認)할 수 있습니다.

☐ cmxIxArc3P() 함수를 사용하는 경우에는 내부적으로 루프를 수행하면서 모션이 완료되기를 기다리는데, 이때 "Blocking Mode"설정에 따라 윈도우 이벤트를 처리하는 방식이 달라집니다. 그러나 일반적으로 윈도우의 작업 쓰레드(Work Thread) 에서는 블록모드를 사용하여, 함수내부에서 지연없이 쓰레드 내부의 작업에 집중할 수 있도록 설정하는 것이 바람직합니다.

☐ cmxIxArc3P() 함수를 사용하는 경우에는 INP 입력신호가 Enable 로 설정되었으면 Command 펄스 출력이 완료되어도 INP 입력이 ON 이 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환되지 않습니다.
스탭 드라이브는 INP 출력이 없는 경우가 일반적인데, 고객(顧客)님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스탭 드라이브를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다.

고객(顧客) 여러분들께서는 스탭 드라이브 사용시에 이점을 주의해주시기를 부탁드립니다.

윈도우 이벤트라는 것은 무엇입니까?
윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

NAME

cmxIxArc3PTo

cmxIxArc3PToStart

- 3 점을 통한 원호보간

INFORMATION

Interpolation Motion

VC++/VB

BCB/Delphi/.NET

Level 3

이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인(確認)합니다.

SYNOPSIS

❒ VT_I4 cmxIxArc3P

([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_R8 P2[], [in] VT_R8 P3[], [in] VT_R8 EndAngle, [in] VT_I4 IsBlocking)

❒ VT_I4 cmxIxArc3PStart

([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_R8 P2[], [in] VT_R8 P3[], [in] VT_R8 EndAngle)

DESCRIPTION

현재 절대(絶對)적 좌표와 두 개의 매개변수로 지원되는 좌표를 통하여 원호보간이동을 수행합니다. cmxIxArc3P() 함수는 모션이 완료되기 전까지 반환되지 않으며, cmxIxArc3PStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.
원호보간은 현재 절대(絶對)적 좌표와 임의의 두 좌표에 대해서 적용됩니다. 세 점을 통해 만들어지는 원에서 매개변수로 주어지는 각의 값만큼 보간 이송을 합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ MapIndex : 맵번호(Map index), 이 맵번호를 사용하기 전에 먼저 cmxIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑되어 있어야 합니다.

▶ P2[] : 두번째 좌표배열입니다.

▶ P3[] : 세번째 좌표배열입니다.

▶ EndAngle: 현재 절대(絶對)적 좌표에서 원하는 위치까지의 각도를 나타냅니다. 현재 좌표에서 세 점을 통해 만들어지는 원위를 이 파라미터 값만큼 보간이송합니다.

▶ IsBlocking : 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다.

Value	Meaning
0 (cmxFALSE)	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
1 (enTRUE)	블록(Blocking)을 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

SEE ALSO

☐ cmxIxArc3PToStart() 함수를 사용하는 경우에는 cmxIxIsDone() 함수나 cmxIxWaitDone() 함수를 사용하여 모션의 완료를 확인(確認)할 수 있습니다.

☐ cmxIxArc3PTo() 함수를 사용하는 경우에는 내부적으로 루프를 수행하면서 모션이 완료되기를 기다리는데, 이때 "Blocking Mode"설정에 따라 윈도우 이벤트를 처리하는 방식이 달라집니다. 그러나 일반적으로 윈도우의 작업 쓰레드(Work Thread) 에서는 블록모드를 사용하여, 함수내부에서 지연없이 쓰레드 내부의 작업에 집중할 수 있도록 설정하는 것이 바람직합니다.

☐ cmxIxArc3PTo() 함수를 사용하는 경우에는 INP 입력신호가 Enable 로 설정되었으면 Command 펄스 출력이 완료되어도 INP 입력이 ON 이 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환되지 않습니다.
스탭 드라이브는 INP 출력이 없는 경우가 일반적인데, 고객(顧客)님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스탭 드라이브를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다.

고객(顧客) 여러분들께서는 스탭 드라이브 사용시에 이점을 주의해주시기를 부탁드립니다.

윈도우 이벤트라는 것은 무엇입니까?
윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

NAME

cmxIxIsDone

- 보간(補間) 모션 완료(完了) 확인(確認)

INFORMATION

- Interpolation Motion
- VC++/VB
- BCB/Delphi/.NET
- Level 3
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxIxIsDone ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [out] VT_PI4 IsDone)

DESCRIPTION

지정한 보간맵에 해당하는 보간작업이 완료됐는지를 확인(確認)합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ MapIndex : 맵번호(Map index), 이 맵번호를 사용하기전에 먼저 cmxIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑되어 있어야 합니다.

▶ IsDone : 이 매개 변수로 인해 모션 작업이 완료되었는지를 판단할 수 있습니다.

Value	Meaning
0	모션작업이 완료되지 않음
1	모션작업이 완료됨

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

EXAMPLE

C/C++

```
#define MAP0 0 //맵번호 (0)

Long BoardID = 0;

cmxIxMapAxes(BoardID, MAP0, cmxX1_MASK | cmxY1_MASK, cmxIX_MODE_LINEAR);
//또는 cmxIxMapAxes(BoardID, MAP0, 0x3, 0x0);
//보간 이동할 축들의 기본속도를 설정합니다.
```

```
//속도 패턴 설정

long nIsDone = 0;
double fDistList[2] = {1000, 1000};

if(cmxIxLine(BoardID, MAP0, fDistList, cmxFALSE) != ERR_NONE){
        // 에러메시지 출력
        return;
}

while (1){
        cmxIxIsDone(BoardID, MAP0, &nIsDone );
        if(nIsDone == cmxTRUE) break;
        else{
                ...
        }
}
```

Visual Basic

```
//BoardID 는 0 으로 선언되었다고 가정함

Dim fDistList As Double
Dim nIsDone As Long

If (IxLineStart(BoardID, MAP0, fDistList(0)) = cmxFALSE) Then
        // 에러메시지 출력
        Exit Sub
End If

While (IxIsDone(BoardID, 3, nIsDone) = cmxFALSE)
…
end
If (Not (ErrGetLastCode(nErrCode) = ERR_NONE)) Then
        // 에러메시지 출력
        Exit Sub
End If
```

NAME

cmxIxWaitDone

- 보간(補間) 모션 완료(完了) 대기(待機)

INFORMATION

- Interpolation Motion
- VC++/VB
- BCB/Delphi/.NET
- Level 3
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxIxWaitDone ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_I4 IsBlocking)

DESCRIPTION

지정한 보간맵에 해당하는 보간작업이 완료(完了)될 때까지 기다립니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶MapIndex : 맵번호(Map index), 이 맵번호를 사용하기전에 먼저 cmxIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑되어 있어야 합니다.

▶ IsBlocking : 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다.

Value	Meaning
0	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
1	블록(Blocking)을 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

REFERENCE

□ INP 입력신호가 Enable 로 설정되었으면 Command 펄스 출력이 완료되어도 INP 입력이 ON 이 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환되지 않습니다.

□ 스탭 드라이브를 사용 중인 고객(顧客)님들께서는 다음을 참조해 주십시요.
스탭 드라이브는 INP 출력이 없는 경우가 일반적인데, 고객(顧客)님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스탭 드라이브를

통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다.
고객(顧客) 여러분들께서는 스탭 드라이브 사용시에 이점을 주의해주시기를 부탁드립니다.

□ 서보 드라이브의 LSP , LSN 신호를 사용 중인 고객(顧客)님께서는 다음을 참조해 주십시요.
서보드라이브의 입력 신호 중 하나인 EL(End of Limit) 신호는 저희 ㈜커미조아 모션 보드 뿐만 아니라 서보드라이브에도 전달 될 수 있도록 설정할 수 있습니다. 통상적으로 LSP 신호와 LSN 신호로 불리어 지는 이 신호는 실제 기구물에서 양의 방향(Positive Direction) 혹은 음의 방향(Negative Direction) 에 장착되어 있는 EL(End of Limit) 신호를 서보 드라이브 측에 전달하기 위한 용도로 사용됩니다.

그러나, 모션 소프트웨어에서 INP 설정이 되어 있는 경우 EL 신호가 검출 된 후에 일부 서보 드라이브에서는 진행 방향에서 정지(停止) 한후 더 이상 움직이지 않는 상황이 발생하며, 이 상황에서 INP 신호가 출력되지 않아, 모션 이송이 완료되지 못하고, 명시적으로 STOP 을 해줘야만 하는 강제적으로 모션 종료가 되는 현상이 발생할 수 있습니다. 이 현상은 EL 모드를 통해 원점 복귀를 하는 상황에서도 발생할 수 있습니다.

따라서, 이러한 경우에는 반드시 원점 복귀나 EL 검출시에 인터럽트 이벤트나 타이머를 통해 INP 를 무기한 대기하는 현상에 대해서 적절히 대처하시거나 INP 신호 사용을 배제 해야 합니다.

윈도우 이벤트라는 것은 무엇입니까?
윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

EXAMPLE

C/C++

```
#define MAP0 0 //맵번호 (0)

// BoardID 는 0 으로 선언되었다고 가정함

cmxIxMapAxes(BoardID, MAP0, cmxX1_MASK | cmY1_MASK, 0 );
//또는 cmxIxMapAxes(BoardID, MAP0, 0x3, 0x0);
//보간 이동할 축들의 기본속도를 설정합니다.

...//속도 패턴 설정

long nIsDone = 0;
double fDistList[2] = {1000, 1000};

if(cmxIxLine(BoardID, MAP0, fDistList, cmxFALSE) != ERR_NONE){
        // 에러메시지 출력
        return;
}

//모션이 완료 될 때 까지 기다립니다.
if(cmxIxWaitDone(BoardID, MAP0, cmxFALSE) != ERR_NONE){
        // 에러메시지 출력
        return ;
}
```

NAME

cmxIxStop

cmxIxStopEmg

보간(補間) 이송 정지(停止)

비상 정지(非常停止)

INFORMATION

Interpolation Motion

VC++/VB

BCB/Delphi/.NET

Level 3

정지(停止) 함수

고속 이송시에 급 정지(停止)(비상정지(停止)를 주의하십시요. 기구물의 손상이나 안전사고에 원인이 될 수 있습니다.

SYNOPSIS

❒ VT_I4 cmxIxStop ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_I4 IsWaitComplete, [in] VT_I4 IsBlocking)

❒ VT_I4 cmxIxStopEmg ([in] VT_I4 BoardID, [in] VT_I4 MapIndex)

DESCRIPTION

지정한 보간맵에 대한 보간작업을 정지(停止)합니다. 정지(停止)시에 cmxIxStop() 함수를 사용하면 감속 후 정지(停止)하며, cmxIxStopEmg()를 사용하면 감속없이 즉시정지(停止)를 수행합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ MapIndex : 맵번호(Map index), 이 맵번호를 사용하기전에 먼저 cmxIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑되어 있어야 합니다.

▶ IsWaitComplete : 완료될때까지 기다리는지의 여부.

▶ IsBlocking : 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다.

Value	Meaning
0	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
1	블록(Blocking)을 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

REFERENCE

□ INP 입력신호가 Enable 로 설정되었으면 Command 펄스 출력이 완료되어도 INP 입력이 ON 이 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환되지 않습니다.

□ 스탭 드라이브를 사용 중인 고객(顧客)님들께서는 다음을 참조해 주십시요.
스탭 드라이브는 INP 출력이 없는 경우가 일반적인데, 고객(顧客)님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스탭 드라이브를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다.
고객(顧客) 여러분들께서는 스탭 드라이브 사용시에 이점을 주의해주시기를 부탁드립니다.

□ 서보 드라이브의 LSP , LSN 신호를 사용 중인 고객(顧客)님께서는 다음을 참조해 주십시요.
서보드라이브의 입력 신호 중 하나인 EL(End of Limit) 신호는 저희 ㈜커미조아 모션 보드 뿐만 아니라 서보드라이브에도 전달 될 수 있도록 설정할 수 있습니다. 통상적으로 LSP 신호와 LSN 신호로 불리어 지는 이 신호는 실제 기구물에서 양의 방향(Positive Direction) 혹은 음의 방향(Negative Direction) 에 장착되어 있는 EL(End of Limit) 신호를 서보 드라이브 측에 전달하기 위한 용도로 사용됩니다.

그러나, 모션 소프트웨어에서 INP 설정이 되어 있는 경우 EL 신호가 검출 된 후에 일부 서보 드라이브에서는 진행 방향에서 정지(停止) 한후 더 이상 움직이지 않는 상황이 발생하며, 이 상황에서 INP 신호가 출력되지 않아, 모션 이송이 완료되지 못하고, 명시적으로 STOP 을 해줘야만 하는 강제적으로 모션 종료가 되는 현상이 발생할 수 있습니다. 이 현상은 EL 모드를 통해 원점 복귀를 하는 상황에서도 발생할 수 있습니다.

따라서, 이러한 경우에는 반드시 원점 복귀나 EL 검출시에 인터럽트 이벤트나 타이머를 통해 INP 를 무기한 대기하는 현상에 대해서 적절히 대처하시거나 INP 신호 사용을 배제 해야 합니다.

윈도우 이벤트라는 것은 무엇입니까?
윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

EXAMPLE

C/C++

```
// BoardID 는 0 으로 선언되었다고 가정함

#define MAP0 0

Void OnStop() {
    if(cmxIxStop(BoardID, MAP0, cmxTRUE, cmxFALSE) != ERR_NONE){
        // 에러메시지 출력
```

```
}
```

```
Delphi
//* BoardID 는 0 으로 선언되었다고 가정함

Const
MAPINDEX = 0;

// * Description :
// *
// *  현재 수행되고 있는 모션 동작에 대해서 감속 후 정지(停止) 합니다.
procedure btnStopClick();
begin
      cmxIxStop(BoardID, MAPINDEX,cmxTRUE, cmxFALSE);
end;
```

```
Visual Basic
'BoardID 는 0 으로 선언되었다고 가정함
'맵 번호 MAP0 은 이미 선언되어 있다고 가정함.

''//Description
''//*
''//현재 수행되고 있는 모션 동작에 대해서 감속후 정지(停止)합니다.
Private Sub btnStop_Click();
Begin
       IxStop(BoardID, MAP0, cmxTRUE, cmxFALSE)
end
```

8.4 원점복귀(Home Return)

이 단원에서는 원점 복귀(Home Return)에 관련된 함수들을 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(Zero)로 초기화됩니다.

원점 복귀 작업을 수행하기 위해서는 ORG(HOME), EZ 및 EL 신호가 참조되는데 이 신호들의 의미 및 작용은 다음과 같습니다.

□ ORG (HOME) 신호

ORG 신호는 구조물이 원점에 복귀했는지를 센서로부터 입력받는 신호입니다. 일반적으로는 근접 센서와 같은 센서들을 이용하여 원점 복귀 여부를 감지하게 됩니다. 이 신호는 'HOME'단자를 통하여 입력되어야 합니다.

□ EZ (C 상) 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 'EZ+'단자와 'EZ-'단자를 통하여 입력되어야 합니다.

□ EL 신호

기계적인 리밋(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 리밋 신호와 (-)방향 리밋 신호의 두 가지 신호가 있습니다. (+)방향 리밋 신호는 '+EL'단자, 그리고 (-)방향 리밋 신호는'-EL'단자를 통하여 입력되어야 합니다.

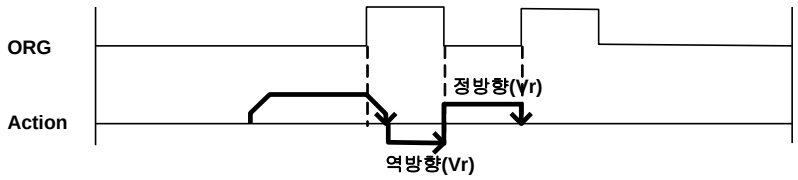
8.4.1 원점복귀모드 안내

㈜커미조아 NEMO-ME 모션제어보드는 다음과 같이 2 가지의 원점 복귀 모드를 제공합니다. 각 원점복귀모드에 따른 동작방식은 아래 설명과 같습니다. 아래의 그림은 모두 속도모드를 Trapezoidal 모드로 설정한 상태임을 가정하여 그려진 것이며, 만일 Constant 속도 모드로 설정된 경우에는 감속이 없이 즉시 정지(停止)하게 됩니다.

□ MODE 1 : ORG ON => Stop => Back (V2) => ORG OFF => Forward(V3) => ORG ON => Stop

MODE 1 에서는 ORG 신호가 OFF 에서 ON 으로 바뀌는 순간에 모션을 감속 후 정지한 후 ORG 신호가 OFF 가 될 때까지 V2(2nd Phase Speed)의 속도로 역방향 회전을 수행합니다. ORG 신호가 OFF 되면 다시 V3(3rd Phase Speed) 의 속도로 정방향 회전을 수행하다가 ORG 신호가 다시 ON 되는 순간에 복귀작업을 종료합니다.

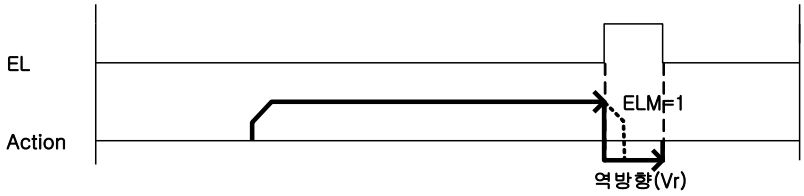
이 모드를 사용할 때 주의할 것은 처음 ORG 신호가 ON 으로 변경되어 감속 후 정지하는 도중에 ORG 센서의 ON 으로 유지되는 시간이 짧아서 이미 OFF 상태로 변경되면 역방향 이동을 생략하고 최종적으로 원점 센서를 찾으려는 단계로 넘어갑니다. 따라서 이러한 경우에는 (-)Limit 위치까지 이동하게 됩니다. 이러한 경우에도 자동으로 다시 탈출하여 정상적인 원점복귀 작업을 다시 수행하지만 의도하지 않게 원점복귀 시간이 길어질 수 있습니다. 이를 방지하는 방법은 구조적으로 ORG 신호가 ON 으로 유지되는 시간을 길게 해주거나 또는 원점복귀 시의 작업속도를 낮추거나 감속도를 크게 해 주어 감속 시 이동되는 거리를 짧게 해주면 됩니다.



Vr : Reverse Speed를 의미합니다.

□ MODE 3 : EL ON => Stop => Back (V2) => EL OFF => Forward(V3) => EL ON => Stop

MODE 3 에서는 EL 신호가 ON 으로 바뀌는 순간 즉시 정지(또는 ELM=1 인 경우에 감속 후 정지합니다. 그리고 반대 방향으로 V2 속도로 회전하다가 EL 신호가 OFF 되면 다시 V3(3rd Phase Speed) 의 속도로 정방향 회전을 수행하다가 EL 신호가 다시 ON 되는 순간에 복귀작업을 종료합니다. 여기서 ELM=1 은 EL 의 "Stop mode"가 "감속 후 정지" 모드로 설정됨을 의미합니다.



Vr : Reverse Speed

8.4.2 자동원점 검색 기능에 대하여

일반적으로 기구물의 위치는 원점센서를 기준으로 (+) 방향의 위치에 있으며, 따라서 (-) 방향으로 원점복귀를 수행하면 됩니다. 하지만 원점복귀를 시작하는 시점에 기구물이 이미 원점센서가 ON 이 되어 있는 위치에 있거나 원점센서와 (-)EL 센서 사이에 위치한 경우에는 원점센서를 기준으로 (+) 방향의 위치까지 탈출한 후에 정상적인 원점복귀 작업을 수행하여야 합니다. 이러한 기능을 "자동원점 검색" 기능이라 하며, ㈜커미조아의 모션제어보드는 이 기능을 자동으로 수행해주므로 기구물의 위치에 관계없이 원점복귀 작업을 수행할 수 있습니다.

기구물과 각 센서들의 위치에 따른 원점복귀 절차는 다음과 같이 약간씩 다릅니다. 단, 원점복귀 작업의 방향을 음의 방향으로 한경우에 대한 것입니다. 만일 원점복귀를 양의 방향으로 한 경우에는 –EL 신호 대신 +EL 신호가 사용됩니다.

□ CASE 1. 원점센서를 기준으로 양의 방향 위치에서 원점복귀를 시작한 경우

정상적인 원점복귀 작업을 수행합니다.

☐ CASE 2. 원점센서가 ON 인 상태에서 원점복귀를 시작한 경우

이러한 경우에는 먼저 원점 탈출거리(Escape distance) 만큼 양의 방향으로 이동한 후 다시 정상적인 원점복귀 작업을 수행합니다. 원점 탈출거리만큼 탈출하였어도 원점센서가 ON 상태이면 원점 탈출거리만큼 탈출하는 작업을 반복하므로 탈출거리가 짧아도 원점을 탈출하는데는 아무 문제가 없습니다.

☐ CASE 3. 원점센서를 기준으로 음의 방향 위치에서 원점복귀를 시작한 경우

이러한 경우에는 먼저 음의 방향으로 이동을 시작합니다. 그리고 -EL(Negative Limit) 센서가 ON 되면 정지(停止) 후 양의 방향으로 이동합니다. 원점센서가 ON 되면 다시 정지(停止) 후 CASE 2 에서와 같이 원점 탈출거리만큼 양의 방향으로 이동한 후 다시 정상적인 원점복귀 작업을 수행합니다.

[그림 3-7]은 원점복귀모드를 0, 속도 패턴을 사다리꼴 방식으로 하고 원점복귀 작업을 수행할 때 위의 각 경우에 대하여 동작하는 방식을 도식적으로 나타낸 것입니다.

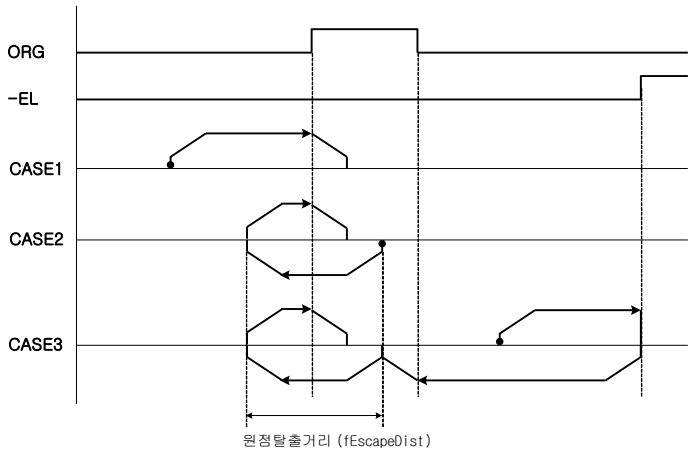


그림 8-3 기구물과 센서의 위치에 따른 원점복귀 작업

8.4.3 함수 요약

원점복귀와 관련된 함수들은 아래의 표와 같습니다. 그리고 속도패턴 설정은 cmxHomeSetSpeedPattern 함수를 사용합니다.

Summary of Functions
❒ VT_I4 cmxHomeSetConfig ([in] VT_I4 BoardID, [in] VT_I4 Channel, [in] VT_I4 ParamId, [in] VT_I4 ParamVal) 대상(對象) 모션 채널에 대해서, 원점복귀(原點復歸)에 대한 환경설정(環境設定)을 구성합니다.
❒ VT_I4 cmxHomeGetConfig ([in] VT_I4 BoardID, [in] VT_I4 Channel, [in] VT_PI4 ParamId, [out] VT_PI4 ParamVal) 대상(對象) 모션 채널에 대해서, 원점복귀(原點復歸)에 대해 설정되어 있는 환경 설정을 반환합니다.
❒ VT_I4 cmxHomeSetOffset ([in] VT_I4 BoardID, [in] VT_I4 Channel, [in] VT_R8 Offset) 원점복귀(原點復歸) 완료 후 이동할 거리를 지정합니다.
❒ VT_I4 cmxHomeGetConfig ([in] VT_I4 BoardID, [in] VT_I4 Channel, [out] VT_PR8 Offset) 원점복귀(原點復歸) 완료 후 이동할 거리를 반환합니다.
❒ VT_I4 cmxHomeSetPosClrMode ([in] VT_I4 BoardID, [in] VT_I4 Channel, [in] VT_I4 PosClrMode) 대상(對象) 모션 채널에 대해서, 원점복귀(原點復歸) 완료 후 발생하는 모션 컨트롤러와 서보 드라이브간의 제어 편차에 의해 발생한 입력 펄스(Feedback Pulse) 에 대한 처리에 대한 환경 설정을 구성합니다.
❒ VT_I4 cmxHomeGetPosClrMode ([in] VT_I4 BoardID, [in] VT_I4 Channel, [out] VT_PI4 PosClrMode) 대상(對象) 모션 채널에 대해서, 원점복귀(原點復歸) 완료 후 발생하는 모션 컨트롤러와 서보 드라이브간의 제어 편차에 의해 발생한 입력 펄스(Feedback Pulse) 에 대한 처리에 대하여, 설정되어 있는 환경 설정을 반환합니다.
❒ VT_I4 cmxHomeSetSpeedPattern ([in] VT_I4 BoardID, [in] VT_I4 Channel, [in] VT_I4 SpeedMode, [in] VT_R8 Vel, [in] VT_R8 Accel, [in] VT_R8 Decel, [in] VT_R8 RevVel) 대상(對象) 모션 채널에 대한, 원점복귀(原點復歸) 속도(速度)를 설정합니다. 이 속도는 원점 복귀 시에만 사용되는 속도(速度)이며, 일반 모션 속도와 개별 적인 원점 복귀 속도(速度)를 지정할 수 있습니다.
❒ VT_I4 cmxHomeGetSpeedPattern ([in] VT_I4 BoardID, [in] VT_I4 Channel, [out] VT_PI4 SpeedMode, [out] VT_PR8 Vel, [out] VT_PR8 Accel, [out] VT_PR8 Decel, [out] VT_PR8 RevVel) 대상(對象) 모션 채널에 대한, 원점복귀(原點復歸) 속도(速度)를 반환합니다. 이 속도는 원점 복귀 시에만 사용되는 속도(速度)이며, 일반 모션 속도와 개별 적인 원점 복귀 속도(速度)를 반환합니다.
❒ VT_I4 cmxHomeSetSpeedPattern_T ([in] VT_I4 BoardID, [in] VT_I4 Channel,[in] VT_I4 PhaseID, [in] VT_I4 SpeedMode, [in] VT_R8 Vel, [in] VT_R8 AccelTime, [in] VT_R8 DecelTime) 대상(對象) 모션 채널에 대한, 원점복귀(原點復歸) 속도(速度)를 설정합니다. 이 속도는 원점 복귀 시에만 사용되는 속도(速度)이며, 일반 모션 속도와 개별 적인 원점 복귀 속도(速度)를 지정할 수 있습니다.

❒ VT_I4 cmxHomeGetSpeedPattern_T ([in] VT_I4 BoardID, [in] VT_I4 Channel,[in] VT_I4 PhaseID, [out] VT_PI4 SpeedMode, [out] VT_PR8 Vel, [out] VT_PR8 AccelTime, [out] VT_PR8 DecelTime)
대상(對象) 모션 채널에 대한, 원점복귀(原點復歸) 속도(速度)를 반환합니다. 이 속도는 원점 복귀 시에만 사용되는 속도(速度)이며, 일반 모션 속도와 개별 적인 원점 복귀 속도(速度)를 반환합니다.

❒ VT_I4 cmxHomeMove ([in] VT_I4 BoardID, [in] VT_I4 Channel, [in] VT_I4 IsBlocking)
대상 단축(單軸) 모션 채널에 대한, 환경 설정을 바탕으로 원점복귀(原點復歸)를 구동합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)되지 않습니다.

❒ VT_I4 cmxHomeMoveStart ([in] VT_I4 BoardID, [in] VT_I4 Channel)
대상 단축(單軸) 모션 채널에 대한, 환경 설정을 바탕으로 원점복귀(原點復歸)를 구동합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)됩니다.

❒ VT_I4 cmxHomeMoveAll ([in] VT_I4 BoardID, [in] VT_I4 NumAxes, [in] VT_PI4 ChannelList, [in] VT_I4 IsBlocking)
대상 다축(多軸) 모션 채널에 대한, 환경 설정을 바탕으로 원점복귀(原點復歸)를 구동합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)되지 않습니다.

❒ VT_I4 cmxHomeMoveAllStart ([in] VT_I4 BoardID, [in] VT_I4 NumAxes, [in] VT_PI4 ChannelList)
대상 다축(多軸) 모션 채널에 대한, 환경 설정을 바탕으로 원점복귀(原點復歸)를 구동합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)됩니다.

❒ VT_I4 cmxHomeIsBusy ([in] VT_I4 BoardID, [in] VT_I4 Channel, [out] VT_PI4 IsBusy)
대상 단축 (單軸) 모션 채널에 대한, 원점복귀(原點復歸) 구동 상태를 반환합니다. 이 함수를 통해 원점 복귀가 진행 중인지를 판단할 수 있습니다.

❒ VT_I4 cmxHomeWaitDone ([in] VT_I4 BoardID, [in] VT_I4 Channel, [in] VT_I4 IsBlocking)
대상 단축 (單軸) 모션 채널에 대한, 원점복귀(原點復歸) 완료 시까지 대기합니다. 이 함수를 통해 원점 복귀 완료 시점까지 대기 할 수 있습니다.

VT_I4 cmxHomeGetSuccess ([in] VT_I4 BoardID, [in] VT_I4 Channel, [out] VT_PI4 IsSuccess)
대상 단축 (單軸) 모션 채널에 대해, 이전에 실행된
원점복귀구동완료(原點復歸驅動完了) 상태를 확인할 수 있습니다. 원점 복귀 완료 상태는 이전에 실행된 원점 복귀 상태이며, 모션 운영 시스템의 하드웨어적인 전원이 차단되지 않는 다는 조건 하에서는 영구히 보존됩니다.

VT_I4 cmxHomeSetSuccess ([in] VT_I4 BoardID, [in] VT_I4 Channel, [in] VT_I4 IsSuccess)
대상 단축 (單軸) 모션 채널에 대해 ,이전에 실행된 원점
복귀구동완료(原點復歸驅動完了) 상태를 설정할 수 있습니다. 원점 복귀 완료 상태는 이전에 실행된 원점 복귀 상태이며, 모션 운영 시스템의 하드웨어적인 전원이 차단되지 않는 다는 조건 하에서는 영구히 보존되며, 이 함수를 통해 이전의 원점 복귀 완료 상태를 변경 할 수 있습니다.

8.4.4 함수 설명

NAME

cmxHomeSetConfig

cmxHomeGetConfig

- 원점 복귀 환경 설정 (原點復歸環境設定)

INFORMATION

Home Return

VC++/VB

BCB/Delphi/.NET

Level 4

다소 주의

원점 복귀 함수의 전체 환경 설정을 하는 함수입니다. 매개변수 및 관련 내용을 반드시 숙지합니다.

SYNOPSIS

❒ VT_I4 cmxHomeSetConfig

([in] VT_I4 BoardID, [in] VT_I4 Channel, [in] VT_I4 ParamId, [in] VT_I4 ParamVal)

❒ VT_I4 cmxHomeGetConfig

([in] VT_I4 BoardID, [in] VT_I4 Channel, [in] VT_I4 ParamId, [out] VT_PI4 ParamVal)

DESCRIPTION

cmxHomeSetConfig() 함수는 원점복귀에 관련된 여러가지 환경을 설정합니다.
cmxHomeGetConfig() 함수는 원점복귀 작업에 관련된 환경 설정값을 확인(確認)합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Channel : 축(채널) 번호. 축번호는 상수값으로 0 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.

▶ ParamId : 설정하고자 하는 파라미터 ID 값을 지정합니다.해당 ID 값의 의미는 다음과 같습니다.

Value	Meaning
0 또는 cmxHPID_HOME_MODE	원점복귀를 수행 할 때 사용할 복귀 방법을 지정합니다.
1 또는 cmxHPID_HOME_DIR	원점복귀를 수행 할 방향을 지정합니다.
2 또는 cmxHPID_HOME_OFFSET	원점복귀 완료 후 추가로 이동할 거리를 지정합니다.

▶ ParamVal : cmxHomeSetConfig 함수의 인자이며, 파라미터 ID 값에 따른 실제 지정하고자 하는 파라미터의 적용 값을 의미합니다. 각값의 적용범위는 파라미터 ID 별로 다르며 해당 값의 의미는 다음 표와 같습니다.

ParamId 가 0 인 경우	
Value	**Meaning**
1 또는 cmxHOME_MODE_ORG	원점센서를 사용한 원점복귀를 시행합니다.
3 또는 cmxHOME_MODE_EL	EL 센서를 사용한 원점복귀를 시행합니다.

ParamId 가 1 인 경우	
Value	**Meaning**
0 또는 cmxDIR_N	음의 방향으로 원점복귀를 시행합니다.
1 또는 cmxDIR_P	양의 방향으로 원점복귀를 시행합니다.

ParamId 가 2 인 경우
Meaning
원점 복귀 위치에서 일정거리 이상을 상대 이동할 필요가 있을 경우, 그 값을 설정합니다. 이것은 원점 복귀 종료 위치를 기준으로 추가 모션 이동을 의미합니다.

▶ ParamVal : cmxHomeGetConfig 함수의 인자이며, 파라미터 ID 값에 따른 실제 지정된 파라미터의 적용 값을 반환합니다.

RETURN VALUE

Value	**Meaning**
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxHomeSetOffset

cmxHomeGetOffset

- 원점 복귀 후 추가 이동거리 지정

INFORMATION

- Home Return
- VC++/VB
- BCB/Delphi/.NET
- Level 2
- 다소 주의

SYNOPSIS

❒ VT_I4 cmxHomeSetOffset
([in] VT_I4 BoardID, [in] VT_I4 Channel, [in] VT_R8 Offset)

❒ VT_I4 cmxHomeGetOffset
([in] VT_I4 BoardID, [in] VT_I4 Channel, [out] VT_PR8 Offset)

DESCRIPTION

cmxHomeSetOffset() 함수는 원점복귀 완료 후 이동 할 거리를 지정합니다.
cmxHomeGetOffset() 함수는 원점복귀 완료 후 이동 할 거리를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Channel : 축(채널) 번호. 축번호는 상수값으로 0 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.

▶ Offset : cmxHomeSetConfig 함수의 인자이며, 원점 복귀 위치에서 일정거리 이상을 상대 이동할 필요가 있을 경우, 그 값을 설정합니다. 이것은 원점 복귀 종료 위치를 기준으로 추가 모션 이동을 의미합니다.

▶ Offset : cmxHomeGetConfig 함수의 인자이며, Offset 으로 설정된 상대 거리 이동값을 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxHomeSetPosClrMode

cmxHomeGetPosClrMode

- 원점 복귀(原點 復歸) 완료(完了) 후 위치(位置) 소거(消去) 모드 설정

INFORMATION

Home Return

VC++/VB

BCB/Delphi/.NET

Level 4

다소 주의

원점 복귀 후 발생할 수 있는 일정량의 위치 편차는 정밀한 모션 제어에서 매우 중요한 부분입니다.

SYNOPSIS

❒ VT_I4 cmxHomeSetPosClrMode ([in] VT_I4 BoardID, [in] VT_I4 Channel, [in] VT_I4 PosClrMode)

❒ VT_I4 cmxHomeGetPosClrMode ([in] VT_I4 BoardID, [in] VT_I4 Channel, [out] VT_PI4 PosClrMode)

DESCRIPTION

cmxHomeSetPosClrMode() 함수는 원점복귀가 완료된 후에 Command 및 Feedback 위치에 대한 처리 모드를 설정하는 함수입니다. cmxHomeGetConfig() 함수는 원점복귀가 완료된 후에 Command 및 Feedback 위치를 소거하는 모드의 설정을 읽어들이는 함수입니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Channel : 축(채널) 번호. 축번호는 상수값으로 0 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.

▶ PosClrMode : cmxHomeSetPosClrMode 함수의 인자이며, 원점복귀가 완료된 후에 Command 및 Feedback 위치가 클리어되는 모드를 결정하는 매개 변수(媒介變數)입니다. PosClrMode 는 다음과 같이 3 가지를 설정할 수 있습니다.

Value	Meaning
0	최종 완료 조건에 해당하는 외부 하드웨어 신호가 입력되는 순간에 Command 및 Feedback 의 위치가 0 으로 소거됩니다. 일정량의 Feedback 위치 편차를 보입니다.
1	원점복귀 모드에 상관없이 하드웨어 신호의 입력 상태와 더불어 소프트웨어적인 모션 완료 확인 동작을 포함한, 전체적인 원점복귀 작업이 모두 완료된 후에 Command 와 Feedback 위치가 모두 자동으로 0 으로 소거됩니다. 일정량의 Feedback 위치 편차를 보입니다.

2	1 차적으로는 cmHPCM_M0 일 때와 동일하게 동작합니다. 단, 원점복귀가 완료된 후에 Feedback 위치와 동일한 값으로 Command 위치를 셋팅하므로써 Command 와 Feedback 을 일치하도록 동작 합니다. 이를 통해 서보드라이브에서 실제 이송한 위치에 대한 양을 Command 위치에 반영시켜서, 다음 모션 동작을 수행할 수 있도록 합니다.

▶ PosClrMode : cmxHomeGetPosClrMode 함수의 인자이며, 원점복귀가 완료된 후에 Command 및 Feedback 위치가 클리어되는 모드를 반환합니다. PosClrMode 는 다음과 같이 3 가지 설정값을 갖습니다.

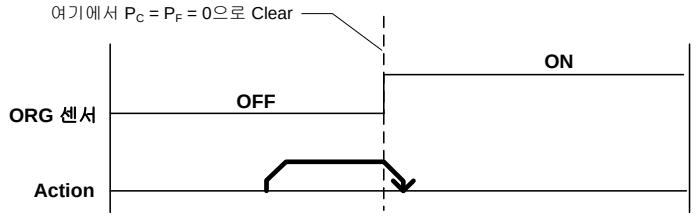
Value	Meaning
0	최종 완료 조건에 해당하는 외부 하드웨어 신호가 입력되는 순간에 Command 및 Feedback 의 위치가 0 으로 소거되는 모드입니다.
1	원점복귀 모드에 상관없이 하드웨어 신호의 입력 상태와 더불어 소프트웨어적인 모션 완료 확인 동작을 포함한, 전체적인 원점복귀 작업이 모두 완료된 후에 Command 와 Feedback 위치가 모두 자동으로 0 으로 소거되는 모드입니다.
2	1 차적으로는 cmHPCM_M0 일 때와 동일하게 동작합니다. 단, 원점복귀가 완료된 후에 Feedback 위치와 동일한 값으로 Command 위치를 셋팅하므로써 Command 와 Feedback 을 일치하도록 동작시키는 모드입니다.

RETURN VALUE

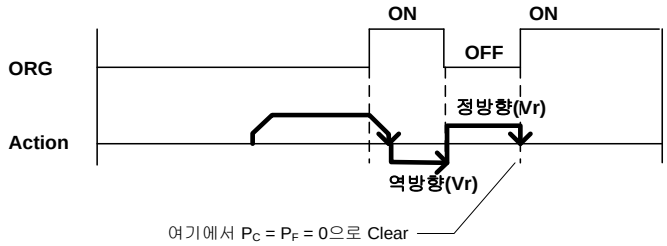
Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

SEE ALSO

□ PosClrMode 가 0 또는 2 로 하면 최종 단계에서 감속 없이 정지(停止)하는 원점복귀 모드(1, 2, 3, 4)에서는 Command 위치가 0 인 상태에서 원점복귀가 완료되고, 나머지 모드에서는 감속을 시작 할때 위치가 0 으로 클리어되며, 감속 하는 동안에 이송한 양만큼 위치가 증가 또는 감소하게 됩니다. 아래의 두 그림은 최종 단계에서 감속을 하는 원점복귀 모드 0 번과 즉시 정지(停止)를 하는 모드 1 번에서의 동작을 예로 든 것입니다.



[원점복귀 모드 0에서의 동작]



[원점복귀 모드 1에서의 동작]

□ 서보모터를 사용하는 경우에는 서보드라이버의 제어 지연 시간 때문에 원점복귀완료 후 모터의 실제 위치는 약간씩 편차가 발생할 수 있습니다. 이는 원점복귀의 오차를 유발하게 됩니다. 그런데 cmHPCM_M0 또는 cmHPCM_M2 인 경우에는 모션제어기의 Feedback 카운터에 해당 편차가 그대로 반영됩니다. 따라서 Command 위치를 Feedback 위치와 일치시킨 후에 절대좌표 이송을 수행하면 이러한 편차에 의한 제어 오차를 제거할 수 있습니다. 이의 원리를 적용한 것이 cmHPCM_M2 입니다.

□ 스텝모터를 사용하는 경우에는 cmHPCM_M2 를 사용하면 안됩니다.

안내

서보모터를 사용하는 경우에는 cmHPCM_M2 로 하였을 때 가장 정확한 원점복귀 작업 결과를 얻을 수 있습니다. 단, 이때 다음과 같은 사항에 주의하여야 합니다.

- Command 방향과 Feedback 방향이 반드시 일치하여야 합니다.
- Command 분해능과 Feedback 분해능이 반드시 일치하여야 합니다.
- 원점복귀가 완료된 후에 Command 와 Feedback 좌표는 일치하지만 0 이 되지는 않습니다. 따라서 경우에 따라서는 원점복귀 완료후에 절대좌표 0 으로 이송하는 명령이 필요할 수도 있습니다.

NAME

cmxHomeSetSpeedPattern

cmxHomeGetSpeedPattern

- 원점 복귀속도설정 (原點復歸速度設定)

INFORMATION

Home Return

VC++/VB

BCB/Delphi/.NET

Level 4

다소 주의

속도설정은 논리적인 속도 단위가 적용됩니다. 기본적으로는 PPS(Pulse per second) 단위를 적용하므로, 비율로 설정되는 단위가 아닙니다.

SYNOPSIS

❒ VT_I4 cmxHomeSetSpeedPattern

([in] VT_I4 BoardID, [in] VT_I4 Channel,[in] VT_I4 PhaseID, [in] VT_I4 SpeedMode, [in] VT_R8 Vel, [in] VT_R8 Accel, [in] VT_R8 Decel)

❒ VT_I4 cmxHomeGetSpeedPattern

([in] VT_I4 BoardID, [in] VT_I4 Channel, [in] VT_I4 PhaseID, [out] VT_PI4 SpeedMode, [out] VT_PR8 Vel, [out] VT_PR8 Accel, [out] VT_PR8 Decel)

DESCRIPTION

cmxHomeSetSpeedPattern() 함수는 설정된 축의 원점복귀 속도 단계와 속도 모드, 가감속도 및 작업속도 등을 설정합니다.
cmxHomeGetSpeedPattern() 함수는 설정된 축의 원점복귀 속도 단계와 속도 모드, 가감속도 및 작업속도 등을 읽어옵니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Channel : 축(채널) 번호. 축번호는 상수값으로 0 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.

▶ PhaseID : 원점복귀의 속도단계을 설정한다.

▶ SpeedMode : cmxHomeSetSpeedPattern 함수의 인자이며, 원점복귀시의 S-Curve 형 또는 선형 가감속, 가감속이 없는 모드등을 선택할 수 있습니다.

Value	Meaning
0 또는 cmxSPEED_CONSTANT	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 또는 cmxSPEED_TRAPEZOIDAL	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 또는 cmxSPEED_SCURVE	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.

▶ SpeedMode : cmxHomeGetSpeedPattern 함수의 인자이며, 원점복귀모드를 반환합니다.

Value	Meaning
0 또는 cmxSPEED_CONSTANT	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 또는 cmxSPEED_TRAPEZOIDAL	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 또는 cmxSPEED_SCURVE	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.

▶ Vel : cmxHomeSetSpeedPattern 함수의 인자이며, 작업 속도를 의미합니다. 기호로는 Vwork 로 표기합니다.

▶ Vel : cmxHomeGetSpeedPattern 함수의 인자이며, 작업 속도를 반환합니다.

▶ Accel : cmxHomeSetSpeedPattern 함수의 인자이며, 홈복귀시의 가속도를 의미합니다.

▶ Accel : cmxHomeGetSpeedPattern 함수의 인자이며, 홈복귀시의 가속도를 반환합니다.

▶ Decel : cmxHomeSetSpeedPattern 함수의 인자이며, 홈복귀시의 감속도를 의미합니다.

▶ Decel : cmxHomeGetSpeedPattern 함수의 인자이며, 홈복귀시의 감속도를 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxHomeSetSpeedPattern _T

cmxHomeGetSpeedPattern_T

- 원점 복귀속도설정 (原點復歸速度設定)

INFORMATION

Home Return

VC++/VB

BCB/Delphi/.NET

Level 4

다소 주의

속도설정은 논리적인 속도 단위가 적용됩니다. 기본적으로는 PPS(Pulse per second) 단위를 적용하므로, 비율로 설정되는 단위가 아닙니다.

SYNOPSIS

❒ VT_I4 cmxHomeSetSpeedPattern

([in] VT_I4 BoardID, [in] VT_I4 Channel, [in] VT_I4 PhaseID, [in] VT_I4 SpeedMode, [in] VT_R8 Vel, [in] VT_R8 AccelTime, [in] VT_R8 DecelTime)

❒ VT_I4 cmxHomeGetSpeedPattern

([in] VT_I4 BoardID, [in] VT_I4 Channel, [in] VT_I4 PhaseID, [out] VT_PI4 SpeedMode, [out] VT_PR8 Vel, [out] VT_PR8 AccelTime, [out] VT_PR8 DecelTime)

DESCRIPTION

cmxHomeSetSpeedPattern_T() 함수는 설정된 축의 원점복귀 속도 단계와 속도 모드, 가감속도 및 작업속도 등을 설정합니다.
cmxHomeGetSpeedPattern_T() 함수는 설정된 축의 원점복귀 속도 단계와 속도 모드, 가감속도 및 작업속도 등을 읽어옵니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Channel : 축(채널) 번호. 축번호는 상수값으로 0 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.

▶ PhaseID : 원점복귀의 속도단계을 설정한다.

▶ SpeedMode : cmxHomeSetSpeedPattern_T 함수의 인자이며, 원점복귀시의 S-Curve 형 또는 선형 가감속, 가감속이 없는 모드등을 선택할 수 있습니다.

Value	Meaning

0 또는 cmxSPEED_CONSTANT	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 또는 cmxSPEED_TRAPEZOIDAL	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 또는 cmxSPEED_SCURVE	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.

▶ SpeedMode : cmxHomeGetSpeedPattern_T 함수의 인자이며, 원점복귀모드를 반환합니다.

Value	Meaning
0 또는 cmxSPEED_CONSTANT	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 또는 cmxSPEED_TRAPEZOIDAL	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 또는 cmxSPEED_SCURVE	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.

▶ Vel : cmxHomeSetSpeedPattern_T 함수의 인자이며, 작업 속도를 의미합니다. 기호로는 Vwork 로 표기합니다.

▶ Vel : cmxHomeGetSpeedPattern_T 함수의 인자이며, 작업 속도를 반환합니다.

▶ AccelTime : cmxHomeSetSpeedPattern_T 함수의 인자이며, 홈복귀시의 가속 시간을 의미합니다.

▶ AccelTime : cmxHomeGetSpeedPattern_T 함수의 인자이며, 홈복귀시의 가속 시간을 반환합니다.

▶ DecelTime : cmxHomeSetSpeedPattern 함수의 인자이며, 홈복귀시의 감속 시간을 의미합니다.

▶ DecelTime : cmxHomeGetSpeedPattern 함수의 인자이며, 홈복귀시의 감속 시간을 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxHomeMove

cmxHomeMoveStart

- 단축 원점 복귀 이송 (單軸原點復歸移送)

INFORMATION

Home Return

VC++/VB

BCB/Delphi/.NET

Level 4

다소 위험

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인(確認)합니다.

SYNOPSIS

- ❒ VT_I4 cmxHomeMove ([in] VT_I4 BoardID, [in] VT_I4 Channel, [in] VT_I4 IsBlocking)
- ❒ VT_I4 cmxHomeMoveStart ([in] VT_I4 BoardID, [in] VT_I4 Channel)

DESCRIPTION

원점복귀 작업을 수행합니다. cmxHomeMove() 함수는 모션이 완료되기 전까지 반환되지 않으며, cmxHomeMoveStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Channel : 축(채널) 번호. 축번호는 상수값으로 0 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.

▶ IsBlocking : 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다.

Value	Meaning
0	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
1	블록(Blocking)을 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
ERR_NONE	수행 성공

SEE ALSO

☐ cmxHomeMoveStart() 함수를 사용하는 경우에는 cmxSxIsDone(), cmxSxWaitDone() 또는 cmxMxIsDone(), cmxMxWaitDone() 함수를 사용하여 모션의 완료를 확인(確認)할 수 있습니다. 그러나 가장 바람직한 방법은 cmxHomeWaitDone() 함수를 사용하는 것이 좋습니다.

☐ cmxHomeMove() 함수를 사용하는 경우에는 내부적으로 루프를 수행하면서 모션이 완료되기를 기다리는데, 이때 "Blocking Mode"설정에 따라 윈도우 이벤트를 처리하는 방식이 달라집니다.

윈도우 이벤트라는 것은 무엇입니까?
윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

☐ INP 입력신호가 Enable 로 설정되었으면 Command 펄스 출력이 완료되어도 INP 입력이 ON 이 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환되지 않습니다.

☐ 스탭 드라이브를 사용 중인 고객(顧客)님들께서는 다음을 참조해 주십시요.
스탭 드라이브는 INP 출력이 없는 경우가 일반적인데, 고객(顧客)님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스탭 드라이브를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다.
고객(顧客) 여러분들께서는 스탭 드라이브 사용시에 이점을 주의해주시기를 부탁드립니다.

☐ 서보 드라이브의 LSP , LSN 신호를 사용 중인 고객(顧客)님께서는 다음을 참조해 주십시요.
서보드라이브의 입력 신호 중 하나인 EL(End of Limit) 신호는 저희 ㈜커미조아 모션 보드 뿐만 아니라 서보드라이브에도 전달 될 수 있도록 설정할 수 있습니다. 통상적으로 LSP 신호와 LSN 신호로 불리어 지는 이 신호는 실제 기구물에서 양의 방향(Positive Direction) 혹은 음의 방향(Negative Direction) 에 장착되어 있는 EL(End of Limit) 신호를 서보 드라이브 측에 전달하기 위한 용도로 사용됩니다.

그러나, 모션 소프트웨어에서 INP 설정이 되어 있는 경우 EL 신호가 검출 된 후에 일부 서보 드라이브에서는 진행 방향에서 정지(停止) 한후 더 이상 움직이지 않는 상황이 발생하며, 이 상황에서 INP 신호가 출력되지 않아, 모션 이송이 완료되지 못하고, 명시적으로 STOP 을 해줘야만 하는 강제적으로 모션 종료가 되는 현상이 발생할 수 있습니다. 이 현상은 EL 모드를 통해 원점 복귀를 하는 상황에서도 발생할 수 있습니다.

따라서, 이러한 경우에는 반드시 원점 복귀나 EL 검출시에 인터럽트 이벤트나 타이머를 통해 INP 를 무기한 대기하는 현상에 대해서 적절히 대처하시거나 INP 신호 사용을 배제 해야 합니다.

EXAMPLE

본 예제는 cmxHomeMoveStart() 함수를 이용하여 X1, Y1 축의 원점복귀를 수행하는 함수입니다. 원점복귀에 대한 환경설정은 이미 이루어진 것으로 가정합니다.

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"
```

```
// BoardID 는 0 으로 선언되었다고 가정함

/************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.
************************************************************/
void OnProgramInitial()
{

	long m_nNumDevices;
	long m_DeviceList[16];
	long m_nNumAxes;
	cmxLoadDll();
	if(cmxGnLoadDevice (&m_nNumDevices, m_DeviceList, &m_nNumAxes)!= ERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		// 에러메시지 출력
		return;
	}
}

/************************************************************
* OnHomeSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
* 호출되는 가상의 함수 입니다. 이때 m_fVwork, m_fAcc, m_fDec 변수를
* 통하여 속도, 가속도 , 감속도 값이 적절하게 전달된다고 가정합니다.
************************************************************/
void OnHomeSetSpeed()
{
	//각 축(Axis)의 기본 속도를 설정 합니다.
	//X1 축의 홈복귀 모드를 설정합니다.
	cmxHomeSetConfig(BoardID, 3, 0, 1, 100, 10);
	//Y1 축의 홈복귀 모드를 설정합니다.
	cmxHomeSetConfig(BoardID, cmY1, 0, 1, 100, 10);
	// X1 축 홈복귀 속도패턴 설정 //
	cmxHomeSetSpeedPattern(BoardID, 3, cmxMODE_S, m_fVwork, m_fAcc, m_fDec,
m_fRevVel );
	// Y1 축 홈복귀 속도패턴 설정 //
	cmxHomeSetSpeedPattern(BoardID, cmY1, cmxMODE_S, m_fVwork, m_fAcc, m_fDec,
m_fRevVel );
}

/************************************************************
* OnHomeReturn : 이 함수는 가상의 함수로서  원점복귀를 실행합니다.
************************************************************/
void OnHomeRetrun()
{
	// X1 축 원점복귀 시작 //
	if(cmxHomeMoveStart(BoardID, 3, cmDIR_N) != ERR_NONE){
		// 에러메시지 출력
		return;
	}
	// Y1 축 원점복귀 시작 //
	if(cmxHomeMoveStart(BoardID, cmY1, cmDIR_N) != ERR_NONE){
		// 에러메시지 출력
		return;
	}
```

```
	// X1&Y1 두축의 원점복귀 작업이 완료될 때까지 기다린다. //
	if(cmxHomeWaitDone(BoardID, 3, cmxFALSE) != ERR_NONE) // 에러메시지 출력
	if(cmxHomeWaitDone(BoardID, cmY1, cmxFALSE) != ERR_NONE) // 에러메시지 출력

	////////////////////////////////////////////////////////
	//위의 cmxSxWaitDone() 함수 대신에 아래와 같이 cmxMxWaitDone()
	//함수를 사용하여 두축의 작업완료를 기다리는 것을 한번에 수행할 수 있다.
	// int nAxes[2] = {0, cmY1};
	// cmxMxWaitDone(BoardID, 2, nAxes, cmxFALSE);

	// 원점복귀의 성공 여부를 확인(確認)하여 처리한다. //
	long dwIsSuccess;
	cmxHomeGetSuccess(BoardID, nAxis, &dwIsSuccess);

	if(dwIsSuccess){
		MessageBox(NULL, "원점복귀를 성공적으로 수행하였습니다.", "Message",
MB_OK);
}else{

	long dwErrCode;
	char szErrMsg[CMX_MAX_STR_LEN_ERR];
	char szErrReason[CMX_MAX_STR_LEN_ERR];

	cmxErrGetLastCode(BoardID, nAxis, &dwErrCode);
	cmxErrGetString(BoardID, dwErrCode, szErrReason, CMX_MAX_STR_LEN_ERR);

	sprintf(szErrMsg, "다음과 같은 이유로 원점복귀에 실패하였습니다.\n%s", szErrReason);
	MessageBox(NULL, szErrMsg, "Motion Error", MB_OK | MB_ICONERROR);

	}

}
```

```
Visual Basic

'BoardID 는 0 으로 선언되었다고 가정함

'============================================================
'GnLoadDevice 함수로 장치를 초기화 합니다.
'============================================================
Private Sub Form_Load()

   Dim nTotalDevices As Long
   Dim DeviceList(16) As Long
   Dim nTotalAxis As Long
   Dim IRetVal As Long

'============================================================
 ' GnLoadDevice 함수로 장치를 초기화합니다.
'============================================================
   IRetVal = GnLoadDevice(nTotalDevices, DeviceList(0), nTotalAxis)

  If IRetVal <> ERR_NONE Then
     MsgBox ("GnLoadDevice has been failed")
  End If
End Sub
```

```
' 버튼 이벤트에 의해서 홈복귀를 시작합니다.
Private Sub btnHome_Click()
   ' HomeSetConfig(BoardID, 대상 축, 홈 복귀 모드, EZCount, EscDist, Offset )
   Call HomeSetConfig(BoardID, 3, 0, 1, 1000, 0)

   ' HomeMove(BoardID, 대상 축, 홈 복귀 방향, 블럭 여부)
   Call HomeMove(BoardID, 3, GetDirection, cmDIR_N, cmxFALSE )
End Sub

' 홈 복귀 동작시 정지(停止)가 필요할 경우 동작합니다.
Private Sub BtnStop_Click()
   Dim IsWaitComplete As Long
   Dim nResult As Long
   IsWaitComplete = True

   ' 정지(停止) 버튼을 누르게 되면 아래와 같이 모션 정지(停止) 함수가 호출되게 됩니다.
   ' 여기서 IsWaitComplete 는 모션 정지(停止)가 완료된 후 반환 할 것인지,
   ' 모션 정지(停止) 명령 수행 후 바로 반환 할 것인지를 결정합니다.

   nResult = SxStop(BoardID, 3, IsWaitComplete, cmxFALSE)
End Sub

Private Sub CfgSpeed(nTotalAxis As Long)


      '=========================================================
       ' 이 함수에서 cmxCfgSetSpeedPattern 함수로 속도를 설정하는 것은
       ' 모든 모션의 기준속도(Standard Spee) 가 됩니다.
       ' 단축 구동을 비롯한 대부분의 모션 동작은 이 기준 속도의 비율로
       ' 동작되게 됩니다.
      ' 아래 함수는 전체 축에 대해서 임의의 기준 속도를 설정하고 있습니다.
      '=========================================================

      Call CfgSetSpeedPattern(BoardID, 3, cmxSMODE_S, 1000, 2000, 2000)

End Sub
```

NAME

cmxHomeMoveAll

cmxHomeMoveAllStart

- 다축 원점 복귀 이송 (多軸原點復歸移送)

INFORMATION

Home Return

VC++/VB

BCB/Delphi/.NET

Level 4

다소 위험

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인(確認)합니다.

다축의 원점 복귀 확인(確認)시에는 cmxMxWaitDone 함수를 사용할 수 있습니다.

SYNOPSIS

❒ VT_I4 cmxHomeMoveAll

([in] VT_I4 BoardID, [in] VT_I4 NumAxes, [in] VT_PI4 ChannelList, [in] VT_I4 IsBlocking)

❒ VT_I4 cmxHomeMoveAllStart

([in] VT_I4 BoardID, [in] VT_I4 NumAxes, [in] VT_PI4 ChannelList)

DESCRIPTION

여러 축에 대한 원점복귀 작업을 동시에 수행합니다. cmxHomeMoveAll() 함수는 모션이 완료되기 전까지 반환되지 않으며, cmxHomeMoveAllStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ NumAxes : 동시에 작업을 수행할 대상 축의 수

▶ ChannelList : 동시에 작업을 수행할 대상 축의 배열 주소값. 이 배열의 크기는 NumAxes 값과 일치하거나 커야 합니다.

▶ IsBlocking : 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다.

Value	Meaning
0	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.

1	블록(Blocking)을 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

SEE ALSO

☐ cmxHomeMoveAllStart() 함수를 사용하는 경우에는 cmxMxIsDone() 함수나 MxWaitDone() 함수를 사용하여 모션의 완료를 확인(確認)할 수 있습니다.

☐ cmxHomeMoveAll() 함수를 사용하는 경우에는 내부적으로 루프를 수행하면서 모션이 완료되기를 기다리는데, 이때 "Blocking Mode"의 전달 인자값에 따라 윈도우 이벤트를 처리하는 방식이 달라집니다.

윈도우 이벤트라는 것은 무엇입니까?

윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

☐ INP 입력신호가 Enable 로 설정되었으면 Command 펄스 출력이 완료되어도 INP 입력이 ON 이 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환되지 않습니다.

☐ 스탭 드라이브를 사용 중인 고객(顧客)님들께서는 다음을 참조해 주십시요.
스탭 드라이브는 INP 출력이 없는 경우가 일반적인데, 고객(顧客)님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스탭 드라이브를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다.
고객(顧客) 여러분들께서는 스탭 드라이브 사용시에 이점을 주의해주시기를 부탁드립니다.

☐ 서보 드라이브의 LSP , LSN 신호를 사용 중인 고객(顧客)님께서는 다음을 참조해 주십시요.
서보드라이브의 입력 신호 중 하나인 EL(End of Limit) 신호는 저희 ㈜커미조아 모션 보드 뿐만 아니라 서보드라이브에도 전달 될 수 있도록 설정할 수 있습니다. 통상적으로 LSP 신호와 LSN 신호로 불리어 지는 이 신호는 실제 기구물에서 양의 방향(Positive Direction) 혹은 음의 방향(Negative Direction) 에 장착되어 있는 EL(End of Limit) 신호를 서보 드라이브 측에 전달하기 위한 용도로 사용됩니다.

그러나, 모션 소프트웨어에서 INP 설정이 되어 있는 경우 EL 신호가 검출 된 후에 일부 서보 드라이브에서는 진행 방향에서 정지(停止) 한후 더 이상 움직이지 않는 상황이 발생하며, 이 상황에서 INP 신호가 출력되지 않아, 모션 이송이 완료되지 못하고, 명시적으로 STOP 을 해줘야만 하는 강제적으로 모션 종료가 되는 현상이 발생할 수 있습니다. 이 현상은 EL 모드를 통해 원점 복귀를 하는 상황에서도 발생할 수 있습니다.

따라서, 이러한 경우에는 반드시 원점 복귀나 EL 검출시에 인터럽트 이벤트나 타이머를 통해 INP 를 무기한 대기하는 현상에 대해서 적절히 대처하시거나 INP 신호 사용을 배제 해야 합니다.

EXAMPLE

본 예제는 cmxHomeMoveAll() 함수를 이용하여 X1, Y1 축의 원점복귀를 동시에 수행하는 함수입니다.

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

// BoardID 는 0 으로 선언되었다고 가정함

/************************************************************
* OnHomeSetSpeed :  이 함수는 속도설정의 변경이 필요할 때
* 호출되는 가상의 함수 입니다. 이때 m_fVwork, m_fAcc, m_fDec 변수를
* 통하여 속도, 가속도 , 감속도 값이 적절하게 전달된다고 가정합니다.
************************************************************/
void OnHomeSetSpeed()
{
	//각 축(Axis)의 기본 속도를 설정 합니다.
	//X1 축의 홈복귀 모드를 설정합니다.
	cmxHomeSetConfig(BoardID, 3, 0, 1, 100, 10);
	//Y1 축의 홈복귀 모드를 설정합니다.
	cmxHomeSetConfig(BoardID, cmY1, 0, 1, 100, 10);
	// X1 축 홈복귀 속도패턴 설정 //
	cmxHomeSetSpeedPattern(BoardID, 3, cmxMODE_S, m_fVwork, m_fAcc, m_fDec,
m_fRevVel );
	// Y1 축 홈복귀 속도패턴 설정 //
	cmxHomeSetSpeedPattern(BoardID, cmY1, cmxMODE_S, m_fVwork, m_fAcc, m_fDec,
m_fRevVel );
}

/************************************************************
* OnHomeReturn : 이 함수는 가상의 함수로서  원점복귀를 실행합니다.
************************************************************/
void OnHomeReturn()
{
	long nAxes[2] = {0, cmY1};
	long nDirList[2] = {cmDIR_N, cmDIR_N};

	if(cmxHomeMoveAll(BoardID, 2, nAxes, nDirList, cmxFALSE) != ERR_NONE){
		// 에러메시지 출력
		return ;
	}
	////////////////////////////////////////////////////////////
	// cmxHomeMoveAll() 함수 대신 아래와 같이 cmxHomeMoveAllStart() 함
	// 수를 사용할 수 있습니다.
	// cmxHomeMoveAllStart(BoardID, 2, nAxes, nDirList, cmxFALSE);
	// cmxMxWaitDone(BoardID, 2, nAxes, cmxFALSE);
}
```

Visual Basic

```
' 홈복귀를 위한 가상 함수를 시작합니다.
'BoardID 는 0 으로 선언되었다고 가정함
```

```
Private Sub OnStart()

    Dim NumChannel(2) As Long
    Dim DirList(2) As Long

    NumChannel(0) = 0
    NumChannel(1) = cmY1

    DirList(0) = cmDIR_N
    DirList(1) = cmDIR_N


    ' HomeSetConfig(BoardID,  대상 축, 홈 복귀 모드, EZCount, EscDist, Offset )
    Call HomeSetConfig(BoardID, NumChannel(0), 0, 0, 1000, 0)
    Call HomeSetConfig(BoardID, NumChannel(1), 0, 0, 1000, 0)

    ' HomeSetSpeedPattern 함수를 통해 원점 복귀 속도를 결정합니다.
    Call HomeSetSpeedPattern(BoardID, NumChannel(0), cmxSMODE _S, 10000, 20000, 20000, 1000)
    Call HomeSetSpeedPattern(BoardID, NumChannel(1), cmxSMODE _S, 10000, 20000, 20000, 1000)

    ' HomeMoveAll(BoardID, 대상 축, 홈 복귀 방향, 블록킹 여부)
    Call HomeMoveAll(BoardID, 2,NumChannel(0),DirList(0), cmxFALSE)

End Sub
```

```
Delphi

// BoardID 는 0 으로 선언되었다고 가정함

procedure btnHomeMoveClick();
var
    arAxes : Array[0..1] of LongInt;
    arDirecton : Array[0..1] of LongInt;
begin

    // cmxHomeSetConfig(BoardID,  대상 축, 홈 복귀 모드, EZCount, EscDist, Offset );

    // X1 축 에 대한 홈 설정을 합니다.
    cmxHomeSetConfig(BoardID, 3, 0, 0, 1000, 0 );

    // Y1 축에 대한 홈 설정을 합니다.
    cmxHomeSetConfig(BoardID, cmY1, 0, 0, 1000, 0 );

    // X1 축의 홈 복귀 속도를 설정합니다.
    cmxHomeSetSpeedPattern(
        BoardID,
        0,
        cmxMODE_S,
        10000,
        20000,
        20000,
        1000);

    // Y1 축의 홈 복귀 속도를 설정합니다.
    cmxHomeSetSpeedPattern(BoardID,
        cmY1,
```

```
      cmxMODE_S,
      10000,
      20000,
      20000,
      1000);

   // 다축의 홈 이송을 시작합니다. 각 축의 홈 복귀 방향은 Negative 로 설정합니다
   arAxes[0] := 0;            // X1 축
   arAxes[1] := cmY1;         // Y1 축
   arDirecton[0] := cmDIR_N;
   arDirecton[1] := cmDIR_N;

   // 인자는 다음과 같습니다.
   // cmxHomeMoveAllStart(BoardID, 홈복귀 대상축, 축의 배열, 방향의 배열)
   // 이 명령은 홈 복귀 명령 실행시 바로 리턴됩니다.
   cmxHomeMoveAllStart(BoardID, 2, @arAxes, @arDirecton);

   // 두 축에 대해서 홈 복귀 완료시까지 대기합니다.
   cmxHomeWaitDone(BoardID, 3, cmxFALSE);
   cmxHomeWaitDone(BoardID, cmY1, cmxFALSE);
end;

end.
```

NAME

cmxHomeIsBusy

- 원점 복귀(原點 復歸) 모션 진행 상태 확인(確認)

INFORMATION

- Home Return
- VC++/VB
- BCB/Delphi/.NET
- Level 4
- ☺ 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxHomeIsBusy ([in] VT_I4 BoardID, [in] VT_I4 Channel, [out] VT_PI4 IsBusy)

DESCRIPTION

지정한 축이 현재 원점복귀를 진행중인지를 IsBusy 버퍼를 통하여 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Channel : 축(채널) 번호. 축번호는 상수값으로 3 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.

▶ IsBusy : 현재 원점복귀가 진행중인지를 알려주는 값을 반환받을 버퍼. 이 값에 반환되는 값의 의미는 다음과 같습니다.

Value	Meaning
0 (cmxFALSE)	지정한 축은 현재 원점복귀가 진행중이지 않습니다.
1 (cmxTRUE)	지정한 축은 현재 원점복귀를 진행하고 있습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

SEE ALSO

□ ComiRTEX 라이브러리에서는 cmxSxIsDone()과 같이 일반적으로 모션이 진행중이냐 아니면 정지(停止)해 있느냐를 확인(確認)할 때, 진행중(Busy)을 확인(確認)하기 보다는 완료(Done)를 확인(確認)하는 방식을 채택합니다. 그러나 원점복귀에서는 cmxHomeGetSuccess() 함수와 혼동될 소지가 있어서 cmxHomeIsDone() 함수 대신에 cmxHomeIsBusy() 함수를 제공하게 되었습니다.

□ cmxHomeIsBusy() 함수가 IsBusy 버퍼에 FALSE 값이 반환되면 원점복귀가 완료되었음을 의미하지만 성공 여부는 알 수가 없습니다. 예를 들어 원점복귀 진행 중에 Limit 또는 Alarm 등과 같은 에러에 의해서 정지(停止)되었거나, Stop 함수에 의해서 강제로 정지(停止)되었을 때도 IsBusy 에는 FALSE 값이 반환됩니다. 따라서 cmxHomeIsBusy()함수를 이용하여 원점복귀의 원점복귀가 완료되었음을 확인(確認)한 후에는 cmxHomeGetSuccess() 함수를 사용하여 원점복귀의 성공여부를 확인(確認)하여 각각의 상황에 대한 처리를 해주는 것이 바람직합니다.

EXAMPLE

C/C++

```
// BoardID 는 0 으로 선언되었다고 가정함

BOOL OnHomeMove(int nAxis)
{
	long dwIsHomming = TRUE;
	cmxHomeMoveStart(BoardID, nAxis, cmDIR_N);

	while(dwIsHomming){

			cmxHomeIsBusy(BoardID, nAxis, &dwIsHomming);

			// 원점복귀 진행여부 읽기
			// 윈도우 메시지를 처리해준다 (단, 쓰레드를 사용하는 경우에는
			// 아래 함수는 생략되어야 한다)
			cmxUtlProcessWndMsgM(BoardID, GetSafeHwnd(), 500, NULL);
	}

	long dwIsSuccess;

	if(cmxHomeGetSuccess(BoardID, nAxis, &dwIsSuccess) != ERR_NONE){
		// 에러메시지 출력
		return FALSE;
	}

	if(dwIsSuccess){
		MessageBox(NULL, "원점복귀를 성공적으로 수행하였습니다.", "Message",
MB_OK);
	}else{
		char szErrMsg[CMX_MAX_STR_LEN_ERR];
		char szErrReason[CMX_MAX_STR_LEN_ERR];

		    long dwErrCode;
		    cmxErrGetLastCode(BoardID, nAxis, &dwErrCode);
		    cmxErrGetString(BoardID, dwErrCode, szErrReason,
CMX_MAX_STR_LEN_ERR);

		    sprintf(szErrMsg, "다음과 같은 이유로 원점복귀에 실패하였습니다.\n%s",
szErrReason);
		MessageBox(NULL, szErrMsg, "Motion Error", MB_OK | MB_ICONERROR);
		   return FALSE;
}

return TRUE;
}
```

Visual Basic

```
'BoardID 는 0 으로 선언되었다고 가정함

Dim dwIsHomming As Long
Dim dwIsSuccess As Long

    dwIsHomming = True

    Call HomeMoveStart(BoardID, 3, cmDIR_N)

    Do While (dwIsHomming)
        Call HomeIsBusy(BoardID, 3, dwIsHomming) '원점 진행여부 확인(確認)
    loop

    If (HomeGetSuccess(BoardID, 3, dwIsSuccess) <> ERR_NONE) Then
        // 에러메시지 출력
    End If

    If (dwIsSuccess) Then
        MsgBox ("원점 복귀를 성공적으로 수행하였습니다.")
    End If
```

NAME

cmxHomeWaitDone

- 원점 복귀 완료 대기(完了待機原點 復歸)

INFORMATION

Home Return
VC++/VB
BCB/Delphi/.NET
Level 4
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxHomeWaitDone ([in] VT_I4 BoardID, [in] VT_I4 Channel, [in] VT_I4 IsBlocking)

DESCRIPTION

cmxHomeWaitDone() 함수는 해당 축에 대해 원점 복귀가 완료(完了)될때까지 기다립니다. 이 함수는 반복문(loop)에서 cmxHomeIsBusy() 함수를 계속 호출하다가 원점복귀가 완료(完了)되면 반복문(loop) 루프를 탈출 하는 용도로 사용됩니다.

cmxHomeIsBusy() 함수를 통해 원점 복귀가 완료된 것을 확인할 수 있으며, 내부적으로 반복문을 통해 원점 복귀 완료를 확인하는 함수가 cmxHomeWaitDone() 입니다. 용도에 따라서 사용하시기 바랍니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Channel : 축(채널) 번호. 축번호는 상수값으로 3 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.

▶ IsBlocking : 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다.

Value	Meaning
0 또는 cmxFALSE	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
1 또는 cmxTRUE	블록(Blocking)을 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

SEE ALSO

윈도우 이벤트라는 것은 무엇입니까?

윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

NAME

cmxHomeGetSuccess

cmxHomeSetSuccess

- 이전 원점 복귀(原點復歸) 완료 확인(確認) 및 완료 설정

INFORMATION

Home Return

VC++/VB

BCB/Delphi/.NET

Level 4

다소 주의

원점 복귀 성공 여부는 응용프로그램의 종료와 관계없이 유지됩니다.

SYNOPSIS

❒ VT_I4 cmxHomeGetSuccess ([in] VT_I4 BoardID, [in] VT_I4 Channel, [out] VT_PI4 IsSuccess)

❒ VT_I4 cmxHomeSetSuccess ([in] VT_I4 BoardID, [in] VT_I4 Channel, [in] VT_I4 IsSuccess)

DESCRIPTION

cmxHomeGetSuccess() 함수는 이 함수가 호출되기 이전에 원점복귀가 성공적으로 완료되었는지를 알려주는 함수입니다.
cmxHomeSetSuccess() 함수는 원점복귀의 성공여부에 대한 플래그 값을 강제로 설정하는 함수입니다. 일반적으로는 이 플래그 값은 원점복귀의 실제 수행에 의해서 셋팅됩니다. 그러나 필요한 경우에 강제로 그 값을 셋(Set) 또는 리셋(Reset)할 수 있습니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Channel : 축(채널) 번호. 축번호는 상수값으로 3 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.

▶ IsSuccess : cmxHomeGetSuccess 함수의 인자이며, 이 함수가 호출된 시점을 기준으로 이전에 원점복귀가 성공적으로 완료된 상태인지를 알려주는 매개 변수(媒介變數)입니다.

Value	Meaning
0 (FALSE)	지정한 축은 현재 원점복귀가 진행 중이거나 또는 비정상적으로 완료되었습니다
1 (TRUE)	지정한 축은 현재 원점복귀가 정상적으로 수행된 상태입니다.

▶ IsSuccess : cmxHomeSetSuccess 함수의 인자이며, 원점복귀의 성공여부에 대한 플레그 값을 강제로 설정합니다.

Value	Meaning

0 (FALSE)	지정한 축을 원점복귀가 진행 중인 상태로 설정합니다.
1 (TRUE)	지정한 축을 원점복귀가 정상적으로 수행된 상태로 설정합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

SEE ALSO

□ 원점복귀의 성공 여부에 대한 플래그 값은 응용프로그램이 종료(終了)되어도 그대로 유지됩니다. 따라서 다시 응용프로그램이 시작되면 이전에 원점복귀를 정상적으로 수행했었는지를 알 수가 있습니다. 단, PC 의 하드웨어적인 전원이 차단되거나 재시작(Rebooting) 되면 그 값은 FALSE 로 리셋됩니다. 따라서 cmxHomeGetSuccess() 함수의 이러한 특성(特性)을 활용하면 프로그램이 종료되었다가 다시 실행될 때 이전의 원점복귀 수행여부를 확인(確認)할 수가 없어서 매번 원점복귀를 수행해야 했던 불편을 보완(補完)할 수 있습니다.

□ IsSuccess 매개 변수(媒介變數)가 FALSE 인 경우는 원점복귀가 진행중인 경우를 의미할 수도 있고 비정상적으로 종료되었음을 의미할 수도 있습니다. 따라서 cmxHomeMoveStart() 함수를 사용한 경우에는 먼저 cmxHomeIsBusy() 함수나 cmxHomeWaitDone() 함수를 선행하여 완료를 확인(確認)한 후에 cmxHomeGetSuccess()를 사용하여 성공여부를 확인(確認)하는 것이 정석입니다.

□ 이전에 원점복귀가 성공적으로 수행되었더라도 해당 축의 원점복귀를 다시 시작하면 원점복귀의 성공 여부에 대한 플래그는 FALSE 로 리셋(Reset)됩니다.

EXAMPLE

아래의 예제에서 OnProgramInitialHome() 함수는 응용프로그램이 시작될 때 자동으로 원점복귀를 수행하기 위하여 호출되는 가상의 함수입니다. 단, 이때 cmxHomeGetSuccess() 함수를 이용하여 이전에 원점복귀가 이미 성공적으로 수행되었는지를 확인(確認)하고, 만일 그러한 경우라면 원점복귀를 생략하도록 합니다.

C/C++ :

```
// BoardID 는 0 으로 선언되었다고 가정함

BOOL OnProgramInitialHome(int nAxis)
{

    long dwAlreadyDone;
    cmxHomeGetSuccess(BoardID, nAxis, &dwAlreadyDone); // 이전에 원점복귀 상태
확인(確認)
    if(dwAlreadyDone){// 이전에 이미 원점복귀가 이루어졌으면 원점복귀 생략
    return TRUE;

}

long dwIsHomming = TRUE;
cmxHomeMoveStart(BoardID, nAxis, cmDIR_N);
while(dwIsHomming){
```

```
        cmxHomeIsBusy(BoardID, nAxis, &dwIsHomming); // 원점복귀 진행여부 읽기

        // 윈도우 메시지를 처리해준다 (단, 쓰레드를 사용하는 경우에는
}

// 원점복귀의 성공여부를 확인(確認)하여 처리한다. //
long dwIsSuccess;
cmxHomeGetSuccess(BoardID, nAxis, &dwIsSuccess);
if(dwIsSuccess){

        MessageBox(NULL, "원점복귀를 성공적으로 수행하였습니다.", "Message", MB_OK);

}else{

        char szErrMsg[CMX_MAX_STR_LEN_ERR];
        char szErrReason[CMX_MAX_STR_LEN_ERR];
        long dwErrCode;
        cmxErrGetLastCode(BoardID, nAxis, &dwErrCode);
        cmxErrGetString(BoardID, dwErrCode, szErrReason, CMX_MAX_STR_LEN_ERR);
        sprintf(szErrMsg, "다음과 같은 이유로 원점복귀에 실패하였습니다.\n%s", szErrReason);
                MessageBox(NULL, szErrMsg, "Motion Error", MB_OK | MB_ICONERROR);

      }
}
```

```
Visual Basic

'BoardID 는 0 으로 선언되었다고 가정함

Dim dwIsHomming As Long
Dim dwIsSuccess As Long

dwIsHomming = True

Call HomeMoveStart(BoardID, 3, cmDIR_N)
Do While (dwIsHomming)

        '원점 진행여부 확인(確認)
        Call HomeIsBusy(BoardID, 3, dwIsHomming)
Loop

If (HomeGetSuccess(BoardID, 3, dwIsSuccess) <> ERR_NONE)
Then
        // 에러메시지 출력

If (dwIsSuccess) Then
        MsgBox ("원점 복귀를 성공적으로 수행하였습니다.")
End If
```

Chapter

9

Advanced Motion Control

커미조아의 모션 세계는 비단 기본 모션제어에서만 국한되지 않습니다. 더 높은 기능과 더 안정적인 기능이 커미조아의 제품을 대변해 주고 있습니다. 고급 모션제어에서는 그 기능에 있어 다양한 기능을 표현하고 있지만, 기능의 응용에서 비로소 진정한 고급 모션제어를 구현하실 수 있습니다. ㈜커미조아의 고급 모션제어서는 1 나노미터의 오차도 허용하지 않는 저희 ㈜커미조아의 모션제품이 고객(顧客) 여러분들을 만족시켜드립니다.

이 단원에서는 속도(速度) 및 위치(位置) 오버라이딩 함수들을 소개합니다. 속도(速度) 오버라이딩은 모션이 진행되고 있는 중에 작업 속도를 변경하는 것을 의미합니다. 위치(位置) 오버라이딩은 상대좌표 모션 이송이나 절대좌표 모션이송과 같이 목적좌표를 향해 추종 모션을 수행하고 있는 중에 최종 목표 거리 또는 최종 목표 좌표를 수정하는 것을 의미 합니다.

9 고급 모션 제어 편

9.1 확장 보간제어 **(Extended Interpolation Motion)**

이 단원에서는 “확장 보간제어”와 관련된 함수들을 소개합니다. “확장 보간제어”에는 헬리컬 보간제어와 스플라인 보간제어가 포함됩니다.

가) 헬리컬 보간제어

헬리컬 보간 기능은 아래 그림과 같이 2 축 원호보간과 1 축 단축이송이 복합적으로 구동되는 기능입니다. 따라서 원호보간과 동기되어 1 축 또는 2 축의 직선보간을 수행할 수 있습니다. 그리고 거꾸로 1 축 또는 2 축 직선보간과 동기되어 원호보간을 수행할 수도 있습니다. 또한 이러한 작업을 리스트모션(Listed Motion) 기능과 연계하여 연속적으로 수행하면 그림 9-1 와 같이 나사산 모양의 작업을 수행할 수 있습니다.

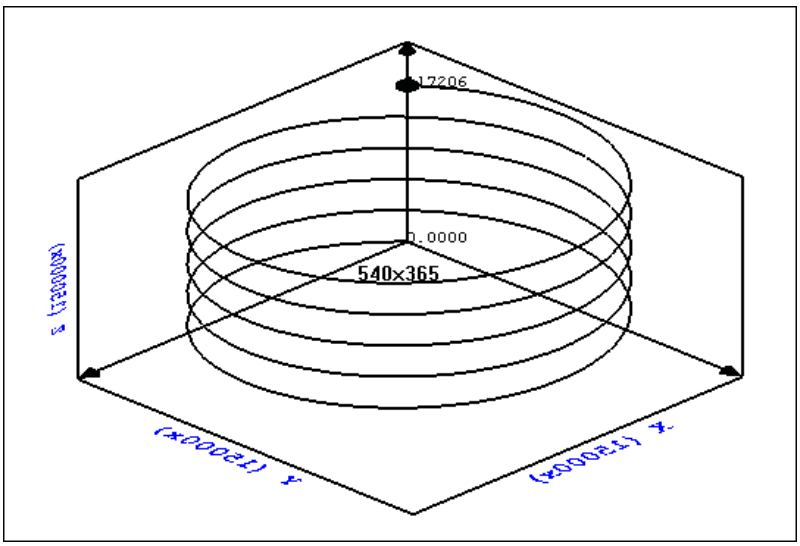


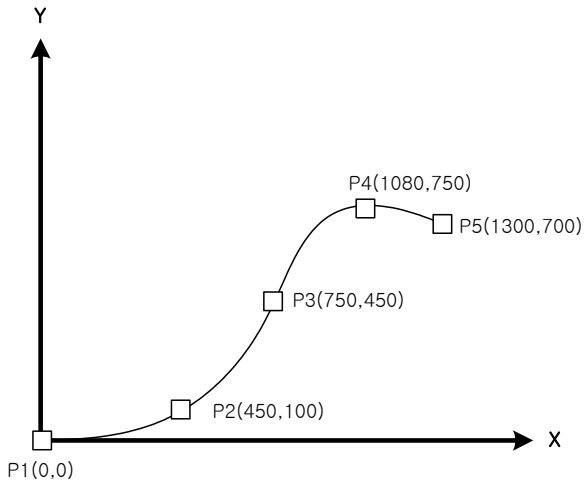
그림 9-1 연속적인 헬리컬 보간작업

□ 헬리컬 보간 기능 사용 시의 축 맵핑(Mapping)

하나의 헬리컬 보간 이송을 수행하는데 있어서 관여되는 축들을 정의하는 것을 축 맵핑이라고 합니다.

나) 스플라인 보간제어

㈜커미조아 모션제어 보드는 Cubic spline interpolation 기능을 지원합니다. Cublic spline interpolation 은 아래 그림과 같이 사용자가 지정하는 데이터 포인트를 자유곡선으로 보간해주는 기능입니다.



9.1.1 함수 요약

헬리컬 보간제어와 관련된 함수들은 다음의 표와 같습니다.

Summary of Functions
❒ _I4 cmxIxHelOnceStart ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_PI4 HelCoord, [in] VT_I4 ArcAngle) 대상(對象) 모션 채널에 대해서, 2 축 원호보간과 1 축 직선 보간 (補間)을 동시에 시작하고 동시에 종료하는 헬리컬 보간(補間) 구동을 동작합니다. 이 구동(驅動) 함수는 구동 시작 후 바로 반환(返還)됩니다.

커미조아의 정밀 스플라인 보간 위치 계산에 관련된 함수는 다음과 같습니다.

Summary of Functions
❒ VT_I4 cmxIxSplineStart ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_R8 InArray[][2], [in] VT_I4 NumInArray, [in] VT_I4 NumOutArray) 전달된 샘플좌표를 통해 Cubic Spline 보간 구동을 수행합니다.

9.1.2 함수 설명

NAME

cmxIxHelOnceStart

- 헬리컬 보간이송(補間移送)

INFORMATION

Extended Interpolation Motion

VC++/VB

BCB/Delphi/.NET

Level 5

이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인(確認)합니다.

SYNOPSIS

❒ VT_I4 cmxIxHelOnceStart

([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_PI4 HelCoord, [in] VT_I4 ArcAngle)

DESCRIPTION

2 축 원호보간과 1 축 직선보간을 동시에 시작하고 동시에 종료하는 헬리컬보간 구동을 시작합니다. cmxIxHelOnceStart () 함수는 모션을 시작시킨 후에 바로 반환됩니다.

헬리컬보간 구동에서 원호보간 이동은 그 속도와 이동량이 U 축과 동기되어 움직입니다. 따라서 U 축은 헬리컬보간에서 반드시 포함되어야 하며, 채널리스트의 마지막 축이 반드시 U 축으로 설정되어야 합니다. 그리고 U 축은 원호보간과 동기되어 움직이므로 이송량이 자동으로 결정됩니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ MapIndex: 맵번호(Map index), 이 맵번호를 사용하기전에 먼저 cmxIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑되어 있어야 합니다.

▶ HelCoord: 좌표 배열 주소. 3 축을 사용하는 경우와 4 축을 사용하는 경우에 이 배열의 구성은 다음과 같이 하면 됩니다.
□ 3 축을 사용하는 경우
nCoordList[0] : 원호 중심의 X 상대좌표값
nCoordList[1] : 원호 중심의 Y 상대좌표값
nCoordList[2] : U 축 방향 (0 또는 음수: 음의 방향, 양수: 양의 방향)

▶ ArcAngle: 원호보간 이동 각도. 이 값이 음수이면 시계방향으로 양수이면 반시계 방향으로 원호를 그리게 되며, 이 값의 절대값은 제한이 없습니다.

NAME

cmxIxSplineStart

- 스플라인(Spline) 보간(補間) 이송

INFORMATION

Extended Interpolation Motion

VC++/VB BCB/Delphi/.NET

Level 5

이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인(確認)합니다.

SYNOPSIS

❒ VT_I4 cmxIxSplineStart

([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [in] VT_R8 InArray[][2], [in] VT_I4 NumInArray, [in] VT_I4 NumOutArray)

DESCRIPTION

사용자가 입력한 데이터를 기반으로 Cubic spline 보간을 수행하여 생성된 곡선 데이터를 사용자가 지정한 배열에 전달합니다. 이 함수는 시간축을 매개변수로 하여 Cubic spline 보간을 수행합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ MapIndex: 맵번호(Map index), 이 맵번호를 사용하기전에 먼저 cmxIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑되어 있어야 합니다.

▶ InArray[][2] : 스플라인 보간을 수행할 샘플 데이터 포인트 배열. 단, 샘플 데이터 포인트는 최대 20 개까지만 가능합니다.

▶ NumInArray : 샘플 데이터의 수.

▶ NumOutArray : 스플라인 보간을 수행하여 자동 생성할 데이터 수. 이 값은 전체 곡선을 몇 개의 데이터 포인트로 곡선화할 것인지를 결정합니다.

9.2 리스트 모션(Listed Motion)

이 단원에서는 리스트 모션(Listed Motion) 제어에 관련된 함수들을 소개합니다. 리스트 모션은 수행해야할 여러 단계의 작업을 리스트로 등록시킨 후에 일괄적으로 처리하는 기능을 말합니다. 리스트 모션의 장점은 하나의 작업과 그 다음 작업간에 지연시간(Delay)이 없이 연속적인 작업을 수행할 수 있도록 한다는 것입니다. 또한 리스트 모션을 사용하면 MoveStart 나 MoveToStart 함수와 같은 In-Position 함수를 사용할 때에도 작업 속도의 연속성을 확보할 수 있습니다.

리스트 모션은 한 보드당 한 개의 리스트모션이 실행가능합니다.

9.2.1 함수 요약

리스트 모션에 관련된 함수들은 다음과 같습니다.

Summary of Functions
❒ VT_I4 cmxLmxStart ([in] VT_I4 BoardID, [in] VT_I4 LmIdx, [in] VT_I4 LmStartMode, [in] VT_I4 AxisMask) Listed Motion 에서 사용되는 모든 축들을 지정하고 모션을 시작합니다.
❒ VT_I4 cmxLmxSuspend ([in] VT_I4 BoardID, [in] VT_I4 LmIdx, [in] VT_I4 SuspendMode) Listed Motion 동작을 일시정지합니다.
❒ VT_I4 cmxLmxResume ([in] VT_I4 BoardID, [in] VT_I4 LmIdx, [in] VT_I4 ResumeMode) 일시정지된 Listed Motion 동작을 다시 재개합니다.
❒ VT_I4 cmxLmxEnd ([in] VT_I4 BoardID, [in] VT_I4 LmIdx) Listed Motion 동작을 종료합니다.
.❒ VT_I4 cmxLmxGetStates ([in] VT_I4 BoardID, [in] VT_I4 LmIdx, [in] VT_I4 LmStsId, [out] VT_PI4 LmxStsVal) Listed Motion 의 상태값을 반환합니다.
❒ VT_I4 cmxLmxSetSeqMode ([in] VT_I4 LmIdx, [in] VT_I4 SeqMode) Extend Listed Motion 수행 중에 새로운 이송 명령을 예약하려 하는데 이미 명령 버퍼(Extend Listed Motion Buffer) 가 이미 꽉 차있는 경우에 어떻게 처리할 지에 대한 모드를 설정합니다.
❒ VT_I4 cmxLmxGetSeqMode ([in] VT_I4 LmIdx, [out] VT_PI4 SeqMode) Extend Listed Motion 수행 중에 새로운 이송 명령을 예약하려 하는데 이미 명령 버퍼(Extend Listed Motion Buffer) 가 이미 꽉 차있는 경우에 어떻게 처리할 지에 대해 설정된 모드를 반환합니다.
❒ VT_I4 cmxLmxSetNextItcmxId ([in] VT_I4 LmIdx, [in] VT_I4 SeqId) Extend Listed Motion 에서 수행할 명령(Itcmx)에 대해 Sequence Itcmx Id 를 설정합니다.
❒ VT_I4 cmxLmxGetNextItcmxId ([in] VT_I4 LmIdx, [out] VT_PI4 SeqId) Extend Listed Motion 에서 다음 수행할 명령(Itcmx)에 해당하는 Sequence Itcmx Id 를 반환합니다.
❒ VT_I4 cmxLmxSetNextItcmxParam ([in] VT_I4 LmIdx, [in] VT_I4 ParamIdx, [in] VT_I4 ParamData) Extend Listed Motion 에서 다음 수행 예정인 명령에 대한 함수 파라미터 설정 값 설정합니다.

❒ VT_I4 cmxLmxGetNextItcmxParam ([in] VT_I4 LmIdx, [in] VT_I4 ParamIdx, [out] VT_PI4 ParamData) Extend Listed Motion 에서 다음 수행 예정인 명령에 대한 함수 파라미터 설정 값 반환합니다.
❒ VT_I4 cmxLmxGetRunItcmxParam ([in] VT_I4 BoardID, [in] VT_I4 LmIdx, [in] VT_I4 ParamIdx, [out] VT_PI4 ParamData) Extend Listed Motion 에서 현재 수행 중인 명령에 대한 함수 파라미터 설정 값 반환합니다.
❒ VT_I4 cmxLmxGetRunItemStaPos([in] VT_I4 LmIdx,[in] VT_I4 Axis, [out] VT_PR8 Position) Extend Listed Motion 수행 중에 현재 수행 중인 명령(Current Sequence Itcmx Id) 이 수행되기 직전에 해당 축의 Command Pulse Position 값을 반환 합니다.
❒ VT_I4 cmxLmxGetRunItemTargPos([in] VT_I4 LmIdx, [in] VT_I4 Axis, [out] VT_PR8 Position) Extend Listed Motion 수행 중에 현재 수행 중인 명령(Current Sequence Itcmx Id) 에 대해 해당 축의 목표 좌표에 해당하는 Command Pulse Position 값을 반환 합니다.
❒ VT_I4 cmxLmxSetSeqId ([in] VT_I4 BoardID, [in] VT_I4 LmIdx, [in] VT_I4 SeqId) Extend Listed Motion 에서 다음 차례에 수행할 Sequence Id 를 설정합니다.
❒ VT_I4 cmxLmxGetSeqId ([in] VT_I4 BoardID, [in] VT_I4 LmIdx, [out] VT_PI4 pSeqId) Extend Listed Motion 에서 다음 차례에 수행할 Sequence Id 를 반환합니다.

9.2.2 함수 설명

NAME

cmxLmxStart

- Listed Motion 대상(對象) 축 그룹(Group) 설정(設定) 및 수행 시작

INFORMATION

Listed Motion

VC++/VB

BCB/Delphi/.NET

Level 6

이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인(確認)합니다.

SYNOPSIS

❒ VT_I4 cmxLmxStart ([in] VT_I4 BoardID, [in] VT_I4 LmIdx, [in] VT_I4 LmStartMode, [in] VT_I4 AxisMask)

DESCRIPTION

이 함수는 리스트모션에서 사용되는 모든 축들을 등록하고 리스트 모션을 수행하는 함수입니다. X,Y 축을 리스트 모션을 이용하여 연속으로 여러 단계의 작업을 수행하면서 동시에 Z 축은 독립적으로 계속 구동되도록하고자 할 때 리스트 모션 작업의 영향으로 Z 축이 중간에 멈춰지는 현상이 벌어질 수 있습니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ LmIdx: 리스트모션의 Map Index 를 의미합니다. ComiRTEX 라이브러리는 동일 보드에 연결된 모든 서보(축)들의 수만큼의 리스트모션 작업이 각각 동시에 수행될 수 있습니다. 그러므로 이들을 서로 구분해줄 인자가 필요한데, LmIdx 가 바로 그 역할을 하는 인자입니다.

▶ LmStartMode: 리스트모션 동작의 시작모드를 결정합니다.

▶ MapMask : 리스트모션에 포함시킬 축에 Mask 값입니다. 32 비트로 이 값의 bit0 ~ bit31 은 각각 Axis0 ~ Axis31 의 리스트모션 포함 여부를 결정합니다. 비트 값이 0 이면 해당 축은 포함하지 않는 것이며, 1 이면 포함하는 것입니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

REFERENCE

□ ComiRTEX2 라이브러리는 동일 PC 에 장착된 모든 보드의 축들을 통합관리하는
통합라이브러리이므로 장치의 수만큼의 리스트모션 작업이 동시에 수행될 수 있습니다.
사용자는 LmIdx 매개 변수(媒介變數)를 사용하여 각각의 리스트모션 작업을 구분합니다.
모든 리스트모션 관련 함수는 LmIdx 를 매개 변수(媒介變數)로 취하고 있습니다.

NAME

cmxLmxSuspend

- Listed Motion 일시 정지

INFORMATION

Listed Motion

VC++/VB

BCB/Delphi/.NET

Level 6

다소 주의

Listed Motion 에서 Suspend 는 Resume 와 서로 짝을 이룹니다.

SYNOPSIS

❒ VT_I4 cmxLmxSuspend ([in] VT_I4 BoardID, [in] VT_I4 LmIdx, [in] VT_I4 SuspendMode)

DESCRIPTION

이 함수는 List Motion 의 동작을 일시정지 시키는 함수입니다. cmxLmResume() 함수를 통해 재개시킬 수 있습니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ LmIndex : 리스트모션의 Map Index 를 의미합니다.

▶ SuspendMode: 리스트모션의 일시정지 모드값을 결정합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

SEE ALSO

cmxLmxResume

NAME

cmxLmxResume

- Listed Motion 수행 재개

INFORMATION

Listed Motion

VC++/VB

BCB/Delphi/.NET

Level 6

다소 주의

Listed Motion 에서 Resume 는 Suspend 와 서로 짝을 이룹니다.

SYNOPSIS

❒ VT_I4 cmxLmxResume ([in] VT_I4 BoardID, [in] VT_I4 LmIdx, [in] VT_I4 ResumeMode)

DESCRIPTION

이 함수는 일시정지된 List Motion 의 동작을 재개시키는 함수입니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ LmIdx : 리스트모션의 Map Index 를 의미합니다.

▶ ResumeMode: 리스트모션의 재개 모드값을 결정합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

SEE ALSO

cmxLmxSuspend

NAME

cmxLmxEnd

- Listed Motion 수행 종료

INFORMATION

- Listed Motion
- VC++/VB
- BCB/Delphi/.NET
- Level 6
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxLmxEnd ([in] VT_I4 BoardID, [in] VT_I4 LmIdx)

DESCRIPTION

이 함수는 리스트 모션 수행을 종료하는 함수입니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ LmIdx : 리스트모션의 Map Index 를 의미합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxLmxGetStates

- Listed Motion 상태 반환

INFORMATION

- Listed Motion
- VC++/VB
- BCB/Delphi/.NET
- Level 6
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxLmxGetStates ([in] VT_I4 BoardID, [in] VT_I4 LmIdx, [in] VT_I4 LmStsId, [out] VT_PI4 LmxStsVal)

DESCRIPTION

이 함수는 리스트 모션의 상태를 반환하는 함수입니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ LmIdx : 리스트모션의 Map Index 를 의미합니다.

▶ LmStsId: 상태 종류를 의미하는 Id 값입니다.

▶ LmxStsVal: 해당 Id 의 상태값을 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxLmxSetSeqMode

cmxLmxGetSeqMode

- Listed Motion 이송명령 예약 시 기존 명령 버퍼 Full 인 경우 처리 방법 설정/반환

INFORMATION

Listed Motion

VC++/VB

BCB/Delphi/.NET

Level 6

위험 요소 없음

SYNOPSIS

- ❒ VT_I4 cmxLmxSetSeqMode ([in] VT_I4 LmIdx, [in] VT_I4 SeqMode)
- ❒ VT_I4 cmxLmxGetSeqMode ([in] VT_I4 LmIdx, [out] VT_PI4 SeqMode)

DESCRIPTION

이 함수는 리스트 모션 수행 중에 새로운 이송 명령을 예약하려 하는 경우에 이미 명령 버퍼(Extend Listed Motion Buffer) 가 꽉 차 있는 경우에 어떻게 처리할 지에 대한 모드를 설정/반환하는 함수입니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ LmIdx : 리스트모션의 Map Index 를 의미합니다.

▶ SeqMode : 예약하려는 명령의 처리 방법입니다.

Value	Meaning
0	새로 예약하려는 명령이 SKIP 됩니다. 함수는 바로 반환됩니다.
1	명령 버퍼에 free space 가 생길 때까지 대기하고 있다가 free space 가 생기면 새로운 명령이 예약되고 함수가 반환 됩니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxLmxSetNextItcmxId

cmxLmxGetNextItcmxId

- Listed Motion 에서 처리할 명령의 Sequence Itcmx ID 설정/반환

INFORMATION

Listed Motion

VC++/VB

BCB/Delphi/.NET

Level 6

위험 요소 없음

SYNOPSIS

- ❒ VT_I4 cmxLmxSetNextItcmxId ([in] VT_I4 LmIdx, [in] VT_I4 SeqId)
- ❒ VT_I4 cmxLmxGetNextItcmxId ([in] VT_I4 LmIdx, [out] VT_PI4 SeqId)

DESCRIPTION

이 함수는 리스트 모션에서 수행할 명령에 대해 Sequence Itcmx Id 를 설정/반환합니다.
이 Sequence Itcmx Id 는 cmxLmxSetSeqId() 함수를 통해 수행할 리스트 모션 명령을 구분하는 데 사용되므로 중복된 Id 를 사용하시면 안되며 고유의 순차적 Id 로 설정되어야 합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ LmIdx : 리스트모션의 Map Index 를 의미합니다.

▶ SeqId : 해당 Sequence 단계에 해당하는 이송 또는 설정 함수에 대해 설정할 순차적 Sequence Itcmx Id

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxLmxSetNextItcmxParam

cmxLmxGetNextItcmxParam

- Listed Motion 에서 다음 수행 예정인 명령의 함수 파라미터 설정값 설정/반환

INFORMATION

Listed Motion

VC++/VB

BCB/Delphi/.NET

Level 6

위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxLmxSetNextItcmxParam ([in] VT_I4 LmIdx, [in] VT_I4 ParamIdx, [in] VT_I4 ParamData)

❒ VT_I4 cmxLmxGetNextItcmxParam ([in] VT_I4 LmIdx, [in] VT_I4 ParamIdx, [out] VT_PI4 ParamData)

DESCRIPTION

이 함수는 리스트 모션에서 다음 수행할 예정인 명령에 대해 함수 파라미터 값을 설정/반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ LmIdx : 리스트모션의 Map Index 를 의미합니다.

▶ ParamIdx : 해당 명령에 대해 값 변경/확인을 원하는 파라미터의 인덱스 값

▶ ParamData : 해당 명령에 대해 설정/반환하고자 하는 파라미터 데이터 값

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxLmxGetRunItcmxParam

- Listed Motion 에서 현재 수행 중인 명령에 대한 함수 파라미터 설정값 반환

INFORMATION

- Listed Motion
- VC++/VB
- BCB/Delphi/.NET
- Level 6
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxLmxGetRunItcmxParam ([in] VT_I4 BoardID, [in] VT_I4 LmIdx, [in] VT_I4 ParamIdx, [out] VT_PI4 ParamData)

DESCRIPTION

이 함수는 리스트 모션에서 현재 수행중인 명령에 대한 함수 파라미터 설정값을 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ LmIdx : 리스트모션의 Map Index 를 의미합니다.

▶ ParamIdx : 해당 명령에 대해 확인을 원하는 파라미터의 인덱스 값

▶ ParamData : 해당 명령에 대해 설정된 파라미터 데이터 값

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxLmxGetRunItemStaPos

cmxLmxGetRunItemTargPos

- Listed Motion 에서 현재 수행 중인 명령의 해당 축 시작/목표 위치 반환

INFORMATION

Listed Motion

VC++/VB

BCB/Delphi/.NET

Level 6

위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxLmxGetRunItemStaPos ([in] VT_I4 LmIdx, [in] VT_I4 Axis, [out] VT_PR8 Position)

❒ VT_I4 cmxLmxGetRunItemTargPos ([in] VT_I4 LmIdx, [in] VT_I4 Axis, [out] VT_PR8 Position)

DESCRIPTION

이 함수는 리스트 모션에서 현재 수행중인 명령에 대해 수행되기 직전의 해당축 위치/수행 완료 후의 해당축 위치값을 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ LmIdx : 리스트모션의 Map Index 를 의미합니다.

▶ Axis : zero-based(0 번축 기준) 로 축 번호를 지정합니다.

▶ Position : 해당 축에 대해 현재 시퀀스 명령이 수행되기 직전의 명령 펄스 위치

▶ Position : 해당 축에 대해 현재 시퀀스 명령의 이송 목표 좌표에 해당하는 명령 펄스 위치

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxLmxSetSeqId

cmxLmxGetSeqId

- Listed Motion 에서 처리할 명령의 Sequence ID 설정/반환

INFORMATION

Listed Motion

VC++/VB

BCB/Delphi/.NET

Level 6

위험 요소 없음

SYNOPSIS

- ❒ VT_I4 cmxLmxSetSeqId ([in] VT_I4 BoardID, [in] VT_I4 LmIdx, [in] VT_I4 SeqId)
- ❒ VT_I4 cmxLmxGetSeqId ([in] VT_I4 BoardID, [in] VT_I4 LmIdx, [out] VT_PI4 SeqId)

DESCRIPTION

이 함수는 리스트 모션에서 수행할 명령의 Sequence Itcmx Id 를 설정하거나 수행중인 명령의 Sequence Itcmx Id 를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ LmIdx : 리스트모션의 Map Index 를 의미합니다.

▶ SeqId : 해당 Sequence 단계에 해당하는 이송 또는 설정 함수의 Sequence Itcmx Id

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

9.3 속도 및 위치 오버라이딩(Overriding)

이 단원에서는 속도 및 위치 오버라이딩 함수들을 소개합니다. 속도 오버라이딩은 모션이 진행되고 있는 중에 작업 속도를 변경하는 것을 의미합니다. 위치 오버라이딩은 Move 나 MoveTo 와 같이 In-Position 모션을 수행하고 있는 중에 목표 거리 또는 목표 좌표를 수정하는 것을 의미 합니다. 일반적인 모션 구동에서 많이 요구되는 기능은 아니지만, 그 기능면에 있어, 타사의 다른 오버라이드 기능보다도 월등한 기능과 성능, 그리고 정확성을 제공하고 있습니다.

9.3.1 함수 요약

㈜ 커미조의속도 및 위치 오버라이딩에 관련된 함수는 다음과 같습니다.

Summary of Functions
❒ VT_I4 cmxOverrideSpeedSet ([in] VT_I4 BoardID, [in] VT_I4 Channel) 단축(單軸) 모션 작업이 진행되고 있는 중에 속도(速度)를 변경합니다.
❒ VT_I4 cmxOverrideMove ([in] VT_I4 BoardID, [in] VT_I4 Channel, [in] VT_R8 NewDistance, [out] VT_PI4 IsIgnored) 단축(單軸) 구동 함수를 통해서 구동되는 단축상대좌표이송(單軸相對座標移送) 모션에 대하여, 상대(相對) 좌표상의 목표 논리 거리 값을 수정합니다.
❒ VT_I4 cmxOverrideMoveTo ([in] VT_I4 BoardID, [in] VT_I4 Channel, [in] VT_R8 NewPosition, [out] VT_PI4 IsIgnored) 단축(單軸) 구동 함수를 통해서 구동되는 단축절대좌표이송(單軸絶對座標移送) 모션에 대하여, 절대(絶對) 좌표상의 목표 논리 거리 값을 수정합니다.

참고적으로, 속도 및 위치 오버라이딩의 함수는 모션이 종료된 시점에서 수행되는 함수가 아닌, 모션의 이송중에 적용되어 지는 함수이기 때문에, 이전에 수행된 모션 명령이 cmxSxMove 나 cmxSxMoveTo 같은 결과적으로 모션이송의 목표 위치에 대한 완료를 동반하여 반환되는 함수에서는 사용하는 것이 배제됩니다. 따라서 오버라이딩 함수의 이전함수는 cmxSxMoveStart 나 cmxSxMoveToStart 와 같은 함수를 통해 이송 명령이 설정된 후에 응용프로그램의 제어를 즉시 반환 받고, 오버라이딩을 수행하게 됩니다.

9.3.2 함수 설명

NAME

cmxOverrideSpeedSet

- 단축속도(單軸速度) 오버라이딩 실행

INFORMATION

Overriding

VC++/VB

BCB/Delphi/.NET

Level 5

다소 주의

함수의 호출전에 속도 지령 함수를 통해 변경하고자 하는 속도를 설정합니다.

SYNOPSIS

❒ VT_I4 cmxOverrideSpeedSet ([in] VT_I4 BoardID, [in] VT_I4 Channel)

DESCRIPTION

이 함수는 단축 모션이 진행되고 있는 중에 속도를 오버라이딩하고자할 때 사용하는 함수입니다. 속도를 오버라이딩하기 위해서는 먼저 cmxCfgSetSpeedPattern() 속도 패턴 설정 함수를 통하여 변경하고자 하는 속도 또는 가속도값을 설정하고나서 이 함수를 수행해야합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Channel : 축(채널) 번호. 축번호는 상수값으로 3 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

SEE ALSO

□ 직선, 원호, 헬리컬 보간작업을 수행하는 경우에는 속도 오버라이딩을 사용할 수 없습니다.

EXAMPLE

본 예제는 cmxOverrideSpeedSet()함수를 사용하여 속도를 오버라이딩하는 것을 예로 보여주는 코드입니다. 본 예제는 "HIGH" 와 "LOW" 로 이름지어진 두 개의 버튼이 있다고 가정하고 "HIGH" 버튼이 눌리면 X1 축의 속도를 20000 으로 설정하고 "LOW" 버튼이 눌리면 속도를 10000 으로 설정하는 예입니다.

C/C++

```
// BoardID 는 0 으로 선언되었다고 가정함

#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

#define V_LOW          10000    // 저속모드 속도
#define V_HIGH  20000           // 고속모드 속도

/*************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.

*************************************************************/
void OnProgramInitial()
{

      long m_nNumDevices;
      long m_DeviceList[16];
      long m_nNumAxes;
      cmxLoadDll();
      if(cmxGnDeviceLoad(&m_nNumDevices, m_DeviceList, &m_nNumAxes)!= ERR_NONE)
      {
            //Handle 은 사용자가 생성한 폼의 핸들 값입니다.
            // 에러메시지 출력
            return;
      }
}

/*************************************************************
* OnStart() : 이 함수는 가상의 함수로서 X 축에 대하여 V-MOVE 모션을 시작합니다.

*************************************************************/
void OnStart()
{
      long nIsDone;
      // 해당축이 작업중이면 정지(停止)하고 다시 시작 //
      cmxSxIsDone(BoardID, 3, &nIsDone);
      if(nIsDone != cmxTRUE) cmxSxStopEmg(0);
      // 속도설정 => 시작은 LOW 속도로 시작 //
      cmxCfgSetSpeedPattern(BoardID, 3, cmxSMODE_S, V_LOW, 50000, 50000,0,0);
      // V-Move start //
      if(cmxSxVMoveStart(BoardID, 3, cmDIR_P)){
            // 에러메시지 출력
            return;
      }
}
/*************************************************************
* OnHighButtonClick() : "HIGH" 버튼 콜백함수 (가상함수)
* "HIGH"버튼이 클릭되면 속도를 V_HIGH 속도로 오버라이드한다.
*************************************************************/
void OnHighButtonClick()
{
      // V_HIGH 속도로 오버라이딩 //
      cmxSxSetSpeedRatio(BoardID, 3, cmxSMODE_S, , 200, 100, 100,0,0);
```

```
	//아래 코드로 대체가 가능합니다.
	//cmxCfgSetSpeedPattern(BoardID, 3, cmxSMODE_S, V_HIGH, 50000,  50000,0,0);

	if(cmxOverrideSpeedSet (0) != ERR_NONE){
		// 에러메시지 출력
		return;
	}
}
/*************************************************************
* OnLowButtonClick() : "LOW" 버튼 콜백함수 (가상함수)
* "LOW"버튼이 클릭되면 속도를 V_LOW 속도로 오버라이드한다.
*************************************************************/
void OnLowButtonClick()
{
	// V_LOW 속도로 오버라이딩 //
	cmxSxSetSpeedRatio(BoardID, 3, cmxSMODE_S, , 100, 100, 100);
	//아래 코드로 대체가 가능합니다.
	//cmxCfgSetSpeedPattern(BoardID, 3, cmxSMODE_S, V_LOW, 50000, 50000,0,0);

	if(cmxOverrideSpeedSet (0) != ERR_NONE){
		// 에러메시지 출력
		return;
	}
}
/*************************************************************
* OnStop() : "Stop"명령시에 호출되는 가상의 함수
*************************************************************/
void OnStop()
{
	cmxSxStopEmg(0);
}
```

Visual Basic

```
'BoardID 는 0 으로 선언되었다고 가정함

'/*************************************************************
'* OnStart() : 이 함수는 가상의 함수로서 X 축에 대하여 V-MOVE 모션을
'*시작합니다.
'*************************************************************/
Private Sub OnStart()

	Dim nIsDone As Long

	'해당축이 작업중이면 정지(停止)하고 다시 시작
	Call SxIsDone(BoardID, 3, nIsDone)
	If (nIsDone <> cmxTRUE) Then
		SxStopEmg (0)
	End If

	'시작은 LOW 속도로 시작
	Call CfgSetSpeedPattern(BoardID, 3, cmxMODE_S, 10000, 50000, 50000)

	If (SxVMoveStart(BoardID, 3, cmDIR_P)) Then
	   // 에러메시지 출력
	End If
```

```
End Sub

'/*************************************************************
'* OnHighButtonClick() : "HIGH" 버튼 콜백함수 (가상함수)
'* "HIGH"버튼이 클릭되면 속도를 V_HIGH 속도로 오버라이드한다.
'*************************************************************/
Private Sub OnHighButtonClick()

    'V_HIGH 속도로 오버라이딩
    Call SxSetSpeedRatio(BoardID, 3, cmxSMODE_S, 200, 100, 100)

    '아래 코드로 대체가 가능합니다.
    'Call CfgSetSpeedPattern(BoardID, 3, cmxSMODE_S, 20000, 50000, 50000)

    If (OverrideSpeedSet(0) <> ERR_NONE) Then
       // 에러메시지 출력
    End If

End Sub

'/*************************************************************
'* OnLowButtonClick() : "LOW" 버튼 콜백함수 (가상함수)
'* "LOW"버튼이 클릭되면 속도를 V_LOW 속도로 오버라이드한다.
'*************************************************************/
Private Sub OnLowButtonClick()

    'V_LOW 속도로 오버라이딩
    Call CfgSetSpeedPattern(BoardID, 3, cmxSMODE_S, 10000, 50000, 50000)

    If (OverrideSpeedSet(0) <> ERR_NONE) Then
       // 에러메시지 출력

End Sub

`/*************************************************************
`* OnStop() : "Stop"명령시에 호출되는 가상의 함수
`*************************************************************/
Private Sub OnStop()

    Call SxStopEmg(0)

End Sub
```

NAME

cmxOverrideMove

- 단축상대위치(單軸相對位置) 오버라이드 이송(移送)

INFORMATION

Overriding

VC++/VB

BCB/Delphi/.NET

Level 5

다소 위험

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인(確認)합니다.

SYNOPSIS

❒ VT_I4 cmxOverrideMove

([in] VT_I4 BoardID, [in] VT_I4 Channel, [in] VT_R8 NewDistance, [out] VT_PI4 IsIgnored)

DESCRIPTION

이 함수는 cmxSxMoveStart() 이송 함수를 통하여 수행되는 상대좌표 In-position 모션에 대하여 상대좌표값, 즉 목표 거리값을 오버라이딩하는 함수입니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ Channel : 축(채널) 번호. 축번호는 상수값으로 3 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.

▶ NewDistance : 새로운 목표 거리값을 지정합니다. 이 값의 기준 위치는 오버라이드하고자 하는 대상이 되는 cmxSxMoveStart() 작업에서 사용한 기준점과 같습니다. 즉, 새로운 목표 거리는 cmxSxMoveStart() 함수를 실행하기 바로직전의 위치를 기준으로 계산하여야 합니다.

▶ IsIgnored : cmxOverrideMove 의 적용 성공/실패 여부를 반환 합니다.

Value	Meaning
0	모션에러가 발생하였거나 이미 이송이 완료되어 위치 오버라이드가 적용되지 않음.
1	위치 오버라이드가 적용됨.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

REFERENCE

□ 위치 오버라이드를 수행하려는 시점에 이미 이송이 완료되어 버린 경우에는 위치 오버라이드는 무시되고 반환값을 0 으로 반환합니다. 따라서 사용자는 반환값이 0 인 경우에는 이미 이송이 완료되어 오버라이드가 적용되지 않은 것으로 인지하여야 하며, 그럼에도 불구하고 목표좌표를 수정해야하는 경우에는 cmxSxMove() 또는 cmxSxMoveTo() 함수를 추가적으로 수행해야 합니다. 이러한 경우에는 오버라이드라는 개념 보다는 추가 이송의 개념을 의미합니다.

NAME

cmxOverrideMoveTo

- 단축절대위치(單軸絶對位置) 오버라이드 이송(移送)

INFORMATION

Overriding

VC++/VB

BCB/Delphi/.NET

Level 5

다소 위험

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인(確認)합니다.

SYNOPSIS

❒ VT_I4 cmxOverrideMoveTo ([in] VT_I4 BoardID, [in] VT_I4 Channel, [in] VT_R8 NewPosition, [out] VT_PI4 IsIgnored)

DESCRIPTION

이 함수는 cmxSxMoveToStart() 함수를 통하여 수행되는 절대좌표 In-position 모션에 대하여 목표 절대좌표값을 오버라이딩하는 함수입니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Channel : 축(채널) 번호. 축번호는 상수값으로 3 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.

▶ NewPosition : 새로운 목표 절대좌표값을 지정합니다.

▶ IsIgnored : cmxOverrideMoveTo 의 적용 성공/실패 여부를 반환 합니다.

Value	Meaning
0	모션에러가 발생하였거나 이미 이송이 완료되어 위치 오버라이드가 적용되지 않음.
1	위치 오버라이드가 적용됨.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

REFERENCE

□ 위치 오버라이드를 수행하려는 시점에 이미 이송이 완료되어 버린 경우에는 위치 오버라이드는 무시되고 반환값을 0 으로 반환합니다. 따라서 사용자는 반환값이 0 인 경우에는 이미 이송이 완료되어 오버라이드가 적용되지 않은 것으로 인지하여야 하며, 그럼에도 불구하고 목표좌표를 수정해야하는 경우에는 cmxSxMoveTo() 함수를 추가적으로 수행해야합니다.

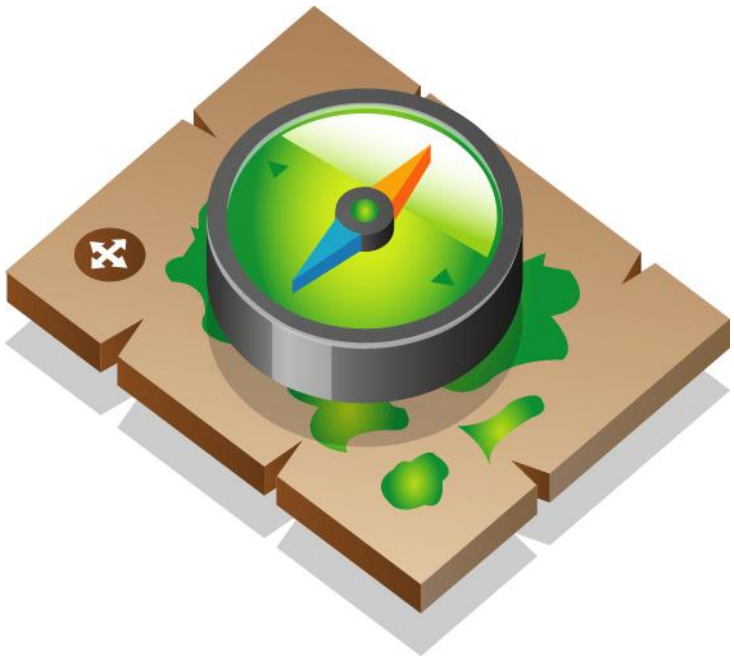
Chapter

10

Monitoring Motion Status

모션제어의 상태를 확인(確認)하는 역할은 고객(顧客)여러분들께서 개발하시는 응용프로그램의 필수 요건 중에 하나입니다. ComiRTEX 에서는 매우 자세하고 효율적인 상태 관리 매커니즘을 가지고 있습니다. ㈜커미조아의 많은 장점 중에 하나인 전문적인 모션 정보 제공 인터페이스를 통해 보다 자세하고 신속한 모션 프로그램의 상태를 구현하시기 바랍니다.

이 단원에서는 모션제어의 상태(狀態) 감시에 관련된 함수들에 대하여 설명합니다. 상태(狀態) 감시 함수들은 모션의 상태를 감시하는데 필요한 함수들을 그룹화한 것입니다. 상태(狀態) 감시에는 모션의 속도, 위치 등을 확인(確認)하는 것을 포함하며 이외에도 현재 모션이 진행되고 있는지를 확인(確認)하고, 현재 진행되고 있는 모션의 단계 및 각 I/O 상태들을 점검하는 기능도 포함합니다.



10 상태감시 편

10.1 모션제어 상태(Status) 감시 및 설정

이 단원에서는 모션제어의 상태 감시에 관련된 함수들에 대하여 설명합니다. 상태 감시 함수들은 모션의 상태를 감시하는데 필요한 함수들을 그룹화한 것입니다. 상태 감시에는 모션의 속도, 위치 등을 확인(確認)하는 것을 포함하며 이외에도 현재 모션이 진행되고 있는지를 확인(確認)하고, 현재 진행되고 있는 모션의 단계 및 각 I/O 상태들을 확인(確認)하는 기능도 포함합니다.

10.1.1 함수 요약

상태 감시 및 제어 함수에 관련된 함수들은 다음과 같습니다.

Summary of Functions
❒ VT_I4 cmxStSetCount ([in] VT_I4 BoardID, [in] VT_I4 Channel, [in] VT_I4 Source, [in] VT_I4 pdwCount) 대상(對象) 모션 채널의 지정한 카운터(Counter)의 값을 전달된 매개변수를 통해 설정합니다. 단, 이때 지정하는 카운터값의 단위(單位)는 펄스수입니다.
❒ VT_I4 cmxStGetCount ([in] VT_I4 BoardID, [in] VT_I4 Channel, [in] VT_I4 Source, [out] VT_PI4 pdwCount) 대상(對象) 모션 채널의 지정한 카운터(Counter)의 값을 전달된 매개변수를 통해 반환합니다. 단, 이때 반환되는 카운터값의 단위(單位)는 펄스수입니다.
❒ VT_I4 cmxStSetPosition ([in] VT_I4 BoardID, [in] VT_I4 Channel, [in] VT_I4 Source, [in] VT_R8 Count) 대상(對象) 채널의 지정한 카운터(Counter)의 값을 전달된 매개변수를 통해 설정합니다. 단, 이때 지정하는 카운터값의 단위는 논리적인 거리 단위(單位)입니다.
❒ VT_I4 cmxStGetPosition ([in] VT_I4 BoardID, [in] VT_I4 Channel, [in] VT_I4 Source, [out] VT_PR8 Count) 대상(對象) 채널의 지정한 카운터(Counter)의 값을 전달된 매개변수를 통해 반환합니다. 단, 이때 반환(返還)되는 카운터값의 단위는 논리적인 거리 단위(單位)입니다.
❒ VT_I4 cmxStGetSpeed ([in] VT_I4 BoardID, [in] VT_I4 Channel, [in] VT_I4 Source, [out] VT_PR8 Speed) 대상(對象) 채널의 Command 또는 Feedback 속도를 확인하여, 전달된 매개 변수를 통해 논리적 속도 단위로 반환(返還)합니다.
❒ VT_I4 cmxStGetTorque ([in] VT_I4 BoardID, [in] VT_I4 Channel, [out] VT_PI4 Torque) 대상(對象) 채널의 토크값을 확인하여, 전달된 매개 변수를 통해 토크값을 반환(返還)합니다.
❒ VT_I4 cmxStReadMioStatuses ([in] VT_I4 BoardID, [in] VT_I4 Channel, [out] VT_PI4 MioStates) 대상(對象) 모션 채널에 대해서, 현재의 모션의 관련 I/O 신호 및 주변 신호(Machine I/O) 상태를 반환(返還)합니다.
❒ VT_I4 cmxStSxReadMotionState ([in] VT_I4 BoardID, [in] VT_I4 Channel, [out] VT_PI4 MotStates) 대상(對象) 모션 채널에 대해서, 현재의 속도 상태를 반환(返還)합니다.
❒ VT_I4 cmxStIxReadMotionState ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [out] VT_PI4 MotStates) 대상(對象) 모션맵에 대해서, 각 보간 모드에 따라 현재의 속도 상태를 반환(返還)합니다.

❒ VT_I4 cmxStGetMotionMode ([in] VT_I4 BoardID, [in] VT_I4 Channel, [out] VT_PI4 Mode)
대상(對象) 모션 채널에 대해서, 현재 모션의 종류를 확인합니다.

❒ VT_I4 cmxStSxGetLastError ([in] VT_I4 BoardID, [in] VT_I4 Channel, [out] VT_PI4 LastError)
단축 구동시의 마지막 발생한 에러코드를 확인합니다.

❒ VT_I4 cmxStIxGetLastError ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [out] VT_PI4 LastError)
보간 구동시의 마지막 발생한 에러코드를 확인합니다.

❒ VT_I4 cmxStSetMultiRevCnt ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 MultiRevCnt)
대상(對象) 모션 채널의 절대 위치를 지정하기 위한 회전수를 지정합니다. 단, 이 때 지정하는 단위(單位)는 펄스 수를 기준으로 한 회전 수 입니다.

❒ VT_I4 cmxStGetMultiRevCnt ([in] VT_I4 BoardID, [in] VT_I4 Axis, [out] VT_PI4 pMultiRevCnt)
대상(對象) 모션 채널의 절대 위치를 확인하기 위한 회전수를 반환합니다. 단, 이때 반환되는 단위(單位)는 펄스 수를 기준으로 한 회전 수 입니다.

❒ VT_I4 cmxStSetOneRevPos ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 OneRevPos)
대상(對象) 모션 채널의 절대 위치를 지정하기 위한 한 회전 내 펄스 수를 지정합니다. 단, 이 때 지정하는 단위(單位)는 펄스 수 입니다.

❒ VT_I4 cmxStGetOneRevPos ([in] VT_I4 BoardID, [in] VT_I4 Axis, [out] VT_PI4 pOneRevPos)
대상(對象) 모션 채널의 절대 위치를 확인하기 위한 한 회전 내 펄스 수를 반환합니다. 단, 이때 반환되는 단위(單位)는 펄스 수 입니다.

10.1.2 함수 설명

NAME

cmxStSetCount

- 사용자정의(使用者定義) 하드웨어 카운트 값 설정(設定)

INFORMATION

- Motion Status
- VC++/VB
- BCB/Delphi/.NET
- Level 7
- ☺ 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxStSetCount ([in] VT_I4 BoardID, [in] VT_I4 Channel, [in] VT_I4 Source, [in] VT_I4 pdwCount)

DESCRIPTION

지정한 채널의 지정한 카운터의 값을 새로이 설정합니다. 단, 이때 지정하는 카운터값의 단위는 펄스수입니다.
이 함수는 카운터의 값을 지정하는 매개 변수(媒介變數)의 단위가 펄스수라는 것을 제외하고는 cmxStSetPosition() 함수와 동일합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Channel : 축(채널) 번호. 축번호는 상수값으로 3 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.

▶ Source: 설정할 카운터 번호. 이 값은 다음의 4 가지 값중의 하나이어야 합니다.

Value	Meaning
0 또는 cmxCNT_COMM	Command Counter
1 또는 cmxCNT_FEED	Feedback Counter
2 또는 cmxCNT_DEV	Deviation Counter : Command 와 Feedback counter 의 편차 카운터
3 또는 cmxCNT_GEN	General Counter : 사용자의 정의에 따라 여러가지 용도로 사용될 수 있는 카운터

▶ pdwCount: 지정한 값으로 대상 카운터의 값을 설정합니다. 이 값은 이 값은 논리단위가 아닌 실제 펄스 카운트값입니다.

SEE ALSO

cmxStGetCount

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxStGetCount

- 사용자정의(使用者定義) 하드웨어 카운트 값 반환(返還)

INFORMATION

Motion Status
VC++/VB
BCB/Delphi/.NET
Level 7
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxStGetCount ([in] VT_I4 BoardID, [in] VT_I4 Channel, [in] VT_I4 Source, [out] VT_PI4 pdwCount)

DESCRIPTION

지정한 채널의 지정한 카운터의 값을 읽어서 반환합니다. 단, 이때 반환되는 값의 단위는 펄스수입니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Channel : 축(채널) 번호. 축번호는 상수값으로 3 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.

▶ Source : 값을 읽을 카운터 번호. 이 값은 다음의 4 가지 값중의 하나이어야 합니다.

Value	Meaning
0 또는 cmxNT_COMM	Command Counter
1 또는 cmxNT_FEED	Feedback Counter
2 또는 cmxNT_DEV	Deviation Counter : Command 와 Feedback counter 의 편차 카운터
3 또는 cmxNT_GEN	General Counter : 사용자의 정의에 따라 여러가지 용도로 사용될 수 있는 카운터

▶ pdwCount : 대상 카운터의 값을 반환합니다. 이 값은 이 값은 논리단위가 아닌 실제 펄스 카운트값입니다.

SEE ALSO

cmxStSetCount

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxStSetPosition

- 사용자정의(使用者定義) 논리적(論理的)

카운트 값 설정

INFORMATION

Motion Status

VC++/VB

BCB/Delphi/.NET

Level 7

위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxStSetPosition ([in] VT_I4 BoardID, [in] VT_I4 Channel, [in] VT_I4 Source, [in] VT_R8 Count)

DESCRIPTION

지정한 채널의 지정한 카운터의 값을 새로이 설정합니다. 단, 이때 지정하는 카운터값의 단위는 "Unit distance"에 의해 정의되는 논리적 거리 단위입니다.
이 함수는 카운터의 값을 지정하는 매개 변수(媒介變數)가 논리거리 단위라는 것을 제외하고는 cmxStSetCount() 함수와 동일합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Channel : 축(채널) 번호. 축번호는 상수값으로 3 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.

▶ Source : 설정할 카운터 번호. 이 값은 다음의 4 가지 값중의 하나이어야 합니다.

Value	Meaning
0 또는 cmxNT_COMM	Command Counter
1 또는 cmxNT_FEED	Feedback Counter
2 또는 cmxNT_DEV	Deviation Counter : Command 와 Feedback counter 의 편차 카운터
3 또는 cmxNT_GEN	General Counter : 사용자의 정의에 따라 여러가지 용도로 사용될 수 있는 카운터

▶ Count : 대상 카운터에 설정될 값. 단, 이 값은 논리적 거리 단위로 설정하여야 합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

SEE ALSO

cmxStGetPosition

REFERENCE

NAME

cmxStGetPosition

- 사용자정의(使用者定義) 논리적(論理的) 카운트 값 반환

INFORMATION

- Motion Status
- VC++/VB
- BCB/Delphi/.NET
- Level 7
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxStGetPosition ([in] VT_I4 BoardID, [in] VT_I4 Channel, [in] VT_I4 Source, [out] VT_PR8 Count)

DESCRIPTION

지정한 채널의 지정한 카운터의 값을 읽어서 반환합니다. 단, 이때 반환되는 값의 단위는 논리적 거리입니다.
이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.
▶ Channel : 축(채널) 번호. 축번호는 상수값으로 3 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.
▶ Source : 대상 카운터 번호. 이 값은 다음의 4 가지 값중의 하나이어야 합니다.

Value	Meaning
0 또는 cmxNT_COMM	Command Counter
1 또는 cmxNT_FEED	Feedback Counter
2 또는 cmxNT_DEV	Deviation Counter : Command 와 Feedback counter 의 편차 카운터
3 또는 cmxNT_GEN	General Counter : 사용자의 정의에 따라 여러가지 용도로 사용될 수 있는 카운터

▶ Count : 전달된 변수를 통해 대상 카운터의 값을 읽어서 논리적 거리 단위로 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

SEE ALSO

cmxStSetPosition

NAME

cmxStGetSpeed

- 논리적(論理的) 속도 반환

INFORMATION

Motion Status

VC++/VB

BCB/Delphi/.NET

Level 7

위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxStGetSpeed ([in] VT_I4 BoardID, [in] VT_I4 Channel, [in] VT_I4 Source, [out] VT_PR8 Speed)

DESCRIPTION

Command 또는 Feedback 속도를 읽어서 논리적 속도 단위로 반환합니다. Source 매개 변수(媒介變數)에 따라서 Command 속도 혹은 Feedback 속도 중 해당하는 속도에 대해서 반환 대상이 결정됩니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Channel : 축(채널) 번호. 축번호는 상수값으로 3 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.

▶ Source : 속도 반환대상이 되는 카운터 번호. 이 값은 다음의 2 가지 값중의 하나이어야 합니다.

Value	Meaning
0 또는 cmxNT_COMM	Command Counter
1 또는 cmxNT_FEED	Feedback Counter

▶ Speed : 전달된 변수를 통해 지정한 Source 의 속도를 읽어서 논리적 속도 단위로 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxStGetTorque

- 논리적(論理的) 토크 반환

INFORMATION

Motion Status

VC++/VB

BCB/Delphi/.NET

Level 7

위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxStReadMotionState ([in] VT_I4 BoardID, [in] VT_I4 Channel, [out] VT_PI4 Torque)

DESCRIPTION

이 함수는 토크를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Channel : 축(채널) 번호. 축번호는 상수값으로 3 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.

▶ Torque: 모터 토크값을 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxStReadMioStatuses

- 서보 관련 상태 반환(狀態返還)

INFORMATION

Motion Status
VC++/VB
BCB/Delphi/.NET
Level 7
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxStReadMioStatuses ([in] VT_I4 BoardID, [in] VT_I4 Channel, [out] VT_PI4 MioStates)

DESCRIPTION

이 함수는 현재 서보와 관련된 여러가지 MIO 상태를 반환합니다. 각 비트별로 할당된 MIO 의 상태를 표시하므로 사용자는 비트마스크를 수행하여 원하는 I/O 의 상태를 확인(確認)하여야 합니다. 범용적인 모션 응용프로그램에서는 MIO(Machine I/O) 상태를 표현하기 위한 용도로 본 함수의 사용 빈도가 매우 높습니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Channel : 축(채널) 번호. 축번호는 상수값으로 3 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.

▶ MioStates : Machine I/O 상태로 구조체(TMechnicalIO)로 되어 있습니다. 각 상태값의 비트 수와 의미는 아래와 같습니다.

Name	Bit 수	Meaning
INP	1	Inposition 상태(1=ON)
Reserved _00	1	예약 공간 #0
HC	1	원점 복귀 완료 상태(1=완료)
TL	1	Torque limite status(1=토크값이 제한치가 되었음)
WARNING	1	서보(AMP)의 경고상태(1=ON)
ALARM	1	서보(AMP)의 알람상태(1=ON)
SVRDY	1	서보(AMP)의 Servo On 준비 상태(1=ON)
SVON	1	Servo On 상태(1=ON)
ELN	1	-EL 센서 신호 상태(1=ON)
ELP	1	+EL 센서 신호 상태(1=ON)
ORG	1	원점 센서 신호 상태(1=ON)
EX_IN1	1	외부 입력 활성상태(1=ON)

EX_IN2	1	외부 입력 활성상태(1=ON)
EX_IN3	1	외부 입력 활성상태(1=ON)
EX_IN4	1	외부 입력 활성상태(1=ON)
EMP_STP	1	외부 입력 신호의 안전에 대한 활성상태(1=ON)
Reserved_01	4	예약 공간 #1
Reserved_02	8	예약 공간 #2
ZSPD	1	Zero Speed(1=정지 상태)
DEN	1	위치 결정 완료 상태(1=ON)
BREAKON	1	정지 상태(1=ON)
ZPOINT	1	Z 상 검출위치 통과 상태

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

EXAMPLE

C/C++

```
// BoardID 는 0 으로 선언되었다고 가정함

long dwMioState = 0;
cmxStReadMioStatuses(BoardID, 3, &dwMioState);

// dwMioState 의 값을 오른쪽으로 쉬프트 연산(Shift Operation) 하여, 해당 상태 값을 얻습니다.
BOOL RDY_State = (dwMioState >> cmIOST_RDY) & 0x1;
BOOL ALM_State = (dwMioState >> cmIOST_ALM) & 0x1;
BOOL ELP_State = (dwMioState >> cmIOST_ELP) & 0x1;
BOOL ELN_State = (dwMioState >> cmIOST_ELN) & 0x1;
…….
…….
```

Delphi

```
//BoardID 는 0 으로 선언되었다고 가정함

Var
    dwMioState : LongInt;
    RDY_State : Boolean;
    ALM_State : Boolean;
    ELP_State : Boolean;
    ELN_State : Boolean;
```

```
begin
	cmxStReadMioStatuses(BoardID, 3,@dwMioState);

	// dwMioState 의 값을 오른쪽으로 쉬프트 연산(Shift Operation) 하여, 해당 상태 값을
얻습니다.
	RDY_State := Boolean((dwMioState shr cmIOST_RDY) and $1);
	ALM_State := Boolean((dwMioState shr cmIOST_ALM) and $1);
	ELP_State := Boolean((dwMioState shr cmIOST_ELP) and $1);
	ELN_State := Boolean((dwMioState shr cmIOST_ELN) and $1);
	……
	……

end;
```

NAME

cmxStSxReadMotionState

- 단축 이송 속도 상태 반환(狀態返還)

INFORMATION

Motion Status
VC++/VB
BCB/Delphi/.NET
Level 7
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxStSxReadMotionState ([in] VT_I4 BoardID, [in] VT_I4 Channel, [out] VT_PI4 MotStates)

DESCRIPTION

이 함수는 단축, 보간 이송시 해당 축의 현재 속도 상태를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Channel : 축(채널) 번호. 축번호는 상수값으로 3 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.

▶ MotStates : 해당 축의 현재 속도 상태를 확인할 수 있습니다.

Value	Meaning
0 또는 cmxMST_STOP	정지상태
1 또는 cmxMST_IN_ACC	가속 상태
2 또는 cmxMST_IN_WORKSPD	정속 상태
3 또는 cmxMST_IN_DEC	감속 상태
4 또는 cmxMST_IN_INISPD	초기 속도 상태
5 또는 cmxMST_IN_WAIT	대기 상태

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxStIxReadMotionState

- 보간 이송 속도 상태 반환(狀態返還)

INFORMATION

Motion Status
VC++/VB
BCB/Delphi/.NET
Level 7
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxStIxReadMotionState ([in] VT_I4 BoardID, [in] VT_I4 MapIndex, [out] VT_PI4 MotStates)

DESCRIPTION

이 함수는 보간 이송시 보간 모드에 따른 보간 이송의 속도 상태를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ MapIndex: 맵번호(Map index), 이 맵번호를 사용하기전에 먼저 cmxIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑되어 있어야 합니다.

▶ MotStates : 보간 모드에 따른 해당 맵의 현재 속도 상태를 확인할 수 있습니다.

- 직선 보간 : 마스터 축의 속도 상태를 반환합니다.
- 원호보간 : 벡터 속도에 대한 속도 상태를 반환합니다.
- 헬리컬 보간 : Z 축 방향의 속도 상태를 반환합니다.
- 스플라인 보간 : 매개변수의 속도상태를 반환합니다.
- 속도 상태값은 아래와 같습니다.

Value	Meaning
0 또는 cmxMST_STOP	정지상태
1 또는 cmxMST_IN_ACC	가속 상태
2 또는 cmxMST_IN_WORKSPD	정속 상태
3 또는 cmxMST_IN_DEC	감속 상태
4 또는 cmxMST_IN_INISPD	초기 속도 상태
5 또는 cmxMST_IN_WAIT	대기 상태

RETURN VALUE

Value	Meaning

음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxStGetMotionMode

- 구동 모션 종류 확인

INFORMATION

Motion Status
VC++/VB
BCB/Delphi/.NET
Level 7
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxStGetMotionMode ([in] VT_I4 BoardID, [in] VT_I4 Channel, [out] VT_PI4 Mode)

DESCRIPTION

이 함수는 대상 채널의 모션 종류를 확인하는 함수입니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Channel : 축(채널) 번호. 축번호는 상수값으로 3 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.

▶ Mode: 각 모션의 값을 반환합니다.

Value	Meaning
0	직선보간
1	원호보간
2	헬리컬보간
3	스플라인보간
4	단축, 다축 구동

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxStSxGetLastError

- 단축 구동시 마지막에 발생한 에러코드 확인

INFORMATION

- Motion Status
- VC++/VB
- BCB/Delphi/.NET
- Level 7
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxStSxGetLastError ([in] VT_I4 BoardID, [in] VT_I4 Channel , [out] VT_PI4 LastError)

DESCRIPTION

단축 구동시에 마지막으로 발생한 에러코드를 확인합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Channel : 축(채널) 번호. 축번호는 상수값으로 3 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.

▶ LastError : 마지막으로 발생한 에러코드 값.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxStIxGetLastError

- 보간 구동시 마지막에 발생한 에러코드 확인

INFORMATION

Motion Status

VC++/VB

BCB/Delphi/.NET

Level 7

위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxStIxGetLastError ([in] VT_I4 BoardID, [in] VT_I4 MapIndex , [out] VT_PI4 LastError)

DESCRIPTION

보간 구동시에 마지막으로 발생한 에러코드를 확인합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ MapIndex : 맵번호(Map index), 이 번호 범위는 0 ~ 31 입니다.

▶ LastError : 마지막으로 발생한 에러코드 값.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

NAME

cmxStSetMultiRevCnt

cmxStGetMultiRevCnt

- 절대 위치(絶對 圍置) 지정 회전 수 설정/반환

INFORMATION

Motion Status

VC++/VB

BCB/Delphi/.NET

Level 7

위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxStSetMultiRevCnt ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 MultiRevCnt)

❒ VT_I4 cmxctGetMultiRevCnt ([in] VT_I4 BoardID, [in] VT_I4 Axis, [out] VT_PI4 pMultiRevCnt)

DESCRIPTION

대상(對象) 모션 채널의 절대 위치를 지정하기 위한 회전수를 설정/반환합니다. 단, 이 때 지정하는 단위(單位)는 펄스 수를 기준으로 한 회전 수 입니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ Axis : 축(채널) 번호. 축번호는 상수값으로 3 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.

▶ MultiRevCnt: 절대 위치 지정에 필요한 회전 수 설정값.
▶ pMultiRevCnt: 절대 위치 지정에 필요한 회전 수.반환값

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

SEE ALSO

cmxStSetOneRevPos, cmxStGetOneRevPos

NAME

cmxStSetOneRevPos

cmxStGetOneRevPos

- 절대 위치(絶對 圍置) 지정 단회전 내 펄스 수 설정/반환

INFORMATION

Motion Status

VC++/VB

BCB/Delphi/.NET

Level 7

위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxStSetOneRevPos ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 OneRevPos)

❒ VT_I4 cmxStGetOneRevPos ([in] VT_I4 BoardID, [in] VT_I4 Axis, [out] VT_PI4 pOneRevPos)

DESCRIPTION

대상(對象) 모션 채널의 절대 위치를 지정하기 위한 단 회전 내 펄스 수를 설정/반환합니다. 단, 이 때 지정하는 단위(單位)는 펄스 수 입니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ Axis : 축(채널) 번호. 축번호는 상수값으로 3 번째 채널을 기준 채널로 임의의 채널을 설정할 수 있습니다.

▶ OneRevPos: 절대 위치 지정에 필요한 한 회전 내의 펄스 수 설정값.
▶ pOneRevPos: 절대 위치 지정에 필요한 한 회전 내의 펄스 수 반환값.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

SEE ALSO

cmxStSetMultiRevCnt, cmxStGetMultiRevCnt

Chapter

11

Digital I/O Control

메모 [.1]:

모션제어의 상태를 확인(確認)하는 역할은 고객(顧客)여러분들께서 개발하시는 응용프로그램의 필수 요건 중에 하나입니다. ComiRTEX 에서는 매우 자세하고 효율적인 상태 관리 매커니즘을 가지고 있습니다. ㈜커미조아의 많은 장점 중에 하나인 전문적인 모션 정보 제공 인터페이스를 통해 보다 자세하고 신속한 모션 프로그램의 상태를 구현하시기 바랍니다.

이 단원에서는 모션제어의 상태(狀態) 감시에 관련된 함수들에 대하여 설명합니다. 상태(狀態) 감시 함수들은 모션의 상태를 감시하는데 필요한 함수들을 그룹화한 것입니다. 상태(狀態) 감시에는 모션의 속도, 위치 등을 확인(確認)하는 것을 포함하며 이외에도 현재 모션이 진행되고 있는지를 확인(確認)하고, 현재 진행되고 있는 모션의 단계 및 각 I/O 상태들을 점검하는 기능도 포함합니다.

11 디지털 입출력 편

이 단원에서는 디지털 입력/출력 모듈에 관련된 함수를 소개합니다. NETIO 디지털 입출력 모듈은 3 가지로 구분되며 각각의 특징 및 사용 가능 함수는 다음과 같습니다.

가. NETIO 디지털 입출력 모듈 특징

모듈 명	디지털 입력/출력 구성	채널 수
ceD16CM	디지털 입력 / 출력 채널 설정	16 Channel Ditital Input/Output
ceDI32N	디지털 입력 채널	32 Channel Digital Input
ceDO32N	디지털 출력 채널	32 Channel Digital Output

나. 디지털 입출력 함수 지원 모듈

함수 명	사용 가능 모듈
cmxDioSetIomode	ceD16CM
cmxDioGetIomode	
cmxDioSetIomodeMulti	
cmxDioGetIomodeMulti	
cmxDioSetLogic	ceD16CM ceDI32N ceDO32N
cmxDioGetLogic	
cmxDioSetLogicMulti	
cmxDioGetLogicMulti	
cmxDioGetOne	
cmxDioPutOne	
cmxDioGetMulti	
cmxDioPutMulti	
cmxDoSetLogic	ceD16CM (디지털 출력 채널 대상) ceDO32N
cmxDoGetLogic	
cmxDoSetLogicMulti	
cmxDoGetLogicMulti	
cmxDoPutOne	
cmxDoGetOne	
cmxDoPutMulti	
cmxDoGetMulti	
cmxDiSetLogic	ceD16CM (디지털 입력 채널 대상) ceDI32N
cmxDiGetLogic	
cmxDiSetLogicMulti	
cmxDiGetLogicMulti	
cmxDiGetOne	
cmxDiGetMulti	

Dio 함수에서 디지털 입력 채널과 출력 채널 번호의 관계는 다음과 같습니다.

1. 디지털 입력 채널과 디지털 출력 채널은 서로 구분되지 않으며 통합하여 채널 번호가 부여됩니다.

2. 슬레이브 내의 모듈 번호를 기준으로 디지털 입력 채널과 디지털 출력 채널 번호가 증가됩니다.

Di/Do 함수에서 디지털 입력 채널과 출력 채널 번호의 관계는 다음과 같습니다.

1. 디지털 입력 채널과 디지털 출력 채널의 번호는 서로 상관관계를 가지지 않습니다. 독립적인 채널 번호로 순서화 됩니다.
2. 슬레이브 내의 모션 모듈 번호를 기준으로 디지털 입력 채널과 디지털 출력 채널 번호가 각각 증가됩니다.

11.1 함수 요약

디지털 입출력 기능과 관련된 함수 는 다음의 표와 같습니다.

Summary of Functions
❒ VT_I4 cmxDiSetLogic([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [in] VT_I4 Logic) 대상 디지털 입력(Digital Input) 채널의 논리(Logic)를 설정합니다.
❒ VT_I4 cmxDiGetLogic ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [out] VT_PI4 InputLogic) 대상 디지털 입력(Digital Input) 채널의 논리(Logic) 설정상태를 반환합니다.
❒ VT_I4 cmxDiSetLogicMulti ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 IniChan, [in] VT_I4 NumChan, [in] VT_I4 LogicMask) 다중 디지털 입력(Digital Input) 채널의 논리(Logic)를 설정합니다.
❒ VT_I4 cmxDiGetLoticMulti ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 IniChan, [in] VT_I4 NumChan, [out] VT_PI4 LogicMask) 다중 디지털 입력(Digital Input) 채널의 논리(Logic) 설정상태를 반환합니다.
❒ VT_I4 cmxDiGetOne ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [out] VT_PI4 State) 대상 디지털 입력(Digital Input) 채널에 대해 입력 상태를 확인합니다.
❒ VT_I4 cmxDiGetMulti ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [out] VT_PI4 InputState) 다중 디지털 입력(Digital Input) 채널에 대해 입력 상태를 확인합니다.
❒ VT_I4 cmxDoSetLogic([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [in] VT_I4 Logic) 대상 디지털 출력(Digital Input/Output) 채널의 논리(Logic)를 설정합니다.
❒ VT_I4 cmxDoGetLogic ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [out] VT_PI4 InputLogic) 대상 디지털 출력(Digital Input/Output) 채널의 논리(Logic) 설정상태를 반환합니다.
❒ VT_I4 cmxDoSetLogicMulti ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 IniChan, [in] VT_I4 NumChan, [in] VT_I4 LogicMask) 다중 디지털 출력(Digital Input/Output) 채널의 논리(Logic)를 설정합니다.
❒ VT_I4 cmxDoGetLoticMulti ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 IniChan, [in] VT_I4 NumChan, [out] VT_PI4 LogicMask) 다중 디지털 출력(Digital Input/Output) 채널의 논리(Logic) 설정상태를 반환합니다.
❒ VT_I4 cmxDoPutOne ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [in] VT_I4 OutState) 단일 디지털 출력(Digital Output) 채널에 대해 출력을 발생합니다.
❒ VT_I4 cmxDoGetOne ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [out] VT_PI4 OutState) 단일 디지털 출력(Digital Output) 채널에 대해 출력 상태를 확인합니다.
❒ VT_I4 cmxDoPutMulti ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [in] VT_I4 OutStates) 다중 디지털 출력(Digital Output) 채널에 대해 출력을 발생합니다.
❒ VT_I4 cmxDoGetMulti ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [out] VT_PI4 OutStates) 다중 디지털 출력(Digital Output) 채널에 대해 출력 상태를 확인합니다.
❒ VT_I4 cmxDioSetIomode([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [in] VT_I4 InOutMode) 대상 디지털 입출력(Digital Input/Output) 채널의 용도(Mode)를 설정합니다.
❒ VT_I4 cmxDioGetIomode ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [out] VT_PI4 InOutMode) 대상 디지털 입출력(Digital Input/Output) 채널의 용도(Mode) 설정상태를 반환합니다.

❒ VT_I4 cmxDioSetIomodeMulti ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 IniChan, [in] VT_I4 NumChan, [in] VT_I4 InOutModeMask) 다중 디지털 입출력(Digital Input/Output) 채널의 용도(Mode)를 설정합니다.
❒ VT_I4 cmxDioGetIomodeMulti ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 IniChan, [in] VT_I4 NumChan, [out] VT_PI4 InOutModeMask) 다중 디지털 입출력(Digital Input/Output) 채널의 용도(Mode) 설정상태를 반환합니다.
❒ VT_I4 cmxDioSetLogic([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [in] VT_I4 Logic) 대상 디지털 입출력(Digital Input/Output) 채널의 논리(Logic)를 설정합니다.
❒ VT_I4 cmxDioGetLogic ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [out] VT_PI4 InputLogic) 대상 디지털 입출력(Digital Input/Output) 채널의 논리(Logic) 설정상태를 반환합니다.
❒ VT_I4 cmxDioSetLogicMulti ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 IniChan, [in] VT_I4 NumChan, [in] VT_I4 LogicMask) 다중 디지털 입출력(Digital Input/Output) 채널의 논리(Logic)를 설정합니다.
❒ VT_I4 cmxDioGetLoticMulti ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 IniChan, [in] VT_I4 NumChan, [out] VT_PI4 LogicMask) 다중 디지털 입출력(Digital Input/Output) 채널의 논리(Logic) 설정상태를 반환합니다.
❒ VT_I4 cmxDioGetOne ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [out] VT_PI4 State) 대상 디지털 입출력(Digital Input/Output) 채널의 디지털 입력 또는 출력 상태를 반환합니다.
❒ VT_I4 cmxDioPutOne ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [in] VT_I4 State) 대상 디지털 출력(Digital Output) 채널을 통해 디지털 출력을 발생시킵니다.
❒ VT_I4 cmxDioGetMulti ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 IniChan, [in] VT_I4 NumChan, [out] VT_PI4 States) 다중 디지털 입출력(Digital Input/Output) 채널의 디지털 입력 또는 출력 상태를 반환합니다.
❒ VT_I4 cmxDioPutMulti ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 IniChan, [in] VT_I4 NumChan, [in] VT_I4 States) 다중 디지털 출력(Digital Output) 채널을 통해 디지털 출력을 발생시킵니다.

NAME

cmxDiSetLogic / cmxDiSetLogic

- 대상 디지털 입력 채널의 논리(Logic) 설정 및 설정 상태 반환

INFORMATION

- DIO Control
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

- VT_I4 cmxDiSetLogic ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [in] VT_I4 Logic)
- VT_I4 cmxDiGetLogic ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [out] VT_PI4 InputLogic)

DESCRIPTION

cmxDiSetLogic()/cmxDiGetLogic() 함수는 대상 디지털 입력 채널의 논리(Logic) 설정 혹은 설정상태를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(Axis 번호는 3 부터 시작합니다.)

▶ Channel : 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 채널수 - 1 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ Logic : 대상 디지털 I/O 채널의 논리(Logic)를 설정 혹은 설정상태를 반환합니다.

Value	Meaning
0 (cmxLOGIC_A)	A 접점 방식 => 평상시 Open, 감지되면 Close 되는 스위치 방식
1 (cmxLOGIC_B)	B 접점 방식 => 평상시 Close, 감지되면 Open 되는 스위치 방식

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

/* 1 번 채널의 로직을 ‘B 접점’ 으로 설정하고 설정된 상태를 확인합니다. */
//BoardID 는 0 으로 가정함

#define CHANNEL 1

long lGetDioLogic =0; // 설정한 디지털 입출력 논리를 반환하기 위한 변수

// 1 번 채널의 로직을 B 접점으로 설정합니다.
if(cmxDiSetLogic (BoardID,  3, CHANNEL, cmxLOGIC_B ) != ERR_NONE )
{
   OutputDebugString("cmxDiSetLogic function Fail");
}

// 1 번 채널의 설정된 로직 상태를 확인합니다.
if(cmxDiGetLogic (BoardID, 3, CHANNEL, &lGetDioLogic) != ERR_NONE )
{
   OutputDebugString ( " cmxDiGetLogic function Fail" );
}
```

NAME

cmxDiSetLogicMulti / cmxDiGetLoticMulti

- 다중(Multi) 디지털 입력 채널의 논리(Logic) 설정 및 설정 상태 반환

INFORMATION

- DIO Control
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxDiSetLogicMulti

([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 IniChan, [in] VT_I4 NumChan, [in] VT_I4 LogicMask)

❒ VT_I4 cmxDiGetLoticMulti

([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 IniChan, [in] VT_I4 NumChan, [out] VT_PI4 LogicMask)

DESCRIPTION

cmxDiSetLogicMulti()/cmxDiGetLoticMulti() 함수는 다중(Multi) 디지털 입출력 채널의 논리(Logic) 설정 및 설정상태를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호. 축 번호는 3 부터 시작합니다.

▶ IniChan : 시작 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 채널수 - 1 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ NumChan : 시작 채널로부터 몇 개의 채널의 상태를 확인할 것인지에 대한 값을 전달합니다. (최대 32 개 채널까지 설정 가능합니다.)

▶ LogicMask : 이 매개변수를 통하여 다중(Multi) 디지털 I/O 채널의 논리(Logic) 설정 및 설정상태를 반환합니다. (32 비트, BIT0 ~ BIT31)

Value	Meaning
0 (cmxLOGIC_A)	A 접점 방식 => 평상시 Open, 감지되면 Close 되는 스위치 방식
1 (cmxLOGIC_B)	B 접점 방식 => 평상시 Close, 감지되면 Open 되는 스위치 방식

RETURN VALUE

Value	Meaning

음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

/* 0~3 번 채널을 디지털 출력 모드로 설정하고 설정된 상태를 확인합니다. */
//BoardID 는 0 으로 가정함

#define INI_CH  0
#define NUM_CH 4

long lGetDioLogicMulti=0;       //사용자가 지정한 채널의 디지털 I/O 논리 상태를 반환하기 위한 변수.

// 0~3 번 채널의 로직을 B 접점으로 설정합니다.
if(cmxDiSetLogicMulti (BoardID, 3,  INI_CH, NUM_CH, 0xF ) != ERR_NONE )
{
   OutputDebugString ( " cmxDiSetLogicMulti function Fail" );
}

// 0~3 번 채널의 설정된 로직 상태를 확인합니다.
if(cmxDiGetLoticMulti (BoardID,  3, INI_CH, NUM_CH, &lGetDioLogicMulti ) != ERR_NONE )
{
   OutputDebugString ( " cmxDiGetLoticMulti function Fail" );
}
```

NAME

cmxDiGetOne

- 단일 채널에 대해 입력 상태 반환

INFORMATION

DIO Control
VC++ (6, 7, 8)/VB
BCB/Delphi
Level 1
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxDiGetOne ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [out] VT_PI4 State)

DESCRIPTION

단일 채널에 대한 디지털 입력 상태를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(Axis 번호는 3 번부터 시작합니다.).

▶ Channel **:** 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 채널수 - 1 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ InputState **:** 해당 채널의 디지털 입력(Digital Input) 상태를 확인합니다.

Value	Meaning
0 (cmxFALSE)	OFF
1 (cmxTRUE)	ON

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

#define DI_CH  0

Long BoardID = 0;
long nDiState  = 0;

// 0 번 채널이 ON 되어 있는 상황이라고 가정한 후,
// 0 번 채널의 디지털 입력 상태를 nDiState 변수로 반환합니다.
if(cmxDiGetOne (BoardID, 3, DI_CH, & nDiState ) != ERR_NONE )
{
   OutputDebugString ( "cmxDiGetOne function Fail" );
}

// cmxDiGetOne ()함수에 의해 읽어 본 0 번 채널의 입력 상태 값을
// 실제 입력 상태 값(1)과 비교합니다.
if( nDiState != 1 )
{
   OutputDebugString ( " cmxDiGetOne function don't read Input State" );
}
```

NAME

cmxDiGetMulti

- 모션 디지털 입력이 지원되는 제품에서 다중 채널에 대해 입력 상태 반환

INFORMATION

- DIO Control
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxDiGetMulti

([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [out] VT_PI4 InputState)

DESCRIPTION

다중 채널에 대해 디지털 입력 상태를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(Axis 번호는 3 번부터 시작합니다.).

▶ IniChannel : 시작 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 채널수 - 1 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ NumChannels : 시작 채널로부터 몇 개의 채널의 상태를 확인할 것인지에 대한 값을 전달합니다. (최대 32 개 채널까지 설정 가능합니다.)

▶ InputState : 이 매개변수를 통하여 다중 디지털 입력 채널의 입력 상태를 반환합니다. (32 비트, BIT0 ~ BIT31).

Value	Meaning
0 (cmxFALSE)	OFF
1 (cmxTRUE)	ON

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

#define INI_CH  0
#define NUM_CH  4

Long BoardID = 0;
long nDiMultiStates=0;

// 0~3 번 채널의 입력 상태가 ON 이라고 가정한 후,
// 0~3 번 채널의 디지털 입력 상태를 nDiMultiStates 변수로 반환합니다.
if(cmxDiGetMulti (BoardID, 3, INI_CH, NUM_CH, & nDiMultiStates ) != ERR_NONE )
{
   OutputDebugString ( " cmxDiGetMulti function Fail" );
}

// cmxDiGetMulti ()함수에 의해 읽어 본 0 번 채널의 입력 상태 값을
// 실제 입력 상태 값(0xF)과 비교합니다.
if( nDiMultiStates!= 0xF )
{
   OutputDebugString ( " cmxDiGetMulti function don't read Input States" );
}
```

NAME

cmxDoSetLogic / cmxDoGetLogic

- 대상 디지털 입출력 채널의 논리(Logic) 설정 및 설정 상태 반환

INFORMATION

- DIO Control
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

- VT_I4 cmxDoSetLogic([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [in] VT_I4 Logic)
- VT_I4 cmxDoGetLogic ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [out] VT_PI4 InputLogic)

DESCRIPTION

cmxDoSetLogic()/cmxDoGetLogic() 함수는 대상 디지털 출력 채널의 논리(Logic) 설정 혹은 설정상태를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(Axis 번호는 3 부터 시작합니다.).

▶ Channel : 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 채널수 - 1 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ Logic : 대상 디지털 I/O 채널의 논리(Logic)를 설정 혹은 설정상태를 반환합니다.

Value	Meaning
0 (cmxLOGIC_A)	A 접점 방식 => 평상시 Open, 감지되면 Close 되는 스위치 방식
1 (cmxLOGIC_B)	B 접점 방식 => 평상시 Close, 감지되면 Open 되는 스위치 방식

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

/* 1 번 채널의 로직을 ‘B 접점’ 으로 설정하고 설정된 상태를 확인합니다. */

#define CHANNEL 1

Long BoardID = 0;
long lGetDioLogic =0; // 설정한 디지털 입출력 논리를 반환하기 위한 변수

// 1 번 채널의 로직을 B 접점으로 설정합니다.
if(cmxDoSetLogic (BoardID,  3, CHANNEL, cmxLOGIC_B ) != ERR_NONE )
{
   OutputDebugString("cmxDoSetLogic function Fail");
}

// 1 번 채널의 설정된 로직 상태를 확인합니다.
if(cmxDoGetLogic (BoardID, 3, CHANNEL, &lGetDioLogic) != ERR_NONE )
{
   OutputDebugString ( " cmxDoGetLogic function Fail" );
}
```

NAME

cmxDoSetLogicMulti / cmxDoGetLoticMulti

- 다중(Multi) 디지털 입출력 채널의 논리(Logic) 설정 및 설정 상태 반환

INFORMATION

- DIO Control
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxDoSetLogicMulti
([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 IniChan, [in] VT_I4 NumChan, [in] VT_I4 LogicMask)

❒ VT_I4 cmxDoGetLoticMulti
([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 IniChan, [in] VT_I4 NumChan, [out] VT_PI4 LogicMask)

DESCRIPTION

cmxDoSetLogicMulti()/cmxDoGetLoticMulti() 함수는 다중(Multi) 디지털 출력 채널의 논리(Logic) 설정 및 설정상태를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(Axis 는 3 부터 시작합니다.).

▶ IniChan : 시작 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 채널수 - 1 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ NumChan : 시작 채널로부터 몇 개의 채널의 상태를 확인할 것인지에 대한 값을 전달합니다. (최대 32 개 채널까지 설정 가능합니다.)

▶ LogicMask : 이 매개변수를 통하여 다중(Multi) 디지털 I/O 채널의 논리(Logic) 설정 및 설정상태를 반환합니다. (32 비트, BIT0 ~ BIT31)

Value	Meaning
0 (cmxLOGIC_A)	A 접점 방식 => 평상시 Open, 감지되면 Close 되는 스위치 방식
1 (cmxLOGIC_B)	B 접점 방식 => 평상시 Close, 감지되면 Open 되는 스위치 방식

RETURN VALUE

Value	Meaning

음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

/* 3~6 번 채널을 디지털 출력 모드로 설정하고 설정된 상태를 확인합니다. */

#define INI_CH  0
#define NUM_CH 4

Long BoardID = 0;
long lGetDioLogicMulti=0;       //사용자가 지정한 채널의 디지털 I/O 논리 상태를 반환하기 위한 변수.

// 0~3 번 채널의 로직을 B 접점으로 설정합니다.
if(cmxDoSetLogicMulti (BoardID,  3, INI_CH, NUM_CH, 0xF ) != ERR_NONE )
{
   OutputDebugString ( " cmxDoSetLogicMulti function Fail" );
}

// 0~3 번 채널의 설정된 로직 상태를 확인합니다.
if(cmxDoGetLoticMulti (BoardID, 3, INI_CH, NUM_CH, &lGetDioLogicMulti ) != ERR_NONE )
{
   OutputDebugString ( " cmxDoGetLoticMulti function Fail" );
}
```

NAME

cmxDoPutOne / cmxDoGetOne

- 단일 채널에 대해 출력 발생 및 출력 상태 확인

INFORMATION

- DIO Control
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

- ❒ VT_I4 cmxDoPutOne ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [in] VT_I4 OutState)
- ❒ VT_I4 cmxDoGetOne ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [out] VT_PI4 OutState)

DESCRIPTION

모션 디지털 출력이 지원되는 제품에서 cmxDoPutOne() 함수는 단일 채널에 대한 디지털 출력 상태를 발생시키며, cmxDoGetOne () 함수는 단일 채널에 대한 디지털 출력 상태를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(Axis 는 3 번 부터 시작합니다.)

▶ Channel : 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 채널수 - 1 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ OutState : 단일 채널에 대한 디지털 출력 상태를 발생 혹은 출력 상태를 반환합니다

Value	Meaning
0 (cmxFALSE)	OFF
1 (cmxTRUE)	ON

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

#define DI_CH  0
#define DI_ON  1

Long BoardID = 0;
// 0 번 채널의 디지털 출력 상태를 1(ON)으로 설정.
if (cmxDoPutOne (BoardID, 3, DI_CH, DI_ON ) != ERR_NONE )
{
   OutputDebugString ( " cmxDoPutOne function Fail" );
}

// 0 번 채널의 디지털 출력 상태를 반환합니다.
Long nGetDoOneState = 0;
if(cmxDoGetOne (BoardID, 3, DI_CH, &nGetDoOneState ) != ERR_NONE )
{
   OutputDebugString ( " cmxDoGetOne function Fail" );
}

// cmxDoGetOne () 함수에 의해 읽어 본 0 번 채널의 입력 상태 값을
// 실제 설정한 값(1)과 비교합니다.
if ( nGetDoOneState != 1 )
{
   OutputDebugString ( " cmxDoGetOne function don't set Correctly" );
}
```

NAME

cmxDoPutMulti / cmxDoGetMulti

- 모션 디지털 출력이 지원되는 제품에서 다중 디지털 출력 채널의 출력 발생 및 출력 상태 확인

INFORMATION

- DIO Control
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxDoPutMulti

([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [in] VT_I4 OutStates)

❒ VT_I4 cmxDoGetMulti

([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [out] VT_PI4 OutStates)

DESCRIPTION

모션 디지털 출력이 지원되는 제품에서 cmxDoPutOne() 함수는 다중 채널 대한 디지털 출력 상태를 발생시키며, cmxDoGetMulti() 함수는 다중 채널에 대한 디지털 출력 상태를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(Axis 는 3 번부터 시작합니다.)

▶ IniChannel : 시작 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 채널수 - 1 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ NumChannels : 시작 채널로부터 몇 개의 채널의 상태를 확인할 것인지에 대한 값을 전달합니다. (최대 32 개 채널까지 설정 가능합니다.)

▶ OutStates : 다중 채널에 대한 출력 상태를 발생 혹은 출력 상태를 반환합니다. 이 출력 상태는 BitMask(비트 마스크) 로 설정되며, 설정된 비트가 1 일 경우 디지털 출력(Digital Out)이 발생합니다.

Value	Meaning
0 (cmxFALSE)	OFF
1 (cmxTRUE)	ON

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

#define INI_CH  0
#define NUM_CH  4

Long BoardID = 0;
long nGetDoOutStates =0;

// 0~3 번 채널의 디지털 출력 상태를 1(ON)으로 설정합니다.
if (cmxDoPutOne (BoardID, 3, INI_CH, NUM_CH, 0xF ) != ERR_NONE )
{
   OutputDebugString ( " cmxDoPutOne function Fail" );
}

// 0~3 번 채널의 디지털 출력 상태를 nGetDoOutStates 변수로 반환합니다.
if (cmxDoGetMulti (BoardID, 3, INI_CH, NUM_CH, &nDoOutStates ) != ERR_NONE )
{
   OutputDebugString ( " cmxDoGetMulti function Fail" );
}

// cmxDoGetMulti ()함수에 의해 읽어 본 0~3 번 채널의 출력 채널 상태 값을
// 실제 설정한 값(0xF)과 비교합니다.
if ( nDoOutStates != 0xF )
{
   OutputDebugString ( " cmxDoGetMulti function don't set Correctly" );
}
```

NAME

cmxDioSetIomode / cmxDioGetIomode

- 대상 디지털 입출력 채널의 용도(Mode) 설정 및 설정 상태 반환

INFORMATION

- DIO Control
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxDioSetIomode([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [in] VT_I4 InOutMode)

❒ VT_I4 cmxDioGetIomode ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [out] VT_PI4 InOutMode)

DESCRIPTION

cmxDioSetIomode()/cmxDioGetIomode() 함수는 지정 채널의 디지털 입출력 채널의 용도(Mode)를 설정 혹은 설정 상태를 반환 합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(Axis 는 3 번부터 시작합니다.)

▶ Channel : 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 채널수 - 1 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ InOutMode : 디지털 입출력 채널의 용도(Mode)를 설정 혹은 설정상태를 반환합니다.

Value	Meaning
0 (cmxFALSE)	Input Mode
1 (cmxTRUE)	Output Mode

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

REFERENCE

본 함수는 ceD16CM 모듈 전용 함수입니다.

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

/* ceD16CM 모듈의 1 번 채널을 디지털 출력 모드로 설정하고 설정된 상태를 확인합니다. */

#define CHANNEL 1

Long BoardID = 0;
long lGetDioMode=0;

// 1 번 채널의 모드를 1(Output Mode)로 설정합니다.
if(cmxDioSetIomode (BoardID, 3, CHANNEL, 1) != ERR_NONE )
{
   OutputDebugString ( " cmxDioSetIomode function Fail" );
}

// 1 번 채널의 모드를 lGetDioMode 변수로 반환합니다.
if(cmxDioGetIomode (BoardID, 3, CHANNEL, &lGetDioMode) != ERR_NONE )
{
   OutputDebugString ( " cmxDioGetIomode function Fail" );
}
```

NAME

cmxDioSetIomodeMulti /

cmxDioGetIomodeMulti

- 다중(Multi) 디지털 입출력 채널의 용도(Mode) 설정 및 설정 상태 반환

INFORMATION

DIO Control

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxDioSetIomodeMulti

([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 IniChan, [in] VT_I4 NumChan, [in] VT_I4 InOutModeMask)

❒ VT_I4 cmxDioGetIomodeMulti

([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 IniChan, [in] VT_I4 NumChan, [out] VT_PI4 InOutModeMask)

DESCRIPTION

cmxDioSetIomodeMulti()/cmxDioGetIomodeMulti() 함수는 다중(Multi) 디지털 입출력 채널의 용도(Mode)를 설정 혹은 설정 상태를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(Axis 는 3 번부터 시작합니다.)

▶ IniChan : 시작 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 채널수 - 1 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ NumChan : 시작 채널로부터 몇 개의 채널의 상태를 확인할 것인지에 대한 값을 전달합니다. (최대 32 개 채널까지 설정 가능합니다.)

▶ InOutModeMask : 이 매개변수를 통하여 다중(Multi) 디지털 I/O 채널의 용도(Mode) 설정 및 설정상태를 반환합니다. (32 비트, BIT0 ~ BIT31)

Value	Meaning
0 (cmxFALSE)	Input Mode
1 (cmxTRUE)	Output Mode

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

REFERENCE

본 함수는 ceD16CM 모듈 전용 함수입니다.

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

/* ceD16CM 모듈의 0~3 번 채널을 디지털 출력 모드로 설정하고 설정된 상태를 확인합니다. */

#define INI_CH  0
#define NUM_CH 4

Long BoardID = 0;
long lGetDioModeMulti = 0;

// 0~3 번 채널을 디지털 출력 모드로 설정합니다.
if(cmxDioSetIomodeMulti (BoardID, 3,  INI_CH, CH_NUM, 0xF ) != ERR_NONE )
{
   OutputDebugString ( " cmxDioSetIomodeMulti function Fail" );
}

// 0~3 번 채널의 입출력 모드 설정 상태를 확인합니다.
if(cmxDioGetIomodeMulti (BoardID, 3, INI_CH, CH_NUM, &lGetDioModeMulti ) != ERR_NONE )
{
   OutputDebugString ( " cmxDioGetIomodeMulti function Fail" );
}
```

NAME

cmxDioSetLogic / cmxDioGetLogic

- 대상 디지털 입출력 채널의 논리(Logic) 설정 및 설정 상태 반환

INFORMATION

- DIO Control
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxDioSetLogic([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [in] VT_I4 Logic)

❒ VT_I4 cmxDioGetLogic ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [out] VT_PI4 InputLogic)

DESCRIPTION

cmxDioSetLogic()/cmxDioGetLogic() 함수는 대상 디지털 입출력 채널의 논리(Logic) 설정 혹은 설정상태를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(Axis 는 3 번부터 시작합니다.)

▶ Channel : 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 채널수 - 1 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ Logic : 대상 디지털 I/O 채널의 논리(Logic)를 설정 혹은 설정상태를 반환합니다.

Value	Meaning
0 (cmxLOGIC_A)	A 접점 방식 => 평상시 Open, 감지되면 Close 되는 스위치 방식
1 (cmxLOGIC_B)	B 접점 방식 => 평상시 Close, 감지되면 Open 되는 스위치 방식

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

/* 1 번 채널의 로직을 ‘B 접점’ 으로 설정하고 설정된 상태를 확인합니다. */

#define CHANNEL 1

Long BoardID = 0;
long lGetDioLogic =0; // 설정한 디지털 입출력 논리를 반환하기 위한 변수

// 1 번 채널의 로직을 B 접점으로 설정합니다.
if(cmxDioSetLogic (BoardID, 3, CHANNEL, cmxLOGIC_B ) != ERR_NONE )
{
   OutputDebugString("cmxDioSetLogic function Fail");
}

// 1 번 채널의 설정된 로직 상태를 확인합니다.
if(cmxDioGetLogic (BoardID, 3, CHANNEL, &lGetDioLogic) != ERR_NONE )
{
   OutputDebugString ( " cmxDioGetLogic function Fail" );
}
```

NAME

cmxDioSetLogicMulti / cmxDioGetLoticMulti

- 다중(Multi) 디지털 입출력 채널의 논리(Logic) 설정 및 설정 상태 반환

INFORMATION

- DIO Control
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxDioSetLogicMulti

([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 IniChan, [in] VT_I4 NumChan, [in] VT_I4 LogicMask)

❒ VT_I4 cmxDioGetLoticMulti

([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 IniChan, [in] VT_I4 NumChan, [out] VT_PI4 LogicMask)

DESCRIPTION

cmxDioSetLogicMulti()/cmxDioGetLoticMulti() 함수는 다중(Multi) 디지털 입출력 채널의 논리(Logic) 설정 및 설정상태를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(Axis 는 3 번부터 시작합니다.)

▶ IniChan : 시작 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 채널수 - 1 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ NumChan : 시작 채널로부터 몇 개의 채널의 상태를 확인할 것인지에 대한 값을 전달합니다. (최대 32 개 채널까지 설정 가능합니다.)

▶ LogicMask : 이 매개변수를 통하여 다중(Multi) 디지털 I/O 채널의 논리(Logic) 설정 및 설정상태를 반환합니다. (32 비트, BIT0 ~ BIT31)

Value	Meaning
0 (cmxLOGIC_A)	A 접점 방식 => 평상시 Open, 감지되면 Close 되는 스위치 방식
1 (cmxLOGIC_B)	B 접점 방식 => 평상시 Close, 감지되면 Open 되는 스위치 방식

RETURN VALUE

Value	Meaning

음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

/* 0~3 번 채널을 디지털 출력 모드로 설정하고 설정된 상태를 확인합니다. */

#define INI_CH  0
#define NUM_CH 4

Long BoardID = 0;
long lGetDioLogicMulti=0;       //사용자가 지정한 채널의 디지털 I/O 논리 상태를 반환하기 위한 변수.

// 0~3 번 채널의 로직을 B 접점으로 설정합니다.
if(cmxDioSetLogicMulti (BoardID, 3, INI_CH, NUM_CH, 0xF ) != ERR_NONE )
{
   OutputDebugString ( " cmxDioSetLogicMulti function Fail" );
}

// 0~3 번 채널의 설정된 로직 상태를 확인합니다.
if(cmxDioGetLoticMulti (BoardID, 3, INI_CH, NUM_CH, &lGetDioLogicMulti ) != ERR_NONE )
{
   OutputDebugString ( " cmxDioGetLoticMulti function Fail" );
}
```

NAME

cmxDioGetOne / cmxDioPutOne

대상 디지털 입출력 채널의 입력/출력 상태를 반환 및 설정

INFORMATION

- DIO Control
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

- ❒ VT_I4 cmxDioGetOne ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [out] VT_PI4 State)
- ❒ VT_I4 cmxDioPutOne ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [in] VT_I4 State)

DESCRIPTION

cmxDioGetOne() 함수는 대상 디지털 채널의 용도(Mode)에 따라 입력 또는 출력 상태를 반환합니다.
cmxDioPutOne() 함수는 대상 디지털 채널의 용도(Mode)에 따라 입력 또는 출력 상태를 설정합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(Axis 는 3 번부터 시작합니다.)

▶ Channel **:** 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 채널수 - 1 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ State **:** 대상 채널의 용도(Mode)에 따라 cmxDioGetOne() 함수에서는 입력 또는 출력 상태를 반환하며, cmxDioPutOne() 함수에서는 입력 또는 출력 상태를 설정합니다.

Value	Meaning
0 (cmxFALSE)	OFF
1 (cmxTRUE)	ON

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

/* 1 번 채널의 출력 상태를 ‘ON’ 으로 설정하고 설정된 상태를 확인합니다. */

#define CHANNEL 1
#define CH_ON 1

Long BoardID = 0;
long lGetDioOneState = 1;

// 0 번 채널을 ON 으로 설정합니다.
if (cmxDioGetOne (BoardID, 3, CHANNEL, CH_ON ) != ERR_NONE )
{
   OutputDebugString ( " cmxDioGetOne function Fail" );
}

// 0 번 채널의 출력 상태를 확인합니다.
if(cmxDioPutOne (BoardID, 3, CHANNEL, &lGetDioOneState ) != ERR_NONE )
{
   OutputDebugString ( " cmxDioPutOne function Fail" );
}
```

NAME

cmxDioGetMulti / cmxDioPutMulti

- 다중 디지털 입출력 채널의 입력/출력 상태 반환 및 설정

INFORMATION

- DIO Control
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxDioGetMulti
([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 IniChan, [in] VT_I4 NumChan, [out] VT_PI4 States)
❒ VT_I4 cmxDioPutMulti
([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 IniChan, [in] VT_I4 NumChan, [in] VT_I4 States)

DESCRIPTION

cmxDioGetMulti() 함수는 다중 디지털 채널의 용도(Mode)에 따라 입력 또는 출력 상태를 반환합니다.
cmxDioPutMulti() 함수는 다중 디지털 채널의 용도(Mode)에 따라 입력 또는 출력 상태를 설정합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(Axis 는 3 번부터 시작합니다.)

▶ IniChan : 시작 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합
채널수 - 1 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ NumChan : 시작 채널로부터 몇 개의 채널의 상태를 확인할 것인지에 대한 값을 전달합니다. (최대 32 개 채널까지 설정 가능합니다.)

▶ States : 대상 채널의 용도(Mode)에 따라 cmxDioGetMulti () 함수에서는 입력 또는 출력 상태를 반환하며, cmxDioPutMulti () 함수에서는 입력 또는 출력 상태를 설정합니다.

Value	Meaning
0 (cmxFALSE)	OFF
1 (cmxTRUE)	ON

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다

0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

/* 0~3 번 채널의 출력 상태를 ‘ON’ 으로 설정하고 설정된 상태를 확인합니다. */

#define INI_CH  0
#define NUM_CH 4

Long BoardID = 0;
long lGetDioMultiState =0xF; //사용자가 지정한 범위 채널의 상태를 확인하기 위한 변수

// 0~3 번 채널을 ON 으로 설정합니다.
if (cmxDioGetMulti (BoardID, 3, INI_CH, NUM_CH, 0xF ) != ERR_NONE )
{
   OutputDebugString ( " cmxDioGetMulti function Fail" );
}

// 0~3 번 채널의 출력 상태를 확인합니다.
if(cmxDioPutMulti (BoardID, 3, INI_CH, NUM_CH, & lGetDioMultiState ) != ERR_NONE )
{
   OutputDebugString ( " cmxDioPutMulti function Fail" );
}
```

Chapter

12

Analog I/O Control

메모 [.2]:

모션제어의 상태를 확인(確認)하는 역할은 고객(顧客)여러분들께서 개발하시는 응용프로그램의 필수 요건 중에 하나입니다. ComiRTEX 에서는 매우 자세하고 효율적인 상태 관리 매커니즘을 가지고 있습니다. ㈜커미조아의 많은 장점 중에 하나인 전문적인 모션 정보 제공 인터페이스를 통해 보다 자세하고 신속한 모션 프로그램의 상태를 구현하시기 바랍니다.

이 단원에서는 모션제어의 상태(狀態) 감시에 관련된 함수들에 대하여 설명합니다. 상태(狀態) 감시 함수들은 모션의 상태를 감시하는데 필요한 함수들을 그룹화한 것입니다. 상태(狀態) 감시에는 모션의 속도, 위치 등을 확인(確認)하는 것을 포함하며 이외에도 현재 모션이 진행되고 있는지를 확인(確認)하고, 현재 진행되고 있는 모션의 단계 및 각 I/O 상태들을 점검하는 기능도 포함합니다.

12 아날로그 입출력 편

이 단원에서는 아날로그 입력(Analog Input), 아날로그 출력(Analog Output) 에 관련된 함수들을 소개합니다.

12.1 아날로그 입력 (Analog Input)

일반적인 A/D 는 아날로그 (Analog) 신호를 입력 받아 디지털(Digital) 값으로 변환 해주는 기능입니다. COMIZOA 의 AI 모듈은 아날로그 신호를 입력 받아 A/D 한 값을 Volt 및 Current, Digit 값으로 변환하여 반환합니다.

12.1.1 함수 요약

아날로그 입력(A/D) 기능에 관련된 함수는 다음과 같습니다.

Summary of Functions
❒ VT_I4 cmxAiSetVoltRangeMode ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [in] VT_I4 RangeMode) 아날로그 입력에 대한 전압 범위를 지정된 모드를 통해 설정합니다.
❒ VT_I4 cmxAiGetVoltRangeMode ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [out] VT_PI4 RangeMode) 아날로그 입력에 대해 설정된 전압 범위에 해당하는 모드를 반환합니다.
❒ VT_I4 cmxAiGetRangeDigit ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [out] VT_PI4 DigitMin, [out] VT_PI4 DigitMax) 아날로그 입력에 대해 설정된 입력 범위를 Digit 값으로 반환합니다.
❒ VT_I4 cmxAiGetDigit ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [out] VT_PI4 Digit) 대상 아날로그 입력 채널에 대하여 AD 결과를 Digit 값으로 반환합니다.
❒ VT_I4 cmxAiGetVolt ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [out] VT_PR8 fVolt) 대상 아날로그 입력 채널에 대하여 AD 결과를 전압(Volt) 값으로 반환합니다.
❒ VT_I4 cmxAiGetCurrent ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [out] VT_PR8 fCurrent) 대상 아날로그 입력 채널에 대하여 AD 결과를 전류(Current)값으로 반환합니다.

12.1.2 함수 설명

NAME

cmxAiSetVoltRangeMode /

cmxAiGetVoltRangeMode

- 대상 아날로그 입력 채널의 전압(Volt) 범위 설정 및 설정 상태 반환

INFORMATION

Analog Input Control

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 1

위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxAiSetVoltRangeMode

([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [in] VT_I4 RangeMode)

❒ VT_I4 cmxAiGetVoltRangeMode

([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [out] VT_PI4 RangeMode)

DESCRIPTION

cmxAiSetVoltRangeMode 함수는 아날로그 입력 채널의 전압 범위를 지정된 모드를 통해 설정합니다.
cmxAiGetVoltRangeMode 함수는 설정된 전압 범위에 해당하는 모드를 반환 합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(Axis 는 3 번부터 시작합니다.)

▶ Channel : 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 채널수 - 1 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ RangeMode : 전압 범위 설정 모드를 설정 혹은 설정상태를 반환합니다. 전압 범위 설정 모드는 다음과 같습니다.

Value	Meaning
0[Default]	-10V ~ 10V
1	-5V ~ 5V
2	-2.5V ~ 2.5V
3	0V ~ 10V (0 ~ 20mA : 전류 모드 사용 시)
4	0V ~ 5V

5	1V ~ 5V
6	1V ~ 5V (4 ~ 20 mA : 전류 모드 사용 시)

0~6 번 모드 모두 전압 모드로 사용할 수 있으나, 전류 모드 사용 시 3 번 또는 6 번 모드로 설정 하셔야 합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

#define AI_CHANNEL  5

Long BoardID = 0;
long nVoltRangeMode = 0;

// 5 번 채널의 전압 범위 모드를 3( 0V ~ +10V )로 설정합니다.
if(cmxAiSetVoltRangeMode (BoardID, 3,  AI_CHANNEL, 3 ) != ERR_NONE )
{
   OutputDebugString ( " cmxAiSetVoltRangeMode function Fail" );
}

// 5 번 채널에 설정된 전압 범위에 해당하는 모드를 반환합니다.
if(cmxAiGetVoltRangeMode (BoardID, 3, AI_CHANNEL, &nVoltRangeMode) != ERR_NONE )
{
   OutputDebugString ( " cmxAiGetVoltRangeMode function Fail" );
}
```

NAME

cmxAiGetRangeDigit

- 대상 아날로그 입력 채널의 설정된 입력 범위를 Digit 값으로 반환

INFORMATION

- Analog Input Control
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxAiGetRangeDigit

([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [out] VT_PI4 DigitMin, [out] VT_PI4 DigitMax)

DESCRIPTION

대상 아날로그 입력 채널에 설정된 입력 Range (전압 범위)를 Digit 값으로 반환합니다. 입력 Range 설정은 cmxAiGetRangeDigit () 함수를 통해 설정합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(Axis 는 3 번부터 시작합니다.)

▶ Channel : 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 채널수 - 1 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ DigitMin : 입력 Range 의 최소 입력 Digit 값.

▶ DigitMax : 입력 Range 의 최대 입력 Digit 값.

입력되는 Digit 값의 범위는 어떻게 되나요?

ceAI08A 모듈의 Digit 범위는 0 ~ 8192 (13 Bit) 입니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다

0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

#define AI_CHANNEL  5

Long BoardID = 0;
long nDigitMin = 0, nDigitMax = 0;

// 5 번 채널의 설정된 입력 Range 를 Digit 값으로 반환합니다.
if(cmxAiGetRangeDigit (BoardID, 3, AI_CHANNEL, &nDigitMin, &nDigitMax ) != ERR_NONE )
{
   OutputDebugString ( " cmxAiGetRangeDigit function Fail" );
}
```

NAME

cmxAiGetDigit

- 대상 아날로그 입력 채널의 입력 Digit 값을 반환

INFORMATION

Analog Input Control
VC++ (6, 7, 8)/VB
BCB/Delphi
Level 1
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxAiGetDigit ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [out] VT_PI4 Digit)

DESCRIPTION

대상 아날로그 입력 채널에 대하여 A/D 변환을 수행하고, 그 값을 Digit 값으로 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

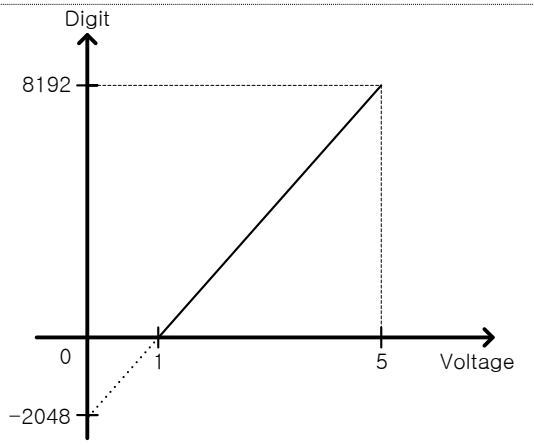
▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(Axis 는 3 번부터 시작합니다.)

▶ Channel : 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 채널수 - 1 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ Digit : A/D 결과값을 Digit 값으로 반환합니다. 13Bit Straight Binery (0~8192)로 구성되어 있습니다.

단, 전압 입력 범위(Range) 모드가 5 번 모드(1~5V), 6 번 모드(1~5V, 4~20mA) 모드인 경우는 입력 전압이 1V 보다 낮은 경우, Digit 값으로 구분해서 표시하기 위해 Digit 값과 Voltage 값의 관계 직선에서 0~1V 에 해당되는 음의 Digit 값을 선형적으로 계산해서 반환합니다.



RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

#define AI_CHANNEL  5

Long BoardID = 0;
long nDigit = 0;

// 5 번 채널의 A/D 결과값을 Digit 값으로 반환합니다.
if(cmxAiGetDigit (BoardID, 3, AI_CHANNEL, &nDigit ) != ERR_NONE )
{
   OutputDebugString ( " cmxAiGetDigit function Fail" );
}
```

NAME

cmxAiGetVolt

- 대상 아날로그 입력 채널의 입력 전압(Volt) 값을 반환

INFORMATION

Analog Input Control
VC++ (6, 7, 8)/VB
BCB/Delphi
Level 1
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxAiGetVolt ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [out] VT_PR8 fVolt)

DESCRIPTION

대상 아날로그 입력 채널에 대하여 A/D 변환을 수행하고, 그 값을 전압(Volt) 값으로 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(Axis 는 3 번부터 시작합니다.)

▶ Channel **:** 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 채널수 - 1 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ fVolt **:** A/D 결과값을 전압(Volt) 값으로 반환.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

#define AI_CHANNEL  5

double fVolt = 0;
long BoardID = 0;
```

```
// 5 번 채널의 A/D 결과값을 Volt 값으로 반환합니다.
if(cmxAiGetVolt (BoardID, 3, AI_CHANNEL, &fVolt ) != ERR_NONE )
{
   OutputDebugString ( " cmxAiGetVolt function Fail" );
}
```

NAME

cmxAiGetCurrent

- 대상 아날로그 입력 채널의 입력 전류(Current) 값을 반환

INFORMATION

- Analog Input Control
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxAiGetCurrent ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [out] VT_PR8 fCurrent)

DESCRIPTION

대상 아날로그 입력 채널에 대하여 A/D 변환을 수행하고, 그 값을 전류(Current) 값으로 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(Axis 는 3 번부터 시작합니다.)

▶ Channel : 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 채널수 - 1 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ fCurrent : A/D 결과값을 전류(Current) 값으로 반환합니다. 단, 전압 범위 설정 모드를 3 번 (0~20mA) 또는 6 번 (4~20mA) 로 설정해야 합니다.

Value	Meaning
-1	전류 입력 모드(3 번, 6 번 모드) 가 아닌 경우의 전달되는 값.
유효한 값	3 번 모드 : 0~20 (mA) 6 번 모드 : 4~20 (mA)

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

#define AI_CHANNEL  5

Long BoardID = 0;
double fCurrent = 0.0f;

// 5 번 채널의 A/D 결과값을 전류(Current) 값으로 반환합니다.
if(cmxAiGetCurrent (BoardID, 3, AI_CHANNEL, &fCurrent ) != ERR_NONE )
{
   OutputDebugString ( " cmxAiGetCurrent function Fail" );
}
```

12.2 아날로그 출력 (Analog Output)

일반적인 D/A 는 사용자가 지정한 전압(Volt) 값을 출력하는 기능입니다. COMIZOA 의 D/A 모듈은 아날로그 출력 채널을 통해 사용자가 지정한 Volt 및 Current, Digit 값을 출력합니다.

12.2.1 함수 요약

아날로그 출력(D/A) 기능에 관련된 함수는 다음과 같습니다.

Summary of Functions
❒ VT_I4 cmxAoOutDigit ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [in] VT_I4 Digit) 대상 아날로그 출력 채널을 통해 Digit 값을 출력합니다.
❒ VT_I4 cmxAoOutVolt ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [in] VT_R8 fVolt) 대상 아날로그 출력 채널을 통해 전압(Volt) 값을 출력합니다.
❒ VT_I4 cmxAoOutCurrent ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [in] VT_R8 fCurrent) 대상 아날로그 출력 채널을 통해 전류(Current) 값을 출력합니다.

12.2.2 함수 설명

NAME

cmxAoOutDigit

- 대상 아날로그 출력 채널을 통해 Digit 값 출력

INFORMATION

- Analog Output Control
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxAoOutDigit ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [in] VT_I4 Digit)

DESCRIPTION

대상 아날로그 출력 채널에 대하여 지정한 Digit 값을 출력합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(Axis 는 3 번부터 시작합니다.).

▶ Channel **:** 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 채널수 - 1 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ Digit **:** Digit 값으로 아날로그 출력. D/A 데이터는 부호를 갖는 16Bit Data (-32768 ~ 32767)로 출력할 수 있습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

#define AO_CHANNEL  1

Long BoardID = 0;
long nDigit = 32767;

// 1 번 아날로그 출력 채널에 10V 를 출력 합니다.
if(cmxAoOutDigit (BoardID, 3, AO_CHANNEL, nDigit ) != ERR_NONE )
{
   OutputDebugString ( " cmxAoOutDigit function Fail" );
}
```

NAME

cmxAoOutVolt

- 대상 아날로그 출력 채널을 통해 전압(Volt) 값 출력

INFORMATION

- Analog Output Control
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxAoOutVolt ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [in] VT_R8 fVolt)

DESCRIPTION

대상 아날로그 출력 채널에 대하여 지정한 전압(Volt) 값을 출력합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID: 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(Axis 는 3 번부터 시작합니다.)

▶ Channel **:** 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 채널수 - 1 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ fVolt **:** 전압(Volt) 값으로 아날로그 출력. 출력 가능한 전압 값의 범위는 -10V ~ 10V 입니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

#define AO_CHANNEL 1

Long BoardID = 0;
double fVolt = 10.0;

// 1 번 아날로그 출력 채널에 10V 를 출력 합니다.
if(cmxAoOutVolt (BoardID, 3, AO_CHANNEL, fVolt ) != ERR_NONE )
{
```

```
    OutputDebugString ( " cmxAoOutVolt function Fail" );
}
```

NAME

cmxAoOutCurrent

- 대상 아날로그 출력 채널을 통해 전류(Current) 값 출력

INFORMATION

- Analog Output Control
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxAoOutCurrent ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Channel, [in] VT_R8 fCurrent)

DESCRIPTION

대상 아날로그 출력 채널에 대하여 지정한 전류(Current) 값을 출력합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ Axis: Axis 번호(Axis 는 3 번부터 시작합니다.)

▶ Channel **:** 채널 번호. 통합 채널로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 채널수 - 1 이하의 값을 채널 번호로 설정할 수 있습니다.

▶ fCurrent **:** 전류(Current) 값으로 아날로그 출력. 출력 가능한 전류 값의 범위는 4mA ~ 20mA 입니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

#define AO_CHANNEL  1

double fCurrent = 20.0;
long BoardID = 0;

// 1 번 아날로그 출력 채널에 20mA 를 출력 합니다.
if(cmxAoOutCurrent (BoardID, 3, AO_CHANNEL, fCurrent ) != ERR_NONE )
```

```
{
   OutputDebugString ( " cmxAoOutCurrent function Fail" );
}
```

Chapter

13

Gnenral Pulse Motion Functions

ComiRTEX 는 기본 함수 이외에도 기타 기본 함수들을 지원합니다. 이 함수의 집합에서 가장 두드러진 기능으로서는 GUI 환경에서 모션의 전체 혹은 부분적인 설정(設定)을 완료하여, 그 설정(設定)을 실제 고객(顧客)님의 응용프로그램에서 사용할 수 있는 기능입니다. 이외에도 다양한 편의 기능과 우수한 기능들이 고객(顧客)님들을 기다리고 있습니다.

본 단원에서는 Pulse Motion 의 General 함수(函數)들을 소개합니다. 이 함수(函數)들은 그 기능에 있어 모든 내용이 반드시 습득(習得) 될 필요는 없지만, 모션 제어를 위해 꼭 필요한 내용의 Pulse Motion General 함수 편으로 구성되어 있습니다. 기본적인 서보 ON 출력 신호제어 기능이 제공됩니다.

13.1 함수 요약

"General Pulse Motion Functions" 에 관련된 함수들은 다음과 같습니다.

Summary of Functions
❒ VT_I4 cmxPmGnSetServoOn ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 IsOn) 해당 모션 축의 서보 드라이버에 대해 SERVO-ON 신호 출력을 제어합니다.
❒ VT_I4 cmxPmGnGetServoOn ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 IsOn) 해당 모션 축의 서보 드라이버에 대해 SERVO-ON 신호 출력 상태를 반환합니다.
❒ VT_I4 cmxPmGnAlarmReset ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 IsReset) 해당 모션 축의 알람 리셋(Alarm Reset) 신호 출력을 제어합니다.

13.2 함수 설명

NAME

cmxPmGnSetServoOn /

cmxPmGnGetServoOn

- 서보 동작 Turn On/Off 신호 출력 제어 및 SERVO-ON 신호의 출력 상태 반환

INFORMATION

General Motion Function

VC++ (6, 7, 8)/VB/ BCB/Delphi

Level 2

다소 위험

서보 동작 Turn ON/OFF 상태를 결정합니다. 서보 동작 시에 반드시 주의 환경을 점검하고, 안전 사항에 유의해야 합니다.

SYNOPSIS

❒ VT_I4 cmxPmGnSetServoOn ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 IsOn)

❒ VT_I4 cmxPmGnGetServoOn ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 IsOn)

DESCRIPTION

cmxPmGnSetServoOn() 함수는 지정한 축의 SERVO-ON 신호 출력을 제어합니다. 서보 드라이버를 사용하실 때는 외부에서 스위치를 이용하여 서보 드라이버의 ON/OFF 를 제어할 수 있도록 하는데, 이를 SERVO-ON 신호라 합니다. 이 함수는 SERVO-ON 신호의 ON/OFF 를 제어하는 함수입니다.

cmxPmGnGetServoOn() 함수는 현재 SERVO-ON 신호의 출력 상태를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***IsOn*** : SERVO-ON 신호의 출력 상태를 설정 혹은 반환 합니다.

Value	Meaning
0 (cmxFALSE)	SERVO-OFF

1 (cmxTRUE)	SERVO-ON

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러 처리' 편을 참고합니다.
0 (ERR_NONE)	수행 성공.

REFERENCE

□ 서보드라이버 Servo-ON 신호의 출력 로직은 cmxPmCfgSetMioProperty(Axis#, ccmxMPID_SVON_LOGIC, cmxLOGIC_A) 또는 cmxPmCfgSetMioProperty(Axis#, ccmxMPID_SVON_LOGIC, ccmxLOGIC_B) 명령을 통해 설정할 수 있습니다.

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

long nChannel = 0;          //0 번 축을 선택합니다.
long nServoState = 0;

/* SERVO ON/OFF 상태를 얻어와 SERVO-OFF 상태면 SERVO-ON 출력 신호를 ON 시킵니다. */
cmxPmGnGetServoOn(BoardID, 3, nChannel, &nServoState );

if( nServoState != cmxFALSE )
{
  cmxPmGnSetServoOn(BoardID, nAxis, cmxTRUE );
}
```

NAME

cmxPmGnAlarmReset

– 알람 리셋(Alarm Reset) 신호 출력 제어

INFORMATION

- General Motion Function
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 1
- ☺ 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxPmGnAlarmReset ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 IsReset)

DESCRIPTION

해당 모션 축의 알람 리셋 신호 출력을 제어하는 함수입니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID***: 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId***: 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 채널 번호. 통합 축으로 관리되는 채널 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***IsReset***: 알람 리셋 신호 출력 여부를 결정합니다.

Value	Meaning
0 (cmxFALSE)	알람 리셋(Alarm Reset) 신호를 출력 하지 않음.
1 (cmxTRUE)	알람 리셋(Alarm Reset) 신호 출력

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러 처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"
```

```
long nChannel = 0;          //제어할 축을 0 번 축으로 설정합니다.

/*일정 시간 동안 알람 리셋 신호를 내보내고, 다시 클리어 시켜줍니다.*/
cmxPmGnAlarmReset(BoardID, 3, nChannel, cmxTRUE );
Sleep(50);
cmxPmGnAlarmReset(BoardID, 3, nChannel, cmxFALSE );
```

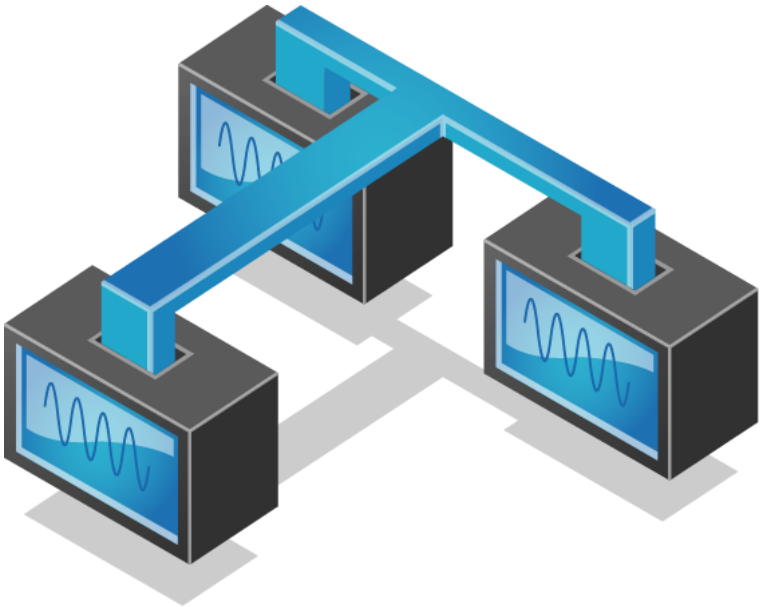
Chapter

14

Pulse Motion Environment Configuration Functions

모션 환경설정(環境設定)의 다양한 함수는 결과적으로 ㈜ 커미조아의 모션 보드의 환경 설정이 얼마나 자세하고 명확한지를 판단하게 합니다. 기본적인 모터 드라이브를 위한 입력 펄스 신호 설정부터, 모션 주변신호까지 다양하게 지원하는 환경 설정함수는 다양한 주변신호에 대한 강력한 노이즈의 필터 기능까지 지원하고 있습니다.

이 단원에서는 모션컨트롤러의 환경(環境)을 설정(設定)하는 함수(函數)들을 소개합니다. 필요에 따라 런타임시에 동적으로 환경(環境)을 변경해야할 필요가 있을 수 있으며 이러한 때에는 본 단원에서 소개하는 함수(函數)들을 이용하여 동적(動的)으로 환경(環境)을 변경할 수 있습니다.



14 모션 환경 설정 함수

14.1 함수 요약

"모션 환경설정" 에 관련된 함수들은 다음과 같습니다.

Summary of Functions
❒ VT_I4 cmxPmCfgSetMioProperty ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 PropId, [in] VT_I4 PropVal) 모션 입출력 신호의 환경설정을 구성합니다.
❒ VT_I4 cmxPmCfgGetMioProperty ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 PropId, [out] VT_PI4 PropVal) 모션 입출력 신호의 환경설정 값을 반환합니다.
❒ VT_I4 cmxPmCfgSetFilter ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 IsEnable) 각종 I/O(Input/Output) 신호에 대해 노이즈 대응 필터(Filter) 기능을 설정합니다.
❒ VT_I4 cmxPmCfgGetFilter ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 IsEnabled) 각종 I/O(Input/Output) 신호에 대해 노이즈 대응 필터(Filter) 기능의 설정 상태를 반환합니다.
❒ VT_I4 cmxPmCfgSetFilterAB ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 Target, [in] VT_I4 IsEnable) EA/EB 또는 PA/PB 신호 입력 회로에 노이즈 대응 필터(Filter) 기능을 설정합니다.
❒ VT_I4 cmxPmCfgGetFilterAB ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 Target, [out] VT_PI4 IsEnabled) EA/EB 또는 PA/PB 신호 입력 회로에 노이즈 대응 필터(Filter) 기능의 설정 상태를 반환합니다.
❒ VT_I4 cmxPmCfgSetInMode ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 InputMode, [in] VT_I4 IsReverse) 인코더의 펄스(Feedback Pulse) 신호의 입력 모드를 설정합니다.
❒ VT_I4 cmxPmCfgGetInMode ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 InputMode, [out] VT_PI4 IsReverse) 인코더의 펄스(Feedback Pulse) 신호의 입력 모드 설정 상태를 반환합니다.
❒ VT_I4 cmxPmCfgSetOutMode ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 OutputMode) 지령 펄스(Command Pulse) 신호의 출력 모드를 설정합니다.
❒ VT_I4 cmxPmCfgGetOutMode ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 OutputMode) 지령 펄스(Command Pulse) 신호의 출력 모드 설정 상태를 반환합니다.
❒ VT_I4 cmxPmCfgSetCtrlMode ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 CtrlMode) 이송 목표 좌표의 기준을 커맨드 위치로 할지 피드백 위치로 할지를 설정하는 함수입니다.
❒ VT_I4 cmxPmCfgGetCtrlMode ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 CtrlMode) 이송 목표 좌표의 기준 설정 상태를 반환합니다.
❒ VT_I4 cmxPmCfgSetInOutRatio ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_R8 Ratio) 입력 펄스(Feedback Pulse)와 출력 펄스(Command Pulse)의 분해능 비율(Resolution Ratio) 을 설정합니다.
❒ VT_I4 cmxPmCfgGetInOutRatio ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PR8 Ratio) 입력 펄스(Feedback Pulse)와 출력 펄스(Command Pulse)의 분해능 비율(Resolution Ratio) 설정 상태를 반환합니다.
❒ VT_I4 cmxPmCfgSetUnitDist ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_R8 UnitDist) 지정된 모션 축에 대한 논리적 거리 단위를 설정합니다.
❒ VT_I4 cmxPmCfgGetUnitDist ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PR8 UnitDist) 지정된 모션 축에 대한 논리적 거리 단위의 설정 상태를 반환합니다.
❒ VT_I4 cmxPmCfgSetUnitSpeed ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_R8 UnitSpeed) 지정된 모션 축에 대한 논리적 속도 단위를 설정합니다.
❒ VT_I4 cmxPmCfgGetUnitSpeed ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PR8 UnitSpeed) 지정된 모션 축에 대한 논리적 속도 단위의 설정 상태를 반환합니다.
❒ VT_I4 cmxPmCfgSetSpeedRange ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 MaxPPS) 지정된 모션 축에 대해 모션 속도를 제한하고, 제한 범위를 설정합니다.
❒ VT_I4 cmxPmCfgGetSpeedRange ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 MaxPPS) 지정된 모션 축에 대해 모션 속도 제한 범위 설정 상태를 반환합니다.
❒ VT_I4 cmxPmCfgSetSpeedPattern ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 SpeedMode, [in] VT_R8 WorkSpeed, [in] VT_R8 Accel, [in] VT_R8 Decel) 모션 이송의 전역 기준속도를 설정합니다. 이 속도의 비율을 통해 모션 이송의 실제 속도를 설정할 수 있습니다.

❒ VT_I4 cmxPmCfgGetSpeedPattern ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 SpeedMode, [out] VT_PR8 WorkSpeed, [out] VT_PR8 Accel, [out] VT_PR8 Decel) 모션 이송의 전역 기준속도를 반환합니다. 반환된 이 속도의 비율을 통해 모션 이송의 실제 속도가 설정 됩니다.
❒ VT_I4 cmxPmCfgSetSpeedPattern_T ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 SpeedMode, [in] VT_R8 WorkSpeed, [in] VT_R8 AccelTime, [in] VT_R8 Decel Time) 모션 이송의 전역 기준속도를 설정합니다. 이 속도의 비율을 통해 모션 이송의 실제 속도를 설정할 수 있습니다.
❒ VT_I4 cmxPmCfgGetSpeedPattern_T ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 SpeedMode, [out] VT_PR8 WorkSpeed, [out] VT_PR8 AccelTime, [out] VT_PR8 Decel Time) 모션 이송의 전역 기준속도를 반환합니다. 반환된 이 속도의 비율을 통해 모션 이송의 실제 속도가 설정 됩니다.
❒ VT_I4 cmxPmCfgSetSoftLimit ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 IsEnable, [in] VT_R8 LimitN, [in] VT_R8 LimitP) 모션의 소프트웨어적인 이송제한범위를 설정하여 이송 범위를 제한합니다.
❒ VT_I4 cmxPmCfgGetSoftLimit ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 IsEnable, [out] VT_PR8 LimitN, [out] VT_PR8 LimitP) 모션의 소프트웨어적인 이송제한범위에 대한 해당 설정을 반환합니다.
❒ VT_I4 cmxPmCfgSetRingCntr ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 TargCntr, [in] VT_I4 IsEnable, [in] VT_R8 CntMax) 지정된 모션 축에 대해 링카운터(Rign-Counter) 기능을 설정합니다.
❒ VT_I4 cmxPmCfgGetRingCntr ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 TargCntr, [out] VT_PI4 IsEnable, [out] VT_PR8 CntMax) 지정된 모션 축에 대해 링카운터(Ring-Counter) 기능에 대한 해당 설정을 반환합니다.
❒ VT_I4 cmxPmCfgSetVelCorrRatio ([in] VT_I4 BoardID, [in] VT_I4 NodeId , [in] VT_I4 CorrRatio) 모션의 작업속도 보정(補正)시에 보정(補正)된 작업속도를 사출하는 비례(比例)값을 설정합니다.
❒ VT_I4 cmxPmCfgGetVelCorrRatio ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [out] VT_PI4 CorrRatio) 모션의 작업속도 보정(補正)시에 보정(補正)된 작업속도를 사출하는 비례(比例)값을 반환합니다.
❒ VT_I4 cmxPmCfgSetSeqMode ([in] VT_I4 BoardID, [in] VT_I4 NodeId , [in] VT_I4 SeqMode) 이전의 이송 작업이 완료되지 않은 축에 새로운 이송 명령이 하달되었을 때의 처리 정책을 설정합니다.
❒ VT_I4 cmxPmCfgGetSeqMode ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [out] VT_PI4 SeqMode) 이전의 이송 작업이 완료되지 않은 축에 새로운 이송 명령이 하달되었을 때의 처리 정책에 대한 설정을 반환합니다.

14.1 함수 설명

NAME

cmxPmCfgSetMioProperty/

cmxPmCfgGetMioProperty

모션 입출력 신호 환경설정 및 반환

INFORMATION

Environment

Configuration Functions

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 2

다소 주의

모션 입출력 신호의 전역 환경설정을 위한 함수 입니다. 반드시 숙지해 주시기 바랍니다.

SYNOPSIS

❒ VT_I4 cmxPmCfgSetMioProperty ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 PropId, [in] VT_I4 PropVal)

❒ VT_I4 cmxPmCfgGetMioProperty ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 PropId, [out] VT_PI4 PropVal)

DESCRIPTION

cmxPmCfgSetMioProperty() 함수는 각종 모션 입출력 신호에 대한 환경을 설정합니다. 이 함수는 다양한 I/O 신호의 환경을 설정하는데 공통적으로 사용하는 함수입니다. PropId 에 따라 어떠한 환경을 설정할 지를 결정하게 됩니다.

cmxPmCfgGetMioProperty() 함수는 각종 모션 입출력 신호에 대하여 현재 설정되어 있는 환경 설정 값을 반환합니다. 어떠한 I/O 의 환경 설정 값을 반환할 지는 PropId 에 따라 결정됩니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***PropId*** : 어떠한 환경에 대하여 설정 혹은 반환할 것인지를 설정합니다. 이 값에 대해서는 아래 표를 참조하시기 바랍니다.

▶ ***PropVal***: PropId 로 지정된 환경에 대해 설정 혹은 반환합니다.

PropId	Meaning & PropVal
ccmxMPID_ALM_LOGIC	Alarm(ALM) 신호의 입력 로직입니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (cmxLOGIC_A) : A 접점 방식 => 평상시 Open, 감지되면 Close 되는 스위치 방식 ▪ 1 (ccmxLOGIC_B) : B 접점 방식 => 평상시 Close, 감지되면 Open 되는 스위치 방식
ccmxMPID_ALM_MODE	Alarm 입력이 ON 되어 해당 축의 모션 작업이 정지할 때 정지되는 방식을 설정합니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 : 즉시 정지 ▪ 1 : 감속 후 정지
ccmxMPID_CMP_LOGIC	위치비교출력(CMP) 신호의 출력 방식을 설정합니다. ▪ 0 (Active low) : 평상시 HIGH 상태를 유지. 트리거 시점에서 LOW 로 떨어졌다가 다시 HIGH 로 올라갑니다. ▪ 1(Active high) : 평상시 LOW 상태를 유지. 트리거 시점에서 HIGH 로 올라갔다가 다시 LOW 로 떨어집니다.
ccmxMPID_DR_LOGIC	-/+ DR 신호의 입력 로직을 설정합니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (cmxLOGIC_A) : A 접점 방식 ▪ 1 (ccmxLOGIC_B) : B 접점 방식
ccmxMPID_EL_LOGIC	-/+EL 신호의 입력 로직을 설정합니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (cmxLOGIC_A) : A 접점 방식 ▪ 1 (ccmxLOGIC_B) : B 접점 방식
ccmxMPID_EL_MODE	-/+ EL 신호가 ON 되어 정지할 때 정지 방식을 설정합니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 : 즉시 정지 ▪ 1 : 감속 후 정지
ccmxMPID_ERC_LOGIC	ERC 신호의 출력 로직을 설정합니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (cmxLOGIC_A) : A 접점 방식 ▪ 1 (ccmxLOGIC_B) : B 접점 방식
ccmxMPID_ERC_OUT	원점복귀 완료 시에 ERC 출력 여부를 설정합니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (cmxFALSE) : 원점복귀 완료 시에 ERC 출력 없음. ▪ 1 (cmxTRUE) : 원점복귀 완료 시에 ERC 출력.
ccmxMPID_EZ_LOGIC	EZ(엔코더 Z 상) 신호의 입력 로직을 설정합니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (cmxLOGIC_A) : A 접점 방식 ▪ 1 (ccmxLOGIC_B) : B 접점 방식
ccmxMPID_INP_EN	INP 신호 입력 활성화 여부를 설정합니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (cmxFALSE) : INP 비활성 ▪ 1 (cmxTRUE) : INP 활성 => Command 출력이 완료되더라도 INP 신호가 ON 되기 전까지는 작업이 완료되지 않은 것으로 간주.
ccmxMPID_INP_LOGIC	INP(Inposition) 신호의 입력 로직을 설정합니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (cmxLOGIC_A) : A 접점 방식 ▪ 1 (ccmxLOGIC_B) : B 접점 방식

PropId	Meaning & PropVal
ccmxMPID_LTC_LOGIC	LTC(Latch) 신호의 입력 로직을 설정합니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (cmxLOGIC_A) : A 접점 방식 ▪ 1 (ccmxLOGIC_B) : B 접점 방식
ccmxMPID_LTC_LTC2SRC	두 번째 LATCH COUNTER 의 대상 카운터를 설정합니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 : Deviation counter value ▪ 1 : Preset speed of command pulse
ccmxMPID_ORG_LOGIC	ORG(원점 센서) 신호의 입력 로직을 설정합니다. 설정 및 반환값의 PropVal 은 다음과 같습니다. ▪ 0 (cmxLOGIC_A) : A 접점 방식 ▪ 1 (ccmxLOGIC_B) : B 접점 방식
ccmxMPID_SD_EN	SD(Start of Deceleration) 신호의 입력 상태를 설정합니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (cmxFALSE) : SD 입력을 비활성화 합니다. ▪ 1 (cmxTRUE) : SD 입력을 활성화 합니다. 활성화 되었을 때 SD 신호에 따른 동작 방식은 ccmxMPID_SD_LATCH 와 ccmxMPID_SD_MODE 설정 값에 의해 결정됩니다.
ccmxMPID_SD_LOGIC	SD(Start of Deceleration) 신호의 입력 로직을 설정합니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (cmxLOGIC_A) : A 접점 방식 ▪ 1 (cmxLOGIC_B) : B 접점 방식
ccmxMPID_SD_LATCH	SD(Start of Deceleration) 신호를 래치(Latch)할 것인지에 대한 속성값입니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. • 0 (cmxFALSE) : SD 가 ON 되어 감속중이거나 초기속도로 운전 중일 때 SD 신호가 다시 OFF 상태로 변경되면 작업속도까지 다시 가속됩니다. • 1 (cmxTRUE) : SD 가 ON 상태에서 OFF 상태로 바뀌어도 작업속도로 가속하지 않습니다.
ccmxMPID_SD_MODE	SD 신호에 따른 동작 모드를 설정합니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. • 0 : SD 신호가 ON 되면 초기속도까지 감속합니다(정지하지 않음). • 1 : SD 신호가 ON 되면 감속 후 정지합니다.
ccmxMPID_STA_MODE	STA 신호 모드를 설정합니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. • 0 : STA 입력 신호는 무시되며, 이동 명령이 내려지면 바로 이동을 시작합니다. • 1 : 이동 명령이 내려져도 바로 이동을 시작하지 않고, STA 신호가 ON 이 되면 이동을 시작합니다.
ccmxMPID_STA_TRG	STA 신호가 ON 되는 형태를 설정합니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. • 0 : Level (LOW) => STA 신호가 LOW LEVEL 일때 ON • 1 : Falling Edge => STA 신호가 HIGH 상태에서 LOW 상태로 천이될 때 ON
ccmxMPID_STP_MODE	STP 신호 모드를 설정합니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. • 0 : Ignore STP => STP 입력 신호 무시 • 1 : Immediate stop => STP 입력이 ON 되면 즉시 정지 • 2 : Stop after decel => STP 입력이 ON 되면 감속 후 정지
ccmxMPID_CLR_CNTR	CLR 신호가 입력되었을 때 CLEAR 되도록 할 모션컨트롤러의 카운터를 선택합니다. 이 값은 4 비트의 값으로 설정하며 각 비트는 다음과 같이 각 카운터의 클리어 여부를 설정합니다. • **Bit 0** : Command counter 의 클리어 여부를 설정 • **Bit 1** : Feedback counter 의 클리어 여부를 설정 • **Bit 2** : Deviation counter 의 클리어 여부를 설정 • **Bit 3** : General counter 의 클리어 여부를 설정

PropId	Meaning & PropVal
ccmxMPID_CLR_SIGTYPE	CLR 신호의 신호 형태를 설정합니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. • 0 : Falling edge => 스위치가 Open 상태에서 Close 되는 순간에 카운터가 클리어 됩니다. • 1 : Rising edge => 스위치가 Close 상태에서 Open 되는 순간에 카운터가 클리어 됩니다. • 2 : Low level => 스위치가 Close 되어 있는 동안에는 계속 카운터가 클리어 됩니다. • 3 : High level => 스위치가 Open 되어 있는 동안에는 계속 카운터가 클리어 됩니다.
ccmxMPID_CMP_PWIDTH	CMP 출력은 One-shot pulse 로 출력되는데, 출력되는 펄스의 폭을 조절할 수 있습니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. • 0 : 트리거 시점의 Command 펄스의 펄스 폭과 동일한 펄스 폭을 가짐 • 양수 : 이 값에 1.5us 가 곱해진 값이 펄스 폭이 됩니다. 즉, 이 값을 1 로 하면 1.5us, 2 로 하면 3us…와 같이 됩니다.
ccmxMPID_ERC_ONTIME	ERC 출력 펄스의 펄스 폭을 설정합니니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. 0 => 12us, 1 => 102us, 2 => 409us, 3 => 1.6ms, 4 => 13ms, 5 => 52ms, 6 => 104ms, 7=> Logic Level Output
ccmxMPID_SVON_LOGIC	서보온(Servo-On) 신호의 출력 로직을 설정합니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (cmxLOGIC_A) : A 접점 방식 ▪ 1 (ccmxLOGIC_B) : B 접점 방식
ccmxMPID_ERC_OUT_EL	-/+ EL 신호가 ON 되어 정지할 때 ERC 출력 발생 여부를 설정합니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (cmxFALSE) : ERC 출력을 발생하지 않습니다. ▪ 1 (cmxTRUE) : ERC 출력을 발생합니다.
ccmxMPID_CNT_D_SRC	Deviation (편차) 카운터의 입력 소스(Input Source) 를 선택합니다. 이 값은 4 비트의 값으로 설정하며 각 비트는 다음과 같습니다. ▪ 0 : 출력 펄스와 EA/EB 단자를 통한 입력 카운트 ▪ 1 : 출력 펄스와 PA/PB 단자를 통한 입력 카운트 ▪ 2 : EA/EB 단자와 PA/PB 단자를 통한 입력 카운트
ccmxMPID_CNT_G_SRC	General (범용) 카운터의 입력 소스(Input Source) 를 선택합니다. 이 값은 4 비트의 값으로 설정하며 각 비트는 다음과 같습니다. ▪ 0 : 출력 펄스 ▪ 1 : EA/EB 단자를 통한 입력 카운트 ▪ 2 : PA/PB 단자를 통한 입력 카운트 ▪ 3 : CLK 카운트를 2 로 나눈 값을 입력 카운트로 설정
ccmxMPID_LTC_TRGMODE	LATCH COUNTER 의 트리거 모드를 설정합니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (ccmxLTM_LTC) : LTC 입력 신호에 의해서 포지션 래치가 수행됩니다. ▪ 1 (ccmxLTM_ORG) : ORG 입력 신호에 의해서 포지션 래치가 수행됩니다.
ccmxMPID_SLIM_EN	Software Limit 기능 사용 여부를 설정합니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 : Software Limit 기능을 사용하지 않습니다. ▪ 1 : Software Limit 기능을 사용합니다.
ccmxMPID_OUT_MODE	Command(지령) 펄스 출력 모드를 선택합니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (ccmxOMODE_PDIR0) : Pulse & Dir 0 ▪ 1 (ccmxOMODE_PDIR1) : Pulse & Dir 1 ▪ 2 (ccmxOMODE_PDIR2) : Pulse & Dir 2 ▪ 3 (ccmxOMODE_PDIR3) : Pulse & Dir 3 ▪ 4 (ccmxOMODE_CWCCW0) : CW & CCW 0 ▪ 5 (ccmxOMODE_CWCCW1) : CW & CCW 1

PropId	Meaning & PropVal
ccmxMPID_IN_MODE	Feedback(인코더) 펄스 입력 모드를 선택합니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (ccmxIMODE_AB1X) : 1X A/B ▪ 1 (ccmxIMODE_AB2X) : 2X A/B ▪ 2 (ccmxIMODE_AB4X) : 4X A/B ▪ 3 (ccmxIMODE_CWCCW) : CW/CCW ▪ 4 (ccmxIMODE_STEP) : Step Mode
ccmxMPID_IN_INV	Feedback Count 값의 UP/DOWN 방향을 반대로 할 것인지를 선택합니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (cmxFALSE) : Feedback 카운터 UP/DOWN 방향 유지. ▪ 1 (cmxTRUE) : Feedback 카운터 UP/DOWN 방향 반대.
ccmxMPID_CEMG_EN	EMG(Emergency) 신호 입력 활성화 여부를 설정합니다. 설정 및 반환되는 PropVal 은 다음과 같습니다. ▪ 0 (cmxFALSE) : EMG 비활성 ▪ 1 (cmxTRUE) : EMG 활성

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러 처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

long nChannel = 0;//0 번 축을 선택합니다.
Long BoardID = 0;
long nElLogic = 0;
long nAlmLogic = 0;
long nOrgLogic = 0;

//0 번 축의 MIO 설정을 변경합니다.
cmxCfgSetMioProperty(BoardID, 3, nChannel, ccmxMPID_EL_LOGIC, ccmxLOGIC_B );     // EL 로직 설정
cmxCfgSetMioProperty(BoardID, 3, nChannel, ccmxMPID_ALM_LOGIC, ccmxLOGIC_B );  // ALM 로직 설정
cmxCfgSetMioProperty(BoardID, 3, nChannel, ccmxMPID_ORG_LOGIC, cmxLOGIC_A );   // ORG 로직 설정
//PropId 를 통해서 변경하고자 하는 옵션을 선택 할 수 있습니다.

//0 번 축의 MIO 설정을 반환합니다.
cmxCfgGetMioProperty(BoardID, 3, nChannel, ccmxMPID_EL_LOGIC, &nElLogic );                    // EL 로직 확인
cmxCfgGetMioProperty(BoardID, 3, nChannel, ccmxMPID_ALM_LOGIC, &nAlmLogic );     // ALM 로직 확인
cmxCfgGetMioProperty(BoardID, 3, nChannel, ccmxMPID_ORG_LOGIC, &nOrgLogic );     // ORG 로직 확인
//PropId 를 통해서 반환할 옵션을 선택 할 수 있습니다.
```

NAME

cmxPmCfgSetFilter/ cmxPmCfgGetFilter

- I/O 신호 노이즈 필터(Noise Filter) 설정 및 반환

INFORMATION

- Environment
  Configuration Functions
- VC++ (6, 7, 8)/VB
  BCB/Delphi
- Level 2
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxPmCfgSetFilter ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 IsEnable)

❒ VT_I4 cmxPmCfgGetFilter ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 IsEnabled)

DESCRIPTION

cmxPmCfgSetFilter()/cmxPmCfgGetFilter() 함수는 각종 I/O 신호에 필터 로직을 적용할지를 설정 혹은 설정 상태를 반환하는 함수입니다. 필터 로직이 적용되면 펄스 폭이 너무 짧은 입력 신호는 무시되게 됩니다. 필터 로직이 적용되는 I/O 신호 및 펄스 폭은 REFERENCE 를 참조하십시오.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***IsEnable*** : 필터 로직의 적용 여부를 설정 혹은 반환합니다.

Value	Meaning
0 (cmxFALSE)	Filter Disable (필터 로직 비활성화).
1 (cmxTRUE)	Filter Enable (필터 로직 활성화).

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러 처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

SEE ALSO

cmxPmCfgSetFilterAB, cmxPmCfgGetFilterAB

REFERENCE

□ 필터 로직 적용이 Enable 됐을 때 필터 로직이 적용되는 신호 및 기준 펄스 폭은 다음과 같습니다.

필터 적용 I/O	필터 기준
+EL, -EL, SD, ORG, ALM, INP	펄스 폭이 4μs 미만은 무시
+DR, -DR	펄스 폭이 3.2ms 미만은 무시

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

long nChannel = 0;//0 번 축을 선택합니다.
Long BoardID = 0;
long nFilterEnable = 0;

/* 노이즈 필터 로직 활성 여부를 확인하여 비활성 상태이면 활성 상태로 설정합니다. */
cmxPmCfgSetFilter(BoardID, 3, nChannel, &nFilterEnable );

if( nFilterEnable != cmxFALSE )
{
  //Filter 를 사용하도록 설정합니다.
  cmxPmCfgSetFilter(BoardID, 3, nChannel, cmxTRUE );
}
```

NAME

cmxPmCfgSetFilterAB /

cmxPmCfgGetFilterAB

- EA/EB, PA/PB 신호 노이즈 필터(Noise Filter) 설정 및 반환

INFORMATION

- Environment
  Configuration Functions
- VC++ (6, 7, 8)/VB
  BCB/Delphi
- Level 2
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxPmCfgSetFilterAB ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 Target, [in] VT_I4 IsEnable)

❒ VT_I4 cmxPmCfgGetFilterAB
([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 Target, [out] VT_PI4 IsEnabled)

DESCRIPTION

cmxPmCfgSetFilterAB()/cmxPmCfgGetFilterAB() 함수는 EA/EB(Encoder Feedback) 신호와 PA/PB(Manual Pulsar) 신호의 입력 회로에 필터 적용을 위해 설정 혹은 해당 설정 매개변수를 반환하는 함수입니다.

필터를 적용하게 되면 펄스의 폭이 308ns 보다 작은 펄스는 노이즈로 간주되어서 필터됩니다. 이는 EA/EB 신호 또는 PA/PB 신호의 입력에 대해서 비교적 노이즈에 강한 처리를 할 수 있도록 합니다. 정리하자면, 노이즈 필터를 적용하게 되면 3.25MHz 이상의 주파수를 가지는 펄스는 노이즈로 간주되므로 무시되어 결과적으로 정상적인 처리가 될 수 있습니다.

필터의 적용 여부는 EA/EB 신호와 PA/PB 신호에 대하여 각각 서로 다르게 설정할 수 있으며, 이는 Target 매개 변수를 통해서 함수의 적용 대상을 구분합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***Target*** : 함수의 적용 대상을 결정합니다. 이 매개변수에 적용 가능한 값은 다음과 같습니다.

Value	Meaning
0 (ccmxAB_ENC)	함수의 적용 대상이 EA/EB 신호임을 의미합니다.
1 (ccmxAB_PULSAR)	함수의 적용 대상이 PA/PB 신호임을 의미합니다.

▶ ***IsEnable*** : 필터 로직의 적용 여부를 설정 혹은 반환합니다.

Value	Meaning
0 (cmxFALSE) **[Default]**	Filter disable
1 (cmxTRUE)	Filter enable

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러 처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

SEE ALSO

cmxPmCfgSetFilter, cmxPmCfgGetFilter

REFERENCE

□ 필터 로직 적용이 Enable 되면 3.25 MHz 이상의 주파수를 가지는 펄스는 노이즈로 간주되어서 카운트되지 않습니다.

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

long nChannel = 0;//0 번 축을 선택합니다.
Long BoardID = 0;
long nFilterABEnable = 0;

//0 번 축의 EA/EB 신호 입력 회로에 필터를 적용합니다.
cmxPmCfgSetFilterAB (BoardID, 3, nChannel, ccmxAB_ENC, cmxTRUE );
//0 번 축의 PA/PB 신호 입력 회로에 필터를 적용합니다.
cmxPmCfgSetFilterAB (BoardID, 3, nChannel, ccmxAB_PULSAR, cmxTRUE );

//0 번 축의 EA/EB 신호 입력 회로의 필터 적용 여부를 확인합니다.
cmxPmCfgGetFilterAB (BoardID, 3, nChannel, ccmxAB_ENC, &nFilterABEnable );
//0 번 축의 PA/PB 신호 입력 회로에 필터 적용 여부를 확인합니다.
cmxPmCfgGetFilterAB (BoardID, 3, nChannel, ccmxAB_PULSAR, &nFilterABEnable );
```

NAME

cmxPmCfgSetInMode/ cmxPmCfgGetInMode

- 인코더 펄스 신호 입력 모드 설정 및 반환

INFORMATION

- Environment
  Configuration Functions
- VC++ (6, 7, 8)/VB
  BCB/Delphi
- Level 2
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxPmCfgSetInMode
([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 InputMode, [in] VT_I4 IsReverse)

❒ VT_I4 cmxPmCfgGetInMode
([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 InputMode, [out] VT_PI4 IsReverse)

DESCRIPTION

cmxPmCfgSetInMode()/cmxPmCfgGetInMode() 함수는 Feedback Pulse 의 입력 모드를 설정 혹은 반환합니다.

Feedback 펄스는 실제 모터 또는 구조물의 구동 거리를 확인하기 위하여 사용되며 거의 대부분 인코더 입력을 사용합니다. 사용자는 4 가지 형태의 Feedback 펄스의 입력 모드를 설정할 수 있습니다.

또한 이 함수는 입력 펄스의 로직(Feedback Count 값의 UP/DOWN 방향)을 설정합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***InputMode*** : Feedback 펄스 입력 모드를 설정 혹은 반환합니다. 설정 가능한 입력 모드는 다음과 같습니다.

Value	Meaning
0 (ccmxIMODE_AB1X)	1X A/B (1 채배 엔코더 입력 모드)

1 (ccmxIMODE_AB2X)	2X A/B (2 채배 엔코더 입력 모드)
2 (ccmxIMODE_AB4X)	4X A/B (4 채배 엔코더 입력 모드)
3 (ccmxIMODE_CWCCW)	CW/CCW (A 펄스 - 카운트 증가, B 펄스 - 카운트 감소)
4 (ccmxIMODE_STEP)	이 모드에서는 Feedback 위치 값을 읽으면 Command 위치값이 바이패스(bypass) 됩니다. 엔코더 피드백이 없는 경우(스텝모터)에 이 모드를 선택합니다.

▶ ***IsReverse*** : Feedback Count 값의 UP/DOWN 방향을 반대로 할 것인지 설정 혹은 반환합니다.

Value	Meaning
0 (cmxFALSE)	Feedback count 의 UP/DOWN 방향이 반대가 아닙니다.
1 (cmxTRUE)	Feedback count 의 UP/DOWN 방향이 반대입니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러 처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

long nChannel = 0;//0 번 축을 선택합니다.
Long BoradID = 0;
long nInputMode = 0;
long nIsReverse = 0;

/*0 번 축의 Input Mode 를 CW/CCW 로 설정하고,
Feedback Count 값의 UP/DOWN 방향을 반대로(Reverse Mode 를 cmxFALSE 로) 설정합니다.*/
cmxPmCfgSetInMode (BoardID, 3, nChannel, ccmxIMODE_CWCCW, cmxFALSE );

/*0 번 축의 Input Mode 를 nInputMode 변수를 통해서 반환 받고,
Reverse Mode 의 설정 여부를 nIsReverse 를 통해서 확인합니다.*/
cmxPmCfgGetInMode (BoardID, 3, nChannel, &nInputMode, &nIsReverse );
```

NAME

cmxPmCfgSetOutMode/

cmxPmCfgGetOutMode

- 지령 펄스(Command Pulse) 신호 출력 모드 설정 및 반환

INFORMATION

- Environment: Configuration Functions
- VC++ (6, 7, 8)/VB: BCB/Delphi
- Level 2
- 다소 주의

SYNOPSIS

❒ VT_I4 cmxPmCfgSetOutMode ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 OutputMode)

❒ VT_I4 cmxPmCfgGetOutMode ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 OutputMode)

DESCRIPTION

cmxPmCfgSetOutMode() 함수는 Command 펄스의 출력 모드를 설정합니다. Command 신호에 대한 하드웨어적인 회로는 별도로 제공되는 "ceMC02P" HW 매뉴얼의 "6.1.1 COMMAND 신호 (CW, CCW)" 단원을 참조하시기 바랍니다.

cmxPmCfgGetOutMode() 함수는 현재 설정된 Command 펄스 출력 모드를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

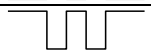
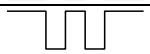
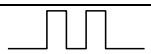
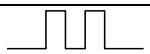
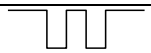
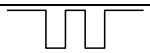
▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

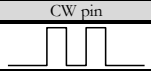
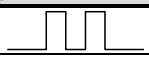
▶ ***OutputMode*** : Command 펄스 출력 모드의 설정 값입니다. 출력 모드는 다음과 같이 6 가지 모드를 가집니다.

가) Pulse & Direction Mode

Value	출력 형태			
	(+) 방향 운전 시		(-) 방향 운전 시	
	CW pin	CCW pin	CW pin	CCW pin

0		(Low)		(High)
1		(Low)		(High)
2		(High)		(Low)
3		(High)		(Low)

나) CW/CCW Mode

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

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러 처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

REFERENCE

□ 모든 경우에 해당하는 부분은 아니지만, 지령 펄스(Command Pulse)의 출력 설정 형태는 일반적으로 서보드라이브에서는 4 혹은 5 번 모드인 CW/CCW 모드를 주로 사용하며, 스텝드라이브에서는 0 번에서 3 번까지의 OUT/DIR 모드를 주로 사용합니다.

□ 일반적인 서보드라이브에서 CW/CCW 혹은 OUT/DIR 모드를 결정하기 위한 지령 펄스(Command Pulse) 입력 설정이 존재합니다. 서보드라이브의 매뉴얼을 반드시 참조하여, ㈜커미조아 모션 모듈과 동일하게 설정하여 주십시오. 동일하게 설정이 되지 않았을 경우에는, 다음과 같은 현상이 발생할 수 있습니다.

1. 서보드라이브를 통해 동작하는 모터의 방향이 한 방향으로만 동작하고 다른 방향으로는 동작하지 않습니다. 즉, 음의 방향(Negative Direction) 으로는 동작하는데 양의 방향(Positive Direction) 으로는 동작하지 않거나 그 반대의 경우를 의미합니다.

2. 방향이 전환 될 때마다 1 펄스(Pulse) 이내의 지령 펄스(Command Pulse) 입력을 무시하게 됩니다. 이 현상을 확인하시려면, 서보드라이브의 Command Pulse Counter 와 ㈜커미조아의 모션 모듈의 Command Pulse Counter 와의 연속적인 방향전환 이동 시에 Pulse Counter 차이를 모니터링(Monitoring) 하시면 됩니다.

이외에도 스텝드라이브의 경우에는 드라이브 입력 부 회로의 사양에 따라 Open Collector 연결을 주로 사용하는데, 결선 방법과 상태에 따라서 일반적인 모터의 움직임에 이상 상황이 발생할 수 있습니다. 이때에는 해당 스텝드라이브의 입력 부 사양을 첨부하여, ㈜커미조아 고객 지원 팀으로 문의해 주시면 친절하게 상담하여 드리겠습니다.

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

long nChannel = 0;//0 번 축을 선택합니다.
long nOutputMode = 0;
long BoardID = 0;

//0 번 축의 Output Mode 를 CWCCW0 로 설정합니다.
cmxPmCfgSetOutMode (BoardID, 3, nChannel, ccmxOMODE_CWCCW0 );

//0 번 축의 Output Mode 상태를 nOutputMode 변수를 통해서 반환 받습니다.
cmxPmCfgGetOutMode (BoardID, 3, nChannel, &nOutputMode );
```

NAME

cmxPmCfgSetCtrlMode/

cmxPmCfgGetCtrlMode

- 각 축의 제어 모드 설정 및 반환

INFORMATION

Environment

Configuration Functions

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 2

다소 주의

전체 이송 목표 좌표의 기준을 변경하는 설정 입니다.

SYNOPSIS

❒ VT_I4 cmxPmCfgSetCtrlMode ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 CtrlMode)

❒ VT_I4 cmxPmCfgGetCtrlMode ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 CtrlMode)

DESCRIPTION

cmxPmCfgSetCtrlMode() 함수는 각 축의 제어모드를 결정합니다. 여기서 제어모드라는 것은 엔코더 피드백 값을 이송 명령에서 어떻게 활용하느냐 하는 것입니다. 이송 명령이 내려졌을 때 목표 좌표를 커맨드(Command) 위치를 기준으로 적용할 것인지, 피드백(Feedback) 위치를 기준으로 적용할 것인지를 결정하는 함수입니다.

cmxPmCfgGetCtrlMode() 함수는 각 축의 제어모드에 대하여 현재 원격 노드에 설정되어 있는 값을 읽어 들이는 함수입니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***CtrlMode*** : cmxPmCfgSetCtrlMode() 함수의 인자이며, 제어 모드를 설정합니다. 이 값의 의미는 다음과 같습니다.

Value	Meaning
0 (ccmxCTRL_OPEN) **[Default]**	Open Loop 제어 모드로 설정합니다. 이 모드에서는 이송명령의 목표 좌표가 항상 커맨드 위치를 기준으로 설정됩니다.
1 (ccmxCTRL_SEMI_C)	Scmxi-Closed Loop 제어 모드로 설정합니다. 이 모드에서는 이송명령의 목표 좌표가 항상 피드백 위치를 기준으로 설정됩니다. 예를 들어서 10000 의 좌표로 이송하라는 명령이 하달되었을 때 커맨드 위치는 무시하고, 피드백 위치가 10000 이 되도록 이송의 목표좌표가 설정됩니다.
2 (ccmxCTRL_FULL_C)	이 값은 향후 버전을 위해서 예약된 값이며, 현재는 사용되지 않습니다.

▶ ***CtrlMode*** : cmxPmCfgGetCtrlMode() 함수의 인자이며, 제어모드를 반환합니다. 반환값의 의미는 다음과 같습니다.

Value	Meaning
0 (ccmxCTRL_OPEN) **[Default]**	Open Loop 제어 모드입니다. 이 모드에서는 이송명령의 목표 좌표가 항상 커맨드 위치를 기준으로 설정됩니다.
1 (ccmxCTRL_SEMI_C)	Scmxi-Closed Loop 제어 모드입니다. 이 모드에서는 이송명령의 목표 좌표가 항상 피드백 위치를 기준으로 설정됩니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러 처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

SEE ALSO

cmxPmSxMoveTo, cmxPmSxMoveToStart, cmxPmIxLineToStart

REFERENCE

□ 여기서 설정하는 제어모드는 절대 좌표 이송 명령에서만 영향을 미치고, 상대 좌표 이송 명령에서는 항상 Open Loop 제어 모드로 운용됩니다.

□ 제어 모드에 따른 차이에 대한 이해를 돕기 위하여 예를 들어 보겠습니다. 현재 커맨드 위치가 5000 , 피드백 위치가 4000 인 상태에서 cmxPmSxMoveTo() 함수를 사용하여 10000 의 위치로 이송하라는 명령을 하달하였을 때

- Open loop 제어모드에서는 현재 위치에서 5000 만큼 (+) 방향으로 이송
- Scmxi-Closed Loop 제어모드에서는 현재 위치에서 6000 만큼 (+) 방향으로 이송

을 하는 서로 다른 이송 결과를 나타낼 수 있습니다.

□ Scmxi-Closed Loop 제어모드로 설정하였을 때 이송 목표 좌표의 계산은 이송 명령을 수행하는 초기에 계산됩니다. 즉, 이송 명령을 실행하는 초기에 현재의 피드백 위치를 읽어서 목표 좌표와의 차이 만큼을 상대좌표 이송하는 방식으로 운용됩니다. 따라서 이송 명령이 끝나는 시점에서 피드백 위치를 확인하여 지령된 목표좌표에 피드백 위치가 항상 일치하는 것을 보장하는 Full-Closed Loop 방식과는 차이가 있습니다.

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

long nChannel = 0;//0 번 축을 선택합니다.
Long BoardID = 0;
long nCtrlMode = 0;

//0 번 축을 Scmxi-Closed Loop 으로 설정합니다.
cmxPmCfgSetCtrlMode(BoardID, 3, nChannel, ccmxCTRL_SEMI_C );

//0 번 축에 설정된 제어모드를 반환합니다.
cmxPmCfgGetCtrlMode(BoardID, 3, nChannel, &nCtrlMode );
```

NAME

cmxPmCfgSetInOutRatio/
cmxPmCfgGetInOutRatio
- 입력 펄스와 출력 펄스의 분해능 비율 (Resolution ratio) 설정 및 반환

INFORMATION

- Environment
  Configuration Functions
- VC++ (6, 7, 8)/VB
  BCB/Delphi
- Level 2
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxPmCfgSetInOutRatio ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_R8 Ratio)

❒ VT_I4 cmxPmCfgGetInOutRatio ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PR8 Ratio)

DESCRIPTION

cmxPmCfgSetInOutRatio() 함수는 Feedback 펄스와 Command 펄스의 분해능 비율 (Resolution ratio)을 설정합니다. 여기서 Feedback 펄스의 분해능이란 엔코더의 1 회전 시에 발생되는 펄스 수를 의미합니다. 그리고 Command 펄스의 분해능이란 모터를 1 회전시키기 위해 필요한 Command 펄스 수를 의미합니다.

cmxPmCfgGetInOutRatio() 함수는 설정되어있는 Feedback 펄스와 Command 펄스의 분해능 비율 (Resolution ratio)을 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***Ratio*** : Feedback 펄스와 Command 펄스의 분해능 비를 설정 혹은 반환합니다. 이 값은 아래와 같이 설정합니다.

Ratio = (Feedback 펄스 분해능)/(Command 펄스의 분해능)

RETURN VALUE

Value	Meaning

음수	수행 실패. 자세한 내용은 '에러 처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

REFERENCE

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

☐ In/Out Ratio 는 cmxPmStGetPosition() 함수와 cmxPmStGetSpeed() 함수에서 카운터를 cmxCNT_FEED 으로 설정한 경우에만 영향을 미칩니다.

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

long nChannel = 0;                    //0 번 축을 선택합니다.
long BoardID = 0;
double fRatio = 2.0f;       //분해능 비율을 2 로 설정합니다.

/* 0 번 축의 분해능 비율을 설정합니다. 분해능 비율은
Ratio = (Feedback 펄스 분해능)/(Command 펄스의 분해능) */
cmxPmCfgSetInOutRatio (BoardID, 3, nChannel, fRatio );

//0 번 축의 현재 분해능 비율을 반환합니다.
cmxPmCfgGetInOutRatio (BoardID, 3, nChannel, &fRatio );
```

NAME

cmxPmCfgSetUnitDist/ cmxPmCfgGetUnitDist

- 논리적 단위 거리 설정 및 반환

INFORMATION

- Environment
  Configuration Functions
- VC++ (6, 7, 8)/VB
  BCB/Delphi
- Level 2
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxPmCfgSetUnitDist ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_R8 UnitDist)

❒ VT_I4 cmxPmCfgGetUnitDist ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PR8 UnitDist)

DESCRIPTION

cmxPmCfgSetUnitDist() 함수는 논리적 단위 거리에 대한 펄스 수를 설정합니다. 여기서 논리적 단위 거리라 함은 Move 함수에서 사용하는 거리 또는 위치에 대한 단위량을 의미합니다. 이 함수를 사용하여 특별히 지정하지 않는 경우에는 초기값인 '1' 로 사용됩니다.

cmxPmCfgGetUnitDist() 함수는 논리적 단위 거리에 대한 펄스 수를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Axis*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***UnitDist*** : 논리적 거리 1 을 이동하기 위해서 출력되어야 하는 펄스 수를 설정 혹은 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러 처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

REFERENCE

☐. Unit distance & Unit speed

모션컨트롤러에서 이동거리는 기본적으로는 Command 펄스 수에 의해 결정되고, 이동속도는 펄스의 주파수에 의해서 결정됩니다. 따라서 이동거리를 물리적인 거리 단위로 하기 위해서는 매번 원하는 물리적인 거리를 이동하기 위해서 필요한 펄스의 수를 계산해야 합니다. 하지만, ㈜커미조아 모션컨트롤러는 모든 이동 함수와 속도 설정 함수에서 사용되는 논리적 거리와 논리적 속도를 사용자가 정의할 수 있도록 하고 있으며, 이 것을 정의하는 것이 "Unit distance"와 "Unit speed" 입니다. 일반적으로 "Unit distance" Du 는 다음과 같이 계산하면 됩니다.

$$D_u = P_r / L_r$$

Pr : 모터 1 회전에 필요한 펄스 수 (모터의 Command 분해능)

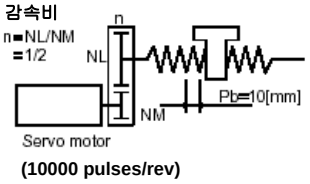
Lr : 모터 1 회전시에 이동되는 기구물의 거리

결과적으로 "Unit distance" 에 설정해야 하는 값은 논리적 거리 '1'을 이동하기 위해서 필요한 출력 펄스의 수가 됩니다. 그리고 특별한 경우가 아니면 "Unit speed" 는 "Unit distance" 와 같은 값을 설정하면 됩니다. 하지만 "Unit speed"와 "Unit distance"는 필요에 따라 서로 다른 단위를 사용할 수 있습니다.

Unit distance 값이 무한소수(나누어 떨어지지 않음)이면 소수점 오차가 누적될 수 있습니다. 따라서 이러한 경우에는 Unit distance 를 '1'로 하고 사용자가 논리적 단위 거리를 처리하는 것이 바람직합니다.

☐. Unit distance & Unit speed 계산 예

다음과 같은 사양의 볼스크류를 사용하는 기구물에서의 경우에 "Unit distance"와 "Unit speed"는 다음과 같이 계산할 수 있습니다.



볼스크류 Lead (Pb) = 10 mm
감속비 (n) = 1/2
모터1회전시 이동거리 (Lr) = 10 *½= 5 mm
모터 Command 분해능 (Pr) = 10000 pulses/rev

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

/*모터 1 회전시 이동거리 (Lr) = 5mm
모터 Command 분해능(Pr) = 10000 pulse/rev
인 기구물이 있다면
Unit distance(Du) = Pr/Lr = 10000/5 = 2000
Unit speed(Vu) = 2000 */

long nChannel = 0;//0 번 축을 선택합니다.
long BoardID = 0;

//Unit distance 를 2000 으로 설정합니다.
cmxPmCfgSetUnitDist (BoardID, 3, nChannel, 2000 );

/*따라서 Unit distance 와 Unit speed 를 2000 으로 설정하면
다음과 같이 모든 이동함수에서 거리의 단위를 mm 단위에 해당하는
값으로 입력할 수 있습니다.*/

//속도 = 100(mm/s), 가/감속도 = 1000(mm/s^2)으로 설정합니다.
cmxPmCfgSetSpeedPattern(BoardID, 3, nChannel, cmxMODE_T, 100, 1000, 1000 );

//(+)방향으로 20mm 이송합니다. (실제는 20 * 2000 = 40000 펄스가 출력됩니다.)
cmxPmSxMove(BoardID, 3, nChannel, 20, cmDIR_P );
```

NAME

cmxPmCfgSetUnitSpeed /

cmxPmCfgGetUnitSpeed

- 논리적 속도 단위 설정 및 반환

INFORMATION

Environment

Configuration Functions

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 2

위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxPmCfgSetUnitSpeed ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_R8 UnitSpeed)

❒ VT_I4 cmxPmCfgGetUnitSpeed ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PR8 UnitSpeed)

DESCRIPTION

cmxPmCfgSetUnitSpeed() 함수는 논리적 단위 속도에 대한 실제 펄스 출력속도(PPS)를 설정합니다. 여기서 논리적 단위 속도라 함은 속도 지정함수에서 사용하는 속도 또는 가속도에 대한 단위량을 의미합니다. 이 함수를 사용하여 특별히 지정하지 않는 경우에는 단위 속도에 대한 펄스 출력속도는 '1' PPS 로 사용됩니다.

cmxPmCfgGetUnitSpeed() 함수는 논리적 단위 속도에 대한 실제 펄스 출력속도(PPS) 설정 상태를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***UnitSpeed*** : 단위 속도에 대한 펄스 출력 속도를 설정 혹은 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러 처리' 편을 참고합니다

0 (ERR_NONE)	수행 성공

REFERENCE

□ 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. cmxPmCfgSetUnitSpeed() 함수는 사용자가 속도의 단위를 결정하도록 하는 함수입니다. 다음의 예를 참고하시기 바랍니다.

Ex 1) 1 회전에 필요한 펄스 수가 3600 펄스인 경우에 속도의 단위를 RPM 으로 하고자 한다면 Unit Distance 값을 3600/60, 즉 60 PPS 로 설정합니다(여기서 60 으로 나누는 것은 RPM 은 분당 회전 수 이므로 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 UnitDist 값을 1000 PPS 로 설정합니다.

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

//1 회전에 3600 펄스가 필요한 모터를 사용한다고 가정합니다.

long nChannel = 0;//0 번 축을 선택합니다.
long BoardID = 0;

//거리의 단위를 각도(1 도)로, 속도의 단위를 rpm 으로 설정합니다.
//1 회전에 3600 펄스가 필요하므로 1 도 회전을 하기 위해서는 10 펄스가 필요합니다.
cmxPmCfgSetUnitDist(BoardID, 3, nChannel, 10 );

//단위 속도를 1rpm 으로 설정합니다. 3600(pulse) / 60(sec) = 60(pps)
cmxPmCfgSetUnitSpeed (BoardID, 3, nChannel, 60 );

/*가속도와 감속도를 각각 200rpm/s, 작업 속도를 100rpm/s 로 설정합니다.
가속및 감속에는 0.5sec 가 소요됩니다.*/
cmxPmCfgSetSpeedPattern(BoardID, 3, nChannel, cmxMODE_T, 100, 200, 200 );

//모터를 720 도 회전 시킵니다. 실제로는 720 * 10 pulse 가 출력됩니다.
cmxPmSxMove(BoardID, 3, nChannel, 720, cmxFALSE );
```

NAME

cmxPmCfgSetSpeedRange/

cmxPmCfgGetSpeedRange

- 모션 속도 제한 범위 설정 및 반환

INFORMATION

Environment

Configuration Functions

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 2

위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxPmCfgSetSpeedRange ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 MaxPPS)

❒ VT_I4 cmxPmCfgGetSpeedRange

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 MaxPPS)

DESCRIPTION

cmxPmCfgSetSpeedRange() 함수는 모션에 적용할 수 있는 최저/최고 속도를 제한합니다. 이 함수는 실제적으로는 출력 펄스의 주파수 범위를 설정하는 역할을 합니다. 출력 펄스의 주파수는 최대 6.5MHz 까지 설정 가능 하며 기본적으로 설정되는 주파수 범위는 10Hz ~ 655,350Hz 입니다. 최저속도는 최대속도 설정에 따라서 자동으로 결정됩니다.

cmxPmCfgGetSpeedRange() 함수는 현재 설정되어 있는 최저/최고 속도의 범위를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***MaxPPS*** : 모션의 최고 속도를 PPS 단위로 설정 혹은 반환합니다.

MaxPPS 값을 설정할 때는 Unit speed 설정에 관계없이 항상 **PPS** 단위로 설정하여야 합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러 처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

long BoardID = 0;
long nChannel = 0;                          //0 번 축을 선택합니다.
double fMaxPPS = 655350;
double fMinPPS = 0;

//최대 PPS 를 655350 으로 설정합니다.
cmxPmCfgSetSpeedRange(BoardID, 3, nChannel, fMaxPPS );

//현재 설정된 최대 PPS 를 반환합니다.
cmxPmCfgGetSpeedRange(BoardID, 3, nChannel, &fMinPPS, &fMaxPPS );
```

NAME

cmxPmCfgSetSpeedPattern/

cmxPmCfgGetSpeedPattern

- 모션 이송 기준 속도 설정 및 반환

INFORMATION

- Environment
  Configuration Functions
- VC++ (6, 7, 8)/VB
  BCB/Delphi
- Level 2
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxPmCfgSetSpeedPattern

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 SpeedMode, [in] VT_R8 WorkSpeed, [in] VT_R8 Accel, [in] VT_R8 Decel)

❒ VT_I4 cmxPmCfgGetSpeedPattern

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 SpeedMode, [out] VT_PR8 WorkSpeed, [out] VT_PR8 Accel, [out] VT_PR8 Decel)

DESCRIPTION

cmxPmCfgSetSpeedPattern() 함수는 지정한 축에 대해 속도 모드, 작업속도, 가속도 및 감속도를 설정합니다. 이 속도는 각 모션제어의 기준 속도로 설정되며, 해당 기준속도에 비율로 각 모션제어를 수행할 수 있습니다.
설정된 기준속도는 단축 제어의 경우 실제 모션 속도 지령 함수인 cmxPmSxSetSpeedRatio() 함수를 통해 설정된 비율로 모션 제어의 속도를 결정할 수 있습니다.

cmxPmCfgGetSpeedPattern() 함수는 지정한 축에 대해 설정되어 있는 속도 설정 환경을 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***SpeedMode*** : cmxPmCfgSetSpeedPattern() 함수의 인자이며, 속도모드의 설정 값입니다. 속도모드는 아래와 같습니다.

Value	Meaning
0 (cmxMODE_C)	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 (cmxMODE_T)	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 (cmxMODE_S)	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.
-1 (ccmxSMODE_KEEP)	이전 속도 모드를 유지합니다.

▶ ***SpeedMode*** : cmxPmCfgGetSpeedPattern() 함수의 인자이며, 속도모드의 반환 값 입니다.

Value	Meaning
0 (cmxMODE_C)	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 (cmxMODE_T)	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 (cmxMODE_S)	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.

▶ ***WorkSpeed*** : 작업 속도를 설정 혹은 반환합니다.

▶ ***Accel*** : 가속도를 설정 혹은 반환합니다.

▶ ***Decel*** : 감속도를 설정 혹은 반환합니다.

SEE ALSO

cmxPmSxSetSpeedRatio, cmxPmSxGetSpeedRatio

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러 처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

REFERENCE

□ 한번 설정한 속도설정은 변경하기 전까지 계속해서 적용됩니다. 따라서 속도를 변경할 필요가 없는 경우에는 이송명령을 수행할 때마다 속도설정을 해줄 필요는 없습니다.

□ 속도와 가/감속도의 단위는 cmxPmCfgSetUnitSpeed() 함수에 의해 결정됩니다.

□ CONSTANT Speed Mode
Constant speed mode 에서는 Motion 을 수행할 때 가속/감속을 적용하지 않고 일정속도로 Motion 을 수행합니다. 여기서 적용되는 일정 속도는 WorkSpeed 에서 주어진 값이 적용됩니다.

□ TRAPEZOIDAL Speed Mode

Trapezoidal speed mode 에서는 Motion 을 수행하는데 있어서 속도의 패턴을 [그림 14-1]과 같이
Linear acceleration -> Working speed(constant) -> Linear deceleration 의 형태로 운용하는 모드입니다.

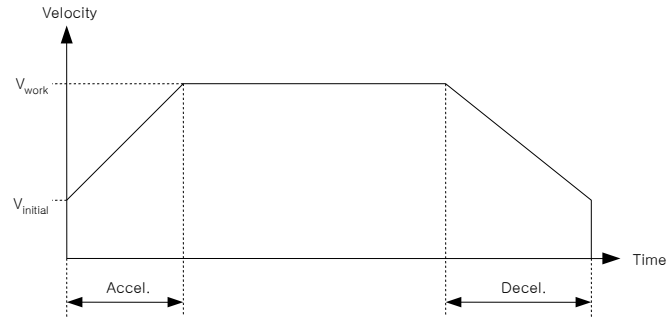


그림 14-1 Trapezoidal speed pattern

여기서 Acceleration time 은 다음과 같습니다.

Tacc = (Vwork – Vinitial)/a

Tacc : Acceleration time
Vwork : Working speed
Vinitial : Initial speed
a : Acceleration setting value

또한, 일반적으로 Vinitial 은 0 이므로, 다음 수식을 만족하며, Deceleration time 또한 위와 같은 계산식이 적용됩니다.
Tacc = Vwork /a

☐ S-CURVE Speed Mode

S-curve speed mode 에서는 Motion 을 수행할 때 S 자 형태로 가속과 감속을 수행합니다. S-curve speed mode 에서 가(감)속 구간은 [그림 7-2] 와 같이 S-curve section 과 Linear acceleration section 으로 구성됩니다.

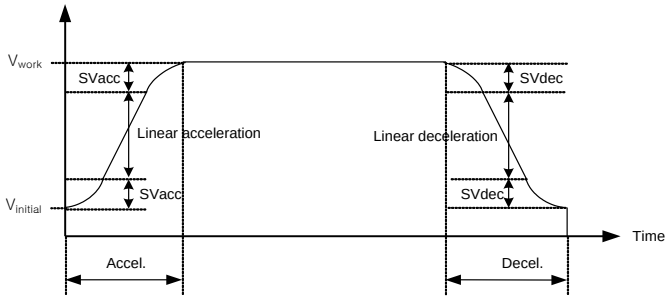


그림 14-2 S-curve speed pattern

※ S-curve speed mode 에서는 설정한 가(감)속 값이 S-curve section 을 포함한 전체 가(감)속 시간을 결정하는 매개 변수로 사용되며 실제 가(감)속도 또는 Jerk 는 자동으로 계산됩니다. 전체 가속 시간 Tacc 는 다음과 같습니다.

Tacc = (Vwork – Vinitial)/a

Tacc : Acceleration time
Vwork : Working speed
Vinitial : Initial speed
a : Acceleration setting value

Deceleration time 또한 위와 같은 계산식이 적용됩니다.

마지막으로 SVacc 의 의미는 가속구간의 S-Curve Section 을 지칭합니다. S-Curve Section 은 모션 라이브러리에서 기본적으로 0 으로 설정되어 있어 완전한 S-Curve 가감속을 수행하도록 설정되어 있습니다. 이 값은 특별한 경우가 아니면 변경하지 않습니다.

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

long BoardID = 0;
long nChannel = 0;//0 번 축을 선택합니다.
long nSpeedMode = 0;
double fVel = 0.0f;
double fAcc = 0.0f;
double fDec = 0.0f;

/*0 번축의 속도 패턴을 S-Curve 로 설정하고,
작업속도를 2000, 가속도를 10000, 감속도를 10000 으로 설정합니다.*/
cmxPmCfgSetSpeedPattern(BoardID, 3, nChannel, cmxMODE_S, 2000, 10000, 10000 );

/*0 번축에 설정된 속도 패턴 및 작업속도, 가속도, 감속도를 반환합니다.*/
cmxPmCfgGetSpeedPattern(BoardID, 3, nChannel, &nSpeedMode, &fVel, &fAcc, &fDec );
```

NAME

cmxPmCfgSetSpeedPattern_T/
cmxPmCfgGetSpeedPattern_T
- 모션 이송 기준 속도 설정 및 반환

INFORMATION

- Environment
  Configuration Functions
- VC++ (6, 7, 8)/VB
  BCB/Delphi
- Level 2
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxPmCfgSetSpeedPattern
([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 SpeedMode, [in] VT_R8 WorkSpeed, [in] VT_R8 AccelTime, [in] VT_R8 DecelTime)

❒ VT_I4 cmxPmCfgGetSpeedPattern
([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 SpeedMode, [out] VT_PR8 WorkSpeed, [out] VT_PR8 AccelTime, [out] VT_PR8 DecelTime)

DESCRIPTION

cmxPmCfgSetSpeedPattern_T() 함수는 지정한 축에 대해 속도 모드, 작업속도, 가속 및 감속 시간을 설정합니다. 이 속도는 각 모션제어의 기준 속도로 설정되며, 해당 기준속도에 비율로 각 모션제어를 수행할 수 있습니다.
설정된 기준속도는 단축 제어의 경우 실제 모션 속도 지령 함수인 cmxPmSxSetSpeedRatio() 함수를 통해 설정된 비율로 모션 제어의 속도를 결정할 수 있습니다.

cmxPmCfgGetSpeedPattern_T() 함수는 지정한 축에 대해 설정되어 있는 속도 설정 환경을 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***SpeedMode*** : cmxPmCfgSetSpeedPattern_T() 함수의 인자이며, 속도모드의 설정 값입니다. 속도모드는 아래와 같습니다.

Value	Meaning
0 (cmxMODE_C)	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 (cmxMODE_T)	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 (cmxMODE_S)	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.
-1 (ccmxSMODE_KEEP)	이전 속도 모드를 유지합니다.

▶ ***SpeedMode*** : cmxPmCfgGetSpeedPattern_T() 함수의 인자이며, 속도모드의 반환 값 입니다.

Value	Meaning
0 (cmxMODE_C)	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 (cmxMODE_T)	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 (cmxMODE_S)	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.

▶ ***WorkSpeed*** : 작업 속도를 설정 혹은 반환합니다.

▶ ***AccelTime*** : 가속 시간을 설정 혹은 반환합니다.

▶ ***Decel*** Time: 감속 시간을 설정 혹은 반환합니다.

SEE ALSO

cmxPmSxSetSpeedRatio, cmxPmSxGetSpeedRatio

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러 처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

REFERENCE

□ 한번 설정한 속도설정은 변경하기 전까지 계속해서 적용됩니다. 따라서 속도를 변경할 필요가 없는 경우에는 이송명령을 수행할 때마다 속도설정을 해줄 필요는 없습니다.

□ 속도와 가/감속도의 단위는 cmxPmCfgSetUnitSpeed() 함수에 의해 결정됩니다.

□ CONSTANT Speed Mode
Constant speed mode 에서는 Motion 을 수행할 때 가속/감속을 적용하지 않고 일정속도로 Motion 을 수행합니다. 여기서 적용되는 일정 속도는 WorkSpeed 에서 주어진 값이 적용됩니다.

□ TRAPEZOIDAL Speed Mode

Trapezoidal speed mode 에서는 Motion 을 수행하는데 있어서 속도의 패턴을 [그림 14-1]과 같이
Linear acceleration -> Working speed(constant) -> Linear deceleration 의 형태로 운용하는 모드입니다.

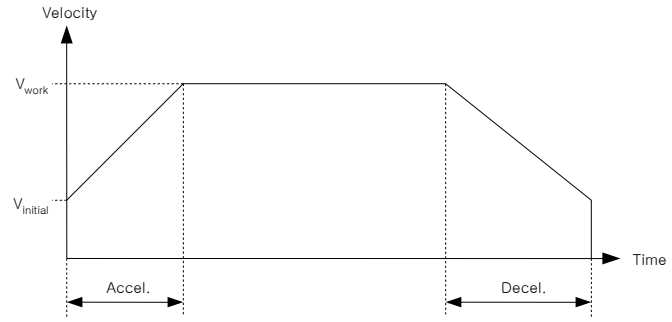


그림 14-3 Trapezoidal speed pattern

여기서 Acceleration time 은 다음과 같습니다.

Tacc = (Vwork – Vinitial)/a

Tacc : Acceleration time
Vwork : Working speed
Vinitial : Initial speed
a : Acceleration setting value

또한, 일반적으로 Vinitial 은 0 이므로, 다음 수식을 만족하며, Deceleration time 또한 위와 같은 계산식이 적용됩니다.
Tacc = Vwork /a

☐ S-CURVE Speed Mode

S-curve speed mode 에서는 Motion 을 수행할 때 S 자 형태로 가속과 감속을 수행합니다. S-curve speed mode 에서 가(감)속 구간은 [그림 7-2] 와 같이 S-curve section 과 Linear acceleration section 으로 구성됩니다.

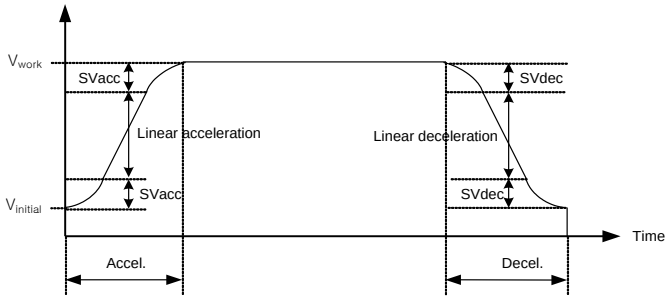


그림 14-4 S-curve speed pattern

※ S-curve speed mode 에서는 설정한 가(감)속 값이 S-curve section 을 포함한 전체 가(감)속 시간을 결정하는 매개 변수로 사용되며 실제 가(감)속도 또는 Jerk 는 자동으로 계산됩니다. 전체 가속 시간 Tacc 는 다음과 같습니다.

Tacc = (Vwork – Vinitial)/a

Tacc : Acceleration time
Vwork : Working speed
Vinitial : Initial speed
a : Acceleration setting value

Deceleration time 또한 위와 같은 계산식이 적용됩니다.

마지막으로 SVacc 의 의미는 가속구간의 S-Curve Section 을 지칭합니다. S-Curve Section 은 모션 라이브러리에서 기본적으로 0 으로 설정되어 있어 완전한 S-Curve 가감속을 수행하도록 설정되어 있습니다. 이 값은 특별한 경우가 아니면 변경하지 않습니다.

NAME

cmxPmCfgSetSoftLimit/

cmxPmCfgGetSoftLimit

- 소프트웨어 이송 제한 범위 설정 및 반환

INFORMATION

- Environment: Configuration Functions
- VC++ (6, 7, 8)/VB: BCB/Delphi
- Level 2
- 다소 주의

SYNOPSIS

❒ VT_I4 cmxPmCfgSetSoftLimit

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 IsEnable, [in] VT_R8 LimitN, [in] VT_R8 LimitP)

❒ VT_I4 cmxPmCfgGetSoftLimit

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 IsEnable, [out] VT_PR8 LimitN, [out] VT_PR8 LimitP)

DESCRIPTION

cmxPmCfgSetSoftLimit() 함수는 소프트웨어 리밋(Limit) 기능을 활성 또는 비활성화 하고 소프트웨어 리밋 범위를 설정합니다. 소프트웨어적인 Limit 은 리밋 센서의 설치가 용이하지 않을 때 안전성을 위하여 모션의 이송 범위를 소프트웨어적인 이송제한범위를 설정하여 제한하는 것입니다.
소프트웨어적인 Limit 은 Command pulse 카운터의 절대값이 지정한 +/- Limit 값보다 같거나 크게 되면 모션을 자동 정지하도록 합니다.

cmxPmCfgGetSoftLimit() 함수는 소프트웨어 리밋(Limit) 활성 상태 및 범위 설정상태를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***IsEnable*** : 소프트웨어 리밋(Limit) 기능의 활성화 여부를 설정 혹은 반환합니다.

▶ ***LimitN*** : (-) 방향 Limit 값을 설정 혹은 반환합니다.

▶ ***LimitP***: (+) 방향 Limit 값을 설정 혹은 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러 처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

REFERENCE

S/W Limit 의 설정에는 항상 Unit Distance 의 값이 고려되지 않는 상황에서 문제가 발생할 수 있습니다. 만약 설정한 Unit Distance 값이 1000 으로 설정되어 있다면, 이 값에 입력된 LimitN 값과 LimitP 값이 28Bit 로 표현할 수 있는 정수 값을 초과해서는 안됩니다.

이 내용을 식으로 표현하면 다음과 같습니다.

Unit Distance * S/W Limit Value < 268,435,456(28bit 정수)

위 의미는 결국 Unit Distance 와 S/W Limit 의 변수 값이 28bit 정수보다 작아야 한다는 의미입니다. 본 함수의 인자가 Double 형이라고 할지라도 이 점을 반드시 주의하시기 바랍니다. 만약 이 값이 28Bit 정수보다 크게 되면, 변수의 값이 Overflow 되어 내부에서 Negative Limit 이 Positive Limit 효과를 가져와, 모터의 축이 +/- 방향으로 움직이지 못하는 현상을 발생시킬 수 있습니다.

소프트웨어 Limit 은 논리적으로 하드웨어적인 Limit 과 동일하게 동작합니다. 또한 Negative Limit 과 Positive Limit 값은 Unit Distance 를 고려하여, 입력 값의 Overflow 를 주의해야 합니다.

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

long BoardID = 0;
long nChannel = 0;                    //0 번 축을 선택합니다.
long nIsEnable = 0;
double fElNeg = 0.0f;
double fElPos = 0.0f;

//0 번 축의 SoftLimit 기능을 활성화 하고, -EL 은 -100000, +EL 은 100000 으로 설정합니다.
cmxPmCfgSetSoftLimit (BoardID, 3, nChannel, cmxTRUE, -100000, 100000 );

//0 번 축의 SoftLimit 의 설정 상태를 반환합니다.
cmxPmCfgGetSoftLimit (BoardID, 3, nChannel, &nIsEnable, &fElNeg, &fElPos );
```

NAME

cmxPmCfgSetRingCntr/

cmxPmCfgGetRingCntr

- 링 카운터(Ring-Counter) 기능 설정 및 반환

INFORMATION

- Environment
  Configuration Functions
- VC++ (6, 7, 8)/VB
  BCB/Delphi
- Level 2
- 다소 주의

SYNOPSIS

❒ VT_I4 cmxPmCfgSetRingCntr

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 TargCntr, [in] VT_I4 IsEnable, [in] VT_R8 CntMax)

❒ VT_I4 cmxPmCfgGetRingCntr

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 TargCntr, [out] VT_PI4 IsEnable, [out] VT_PR8 CntMax)

DESCRIPTION

cmxPmCfgSetRingCntr() 함수는 링 카운터 기능을 활성 또는 비활성화 하고 링 카운터 범위를 설정합니다. 해당 모션 축(Axis) 의 Command 또는 Feedback 카운터를 대상으로 링 카운터를 설정합니다.

cmxPmCfgGetRingCntr() 함수는 링 카운터 기능의 설정 상태를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***TargCntr*** : 링 카운터 기능 대상 카운터를 설정 혹은 반환합니다.

Value	Meaning
0 (cmxCNT_COMM)	Command Counter
1 (cmxCNT_FEED)	Feedback Counter

▶ ***IsEnable*** : 링 카운터 기능 활성/비활성 여부를 설정 혹은 반환합니다.

Value	Meaning
0 (cmxFALSE)	링 카운터 기능을 사용하지 않습니다.
1 (cmxTRUE)	링 카운터 기능을 사용합니다.

▶ ***CntMax*** : 링 카운터 범위(0~ 359)를 설정 혹은 반환합니다. 링 카운터 기능이 활성화되면 지정한 카운터는 0~CntMax 사이의 값에서만 카운트 됩니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러 처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

long BoardID = 0;
long nChannel = 0;//0 번 축을 선택합니다.
long nIsEnable = 0;
long nTrgtCntr = 0;
double fCntMax = 0.0f;

/*0 번 축의 커맨드 카운터를 대상으로 링카운터 기능을 활성화합니다.
40000 펄스가 되면 커맨드 카운터가 다시 '0'부터 카운트 됩니다.*/
cmxPmCfgSetRingCntr (BoardID, 3, nChannel, cmxCNT_COMM, cmxTRUE, 40000 );

//0 번 축의 커맨드 카운터의 링카운터 사용 여부 및 설정 상태를 반환합니다.
cmxPmCfgGetRingCntr (BoardID, 3, nChannel, cmxCNT_COMM, &nIsEnable, &fCntMax );
```

NAME

cmxPmCfgSetVelCorrRatio/

cmxPmCfgGetVelCorrRatio

- 작업 속도 보정에 대한 속도 산출 비례 설정(設定) 및 반환(返還)

INFORMATION

Environment Configuration Functions

VC++ (6, 7, 8)/VB BCB/Delphi

Level 2

다소 주의

특별한 경우가 아니면 이 함수는 사용하지 않습니다.

SYNOPSIS

❒ VT_I4 cmxPmCfgSetVelCorrRatio ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 CorrRatio)

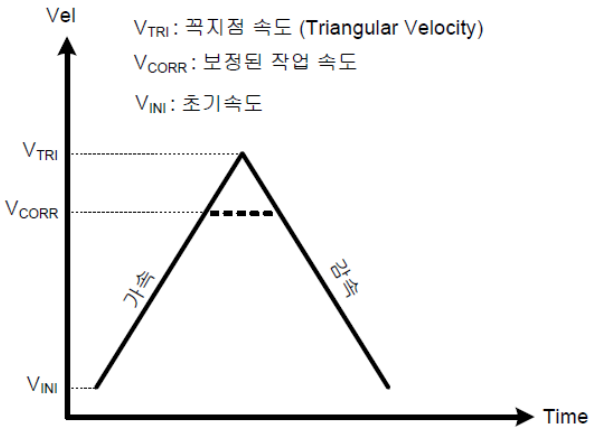
❒ VT_I4 cmxPmCfgGetVelCorrRatio ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [out] VT_PI4 CorrRatio)

DESCRIPTION

cmxPmCfgSetVelCorrRatio() 함수는 작업속도 보정시에 보정(補正)된 작업속도를 산출하는 비례값을 설정합니다. 이 값은 기본적으로 92%로 설정됩니다.

이송거리가 짧은 경우에는 지정한 속도패턴을 구현할 수 없는 경우가 나타납니다. 예를 들면 이송거리가 충분하지 못하면 가속구간 중에 목표지점에 도달한다든지, 감속이 들어가기 전에 목표지점에 도달하는 경우가 발생할 수 있습니다. 이러한 경우에는 급정지(停止)를 하거나 급격한 속도 변화가 발생하여 모터나 기구물에 진동을 유발하고 수명을 단축시킬 수 있습니다.

따라서 ceSDK 에서는 이러한 현상이 발생하지 않도록 하기 위해서 이송거리가 짧은 경우에는 자동으로 작업속도를 보정(補正)하여 이송합니다. 이를 "Work Velocity Correction (WVC)" 이라고 합니다. 기본적으로 WVC 보정은 사용자가 지정한 작업속도가 꼭지점 속도보다 높은 경우에 꼭지점 속도의 92% 수준의 속도로 작업속도를 보정합니다. 여기서 꼭지점 속도란 가속구간과 감속구간만으로 목표지점에 도달할 수 있는 작업속도를 의미합니다. 즉, 꼭지점 속도로 작업속도를 설정하면 가속이 끝나자마자 바로 감속을 시작하게 됩니다.
이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.



PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***CorrRatio*** : cmxPmCfgSetVelCorrRatio 함수의 인자이며, 수정하고자 하는 속도 보정 비율값을 % 단위로 전달합니다.

▶ ***CorrRatio*** : cmxPmCfgGetVelCorrRatio 함수의 인자이며, 수정하고자 하는 속도 보정 비율값을 % 단위로 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러 처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

NAME

cmxPmCfgSetSeqMode/

cmxPmCfgGetSeqMode

- 시퀀스(Sequence) 모드 설정 및 반환

INFORMATION

Environment

Configuration Functions

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 2

위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxPmCfgSetSeqMode ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 SeqMode)

❒ VT_I4 cmxPmCfgGetSeqMode ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [out] VT_PI4 SeqMode)

DESCRIPTION

cmxPmCfgSetSeqMode() 함수는 현재 이송이 진행되고 있는 축에 새로운 이송 명령이 하달되었을 때, 처리를 어떻게 할 것인지에 대한 모드를 설정합니다. ceSDK 에서는 현재 이송 명령이 진행되고 있으므로 에러로 처리하는 모드와 이전 이송 명령이 완료될 때까지 내부적으로 루프를 돌면서 기다리다가 이전 명령이 완료되면 새로운 명령을 실행하는 모드를 지원합니다.

시퀀스(Sequence) 모드 설정은 모든 축에 공통적으로 적용됩니다.

cmxPmCfgGetSeqMode() 함수는 설정되어 있는 시퀀스 모드를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***SeqMode*** : 현재 이송이 진행되고 있는 축에 새로운 이송 명령이 하달되었을 때 이의 처리를 어떻게 할 것인지에 대한 시퀀스(Sequence) 모드를 설정 혹은 반환합니다. 이 값의 의미는 다음과 같습니다.

Value	Meaning
0 (ccmxSEQM_SKIP_RUN) **[Default]**	현재 이송이 진행되고 있는 축에 새로운 이송 명령이 하달되면 -5170 (ccmxERR_MOT_SEQ_SKIPPED) 에러 값과 함께 곧바로 반환됩니다. 다양한 에러 코드의 확인 본 매뉴얼의 부록 편에 명시되어 있습니다.
1 (ccmxSEQM_WAIT_RUN)	현재 이송이 진행되고 있는 축에 새로운 이송 명령이 하달되면 이송 함수 내부에서 루프를 돌면서 이전 이송이 완료되기를 기다리다가 이전 이송이 완료되면 현재

	하달된 이송 명령을 수행합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러 처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

REFERENCE

시퀀스(Sequence) 모드가 ccmxSEQM_SKIP_RUN[Default]으로 설정된 경우에 이전 이송 명령이 아직 끝나지 않은 상태에서 새로운 이송 명령이 하달되면 에러 처리되고 해당 이송 명령은 실행되지 않습니다. 따라서 이 모드에서 이송 명령을 내릴 때 사용자는 이전의 이송 명령이 완료되었음을 확인하는 것이 바람직합니다.

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

long BoardID = 0;
long nSeqMode = 0;

//시퀀스 모드를 SKIP RUN 모드로 설정합니다.
cmxPmCfgSetSeqMode(BoardID, 3, ccmxSEQM_SKIP_RUN );

//현재 설정된 Sequence 모드를 반환합니다.
cmxPmCfgGetSeqMode(BoardID, 3, &nSeqMode );
```

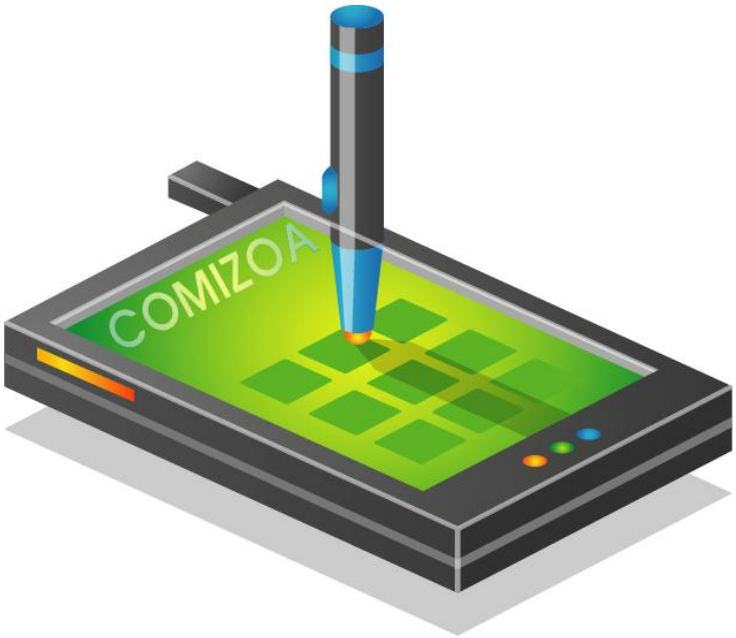
Chapter

15

Basic Pulse Motion Control Functions

기본(基本) 단축(單軸)과 다축(多軸) 모션 제어는 모션 제어에 있어 모터 구동의 첫걸음 이자, 가장 중요한 부분입니다. 고객(顧客) 여러분들께서는 단축(單軸) 모션을 활용하여, 다축 모션과 각종 보간 제어, 특수 조건 모션 제어를 구현하실 수 있습니다. 모션제어에 필요한 속도 설정과 기본적인 모션 제어를 위한 첫단계인 본 장을 잘 활용하시기 바랍니다.

기본 모션제어에 관련된 함수들을 소개(紹介)합니다. 이 장에서는 단축 모션 제어부터 다축(多軸) 모션제어, 기본 보간 제어, 원점복귀(原點復歸)등의 내용으로 구성되어 있습니다. 단축 제어는 단일 축을 독립적으로 제어하는 작업을 의미합니다. 다축(多軸) 제어는 다수의 복수(複數) 축을 제어하는 것을 의미하며, 보간제어는 직선 보간과 원호 보간(補間) 기능(機能)으로 구성되어 있습니다. 원점복귀(原點復歸) 기능을 통해 다양한 초기 위치를 결정할 수 있습니다.



15 기본 모션 제어 편

15.1 단축(Single-Axis) 모션 제어

각 축의 속도를 설정하고 이송 함수를 사용하여 이송 작업을 수행합니다. 그리고 필요에 따라 정지 함수를 사용하여 모션을 정지합니다.

15.1.1 함수 요약

"단축 모션 제어" 에 관련된 함수는 다음과 같습니다.

Summary of Functions
❒ VT_I4 cmxPmSxSetSpeedRatio ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 SpeedMode, [in] VT_I4 VelRatio, [in] VT_I4 AccRatio, [in] VT_I4 DecRatio) 단축 구동 시 해당 축에 대해 속도 방식 및 속도 비율을 설정합니다.
❒ VT_I4 cmxPmSxGetSpeedRatio ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 SpeedMode, [out] VT_PI4 VelRatio, [out] VT_PI4 AccRatio, [out] VT_PI4 DecRatio) 단축 구동 시 해당 축에 대해 속도 방식 및 속도 비율의 설정 상태를 반환합니다.
❒ VT_I4 cmxPmSxMoveStart ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_R8 Distance) 단축 상대 좌표 이송을 시작합니다. 이 함수는 모션 시작 후 바로 반환됩니다.
❒ VT_I4 cmxPmSxMove ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_R8 Distance, [in] VT_I4 IsBlocking) 단축 상대 좌표 이송을 시작합니다. 이 함수는 모션이 완료되기 전까지 반환되지 않습니다.
❒ VT_I4 cmxPmSxMoveToStart ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_R8 Position) 단축 절대 좌표 이송을 시작합니다. 이 함수는 모션 시작 후 바로 반환됩니다.
❒ VT_I4 cmxPmSxMoveTo ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_R8 Position, [in] VT_I4 IsBlocking) 단축 절대 좌표 이송을 시작합니다. 이 함수는 모션이 완료되기 전까지 반환되지 않습니다.
❒ VT_I4 cmxPmSxVMoveStart ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 Direction) 단축 연속 속도 이송을 시작합니다. 이 함수는 모션 시작 후 바로 반환됩니다.
❒ VT_I4 cmxPmSxStop ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 IsWaitComplete, [in] VT_I4 IsBlocking) 단축 이송을 감속 후 정지합니다. 이 정지함수는 이송완료시 까지 대기 할 수 있습니다.
❒ VT_I4 cmxPmSxStopEmg ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel) 단축 이송을 비상 정지 합니다. 이 정지 함수는 감속을 무시합니다.
❒ VT_I4 cmxPmSxIsDone ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 IsDone) 단축 이송의 완료를 확인합니다.
❒ VT_I4 cmxPmSxWaitDone ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel) 단축 이송의 완료 시점까지 대기합니다.
❒ VT_I4 cmxPmSxGetTargetPos ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PR8 Position) 해당 통합 축에 대해 마지막으로 이송한(상대 or 절대 좌표) 위치를 반환합니다.
❒ VT_I4 cmxPmSxSetOptIniSpeed ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_R8 IniSpeed) 단축 모션의 초기속도를 설정합니다.
❒ VT_I4 cmxPmSxGetOptIniSpeed ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PR8 IniSpeed) 단축 모션의 초기속도 설정을 반환합니다.
❒ VT_I4 cmxPmSxSetOptRdpOffset ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_R8 OffsetDist) 감속시작의 상대(상대) 위치를 설정합니다. RDP(Ramping Down Point) 의 오프셋(Offset)에 대해 설정합니다.
❒ VT_I4 cmxPmSxGetOptRdpOffset ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PR8 OffsetDist) 감속시작의 상대(상대) 위치를 반환합니다. RDP(Ramping Down Point) 의 오프셋(Offset)에 대해 반환합니다.
❒ VT_I4 cmxPmSxSetCorrection ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 CorrMode, [in] VT_R8 CorrAmount, [in] VT_R8 CorrVel, [in] VT_I4 CntrMask) 단축 모션의 백래쉬 혹은 슬립 보정을 위해 설정하는 함수입니다.

❒ VT_I4 cmxPmSxGetCorrection ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 CorrMode, [out] VT_PR8 CorrAmount, [out] VT_PR8 CorrVel, [out] VT_PI4 CntrMask) 단축 모션의 백래쉬 혹은 슬립 보정의 설정을 반환하는 함수입니다.
❒ VT_I4 cmxPmSxSetOptSyncMode ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 Mode, [in] VT_I4 RefAxis, [in] VT_I4 Condition) 지정한 다른 축(Other Axis)과 동기 시작 환경을 구성(構成)합니다.
❒ VT_I4 cmxPmSxGetOptSyncMode ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 Mode, [out] VT_PI4 RefAxis, [out] VT_PI4 Condition) 지정한 다른 축(Other Axis)과 동기 시작에 대한 설정(設定)을 반환(返還)합니다.
❒ VT_I4 cmxPmSxSetOptSyncOut ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 Mode, [in] VT_I4 DoChan_local, [in] VT_I4 DoLogic) 모션 속도 구간 별 디지털 출력(出力)에 대한 설정합니다. 지정한 축의 속도구간(速度區間)에서 머신비젼(Machine Vision) 등의 동기 시작 신호(信號)로 사용될 수 있습니다.
❒ VT_I4 cmxPmSxGetOptSyncOut ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 Mode, [out] VT_PI4 DoChan_local, [out] VT_PI4 DoLogic) 모션 속도 구간 별 디지털 출력(出力)에 대한 설정을 반환합니다. 지정한 축의 속도구간(速度區間)에서 머신비젼(Machine Vision) 등의 동기 시작 신호(信號)로 사용 되는 설정에 대해 반환합니다.

15.1.2 함수 설명

NAME

cmxPmSxSetSpeedRatio/

cmxPmSxGetSpeedRatio

- 단축이송 속도 비율 설정 및 반환

INFORMATION

Single Axis Control

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 3

다소 주의

SYNOPSIS

❒ VT_I4 cmxPmSxSetSpeedRatio

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 SpeedMode, [in] VT_I4 VelRatio, [in] VT_I4 AccRatio, [in] VT_I4 DecRatio)

❒ VT_I4 cmxPmSxGetSpeedRatio

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 SpeedMode, [out] VT_PI4 VelRatio, [out] VT_PI4 AccRatio, [out] VT_PI4 DecRatio)

DESCRIPTION

cmxPmSxSetSpeedRatio() 함수는 단축 구동 시 해당 축에 대한 속도 모드 및 속도 비율을 설정 합니다. 모션 속도는 기준속도의 비율로 설정이 가능하며, 이것은 cmxPmCfgSetSpeedPattern() 함수에 의해서 설정된 기준 속도를 의미합니다.

cmxPmSxGetSpeedRatio() 함수는 단축 구동 시 해당 축에 대해 설정되어 있는 속도 모드 및 속도 비율을 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***SpeedMode*** : cmxPmSxSetSpeedRatio() 함수의 인자이며, 속도모드의 설정 값 입니다. 다음과 같은 설정 값을 가집니다.

Value	Meaning

0 (cmxMODE_C)	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 (cmxMODE_T)	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 (cmxMODE_S)	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.
-1 (ccmxSMODE_KEEP)	이전 속도 모드를 그대로 유지합니다. 즉, 속도모드를 변경하지 않습니다.

▶ ***SpeedMode*** : cmxPmSxGetSpeedRatio() 함수의 인자이며, 설정된 속도모드를 반환합니다.

Value	Meaning
0 (cmxMODE_C)	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 (cmxMODE_T)	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 (cmxMODE_S)	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.

▶ ***VelRatio*** : 작업 속도 비율(Ratio) 을 설정 혹은 반환합니다. 이 값의 단위는 %입니다.

▶ ***AccRatio*** : 가속도 비율(Ratio) 을 설정 혹은 반환합니다. 이 값의 단위는 %입니다.

▶ ***DecRatio*** : 감속도 비율(Ratio) 을 설정 혹은 반환합니다. 이 값의 단위는 %입니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러 처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

SEE ALSO

cmxPmCfgSetSpeedPattern, cmxPmCfgGetSpeedPattern

REFERENCE

속도 비율(Ratio) 의 정확한 의미를 알고 싶습니다.

속도 설정은 비율로 설정이 됩니다. 비율에 의한 속도 값은 기준 값에 배수(Multiplication)가 되거나 제법(Division) 이 된 속도 값을 의미합니다. 기준이 되는 속도 값은 cmxPmCfgSetSpeedPattern() 으로 설정됩니다.

㈜ 커미조아 ceSDK 에서는 기준 속도(Standard Speed) 개념을 이용하고 있습니다. 전체 모션 속도는 기준속도의 비율로 설정이 가능하며, 이것은 cmxPmCfgSetSpeedPattern() 함수에 의해서 설정된 기준 속도를 의미합니다. ㈜ 커미조아의 ceSDK 는 기준속도의 값을 비율(Ratio) 로 설정할 수 있는 커다란 이점을 가지고 있습니다.

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

long BoardID = 0;
long nChannel = 0;          //0 번 축으로 설정합니다.

/*단축 모션의 작업 속도 비율을 설정합니다.
cmxPmCfgSetSpeedPattern()함수를 통해서 설정된 속도를 기준으로 비율이 적용됩니다.*/
cmxPmSxSetSpeedRatio(BoardID, 3, nChannel, cmxMODE_T, 100, 100, 100 );

long nSpeedMode = 0;
double fVelRatio = 0.0f, fAccRatio = 0.0f, fDecRatio = 0.0f;

//0 번 축의 설정된 단축 모션 작업 속도 비율을 반환합니다.
cmxPmSxGetSpeedRatio (BoardID, 3, nChannel, &nSpeedMode, &fVelRatio, &fAccRatio, &fDecRatio );
```

NAME

cmxPmSxMove / cmxPmSxMoveStart

- 단축 상대좌표 이송

INFORMATION

Single Axis Control

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 3

이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인 합니다.

SYNOPSIS

❒ VT_I4 cmxPmSxMove ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_R8 Distance, [in] VT_I4 IsBlocking)

❒ VT_I4 cmxPmSxMoveStart ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_R8 Distance)

DESCRIPTION

cmxPmSxMove()/cmxPmSxMoveStart() 함수는 하나의 축에 대하여 현재의 위치에서 지정한 거리(상대위치)만큼 이송을 수행합니다.

cmxPmSxMove() 함수는 모션이 완료되기 전까지 반환되지 않으며, cmxPmSxMoveStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***Distance*** : 이송할 거리를 지정합니다. 이 값은 현재의 위치에 대한 상대 좌표이며, 거리의 단위는 논리적 거리(Unit distance) 단위를 사용합니다.
"Unit distance"를 '1'로 한 경우에 거리의 단위는 Pulse 수가 됩니다. 즉, Distance 값 '1'은 1 Pulse 출력을 의미합니다.

▶ ***IsBlocking*** : cmxPmSxMove() 함수의 인자이며, 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다. 단, 쓰레드 내에서 실행할 때는 이 값을 1(cmxTRUE)로 설정해 주어야 합니다.

Value	Meaning
0 (cmxFALSE)	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
1 (cmxTRUE)	블록(Blocking)을 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러 처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

SEE ALSO

cmxPmSxMoveTo, cmxPmSxMoveToStart, cmxPmSxVMoveStart

REFERENCE

□ 논리적 단위 거리는 cmxPmCfgSetUnitDist() 함수에 의해 결정됩니다.

□ cmxPmSxMoveStart() 함수를 사용하는 경우에는 cmxPmSxIsDone() 함수나 cmxPmSxWaitDone() 함수를 사용하여 모션의 완료를 확인할 수 있습니다.

□ cmxPmSxMove() 함수를 사용하는 경우에는 내부적으로 루프를 수행하면서 모션이 완료되기를 기다리는데, 이때 "Blocking Mode" 설정에 따라 윈도우 이벤트를 처리하는 방식이 달라집니다. 그러나 일반적으로 윈도우의 작업 쓰레드(Work Thread) 에서는 블록모드를 사용하여, 함수 내부에서 지연 없이 쓰레드 내부의 작업에 집중할 수 있도록 설정하는 것이 바람직합니다.

□ INP 입력신호가 Enable 로 설정되었으면 Command 펄스 출력이 완료되어도 INP 입력이 ON 이 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환되지 않습니다.

□ 스텝 드라이브를 사용 중인 고객님들께서는 다음을 참조해 주십시오.
스텝 드라이브는 INP 출력이 없는 경우가 일반적인데, 고객님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스텝 드라이브를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다. 고객 여러분들께서는 스텝 드라이브 사용시에 이점을 주의하시기 바랍니다.

□ 서보 드라이브의 LSP , LSN 신호를 사용 중인 고객님께서는 다음을 참조해 주십시오.
서보드라이브의 입력 신호 중 하나인 EL(End of Limit) 신호는 저희 ㈜ 커미조아 모션 컨트롤러뿐만 아니라 서보드라이브에도 전달 될 수 있도록 설정할 수 있습니다. 통상적으로 LSP 신호와 LSN 신호로

불리어 지는 이 신호는 실제 기구물에서 양의 방향(Positive Direction) 혹은 음의 방향(Negative Direction) 에 장착되어 있는 EL(End of Limit) 신호를 서보 드라이브 측에 전달하기 위한 용도로 사용됩니다.

그러나, 모션 소프트웨어에서 INP 설정이 되어 있는 경우 EL 신호가 검출 된 후에 일부 서보 드라이브에서는 진행 방향에서 정지한 후 더 이상 움직이지 않는 상황이 발생하며, 이 상황에서 INP 신호가 출력되지 않아, 모션 이송이 완료되지 못하고, 명시적으로 STOP 명령을 통해 강제적으로 모션 종료가 되는 현상이 발생할 수 있습니다. 이 현상은 EL 모드를 통해 원점 복귀를 하는 상황에서도 발생할 수 있습니다.
따라서, 이러한 경우에는 반드시 원점 복귀나 EL 검출 시에 인터럽트 이벤트나 타이머를 통해 INP 를 무기한 대기하는 현상에 대해서 적절히 대처하시거나 INP 신호 사용을 배제 해야 합니다.

	윈도우 이벤트라는 것은 무엇입니까?
	윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용 프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

long nChannel = 0; //0 번 축으로 설정합니다.
long BoardID = 0;

//0 번 축의 기본 속도를 설정합니다.
cmxPmCfgSetSpeedPattern(BoardID, 3, nChannelv, cmxMODE_T, 10000, 50000, 50000 );

//0 번 축의 속도 비율을 설정합니다.
//cmxPmCfgSetSpeedPattern()에서 설정된 값의 80%로 구동합니다.
//속도 패턴은 설정된 값을 변경하지 않고 그대로 사용합니다.
cmxPmSxSetSpeedRatio(BoardID, 3, nChannel, ccmxSMODE_KEEP, 80, 80, 80 );

//0 번 축을 현재 위치에서 10000 만큼 이송합니다.
cmxPmSxMoveStart(BoardID, 3, nChannel, 10000 );

//블록 모드를 cmxFALSE 로 하면 UI 메시지 처리가 가능합니다.
cmxPmSxWaitDone(BoardID, 3, nChannel, cmxFALSE );

//cmxPmSxMoveStart(), cmxPmSxWaitDone()을 아래 코드로 대체 할 수도 있습니다.
//cmxPmSxMove(BoardID, 3, nChannel, 10000, cmxFALSE );
```

NAME

cmxPmSxMoveTo / cmxPmSxMoveToStart

- 단축 절대좌표 이송

INFORMATION

Single Axis Control

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 3

이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인 합니다.

SYNOPSIS

- VT_I4 cmxPmSxMoveTo ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_R8 Position, [in] VT_I4 IsBlocking)
- VT_I4 cmxPmSxMoveToStart ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_R8 Position)

DESCRIPTION

cmxPmSxMoveTo()/cmxPmSxMoveToStart() 함수는 하나의 축에 대하여 지정한 절대 좌표로의 이송을 수행합니다.

cmxPmSxMoveTo() 함수는 모션이 완료되기 전까지 반환되지 않으며, cmxPmSxMoveToStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***Position*** : 이동할 절대 좌표 값을 지정합니다. 거리의 단위는 논리적 거리(Logic distance) 단위를 사용합니다.
"Unit distance"를 '1'로 한 경우에 거리의 단위는 Pulse 수가 됩니다. 즉, Distance 값 '1'은 1 Pulse 출력을 의미합니다.

▶ ***IsBlocking*** : cmxPmSxMoveTo() 함수의 인자이며, 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다. 단, 쓰레드 내에서 실행할 때는 이 값을 1(cmxTRUE)로 설정해 주어야 합니다.

Value	Meaning
0 (cmxFALSE)	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
1 (cmxTRUE)	블록(Blocking)을 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러 처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

SEE ALSO

cmxPmSxMove, cmxPmSxMoveStart, cmxPmSxVMoveStart

REFERENCE

□ 논리적 단위 거리는 cmxPmCfgSetUnitDist() 함수에 의해 결정됩니다.

□ cmxPmSxMoveToStart() 함수를 사용하는 경우에는 cmxPmSxIsDone() 함수나 cmxPmSxWaitDone() 함수를 사용하여 모션의 완료를 확인할 수 있습니다.

□ cmxPmSxMoveTo() 함수를 사용하는 경우에는 내부적으로 루프를 수행하면서 모션이 완료되기를 기다리는데, 이때 "Blocking Mode" 설정에 따라 윈도우 이벤트를 처리하는 방식이 달라집니다. 그러나 일반적으로 윈도우의 작업 쓰레드(Work Thread) 에서는 블록모드를 사용하여, 함수 내부에서 지연 없이 쓰레드 내부의 작업에 집중할 수 있도록 설정하는 것이 바람직합니다.

□ INP 입력신호가 Enable 로 설정되었으면 Command 펄스 출력이 완료되어도 INP 입력이 ON 이 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환되지 않습니다.

□ 스텝 드라이브를 사용 중인 고객님들께서는 다음을 참조해 주십시오.
스텝 드라이브는 INP 출력이 없는 경우가 일반적인데, 고객님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스텝 드라이브를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다. 고객 여러분들께서는 스텝 드라이브 사용시에 이점을 주의해 주시기를 부탁 드립니다.

□ 서보 드라이브의 LSP , LSN 신호를 사용 중인 고객님께서는 다음을 참조해 주십시오.
서보드라이브의 입력 신호 중 하나인 EL(End of Limit) 신호는 저희 ㈜ 커미조아 모션 컨트롤러뿐만 아니라 서보드라이브에도 전달 될 수 있도록 설정할 수 있습니다. 통상적으로 LSP 신호와 LSN 신호로 불리어 지는 이 신호는 실제 기구물에서 양의 방향(Positive Direction) 혹은 음의 방향(Negative Direction) 에 장착되어 있는 EL(End of Limit) 신호를 서보 드라이브 측에 전달하기 위한 용도로 사용됩니다.

그러나, 모션 소프트웨어에서 INP 설정이 되어 있는 경우 EL 신호가 검출 된 후에 일부 서보 드라이브에서는 진행 방향에서 정지 한 후 더 이상 움직이지 않는 상황이 발생하며, 이 상황에서 INP 신호가 출력되지 않아, 모션 이송이 완료되지 못하고, 명시적으로 STOP 명령을 통해 강제적으로 모션 종료가 되는 현상이 발생할 수 있습니다. 이 현상은 EL 모드를 통해 원점 복귀를 하는 상황에서도 발생할 수 있습니다.

따라서, 이러한 경우에는 반드시 원점 복귀나 EL 검출 시에 인터럽트 이벤트나 타이머를 통해 INP 를 무기한 대기하는 현상에 대해서 적절히 대처하시거나 INP 신호 사용을 배제 해야 합니다.

윈도우 이벤트라는 것은 무엇입니까?

윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용 프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

long BoardID = 0;
long nChannel = 0; //0 번 축으로 설정합니다.

//0 번 축의 기본 속도를 설정합니다.
cmxPmCfgSetSpeedPattern(BoardID, 3, nChannel, cmxMODE_T, 10000, 50000, 50000 );

/*0 번 축의 속도 비율을 설정합니다.
cmxPmCfgSetSpeedPattern()에서 설정된 값의 80%로 구동합니다.
속도 패턴은 설정된 값을 변경하지 않고 그대로 사용합니다.*/
cmxPmSxSetSpeedRatio(BoardID, 3, nChannel, ccmxSMODE_KEEP, 80, 80, 80 );

//0 번 축을 절대좌표 10000 으로 이송합니다.
cmxPmSxMoveToStart(BoardID, 3, nChannel, 10000 );

//블록 모드를 cmxFALSE 로 하면 UI 메시지 처리가 가능합니다.
cmxPmSxWaitDone(BoardID, 3, nChannel, cmxFALSE );

//cmxPmSxMoveToStart(), cmxPmSxWaitDone()을 아래 코드로 대체 할 수도 있습니다.
//cmxPmSxMoveTo(BoardID, 3, nChannel, 10000, cmxFALSE );
```

NAME

cmxPmSxVMoveStart

- 단축 연속 속도 이송

INFORMATION

Single Axis Control

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 3

☺ 이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인 합니다.

SYNOPSIS

❒ VT_I4 cmxPmSxVMoveStart ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 Direction)

DESCRIPTION

cmxPmSxVMoveStart() 함수는 작업속도까지 가속한 후에 작업속도를 유지하며 정지 함수가 호출될 때까지 지정한 방향으로의 모션을 계속 수행합니다. 이 함수는 모션을 시작시킨 후에 바로 반환됩니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***Direction*** : 모션의 방향을 설정합니다.

Value	Meaning
0 (cmDIR_N)	(-) 방향 => Negative direction
1 (cmDIR_P)	(+) 방향 => Positive direction

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

long BoardID = 0;
long nChannel = 0; //0 번 축으로 설정합니다.

//0 번 축의 기본 속도를 설정합니다.
cmxPmCfgSetSpeedPattern(BoardID, 3, nChannel, cmxMODE_T, 10000, 50000, 50000 );

//0 번 축의 속도 비율을 50%로 설정합니다.
//속도 패턴은 cmxPmCfgSpeedPatternset() 함수를 통해 설정된 값을 변경하지 않습니다.
cmxPmSxSetSpeedRatio(BoardID, 3, nChannel, ccmxSMODE_KEEP, 50, 50, 50 );

/*(-)방향으로 0 번 축을 이송합니다.
cmxPmSxStop() 혹은 cmxPmSxStopEmg()함수를 호출하기 전까지 계속 이송합니다.*/
cmxPmSxVMoveStart(BoardID, 3, nChannel, cmDIR_N );
```

NAME

cmxPmSxStop / cmxPmSxStopEmg

- 단축 이송 정지
- 단축 비상 정지

INFORMATION

Single Axis Control

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 3

정지 함수

고속 이송시에 급 정지, 비상 정지를 주의 하십시오. 기구물의 손상이나 안전사고에 원인이 될 수 있습니다.

SYNOPSIS

- VT_I4 cmxPmSxStop ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 IsWaitComplete, [in] VT_I4 IsBlocking)
- VT_I4 cmxPmSxStopEmg ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel)

DESCRIPTION

cmxPmSxStop()/cmxPmSxStopEmg() 함수는 지정한 축에 대한 모션을 정지합니다. cmxPmSxStop() 함수는 정지 시에 감속 후 정지를 수행하며, cmxPmSxStopEmg() 함수는 감속 없이 즉시 정지를 수행합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***IsWaitComplete*** : cmxPmSxStop() 함수의 인자이며, 이 동작이 완료될 때까지 함수를 반환할 것인지를 결정합니다.

Value	Meaning
0 (cmxFALSE)	대상 축의 정지가 완료될 때까지 대기하지 않고 함수를 벗어납니다.
1 (cmxTRUE)	대상 축의 정지가 완료될 때까지 대기합니다.

▶ ***IsBlocking*** : cmxPmSxStop() 함수의 인자이며, 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다.

Value	Meaning
0 (cmxFALSE)	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
1 (cmxTRUE)	블록(Blocking)을 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

long BoardID = 0;
long nChannel = 0; //0 번 축으로 설정합니다.

//0 번 축을 (+)방향으로 이송합니다.
cmxPmSxVMoveStart(BoardID, nAxis, cmDIR_P );

Sleep(3000);

/*0 번 축을 정지 시킵니다.
IsWaitComplete 인자를 cmxTRUE 로 설정하면, 대상 축이 정지를 완료 하기 전까지
함수가 반환되지 않습니다.*/
cmxPmSxStop(BoardID, 3, nChannel, cmxTRUE, cmxFALSE );

//cmxPmSxStopEmg() 함수를 사용할 경우 감속 없이 즉시 정지합니다.
//cmxPmSxStopEmg(BoardID, 3, nChannel );
```

NAME

cmxPmSxIsDone

- 단축 모션 완료 확인

INFORMATION

- Single Axis Control
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 3
- ☺ 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxPmSxIsDone ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 IsDone)

DESCRIPTION

cmxPmSxIsDone() 함수는 단일 축에 대하여 모션 완료를 확인합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***IsDone*** : 이 매개 변수를 통해 모션 작업이 완료되었는지를 판단할 수 있습니다.

Value	Meaning
0 (cmxFALSE)	모션작업이 완료되지 않음.
1 (cmxTRUE)	모션작업이 완료 됨.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

SEE ALSO

cmxPmSxWaitDone

REFERENCE

□ INP 입력신호가 Enable 로 설정되었으면 Command 펄스 출력이 완료되어도 INP 입력이 ON 이 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환되지 않습니다.

□ 스텝 드라이브를 사용 중인 고객님들께서는 다음을 참조해 주십시오.
스텝 드라이브는 INP 출력이 없는 경우가 일반적인데, 고객님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스텝 드라이브를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다. 고객 여러분들께서는 스텝 드라이브 사용시에 이점을 주의해 주시기를 부탁 드립니다.

□ 서보 드라이브의 LSP , LSN 신호를 사용 중인 고객님께서는 다음을 참조해 주십시오.
서보드라이브의 입력 신호 중 하나인 EL(End of Limit) 신호는 저희 ㈜ 커미조아 모션 컨트롤러뿐만 아니라 서보드라이브에도 전달 될 수 있도록 설정할 수 있습니다. 통상적으로 LSP 신호와 LSN 신호로 불리어 지는 이 신호는 실제 기구물에서 양의 방향(Positive Direction) 혹은 음의 방향(Negative Direction) 에 장착되어 있는 EL(End of Limit) 신호를 서보 드라이브 측에 전달하기 위한 용도로 사용됩니다.

그러나, 모션 소프트웨어에서 INP 설정이 되어 있는 경우 EL 신호가 검출 된 후에 일부 서보 드라이브에서는 진행 방향에서 정지 한 후 더 이상 움직이지 않는 상황이 발생하며, 이 상황에서 INP 신호가 출력되지 않아, 모션 이송이 완료되지 못하고, 명시적으로 STOP 명령을 통해 강제적으로 모션 종료가 되는 현상이 발생할 수 있습니다.. 이 현상은 EL 모드를 통해 원점 복귀를 하는 상황에서도 발생할 수 있습니다.

따라서, 이러한 경우에는 반드시 원점 복귀나 EL 검출 시에 인터럽트 이벤트나 타이머를 통해 INP 를 무기한 대기하는 현상에 대해서 적절히 대처하시거나 INP 신호 사용을 배제 해야 합니다.

윈도우 이벤트라는 것은 무엇입니까?

윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용 프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

long BoardID = 0;
long nChannel = 0; //0 번 축으로 설정합니다.

//0 번 축을 현재 위치에서 10000 펄스 만큼 이송합니다.
cmxPmSxMoveStart(BoardID, 3, nChannel, 10000 );

long nIsDone = cmxFALSE;

while( !nIsDone )
```

```
{
    //0 번 축의 모션 완료 여부를 판단합니다.
    cmxPmSxIsDone(BoardID, 3, nChannel, &nIsDone );
}
```

NAME

cmxPmSxWaitDone

- 단축 모션 완료 대기

INFORMATION

- Single Axis Control
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 3
- ☺ 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxPmSxWaitDone ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel)

DESCRIPTION

cmxPmSxWaitDone() 함수는 단일 축에 대하여 모션이 완료될 때까지 기다립니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

SEE ALSO

cmxPmSxIsDone

REFERENCE

□ INP 입력신호가 Enable 로 설정되었으면 Command 펄스 출력이 완료되어도 INP 입력이 ON 이 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환되지 않습니다.

□ 스텝 드라이브를 사용 중인 고객님들께서는 다음을 참조해 주십시오.
스텝 드라이브는 INP 출력이 없는 경우가 일반적인데, 고객님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스텝 드라이브를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다. 고객 여러분들께서는 스텝 드라이브 사용시에 이점을 주의해 주시기를 부탁 드립니다.

□ 서보 드라이브의 LSP , LSN 신호를 사용 중인 고객님께서는 다음을 참조해 주십시오.
서보드라이브의 입력 신호 중 하나인 EL(End of Limit) 신호는 저희 ㈜ 커미조아 모션 컨트롤러뿐만 아니라 서보드라이브에도 전달 될 수 있도록 설정할 수 있습니다. 통상적으로 LSP 신호와 LSN 신호로 불리어 지는 이 신호는 실제 기구물에서 양의 방향(Positive Direction) 혹은 음의 방향(Negative Direction) 에 장착되어 있는 EL(End of Limit) 신호를 서보 드라이브 측에 전달하기 위한 용도로 사용됩니다.

그러나, 모션 소프트웨어에서 INP 설정이 되어 있는 경우 EL 신호가 검출 된 후에 일부 서보 드라이브에서는 진행 방향에서 정지 한 후 더 이상 움직이지 않는 상황이 발생하며, 이 상황에서 INP 신호가 출력되지 않아, 모션 이송이 완료되지 못하고, 명시적으로 STOP 명령을 통해 강제적으로 모션 종료가 되는 현상이 발생할 수 있습니다.. 이 현상은 EL 모드를 통해 원점 복귀를 하는 상황에서도 발생할 수 있습니다.

따라서, 이러한 경우에는 반드시 원점 복귀나 EL 검출 시에 인터럽트 이벤트나 타이머를 통해 INP 를 무기한 대기하는 현상에 대해서 적절히 대처하시거나 INP 신호 사용을 배제 해야 합니다.

윈도우 이벤트라는 것은 무엇입니까?

윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용 프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

long BoardID = 0;
long nChannel = 0; //0 번 축으로 설정합니다.

//0 번 축을 현재 위치에서 10000 만큼 이송합니다.
if( cmxPmSxMoveStart(BoardID, 3, nChannel, 10000 ) != ERR_NONE )
{
    OutputDebugString( "SxMoveStart fail!" );
    return;
}

//모션이 완료 될 때까지 기다립니다.
if( cmxPmSxWaitDone(BoardID, 3, nChannel, cmxFALSE ) != ERR_NONE )
{
    OutputDebugString( "SxWaitDone fail!" );
    return;
}
```

NAME

cmxPmSxGetTargetPos

- 마지막 이송 (상대 or 절대 좌표) 위치 반환

INFORMATION

- Single Axis Control
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 3
- ☺ 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxPmSxGetTargetPos

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PR8 Position)

DESCRIPTION

대상 축에 대하여 마지막으로 수행한 이송 명령의 위치(상대 or 절대 좌표)를 반환합니다. 대상 함수는 cmxPmSxMove, cmxPmSxMoveStart, cmxPmSxMoveTo, cmxPmSxMoveToStart 입니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***Position*** : 대상 축에 대하여 마지막으로 이송한 (상대 혹은 절대 좌표) 위치를 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

long BoardID = 0;
long nChannel = 0; //0 번 축으로 설정합니다.
//0 번 축을 현재 위치에서 1000 만큼 이송합니다.
```

```
cmxPmSxMove(BoardID, 3, nChannel, 1000, cmxFALSE );

double fPosition = 0.0f;
//0 번 축이 마지막으로 이송한 위치를 반환합니다.
cmxPmSxGetTargetPos(BoardID, 3, nChannel, &fPosition );
```

NAME

cmxPmSxSetOptIniSpeed/

cmxPmSxGetOptIniSpeed

- 단축 모션 초기 속도 설정 및 반환

INFORMATION

- Single Axis Control
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 3
- ☺ 위험 요소 없음

SYNOPSIS

- ❒ VT_I4 cmxPmSxSetOptIniSpeed ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_R8 IniSpeed)
- ❒ VT_I4 cmxPmSxGetOptIniSpeed ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PR8 IniSpeed)

DESCRIPTION

cmxPmSxSetOptIniSpeed()/cmxPmSxGetOptIniSpeed() 함수는 모션의 초기 속도를 설정하거나 설정 값을 얻어 옵니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***IniSpeed*** : 모션의 초기 속도를 설정 혹은 반환하기 위한 매개변수입니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"
```

```
long BoardID = 0;
long nChannel = 0; //0 번 축으로 설정합니다.

//0 번 축의 초기 속도를 설정합니다.
cmxPmSxSetOptIniSpeed(BoardID, 3, nChannel, 100 );

double fIniSpeed = 0.0f;
//0 번 축의 초기 속도 설정 값을 반환합니다.
cmxPmSxGetOptIniSpeed(BoardID, 3, nChannel, &fIniSpeed );
```

NAME

cmxPmSxSetOptRdpOffset/

cmxPmSxGetOptRdpOffset

- 감속(減速) 시작 산대 위치 설정

INFORMATION

Single Axis Control

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 3

다소 주의

응용 분야가 상당히 많은 함수 이므로, ㈜ 커미조아를 통해 기술지원을 받으시면 더욱 자세하게 이해(理解) 하실 수 있습니다.

SYNOPSIS

- ❒ VT_I4 cmxPmSxSetOptRdpOffset ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_R8 OffsetDist)
- ❒ VT_I4 cmxPmSxGetOptRdpOffset ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PR8 OffsetDist)

DESCRIPTION

cmxPmSxSetOptSetOptRdpOffset() 함수는 RDP(Ramping Down Point)의 오프셋(Offset)을 설정하는 함수입니다. 여기서 RDP(Ramping Down point)는 감속을 시작하는 위치를 의미합니다. 기본적으로 이 오프셋값은 0 으로 설정됩니다. 따라서 목표좌표에 도달하는 시점에 감속이 완료되게 됩니다.
그런데, RDP 오프셋을 양의 값으로 설정하면 지정한 오프셋 위치만큼 감속을 일찍 시작하게됩니다. 그러면 감속이 완료되는 시점에 목표좌표보다 모자란 위치가 되므로 나머지 잔여 이송은 초기속도로 이송하게 됩니다. 이러한 모션은 프레스 장비와 같이 이송의 마지막 순간에 저속으로 이송해야하는 경우에 유용하게 사용할 수 있습니다. 반대로, RDP 오프셋을 음의 값으로 설정하면 지정한 오프셋 위치만큼 감속을 늦게 시작하게 됩니다. 따라서 목표좌표에 도달하는 시점에 초기속도보다 높은 속도에서 감속이 완료되게 됩니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***OffsetDist*** : cmxPmSxSetOptRdpOffset 함수의 인자 이며, RdpOffset 을 적용할 거리를 설정합니다. 이 거리는 논리적 거리 단위입니다.

▶ ***OffsetDist*** : cmxPmSxGetOptRdpOffset 함수의 인자 이며, RdpOffset 을 적용할 거리를 반환합니다. 이 거리는 논리적 거리 단위입니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

NAME

cmxPmSxSetCorrection/

cmxPmSxGetCorrection

- 백래쉬 / 슬립 보정 설정 및 반환

INFORMATION

Single Axis Control

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 3

위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxPmSxSetCorrection

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 CorrMode, [in] VT_R8 CorrAmount, [in] VT_R8 CorrVel, [in] VT_I4 CntrMask)

❒ VT_I4 cmxPmSxGetCorrection

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 CorrMode, [out] VT_PR8 CorrAmount, [out] VT_PR8 CorrVel, [out] VT_PI4 CntrMask)

DESCRIPTION

cmxPmSxSetCorrection () 함수는 백래쉬(Backlash) 또는 슬립(Slip)에 대한 보정 환경을 설정합니다. 구조적으로 백래쉬나 슬립 현상이 심하게 일어나는 경우에는 이에 대한 보정이 필요할 수 있습니다. 백래쉬는 일반적으로 모터의 구동 방향이 바뀔 때 발생됩니다. 따라서 ㈜커미조아 모션컨트롤러의 백래쉬 보정은 모터의 제어 방향이 바뀔 때에만 적용됩니다. 백래쉬 보정을 활성화하면 모션컨트롤러에서 지령되는 이동 명령의 이동 방향이 이전의 이동 방향과 다른 경우에 자동적으로 백래쉬 보정 설정에 따라 보정 펄스가 출력된 후에 지정된 이동을 수행합니다. 이 설정은 단축 구동뿐 아니라, 보간 구동에서도 적용됩니다.

슬립은 일반적으로 정지 후 재 기동 시에 발생합니다. 따라서 슬립 보정은 이동방향에 상관없이 기동 시에 보정 펄스가 출력됩니다. 슬립 보정을 활성화한 후에 이동명령이 하달되면 모션컨트롤러는 슬립 보정 설정에 따라 보정 펄스가 출력된 후에 지정된 이동을 수행합니다.

cmxPmSxGetCorrection () 함수는 백래쉬(Backlash) 또는 슬립(Slip) 보정 설정 상태를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***CorrMode*** : 보정 모드를 설정 혹은 반환합니다. 보정 모드의 의미는 다음과 같습니다.

BIT No.	Meaning
0 (ccmxCORR_DIS)	보정기능을 비활성화합니다.
1 (ccmxCORR_BACK)	보정모드를 백래쉬 보정모드로 설정합니다.
2 (ccmxCORR_SLIP)	보정모드를 슬립 보정모드로 설정합니다.

▶ ***CorrAmount*** : 보정 펄스의 수를 설정 혹은 반환합니다. 단, 이 값은 논리적 단위 거리로 설정해야 합니다. 따라서 "Unit distance"(Du)를 '1'이 아닌 값으로 설정한 경우에 실제 출력되는 보정 펄스 수(Nc)는 다음과 같습니다.

$$Nc = CorrAmount * Du$$

그리고 보정펄스 수(Nc)는 0 ~ 4095 의 값이어야 합니다.

▶ ***CorrVel*** : 보정펄스의 출력 주파수를 설정 혹은 반환합니다. 단, 이 값은 논리적 속도 단위로 설정해야 합니다. 따라서 "Unit speed"(Vu)를 '1'이 아닌 값으로 설정한 경우에 실제 출력 주파수(Fc)는 다음과 같습니다.

$$Fc\ (PPS) = CorrVel * Vu$$

그리고, 하드웨어적으로 보정펄스 출력 주파수를 설정하는 레지스터는 원점복귀 특정 모드의 Reverse Velocity (Vr)과 같은 레지스터를 사용합니다. 따라서 원점복귀 Vr 이 보정펄스 출력시의 속도와 다른 경우에는 원점복귀를 수행한 후에 이 함수를 다시 수행해 주어야 합니다.

▶ ***CntrMask*** : 보정펄스가 출력되는 동안에 각 카운터의 동작 여부를 아래의 표와 같이 각 비트 별로 설정 혹은 반환합니다. 예를 들어 이 값의 BIT0 을 1 로 하면 보정펄스가 출력되는 동안에도 Command Counter 의 값은 증가 또는 감소합니다. 그리고 BIT0 을 0 으로 하면 보정펄스가 출력되는 동안에는 Command Counter 의 값이 변화하지 않습니다.

Value	Meaning
BIT0	1 : 보정펄스 출력 시에 Command Counter 가 동작하는 모드입니다.
BIT1	1 : 보정펄스 출력 시에 Feedback Counter 가 동작하는 모드입니다.
BIT2	1 : 보정펄스 출력 시에 Deviation Counter 가 동작하는 모드입니다.
BIT3	1 : 보정펄스 출력 시에 General Counter 가 동작하는 모드입니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

long BoardID = 0;
long nChannel = 0; //0 번 축으로 설정합니다.

//0 번 축에 대해서 백래쉬 보정 모드로 설정 합니다.
//( 보정 펄스 수 : 1000, 보정 펄스 출력 속도 : 1000 PPS )
cmxPmSxSetCorrection (BoardID, 3, nChannel, ccmxCORR_BACK, 1000, 1000, 0x0 );

//이전에 (-)방향으로 이동을 수행하였다면 (+)방향으로 이동할 때
//(방향이 바뀔 때) 백래쉬 보정을 수행합니다.
//1000 PPS 의 속도로 (+)1000 펄스를 출력한 후에 지정한 SxMove()가 수행됩니다.

cmxPmSxMove(BoardID, 3, nChannel, 10000, cmxFALSE );

//이동 방향이 이전과 동일하므로 아래에서는 백래쉬 보정을 하지 않습니다.
cmxPmSxMove(BoardID, 3, nChannel, 10000, cmxFALSE );

//이동 방향이 전환되므로 백래쉬 보정을 수행합니다.
//1000PPS 의 속도로 (-)1000 펄스를 출력 한 후에 지정한 SxMove()가 수행됩니다.
cmxPmSxMove(BoardID, 3, nChannel, -10000, cmxFALSE );

long nCorrMode = 0, nCntrMask = 0;
double fCorrAmount = 0.0f, fCorrVel = 0.0f;

//0 번 축에 설정된 보정 옵션을 반환합니다.
cmxPmSxGetCorrection (BoardID, 3, nChannel, &nCorrMode, &fCorrAmount, &fCorrVel, &nCntrMask );
```

NAME

cmxPmSxSetOptSyncMode/

cmxPmSxGetOptSyncMode

- 다른 축 동기 구동 설정

INFORMATION

- Single Axis Control
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 3
- ☺ 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxPmSxSetOptSyncMode

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 Mode, [in] VT_I4 RefAxis, [in] VT_I4 Condition)

❒ VT_I4 cmxPmSxGetOptSyncMode

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 Mode, [out] VT_PI4 RefAxis, [out] VT_PI4 Condition)

DESCRIPTION

지정한 축에 대하여 이송명령을 전달되었을 때 이송동작의 시작이 다른축(Other Axis)의 동작 상황에 동기되어 시작되도록 할 때 사용하는 함수입니다. 예를 들어 X 축의 이송시작이 Y 축의 가속이 완료되는 시점 또는 감속이 시작되는 시점에 수행되도록 하고자할 때 이 함수를 사용하면 해당 동작을 구현할 수 있습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***Mode*** : cmxPmSxSetOptSyncMode 함수의 인자이며, 동기모드를 지정합니다. 이 값의 의미는 다음과 같습니다..

Value	Meaning
0	동기모드를 사용하지 않습니다. 따라서 이송명령이 하달되면 바로 이송을 시작합니다. ※ 이 모드에서는 RefAxis 와 Condition 매개 변수(媒介變數)가 무시됩니다.
1	"Start by Internal Synch. Signal" 모드: 내부동기 신호에 의하여 이송이 시작됩니다. 따라서 이송명령이 하달되면 바로 이송을 시작하지 않고 내부동기 신호가 발생할 때 실제 이송이 시작됩니다. 내부동기 신호는 Condition 매개 변수(媒介變數)에 의해서 지정된 조건이 만족되면 자동으로 발생됩니다. ※ 이 모드에서는 RefAxis 와 Condition 매개 변수(媒介變數)가 모두 참조됩니다.

2	"Start by Other Axis/Axes Stop"모드: RefAxis 매개 변수(媒介變數)에 의하여 지정된 축의 이송이 완료될 때 이송을 시작합니다. 따라서 RefAxis 에 의하여 지정된 축이 이송이 완료되지 않은상태에서 이송명령이 하달되면 RefAxis 축이 정지(停止)하는 순간에 실제 이송이 시작됩니다. ※ 이 모드에서는 RefAxis 만 참조 되며, Condition 매개 변수(媒介變數)는 무시됩니다.

▶ ***Mode*** : cmxPmSxGetOptSyncMode 함수의 인자이며, 동기모드를의 상태를 반환합니다. 이 값의 의미는 다음과 같습니다..

Value	Meaning
0	동기모드를 사용하지 않는 상태입니다.
1	"Start by Internal Synch. Signal" 모드: 내부동기 신호에 의하여 이송이 시작되는 상태입니다..
2	"Start by Other Axis/Axes Stop"모드: RefAxis 매개 변수(媒介變數)에 의하여 지정된 축의 이송이 완료될 때 이송을 시작하는 상태입니다..

▶ ***RefAxis*** : cmxPmSxSetOptSyncMode 함수의 인자이며, 내부 동기신호를 발생할 때 참조할 축 번호 또는 마스크를지정합니다. 이 값을 지정하는 방식은 Mode 매개 변수(媒介變數)의 값에 따라서 아래와 같이 달라집니다.

- Mode 매개 변수(媒介變數)가 "1"인 경우: 이 모드에서는 이 값에 축 번호를 지정합니다. 단, 주의할 것은 Axis 매개 변수(媒介變數)가 0 ~ 3 사이의 축인 경우에는 이 값도 0 ~ 3 이어야 합니다. 그리고 Axis 매개 변수(媒介變數)가 4 ~ 7 사이의 축인 경우에는 이 값도 4 ~ 7 이어야 합니다.
- Mode 매개 변수(媒介變數)가 "2"인 경우: 이 모드에서는 이 값에 축 마스크를 지정합니다. 이 모드에서는 참조 축을 여러 개 설정할 수 있으며, 각 비트별로 값이 1 인 경우 해당 축이 참조됩니다. 예를 들어 Axis0 과 Axis2 의 두축이 모두 정지(停止)하는 시점에 출발하고자 한다면 RefAxis 값은 0x5 (bit0 과 bit2 를 1 로 만듬)로 설정합니다. 단, 주의할 것은 Axis 매개 변수(媒介變數)가 0 ~ 3 사이의 축인 경우에는 이 값도 BIT0 ~ BIT3 만 사용할 수 있으며, Axis 매개 변수(媒介變數)가 4 ~ 7 사이의 축인 경우에는 이 값도 BIT4 ~ BIT7 만 사용할 수 있습니다.

▶ ***RefAxis*** : cmxPmSxGetOptSyncMode 함수의 인자이며, 내부 동기신호를 발생할 때 참조하는 축 번호 또는 마스크를 반환합니다.

▶ ***Condition*** : cmxPmSxSetOptSyncMode 함수의 인자이며, 이 값은 Mode 매개 변수(媒介變數)가 "1"로 지정되었을 때에만의미를 가집니다. 단, 이때 각 조건의 주체가 되는 축은 RefAxis 매개 변수(媒介變數)에 의해서 지정된 축입니다.

Value	Meaning
0	범용비교기의 조건이 충족되었을 때 이송을 시작합니다. 단, 이 방법을 사용하려면 cmmCmpGenSetConfig() 함수의 CmpAction 매개 변수(媒介變數)를 0 으로 하여야 합니다.
1	참조축이 가속을 시작할 때 이송을 시작합니다.
2	참조축이 가속을 끝내고 정속모드로 들어서는 순간에 이송을 시작합니다.
3	참조축이 감속을 시작할 때 이송을 시작합니다.
4	참조축이 감속을 끝낼 때 이송을 시작합니다.
5	- SL 신호가 검출되었을 때 이송을 시작합니다.
6	+SL 신호가 검출되었을 때 이송을 시작합니다.
7	범용 비교기에 설정된 조건이 만족되었을 때 이송을 시작합니다.
8	TRG-CMP 조건이 만족되었을 때 이송을 시작합니다.

▶ ***Condition*** : cmxPmSxGetOptSyncMode 함수의 인자이며, 변환하는 값의 의미는 다음과 같습니다.

Value	Meaning
0	범용비교기의 조건이 충족되었을 때 이송을 시작합니다.
1	참조축이 가속을 시작할 때 이송을 시작합니다.
2	참조축이 가속을 끝내고 정속모드로 들어서는 순간에 이송을 시작합니다.
3	참조축이 감속을 시작할 때 이송을 시작합니다.
4	참조축이 감속을 끝낼 때 이송을 시작합니다.
5	- SL 신호가 검출되었을 때 이송을 시작합니다.
6	+SL 신호가 검출되었을 때 이송을 시작합니다.
7	범용 비교기에 설정된 조건이 만족되었을 때 이송을 시작합니다.
8	TRG-CMP 조건이 만족되었을 때 이송을 시작합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

NAME

cmxPmSxSetOptSyncOut/

cmxPmSxGetOptSyncOut

- 속도(速度)구간 별 디지털 출력(出力) 설정

INFORMATION

Single Axis Control

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 3

다소 주의

함수의 전달 인자 중 'DoChan_local'은 전역 채널이 아닙니다. 이 의미는 개별 모션 보드 상의 디지털 출력채널을 의미하므로, 로컬 채널임을 반드시 주의 합니다.

SYNOPSIS

❒ VT_I4 cmxPmSxSetOptSyncOut

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 Mode, [in] VT_I4 DoChan_local, [in] VT_I4 DoLogic)

❒ VT_I4 cmxPmSxGetOptSyncOut

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 Mode, [out] VT_PI4 DoChan_local, [out] VT_PI4 DoLogic)

DESCRIPTION

cmxPmSxSetOptSyncOut()함수는 지정한 축의 각 속도 구간에서 고속 디지털 출력을 발생할 수 있도록 합니다. 이 출력은 Machine Vision 등의 동기 시작 트리거(Trigger) 신호로 사용될 수 있습니다. 이 함수를 통해 가속/정속/감속 구간의 시작과 끝에 동기되어 특정 디지털 출력을 발생시킬 수 있습니다. cmxPmSxGetOptSyncOut() 함수는 지정한 축의각 속도 구간에서 고속 디지털 출력의 발생에 대한 설정을 반환합니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***Mode*** : cmxPmSxSetSyncOut 함수의 인자이며, 고속 디지털 출력 발생 모드를 설정합니다.

Value.	Meaning
0	이 함수의 기능을 사용하지 않습니다.
1	가속 시작시에 출력을 발생후 종료시에 출력을 종료합니다.
2	정속 시작시에 출력을 발생후 종료시에 출력을 종료합니다.
3	감속 시작시에 출력을 발생후 종료시에 출력을 종료합니다

▶ ***Mode*** : cmxPmSxGetSyncOut 함수의 인자이며, 고속 디지털 출력 발생 모드의 설정상태를 반환합니다.

Value.	Meaning
0	이 함수의 기능을 사용하지 않습니다.
1	가속 시작시에 출력을 발생후 종료시에 출력을 종료합니다.
2	정속 시작시에 출력을 발생후 종료시에 출력을 종료합니다.
3	감속 시작시에 출력을 발생후 종료시에 출력을 종료합니다

▶ ***DoChan_local*** : cmxSxSetOptSyncOut 함수의 인자이며, 범용 디지털 출력으로 사용할 디지털 출력 채널을 설정합니다. 이 채널은 반드시 해당 모션 보드의 로컬(Local) 채널로 설정해야 합니다. 로컬(Local) 채널이라는 것은 ceSDK 가 관리하는 전체 채널이 아닌 각 모션 보드내에서의 채널번호를 의미합니다. 즉, 장치의 순서에 관계없이 해당 장치내에서의 디지털출력 채널만을 고려한 채널번호를 설정하여야 합니다. 예를 들어서 COMI-LX504 제품의 경우에는 디지털출력 채널이 6 개 제공되므로 DoChannel_local 매개변수(媒介變數)에 사용될 수 있는 번호는 장치의 순서에 관계없이 0 ~ 5 가 되는 것입니다.

▶ ***DoChan_local*** : cmxSxGetOptSyncOut 함수의 인자이며, 범용 디지털 출력으로 사용되는 디지털 출력 채널을 반환합니다.

▶ ***DoLogic*** : cmxSxSetOptSyncOut 함수의 인자이며, 디지털 출력 채널의 로직을 설정합니다.

Value	Meaning
0 또는 cmxPM_LOGIC_A	A 접점 방식 => 평상시 Open, 감지되면 Close 되는 스위치 방식
1 또는 cmxPM_LOGIC_B	B 접점 방식 => 평상시 Close, 감지되면 Open 되는 스위치 방식

▶ ***DoLogic*** : cmxSxGetOptSyncOut 함수의 인자이며, 디지털 출력 채널의 로직을 반환합니다.

Value	Meaning
0 또는 cmxPM_LOGIC_A	A 접점 방식 => 평상시 Open, 감지되면 Close 되는 스위치 방식
1 또는 cmxPM_LOGIC_B	B 접점 방식 => 평상시 Close, 감지되면 Open 되는 스위치 방식

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

15.2 다축(Multi-Axes) 동시제어

이 단원에서는 다축 동시제어에 관련된 함수들을 소개합니다. 다축 동시제어는 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.

다축 동시제어 기능은 Velocity Move 와 In-position Move 모두에 적용 가능합니다. 이와 관련된 함수들은 다음과 같습니다.

※ 다축 동시제어 함수들은 단축(Single Axis) 모션제어 관련 함수들과 혼용이 가능합니다. 특히, 다축 동시제어시에도 각 축에 대한 기준 속도에 대한 설정은 cmxCfgSetSpeedPattern() 함수를 사용하여야 합니다.

15.2.1 함수 요약

Summary of Functions
❒ VT_I4 cmxPmMxMove ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 NumChannel, [in] VT_PI4 NumChannel, [in] VT_PR8 DistList, [in] VT_I4 IsBlocking) 다축(多軸) 상대좌표이송(相對座標移送) 을 시작합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)되지 않습니다.
❒ VT_I4 cmxPmMxMoveStart ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 NumChannel, [in] VT_PI4 NumChannel, [in] VT_PR8 DistList) 다축(多軸) 상대좌표이송(相對座標移送) 을 시작합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)됩니다.
❒ VT_I4 cmxPmMxMoveTo ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 NumChannel, [in] VT_PI4 NumChannel, [in] VT_PR8 PosList, [in] VT_I4 IsBlocking) 다축(多軸) 절대좌표이송(絶對座標移送) 을 시작합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)되지 않습니다.
❒ VT_I4 cmxPmMxMoveToStart ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 NumChannel, [in] VT_PI4 NumChannel, [in] VT_PR8 PosList) 다축(多軸) 절대좌표이송(絶對座標移送) 을 시작합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)됩니다.
❒ VT_I4 cmxPmMxVMoveStart ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 NumChannel, [in] VT_PI4 NumChannel, [in] VT_PI4 DirList) 다축(多軸) 연속속도이송(連續速度移送) 을 시작합니다. 이 구동 함수는 구동 시작 후 바로 반환(返還)됩니다.
❒ VT_I4 cmxPmMxStop ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 NumChannel, [in] VT_PI4 NumChannel, [in] VT_I4 IsWaitComplete) 다축(多軸) 이송을 감속 후 정지(停止) 합니다. 이 정지(停止) 함수는 이송완료 (移送完了)시 까지 대기(待機) 할 수 있습니다.
❒ VT_I4 cmxPmMxStopEmg ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 NumChannel, [in] VT_PI4 NumChannel) 다축(多軸) 이송을 비상정지(非常停止) 합니다. 이 정지(停止) 함수는 감속(減速)을 무시(無視) 합니다.
❒ VT_I4 cmxPmMxIsDone ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 NumChannel, [in] VT_PI4 NumChannel, [out] VT_PI4 IsDone) 다축(多軸) 이송의 완료(完了) 를 확인(確認)합니다.
❒ VT_I4 cmxPmMxWaitDone ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 NumChannel, [in] VT_PI4 NumChannel) 다축(多軸) 이송의 완료(完了) 시점까지 대기(待機)합니다.

15.2.2 함수 설명

NAME

cmxPmMxMove

cmxPmMxMoveStart

- 다축(多軸) 상대 좌표 이송(相對座標移送)

INFORMATION

Multi Axes Control

VC++/VB

BCB/Delphi/.NET

Level 3

이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인(確認)합니다.

SYNOPSIS

❒ VT_I4 cmxPmMxMove

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 NumChannel, [in] VT_PI4 NumChannel, [in] VT_PR8 DistList, [in] VT_I4 IsBlocking)

❒ VT_I4 cmxPmMxMoveStart

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 NumChannel, [in] VT_PI4 NumChannel, [in] VT_PR8 DistList)

DESCRIPTION

여러 개의 축에 대하여 현재의 위치에서 지정한 거리만큼 이동을 동시에 시작합니다. 이 함수를 사용하면 여러 개의 축이 동시에 작업을 시작합니다. 따라서 이 함수는 여러축이 동기를 맞추어 작업을 시작해야하는 경우에 유용하게 사용될 수 있습니다.
cmxPmMxMove() 함수는 지정한 모든 축의 모션이 완료되기 전까지 반환되지 않으며, cmxPmMxMoveStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ NodeId : 노드번호. 노드번호는 3 부터 시작합니다.

▶ NumAxes : 동시에 작업을 수행할 대상 축의 수

▶ NumChannel : 동시에 작업을 수행할 대상 축의 배열 주소값. 이 배열의 크기는 NumAxes 값과 일치하거나 커야 합니다.

▶ DistList : 이동할 거리값의 배열 주소값. 이 배열의 크기는 NumAxes 값과 일치하거나 커야 합니다. 이동 거리값은 현재의 위치에 대한 상대 좌표이며 거리의 단위는 논리적 거리(Logic

distance) 단위를 사용합니다. "Unit distance"를 1 로 한 경우에 거리의 단위는 Pulse 수가 됩니다. 즉, Distance 값 1 은 1 Pulse 출력을 의미합니다.

▶ IsBlocking : 완료될 때까지 기다리는 동안 윈도우 메시지를 블록(Blocking)할 것인지를 결정합니다.

Value	Meaning
cmxFALSE	블록(Blocking)을 하지 않습니다. 따라서 해당 모션이 완료되는 동안에도 윈도우 이벤트를 처리합니다.
cmxTRUE	블록(Blocking)을 합니다. 따라서 해당 모션이 완료되는 동안에는 윈도우 이벤트가 처리되지 않습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

REFERENCE

□ cmxPmMxMoveStart 함수를 사용하는 경우에는 cmxPmMxIsDone() 함수나 cmxPmMxWaitDone() 함수를 사용하여 모션의 완료를 확인(確認)할 수 있습니다.

□ cmxPmMxMove 함수를 사용하는 경우에는 내부적으로 루프를 수행하면서 모션이 완료되기를 기다리는데, 이때 "Blocking Mode"설정에 따라 윈도우 이벤트를 처리하는 방식이 달라집니다. 그러나 일반적으로 윈도우의 작업 쓰레드(Work Thread) 에서는 블록모드를 사용하여, 함수내부에서 지연없이 쓰레드 내부의 작업에 집중할 수 있도록 설정하는 것이 바람직합니다.

□ INP 입력신호가 Enable 로 설정되었으면 Command 펄스 출력이 완료되어도 INP 입력이 ON 이 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환되지 않습니다.

□ 스탭 드라이브를 사용 중인 고객(顧客)님들께서는 다음을 참조해 주십시요.
스탭 드라이브는 INP 출력이 없는 경우가 일반적인데, 고객(顧客)님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스탭 드라이브를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다.
고객(顧客) 여러분들께서는 스탭 드라이브 사용시에 이점을 주의해주시기를 부탁드립니다.

□ 서보 드라이브의 LSP , LSN 신호를 사용 중인 고객(顧客)님께서는 다음을 참조해 주십시요.
서보드라이브의 입력 신호 중 하나인 EL(End of Limit) 신호는 저희 ㈜커미조아 RTEX 보드 뿐만 아니라 서보드라이브에도 전달 될 수 있도록 설정할 수 있습니다. 통상적으로 LSP 신호와 LSN 신호로 불리어지는 이 신호는 실제 기구물에서 양의 방향(Positive Direction) 혹은 음의 방향(Negative Direction) 에 장착되어 있는 EL(End of Limit) 신호를 서보 드라이브 측에 전달하기 위한 용도로 사용됩니다.

그러나, 모션 소프트웨어에서 INP 설정이 되어 있는 경우 EL 신호가 검출 된 후에 일부 서보 드라이브에서는 진행 방향에서 정지(停止) 한후 더 이상 움직이지 않는 상황이 발생하며, 이 상황에서 INP 신호가 출력되지 않아, 모션 이송이 완료되지 못하고, 명시적으로 STOP 을 해줘야만 하는 강제적으로 모션 종료가 되는 현상이 발생할 수 있습니다. 이 현상은 EL 모드를 통해 원점 복귀를 하는 상황에서도 발생할 수 있습니다.

따라서, 이러한 경우에는 반드시 원점 복귀나 EL 검출시에 인터럽트 이벤트나 타이머를 통해 INP 를 무기한 대기하는 현상에 대해서 적절히 대처하시거나 INP 신호 사용을 배제 해야 합니다.

윈도우 이벤트라는 것은 무엇입니까?

윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

EXAMPLE

```
C/C++
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

/************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.
************************************************************/
long BoardID = 0;

void OnProgramInitial()
{
	long m_nNumDevices;
	long m_DeviceList[16];
	long m_nNumAxes;
	cmxLoadDll();
	if(cmxGnLoadDevice (&m_nNumDevices, m_DeviceList, &m_nNumAxes)  != ERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		// 에러메시지 출력
		return;
	}
}

/************************************************************
* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
* 호출되는 가상의 함수 입니다. 이때 m_fVwork, m_fAcc, m_fDec 변수를
* 통하여 속도, 가속도 , 감속도 값이 적절하게 전달된다고 가정합니다.
************************************************************/
void OnSetSpeed()
{
	//각 축(Axis)의 기본 속도를 설정 합니다.
	cmxPmCfgSetSpeedPattern(BoardID, 3, 0, cmxMODE_S, m_fVwork, m_fAcc, m_fDec,0,0 );

	cmxPmCfgSetSpeedPattern(BoardID, 3, cmY1, cmxMODE_S, m_fVwork, m_fAcc,
m_fDec ,0,0);
}

/************************************************************
* DoMotion : 작업명령시에 호출되는 가상의 함수 입니다.
************************************************************/
void DoMotion()
```

```
{
	long nNumChannel[2] = {0, cmY1};//움직일 축의 목록입니다.
	double fDistList[2] = {5000, 5000};//각축의 이동 거리입니다.

	//두 개의 축을 상대거리 5000 만큼 이동 시킵니다.
	cmxPmMxMove(BoardID, 3, 2, nNumChannel, fDistList, cmxFALSE);

	//반대방향으로 5000 만큼 움직이기 위해서 거리 값들을 -5000 으로 변경합니다.
	fDistList[0] = -5000; fDistList[1] = -5000;

	//두 개의 축을 상대거리 -5000 만큼 이동 시킵니다.
	cmxPmMxMove(BoardID, 3, 2, nNumChannel, fDistList, cmxFALSE);

	//위의 cmxPmMxMove()함수 대신에 cmxPmMxMoveStart() 함수를 사용하려면
	//아래코드를 사용합니다.
	//cmxPmMxMoveStart(BoardID, 3, 2, nNumChannel, fDistList);
	//cmxPmMxWaitDone(BoardID, 3, 2, NumChannel, cmxFALSE);
	//fDistList[0] = -5000; fDistList[1] = -5000;
	//cmxPmMxMoveStart(BoardID, 3, 2, nNumChannel, fDistList);
	//cmxPmMxWaitDone(BoardID, 3, 2, NumChannel, cmxFALSE);
}
```

```
Visual Basic

‘=============================================================
‘GnLoadDevice 함수로 장치를 초기화 합니다.
‘=============================================================

Private Sub Form_Load()
   Dim BoardID As Long
   Dim nTotalDevices As Long
   Dim DeviceList(16) As Long
   Dim nTotalAxis As Long
   Dim IRetVal As Long

'=============================================================
 ' GnDeviceLoad 함수로 장치를 초기화합니다.
  BoardID = 0
  IRetVal = GnLoadDevice (nTotalDevices, DeviceList(0), nTotalAxis)

  If IRetVal <> ERR_NONE Then
     MsgBox ("GnLoadDevice has been failed")
  End If

'=============================================================
End Sub

Private Sub PmCfgSpeed(nTotalAxis As Long)

	 Dim i As Integer
	'=============================================================
	' 이 함수에서 PmCfgSetSpeedPattern 함수로 속도를 설정하는 것은
	‘ 모든 모션의 기준속도(Standard Speed) 가 됩니다.
	' 단축 구동을 비롯한 대부분의 모션 동작은 이 기준 속도의 비율로 동작되게
	‘ 됩니다.
```

```
        ' 아래 함수는 전체 축에 대해서 임의의 기준 속도를 설정하고 있습니다.
        '=========================================================

    For i = 0 To nTotalAxis-1
        Call PmCfgSetSpeedPattern(BoardID, 3, i, cmxMODE_S, 10000, 20000, 20000,0,0)
    Next

End Sub

Private Sub btnMove_Click()

    Dim DistanceList(2) As Double
    Dim NumChannel(2) As Long

    NumChannel(0) = 0
    NumChannel(1) = 1

    DistanceList(0) = 1000
    DistanceList(1) = 1000

    ' 각 인자는 순서대로
    ' 대상축의 갯수, 대상 축의 배열, 대상 거리의 배열, 블록 여부입니다.
     Call PmMxMove(BoardID, 3, 2, NumChannel(0), DistanceList(0), cmxFALSE)

End Sub
```

```
Delphi

// * 이 함수는 폼이 생성될때 이벤트에 의해 불려지며 , 장치를 로드하는 함수입
// * 니다.
procedure OnCreate();
var
        BoardID : LongInt;
        g_nDevs : LongInt;
        DevList : Array[0..15] of LongInt
        g_nAxis : LongInt;
begin

        BoardID := 0;
        // Load ComiRTEX(DLL) Library
        if (cmxGnLoadDevice (@g_nDevs, @DevList, @g_nAxis) <> ERR_NONE ) then
        begin
                // 마지막에 발생한 에러를 화면에 표시합니다.
                // 함수 인자로는 Form 의 Handle 이 전달됩니다.
                // 에러메시지 출력
                exit;
        end
end;

// * Description : 속도를 설정 하는 함수
procedure btnSetSpeedClick();
var
    fAccelSpeed : Double;
    fDecelSpeed : Double;
    fWorkSpeed : Double;
    nSMODE : LongInt;
```

```
begin
        //각 변수들의 값을 설정 합니다.
        fWorkSpeed := 50000;
        fAccelSpeed := 100000;
        fDecelSpeed := 100000;
        nSMODE := cmxPM_MODE_S;
        // 설정된 기준 속도를 실제 SDK 함수에 전달합니다.


cmxPmCfgSetSpeedPattern(
      0,                // 현재 Board 의 ID 를 입력합니다.
      0,                // 현재 활성화 되어 있는 슬레이브를 선택합니다.
      0,                // 현재 활성화 되어 있는 채널을 선택합니다.
      nSMODE,           // 가감속이 없는 모드와 선형 가감속, S-CURVE 가감속을 설정합니다.
      fWorkSpeed,       // 작업 속도를 설정합니다.
      fAccelSpeed,      // 가속도를 설정합니다.
      fDecelSpeed,      // 감속도를 설정합니다.
      0,                //초기속도를 설정합니다.
      0) ;              //최종속도를 설정합니다.

     cmxPmCfgtSpeedPattern(
      0,                // 현재 Board 의 ID 를 입력합니다.
      cmY1,             // 현재 활성화 되어 있는 채널을 선택합니다.
      nSMODE,           // 가감속이 없는 모드와 선형 가감속,
                        //S-CURVE 가감속 모드를 설정합니다.
      fWorkSpeed,       // 작업 속도를 설정합니다.
      fAccelSpeed,      // 가속도를 설정합니다.
      fDecelSpeed,      // 감속도를 설정합니다.
      0,                //초기속도를 설정합니다.
      0) ;              //최종속도를 설정합니다.

   end;
end;

// * Description :
// * 이 함수는 버튼 이벤트에 설정된 거리만큼 이동하는 함수입니다.

Procedure btnPositiveClick();
var
   fWorkSpeedRatio : Double;
   fAccelSpeedRatio : Double;
   fDecelSpeedRatio : Double;

      NumChannel : Array[0..1] of LongInt;
      DistanceList : Array[0..1] of Double;

begin
      NumChannel[0] := 0;
      NumChannel[1] := cmY1;
      DistanceList[0] := 1000;
      DistanceList[1] := 2000;

      cmxPmMxMove(BoardID, 3, g_nTargetAxis, @NumChannel, @DistanceList, cmxFALSE);

end;
```

NAME

cmxPmMxMoveTo

cmxPmMxMoveToStart

- 다축(多軸) 절대 좌표 이송(絶對座標移送)

INFORMATION

Multi Axes Control

VC++/VB

BCB/Delphi/.NET

Level 3

이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인(確認)합니다.

SYNOPSIS

❒ VT_I4 cmxPmMxMoveTo

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 NumChannel, [in] VT_PI4 NumChannel, [in] VT_PR8 PosList, [in] VT_I4 IsBlocking)

❒ VT_I4 cmxPmMxMoveToStart

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 NumChannel, [in] VT_PI4 NumChannel, [in] VT_PR8 PosList)

DESCRIPTION

여러 개의 축에 대하여 지정한 절대좌표로의 이동을 시작합니다. 이 함수를 사용하면 여러 개의 축이 동시에 작업을 시작합니다. 따라서 이 함수는 여러축이 동기를 맞추어 작업을 시작해야하는 경우에 유용하게 사용될 수 있습니다.
cmxPmMxMoveTo() 함수는 지정한 모든 축의 모션이 완료되기 전까지 반환되지 않으며, cmxPmMxMoveToStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ NodeId : 노드번호. 노드번호는 3 부터 시작합니다.

▶ NumAxes : 동시에 작업을 수행할 대상 축의 수

▶ NumChannel : 동시에 작업을 수행할 대상 축의 배열 주소값. 이 배열의 크기는 NumAxes 값과 일치하거나 커야 합니다.

▶ PosList : 절대좌표값의 배열 주소값. 이 배열의 크기는 NumAxes 값과 일치하거나 커야 합니다. 절대좌표의 단위는 논리적 거리(Logic distance) 단위를 사용합니다. "Unit distance"를 1 로 한 경우에 거리의 단위는 Pulse 수가 됩니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

REFERENCE

☐ cmxPmMxMoveToStart() 함수를 사용하는 경우에는 cmxPmMxIsDone() 함수나 cmxPmMxWaitDone() 함수를 사용하여 모션의 완료를 확인(確認)할 수 있습니다.

☐ cmxPmMxMoveTo() 함수를 사용하는 경우에는 내부적으로 루프를 수행하면서 모션이 완료되기를 기다리는데, 이때 "Blocking Mode"설정에 따라 윈도우 이벤트를 처리하는 방식이 달라집니다. 그러나 일반적으로 윈도우의 작업 쓰레드(Work Thread) 에서는 블록모드를 사용하여, 함수내부에서 지연없이 쓰레드 내부의 작업에 집중할 수 있도록 설정하는 것이 바람직합니다.

☐ INP 입력신호가 Enable 로 설정되었으면 Command 펄스 출력이 완료되어도 INP 입력이 ON 이 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환되지 않습니다.

☐ 스탭 드라이브를 사용 중인 고객(顧客)님들께서는 다음을 참조해 주십시요.
스탭 드라이브는 INP 출력이 없는 경우가 일반적인데, 고객(顧客)님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스탭 드라이브를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다.
고객(顧客) 여러분들께서는 스탭 드라이브 사용시에 이점을 주의해주시기를 부탁드립니다.

☐ 서보 드라이브의 LSP , LSN 신호를 사용 중인 고객(顧客)님께서는 다음을 참조해 주십시요.
서보드라이브의 입력 신호 중 하나인 EL(End of Limit) 신호는 저희 ㈜커미조아 모션 보드 뿐만 아니라 서보드라이브에도 전달 될 수 있도록 설정할 수 있습니다. 통상적으로 LSP 신호와 LSN 신호로 불리어 지는 이 신호는 실제 기구물에서 양의 방향(Positive Direction) 혹은 음의 방향(Negative Direction) 에 장착되어 있는 EL(End of Limit) 신호를 서보 드라이브 측에 전달하기 위한 용도로 사용됩니다.

그러나, 모션 소프트웨어에서 INP 설정이 되어 있는 경우 EL 신호가 검출 된 후에 일부 서보 드라이브에서는 진행 방향에서 정지(停止) 한후 더 이상 움직이지 않는 상황이 발생하며, 이 상황에서 INP 신호가 출력되지 않아, 모션 이송이 완료되지 못하고, 명시적으로 STOP 을 해줘야만 하는 강제적으로 모션 종료가 되는 현상이 발생할 수 있습니다. 이 현상은 EL 모드를 통해 원점 복귀를 하는 상황에서도 발생할 수 있습니다.

따라서, 이러한 경우에는 반드시 원점 복귀나 EL 검출시에 인터럽트 이벤트나 타이머를 통해 INP 를 무기한 대기하는 현상에 대해서 적절히 대처하시거나 INP 신호 사용을 배제 해야 합니다.

윈도우 이벤트라는 것은 무엇입니까?

	윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

EXAMPLE

본 예제는 cmxPmMxMoveTo 를 사용하여 X1,Y1 축을 절대좌표 (1000,1000) 지점으로 이동한 후 다시 절대좌표 (BoardID, 3,0,0) 지점으로 이동하는 예입니다. 이때 각축의 속도와 이동거리가 같으므로 동시에 종료될 것입니다. 하지만 속도 설정이 서로 다르거나 이동거리가 서로 다른 경우에는 각축이 동시에 시작해도 종료시점은 다를 수 있습니다.

```
C/C++
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

/*************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.
*************************************************************/
long BoardID = 0;
void OnProgramInitial()
{
	long m_nNumDevices;
	long m_DeviceList[16];
	long m_nNumAxes;
	cmxLoadDll();
	if(cmxGnLoadDevice (&m_nNumDevices, m_DeviceList, &m_nNumAxes)!= ERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		// 에러메시지 출력
		return;
	}
}

/*************************************************************
* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
* 호출되는 가상의 함수 입니다. 이때 m_fVwork, m_fAcc, m_fDec 변수를
* 통하여 속도, 가속도 , 감속도 값이 적절하게 전달된다고 가정합니다.
*************************************************************/
void OnSetSpeed()
{
	//각 축(Axis)의 기본 속도를 설정 합니다.
	cmxPmCfgSetSpeedPattern(BoardID, 3, 0, cmxPM_SMODE_S, m_fVwork, m_fAcc,
m_fDec ,0,0);  cmxPmCfgSetSpeedPattern(BoardID, 3, cmY1, cmxPM_SMODE_S, m_fVwork,
m_fAcc, m_fDec ,0, 0);
}

/*************************************************************
* DoMotion : 작업명령시에 호출되는 가상의 함수 입니다.
*************************************************************/
void DoMotion()
{
	long nNumChannel[2] = {0, cmY1};//움직일 축의 목록입니다.
```

```
        double fPosList[2] = {1000, 1000};//각축의 이동할 거리입니다.

        //세개의 축을 절대좌표 5000 으로 이동 시킵니다.
        cmxPmMxMoveTo(BoardID, 3, 2, NumChannel, fPosList, cmxFALSE);

        //각 축들을 절대 좌표 0 으로 움직이기 위해서 위치 값들을 0 으로 바꿉니다.
        fPosList[0] = 0; fPosList[1] = 0;

        //세개의 축을 절대좌표 0 으로 이동 시킵니다.
        cmxPmMxMoveTo(BoardID, 3, 2, NumChannel, fPosList, cmxFALSE);

}
```

```
Visual Basic

‘=============================================================
‘GnLoadDevice 함수로 장치를 초기화 합니다.
‘=============================================================

Private Sub Form_Load()

        Dim BoardID As Long
        Dim nTotalDevices As Long
        Dim DeviceList(16) As Long
        Dim nTotalAxis As Long
        Dim IRetVal As Long

'=============================================================
        BoardID = 0
         ' GnLoadDevice 함수로 장치를 초기화합니다.
          IRetVal = GnLoadDevice (nTotalDevices, DeviceList(0), nTotalAxis)

          If IRetVal <> ERR_NONE Then
             MsgBox ("GnLoadDevice has been failed")
          End If

'=============================================================

End Sub

Private Sub PmCfgSpeed(nTotalAxis As Long)
           Dim i As Integer

        '=========================================================
         ' 이 함수에서 PmCfgSetSpeedPattern 함수로 속도를 설정하는 것은
        ‘ 모든 모션의 기준속도(Standard Speed) 가 됩니다.
        ' 단축 구동을 비롯한 대부분의 모션 동작은 이 기준 속도의 비율로 동작되게
        ‘ 됩니다.
         ' 아래 함수는 전체 축에 대해서 임의의 기준 속도를 설정하고 있습니다.
          '=========================================================

        For i = 0 To nTotalAxis-1
              Call PmCfgSetSpeedPattern(BoardID, 3, i, cmxPM_SMODE _S, 10000, 20000, 20000,0,0)
        Next

End Sub
```

```
‘ IsBlocking 은 함수 실행 시간에 윈도우 메시지를 처리할 것인지에 대한 플래그를 말합니다.
‘ 아래는 IsBlocking 을 반환하는 가상의 함수입니다. 실제의 코드에서는 사용자나 환경
‘ 설정에서 이 설정을 반환받는 것이 옳은 방법입니다.
Dim IsBlocking As Long

Private Function GetIsBlocking() As Long

   GetIsBlocking = IsBlocking

End Function

Private Sub btnMove_Click()

      Dim DistanceList(2) As Double
      Dim NumChannel(2) As Long
      Dim nRetVal As Long

      NumChannel(0) = 0
      NumChannel(1) = 1

      DistanceList(0) = 1000
      DistanceList(1) = 1000

      ' 각 인자는 순서대로
      ' 대상축의 갯수, 대상 축의 배열, 대상 거리의 배열, 블록 여부입니다.

      nRetVal = PmMxMoveTo(BoardID, 3, 2, NumChannel(0), DistanceList(0), GetIsBlocking())


End Sub
```

```
Delphi

// * Description :
// * CME 빌더를 통한 모션 환경설정이 되었다는 가정하에 진행합니다.
// *
// * 이 함수는 폼이 생성될때 이벤트에 의해 불려지며 , 장치를 로드하는 함수입
// * 니다.

procedure OnCreate();
var
      BoardID : LongInt;
      g_nDevs : LongInt;
      DevList : Array[0..15] of LongInt
      g_nAxis : LongInt;
begin

      BoardID := 0;
      // Load ComiRTEX(DLL) Library
      if (cmxGnLoadDevice(@g_nDevs, @DevList, @g_nAxis) <> ERR_NONE ) then
      begin
            // 마지막에 발생한 에러를 화면에 표시합니다.
            // 함수 인자로는 Form 의 Handle 이 전달됩니다.
            // 에러메시지 출력
            exit;
```

```
        end
end;

// * Description : 속도를 설정 하는 함수
procedure btnSetSpeedClick();
var
   fAccelSpeed : Double;
   fDecelSpeed : Double;
   fWorkSpeed : Double;
   nSMODE : LongInt;

begin
                //각 변수들의 값을 설정 합니다.
                fWorkSpeed := 50000;
                fAccelSpeed := 100000;
                fDecelSpeed := 100000;
                nSMODE := cmxMODE_S;
                // 설정된 기준 속도를 실제 SDK 함수에 전달합니다.

                cmxCfgSetSpeedPattern(
                        0,              // 현재 Board 의 ID 를 입력합니다.
                        0,              // 현재 활성화 되어 있는 채널을 선택합니다.
                        nSMODE,         // 가감속이 없는 모드와 선형 가감속,
                                        //S-CURVE 가감속을 설정합니다.
                        fWorkSpeed,     // 작업 속도를 설정합니다.
                        fAccelSpeed,    // 가속도를 설정합니다.
                        fDecelSpeed);   // 감속도를 설정합니다.
                        0,              //초기속도를 설정합니다.
                        0) ;            //최종속도를 설정합니다.


               cmxCfgSetSpeedPattern(
                        0,              // 현재 Board 의 ID 를 입력합니다.
                        cmY1,           // 현재 활성화 되어 있는 채널을 선택합니다.
                        nSMODE,         // 가감속이 없는 모드와 선형 가감속,
                                        //S-CURVE 가감속을 설정합니다.
                        fWorkSpeed,     // 작업 속도를 설정합니다.
                        fAccelSpeed,    // 가속도를 설정합니다.
                        fDecelSpeed);   // 감속도를 설정합니다.
                        0,              //초기속도를 설정합니다.
                        0) ;            //최종속도를 설정합니다.
end;

// * Description :
// * 이 함수는 버튼 이벤트에 설정된 거리만큼 이동하는 함수입니다.
// *

Procedure btnPositiveClick();
var
    fWorkSpeedRatio : Double;
    fAccelSpeedRatio : Double;
    fDecelSpeedRatio : Double;

      NumChannel : Array[0..1] of LongInt;
      DistanceList : Array[0..1] of Double;
```

```
begin
        NumChannel[0] := 0;
        NumChannel[1] := cmY1;
        DistanceList[0] := 1000;
        DistanceList[1] := 2000;

         cmxPmMxMoveTo(BoardID, 3, 2, @NumChannel, @DistanceList, cmxFALSE);

end;
```

NAME

cmxPmMxVMoveStart

- 다축(多軸) 연속속도이송(連續速度移送)

INFORMATION

Multi Axes Control

VC++/VB

BCB/Delphi/.NET

Level 3

이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인(確認)합니다.

SYNOPSIS

❒ VT_I4 cmxPmMxVMoveStart

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 NumChannel, [in] VT_PI4 NumChannel, [in] VT_PI4 DirList)

DESCRIPTION

여러 개의 축에 대하여 Velocity Move 작업을 동시에 시작합니다. Velocity Move 는 작업속도까지 가속한 후에 작업속도를 유지하며 정지(停止)함수가 호출될 때까지 지정한 방향으로의 모션을 계속 수행합니다. 이 함수를 사용하면 여러 개의 축이 동시에 작업을 시작합니다. 따라서 이 함수는 여러축이 동기를 맞추어 작업을 시작해야하는 경우에 유용하게 사용될 수 있습니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ NodeId : 노드번호. 노드번호는 3 부터 시작합니다.

▶ NumAxes : 동시에 작업을 수행할 대상 축의 수

▶ NumChannel : 동시에 작업을 수행할 대상 축의 배열 주소값. 이 배열의 크기는 NumAxes 값과 일치하거나 커야 합니다.

▶ DirList : 방향을 지시하는 값의 배열 주소값. 이 배열의 크기는 NumAxes 값과 일치하거나 커야 합니다. 모션의 방향을 지시하는 값은 다음과 같습니다.

Value	Meaning
0 또는 cmxDIR_N	(-) 방향
1 또는 cmxDIR_P	(+) 방향

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

EXAMPLE

```
C/C++
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

/*************************************************************
* OnProgramInitial : 이 함수는 가상의 함수로서 프로그램 초기화 루틴이
* 적용되는 부분을 의미합니다.
*************************************************************/
long BoardID = 0;

void OnProgramInitial()
{
	long m_nNumDevices;
	long m_DeviceList[16];
	long m_nNumAxes;

	cmxLoadDll();
	if(cmxGnLoadDevice (&m_nNumDevices, m_DeviceList, &m_nNumAxes)!= ERR_NONE)
	{
		//Handle 은 사용자가 생성한 폼의 핸들 값입니다.
		// 에러메시지 출력
		return;
	}
}

/*************************************************************
* OnSetSpeed : 이 함수는 속도설정의 변경이 필요할 때
* 호출되는 가상의 함수 입니다. 이때 m_fVwork, m_fAcc, m_fDec 변수를
* 통하여 속도, 가속도 , 감속도 값이 적절하게 전달된다고 가정합니다.
*************************************************************/
void OnSetSpeed()
{
	//각 축(Axis)의 기본 속도를 설정 합니다.
	cmxPmCfgSetSpeedPattern(BoardID, 3, 0, cmxPM_SMODE_S, m_fVwork, m_fAcc,
m_fDec ,0,0);
	cmxPmCfgSetSpeedPattern(BoardID, 3, cmY1, cmxPM_SMODE_S, m_fVwork, m_fAcc,
m_fDec ,0,0);
}

/*************************************************************
* OnDoMotion() : 작업명령시에 호출되는 가상의 함수
* 이 함수에서는 X1, Y1, Z1 축에 대하여 Velocity Move 를 시작합니다.
*************************************************************/
void OnDoMotion()
{
	long nNumChannel[2] = {0, cmY1};
	long nDirList[2] = {cmDIR_P, cmDIR_P}; //Positive dir

	// Start V-Move of X1&Y1&Z1 //
	if(cmxPmMxVMoveStart(BoardID, 3, 3, nNumChannel, nDirList) != ERR_NONE){
```

```
                // 에러메시지 출력
                return;
        }
}
```

Visual Basic

```
Private Sub PmCfgSpeed(nTotalAxis As Long)

    Dim i As Integer
    Dim BoardID As Long
    '=============================================================
    ' 이 함수에서 CfgSetSpeedPattern 함수로 속도를 설정하는 것은
    ‘ 모든 모션의 기준속도(Standard Speed) 가 됩니다.
    ' 단축 구동을 비롯한 대부분의 모션 동작은 이 기준 속도의 비율로 동작되게
    ‘ 됩니다.
    ' 아래 함수는 전체 축에 대해서 임의의 기준 속도를 설정하고 있습니다.
    '=============================================================
    BoardID = 0
    For i = 0 To nTotalAxis-1
        Call PmCfgSetSpeedPattern(BoardID, 3, i, cmxPM_SMODE_S, 10000, 20000, 20000,0,0)
    Next

End Sub

Private Sub btnMove_Click()

    Dim Direction(2) As Long
    Dim NumChannel(2) As Long
    Dim nRetVal As Long

    NumChannel(0) = 0
    NumChannel(1) = 1

    Direction(0) = cmDIR_P
    Direction(1) = cmDIR_P

    ' 이 함수는 지정된 속도로 정지(停止) 함수가 호출 될때까지 계속 이동합니다.
    nRetVal = PmMxVMoveStart(BoardID, 3, 2, NumChannel(0), Direction(0))


End Sub
```

Delphi

```
Procedure btnSetSpeedClick();
var
    BoardID : LongInt;
    DevList : Array[0..15] of LongInt
    fAccelSpeed : Double;
    fDecelSpeed : Double;
    fWorkSpeed : Double;
    nSMODE : LongInt;

begin
```

```
        BoardID := 0;
        fAccelSpeed := 30000;
        fDecelSpeed := 30000;
        fWorkSpeed := 10000;
        nSMODE := cmxMODE_S;

        // 0 을 위해 설정된 기준 속도를 실제 SDK 함수에 전달합니다.
        cmxCfgSetSpeedPattern(
        0,                  // 현재 Board 의 ID 를 입력합니다.
        0,              // 현재 활성화 되어 있는 채널을 선택합니다.
        nSMODE,         // 가감속이 없는 모드와 선형 가감속, S-CURVE 가감속을 설정합니다.
        fWorkSpeed,     // 작업 속도를 설정합니다.
        fAccelSpeed,    // 가속도를 설정합니다.
        fDecelSpeed,    // 감속도를 설정합니다.
        0,              //초기속도를 설정합니다.
        0) ;            //최종속도를 설정합니다.


        // cmY1 을 위해 설정된 기준 속도를 실제 SDK 함수에 전달합니다.
        cmxCfgSetSpeedPattern(
        0,                  // 현재 Board 의 ID 를 입력합니다.
        cmY1,               // 현재 활성화 되어 있는 채널을 선택합니다.
        nSMODE,         // 가감속이 없는 모드와 선형 가감속, S-CURVE 가감속을 설정합니다.
        fWorkSpeed,     // 작업 속도를 설정합니다.
        fAccelSpeed,    // 가속도를 설정합니다.
        fDecelSpeed,    // 감속도를 설정합니다.
        0,              //초기속도를 설정합니다.
        0) ;            //최종속도를 설정합니다.
end;

procedure FormCreate();
var
        g_nDevs : LongInt;
        DevList : Array[0..15] of LongInt
        g_nAxis : LongInt;

begin

        // Load ComiRTEX(DLL) Library
        if (cmxGnLoadDevice (@g_nDevs, @DevList, @g_nAxis) <> ERR_NONE ) then
        begin
                // 마지막에 발생한 에러를 화면에 표시합니다.
                // 함수 인자로는 Form 의 Handle 이 전달됩니다.
                // 에러메시지 출력
                exit;
        end
end;

// * Description :
// *
// *   속도 모드로 반대 방향으로 다축(Multi Axes) 이동을 시작합니다.
procedure TForm1.btnNegativeClick(Sender: TObject);
var

  fWorkSpeedRatio : Double;
  fAccelSpeedRatio : Double;
```

```
  fDecelSpeedRatio : Double;

  NumChannel : Array[0..1] of LongInt;
  DirList : Array[0..1] of LongInt;

  i : LongInt;
begin

      // 이송 버튼을 비활성화합니다.
      btnPositive.Enabled := FALSE;
      btnNegative.Enabled := FALSE;

      For i:=0 to 1 do begin
            DirList[i] := cmDIR_N;   // 역방향
      end;

      NumChannel[0] := 0;
      NumChannel[1] := cmY1;

      //다축을 대상으로 속도제어(지정한 속도로 정지(停止)명령이 있을 때 까지 이동)
      // 에는 다음과 같이 cmxPmMxVMoveStart(...) 함수를 사용합니다.
      cmxPmMxVMoveStart(BoardID, 3,2,@NumChannel,@DirList);

end;
```

NAME

cmxPmMxStop

cmxPmMxStopEmg

- 다축(多軸) 이송 정지(停止)
- 다축비상 정지(非常停止)

INFORMATION

Multi Axes Control

VC++/VB

BCB/Delphi/.NET

Level 3

정지(停止) 함수

고속 이송시에 급 정지(停止)(비상정지(停止)를 주의하십시요. 기구물의 손상이나 안전사고에 원인이 될 수 있습니다.

SYNOPSIS

❒ VT_I4 cmxPmMxStop ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 NumChannel, [in] VT_I4 ChannelMask, [in] VT_I4 IsWaitComplete)

❒ VT_I4 cmxPmMxStopEmg ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 NumChannel, [in] VT_PI4 ChannelMask)

DESCRIPTION

지정한 모든 축에 대한 모션을 정지(停止)합니다. cmxPmMxStop() 함수는 정지(停止)시에 감속 후 정지(停止)를 수행하며, cmxPmMxStopEmg() 함수는 감속없이 즉시정지(停止)를 수행합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ NodeId : 노드번호. 노드번호는 3 부터 시작합니다.

▶ NumAxes : 동시에 작업을 수행할 대상 축의 수

▶ ChannelMask: 동시에 작업을 수행할 대상 축의 마스크

▶ IsWaitComplete: 정지가 완료될 때까지 기다릴지의 여부를 결정한다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

REFERENCE

윈도우 이벤트라는 것은 무엇입니까?
윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

EXAMPLE

C/C++

```
void CmxMotionDlg::OnStop()
{
	long BoardID = 0;
	long nAxes=0x3; //0,1 번 축을 정지시킵니다.
	GetDlgItcmx(IDC_btnStop)->EnableWindow(FALSE);
	cmxPmMxStop(BoardID, 3, 2, nAxes, TRUE, FALSE);
	GetDlgItcmx(IDC_btnStop)->EnableWindow(TRUE);
}
```

Visual Basic

```
`BoardID 는 0 이라고 가정합니다.

Private Sub btnStop_Click()
	Dim nRetVal As Long

	Dim NumChannel(2) As Long

	NumChannel(0) = 0
	NumChannel(1) = 1

	' PmMxStop(BoardID,  축 갯수, 배열, 완료대기여부, 블록여부)
	nRetVal = PmMxStop(BoardID, 3, 2, NumChannel(0), True, GetIsBlocking())

End Sub
```

Delphi

```
// * Description :
// * 이 함수는 버튼 이벤트에 의해 모션 동작을 정지(停止)하는 함수입니다.
// * BoardID 는 0 이라고 가정합니다.


procedure btnStopClick();
var
	NumChannel : Array[0..1] of LongInt;
	gnTargetAxis : LongInt;
begin
	NumChannel[0] := 0;
	NumChannel[1] := cmY1;
	gnTargetAxis := 2;
```

```
    // 정지(停止) 함수의 원형은 cmxMxSxStop([TargetAxis], [IsWaitComplete],
    // [IsBlocking]) 입니다.
    // TargetAxis : 정지(停止) 할 대상 축을 설정합니다.
    // IsWaitComplete : 대상 축이 완전히 정지(停止)할 때 까지
    //함수 반환을 하지 않습니다.
    // IsBlocking : 함수의 동작시 윈도우 메시지 처리의 여부를 판단합니다.
    // cmxFALSE 시에는 윈도우 메시지를 함수 내부에서 처리해주게 됩니다.
    //  보다 자세한 설명은 매뉴얼을 참고해주시기 바랍니다.

    cmxPmMxStop(BoardID, 3, gnTargetAxis, @NumChannel, cmxTRUE,  cmxFALSE);
end;
```

NAME

cmxPmMxIsDone

- 다축(多軸) 모션 완료 확인(確認)

INFORMATION

Multi Axes Control
VC++/VB
BCB/Delphi/.NET
Level 3
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxPmMxIsDone ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 NumChannel, [in] VT_PI4 NumChannel, [out] VT_PI4 IsDone)

DESCRIPTION

여러 개의 축에 대하여 지정한 모든 축의 모션이 완료됐는지를 확인(確認)합니다. 이 함수는 다축제어뿐 아니라 원점복귀나 단축모션제어 작업시에도 활용할 수 있습니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ NodeId : 노드번호. 노드번호는 3 부터 시작합니다.

▶ NumAxes : 동시에 작업을 수행할 대상 축의 수

▶ NumChannel : 작업완료를 확인(確認)할 대상 축의 배열 주소값. 이 배열의 크기는 NumAxes 값과 일치하거나 커야 합니다.

▶ IsDone : 다축구동 완료 여부를 판단할 수 있는 매개변수 입니다.

Value	Meaning
cmxFALSE	모션작업이 완료되지 않음
cmxTRUE	모션작업이 완료됨

RETURN VLAUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

SEE ALSO

cmxPmMxWaitDone

EXAMPLE

C/C++

```
// BoardID 는 0 이라고 가정합니다.

long nIsDone;
long nNumChannel[2] = {0, cmY1};
double fDistList[2] = {1000, 1000};

if(cmxPmMxMove(BoardID, 3, 2, nNumChannel, fDistList, cmxFALSE) != ERR_NONE){
        //Handle 은 사용자가 생성한 폼의 핸들 값입니다.
        // 에러메시지 출력
        return;
}

while (1){
    cmxPmMxIsDone(BoardID, 3, 2, nNumChannel, &nIsDone);
    if(nIsDone == cmxTRUE) break;
    else{
        // 다축 모션이 종료되지 않은 경우입니다. 적절한 처리를 합니다.
    }
}
```

Visual Basic

```
'BoardID 는 0 이라고 가정합니다.

Dim nNumChannel(2) As Long
Dim fDistList(2) As Double

' 대상 축 설정
nNumChannel(0) = 0
nNumChannel(1) = cmY1

' 대상 축에 대한 이송 거리 설정
fDistList(0) = 1000
fDistList(1) = 1000

If(PmMxMove(BoardID, 3, 2, nNumChannel(0), fDistList(0), cmxFALSE) <> ERR_NONE) Then
    // 에러메시지 출력
    Exit Sub
End If

While(PmMxIsDone(BoardID, 3, 2, nNumChannel(0), cmxTRUE) <> ERR_NONE) Then
    // 에러메시지 출력
    Exit Sub;
End If
```

Delphi

```
// 대상 축 설정
// BoardID 는 0 이라고 가정합니다.

nNumChannel[0] := 0;
nNumChannel[1] := cmY1;
```

```
// 대상 축에 대한 이송 거리 설정
fDistList[0] := 1000;
fDistList[1] := 1000;

if(cmxPmMxMove(BoardID, 3, 2, @nNumChannel, @fDistList) <> ERR_NONE) then begin
	// 에러메시지 출력
end;

while(cmxPmMxIsDone(BoardID, 3, 2, @nNumChannel, @IsDone) <> ERR_NONE) do begin

	// 여기서 IsDone 이 cmxTRUE 이면 Loop 를 탈출하면 됩니다.
	…
end;

if(cmxErrGetLastCode() <> ERR_NONE) then begin
	// 에러메시지 출력
end;
```

NAME

cmxPmMxWaitDone

- 다축(多軸) 모션 완료대기(完了待機)

INFORMATION

- Multi Axes Control
- VC++/VB
- BCB/Delphi/.NET
- Level 3
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxPmMxWaitDone ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 NumChannel, [in] VT_PI4 NumChannel)

DESCRIPTION

여러 축에 대하여 모션이 완료(完了)될 때까지 기다립니다. 이 함수는 다축제어뿐 아니라 원점복귀(原點復歸)나 단축(單軸)모션제어 작업시에도 활용할 수 있습니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ BoardID : 사용자가 설정한 디바이스(보드) ID.

▶ NodeId : 노드번호. 노드번호는 3 부터 시작합니다.

▶ NumAxes : 동시에 작업을 수행할 대상 축의 수

▶ NumChannel : 동시에 작업을 수행할 대상 축의 배열 주소값. 이 배열의 크기는 NumAxes 값과 일치하거나 커야 합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
ERR_NONE	수행 성공

SEE ALSO

cmxPmMxIsDone

REFERENCE

□ INP 입력신호가 Enable 로 설정되었으면 Command 펄스 출력이 완료되어도 INP 입력이 ON 이 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환되지 않습니다.

□ 스탭 드라이브를 사용 중인 고객(顧客)님들께서는 다음을 참조해 주십시요.

스탭 드라이브는 INP 출력이 없는 경우가 일반적인데, 고객(顧客)님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스탭 드라이브를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다. 고객(顧客) 여러분들께서는 스탭 드라이브 사용시에 이점을 주의해주시기를 부탁드립니다.

□ 서보 드라이브의 LSP , LSN 신호를 사용 중인 고객(顧客)님께서는 다음을 참조해 주십시요. 서보드라이브의 입력 신호 중 하나인 EL(End of Limit) 신호는 저희 ㈜커미조아 모션 보드 뿐만 아니라 서보드라이브에도 전달 될 수 있도록 설정할 수 있습니다. 통상적으로 LSP 신호와 LSN 신호로 불리어 지는 이 신호는 실제 기구물에서 양의 방향(Positive Direction) 혹은 음의 방향(Negative Direction) 에 장착되어 있는 EL(End of Limit) 신호를 서보 드라이브 측에 전달하기 위한 용도로 사용됩니다.

그러나, 모션 소프트웨어에서 INP 설정이 되어 있는 경우 EL 신호가 검출 된 후에 일부 서보 드라이브에서는 진행 방향에서 정지(停止) 한후 더 이상 움직이지 않는 상황이 발생하며, 이 상황에서 INP 신호가 출력되지 않아, 모션 이송이 완료되지 못하고, 명시적으로 STOP 을 해줘야만 하는 강제적으로 모션 종료가 되는 현상이 발생할 수 있습니다. 이 현상은 EL 모드를 통해 원점 복귀를 하는 상황에서도 발생할 수 있습니다.

따라서, 이러한 경우에는 반드시 원점 복귀나 EL 검출시에 인터럽트 이벤트나 타이머를 통해 INP 를 무기한 대기하는 현상에 대해서 적절히 대처하시거나 INP 신호 사용을 배제 해야 합니다.

윈도우 이벤트라는 것은 무엇입니까?

윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

EXAMPLE

C/C++

```
// BoardID 는 0 이라고 가정합니다.

long nNumChannel[2] = {0, cmY1};
double fDistList[2] = {1000, 1000};

if(cmxPmMxMoveStart(BoardID, 3, 2, nNumChannel, fDistList) != ERR_NONE){
        //Handle 은 사용자가 생성한 폼의 핸들 값입니다.
        // 에러메시지 출력
        return ;
}

//모션이 완료 될 때 까지 기다립니다.
if(cmxPmMxWaitDone(BoardID, 3, 2, nNumChannel, cmxFALSE) != ERR_NONE){
        // 에러메시지 출력
        return ;
}
```

Visual Basic

```
'BoardID 는 0 이라고 가정합니다.
```

```
Dim nNumChannel(1)
Dim fDistList(1)

‘ 대상 축 설정
nNumChannel(0) = 0
nNumChannel(1) = cmY1

‘ 대상 축에 대한 이송 거리 설정
fDistList(0) = 1000
fDistList(1) = 1000

if(PmMxMove(BoardID, 3, 2, nNumChannel(0), fDistList(0), cmxFALSE) <> ERR_NONE) then
	// 에러메시지 출력
	exit sub
end if

‘Wait till motion done
if(PmMxWaitDone(BoardID, 3, 2, nNumChannel(0), cmxFALSE) <> ERR_NONE) then
	// 에러메시지 출력
	exit sub
end if
```

```
Delphi

// BoardID 는 0 이라고 가정합니다.
// 대상 축 설정

nNumChannel[0] := 0;
nNumChannel[1] := cmY1;

// 대상 축에 대한 이송 거리 설정
fDistList[0] := 1000;
fDistList[1] := 1000;
if(cmxPmMxMove(BoardID, 3, 2, @nNumChannel, @fDistList, cmxFALSE) <> ERR_NONE) then
begin
	// 에러메시지 출력
	// 적절한 에러처리를 수행합니다.
end;
// Wait till motion done //
if(cmxPmMxWaitDone(BoardID, 3, 2, @nNumChannel, cmxFALSE) <> ERR_NONE) then begin
	// 에러메시지 출력
	exit;
end;
```

15.3 기본 보간 제어 (Interpolation Motion)

이 단원에서는 보간(Interpolation) 모션제어에 관련된 함수들을 소개합니다. 보간 모션제어란 두 축 이상의 축이 연동되어 직선 보간(Linear Interpolation), 원호 보간(Circular Interpolation) 등의 모션을 수행하는 것을 의미합니다.

보간 제어는 사용자가 원하는 경로를 추종하기 위해서 보간 이송에 관련된 각 축의 속도가 자동으로 조절되면서 이송을 수행합니다.

㈜커미조아의 모션 모듈은 해당 원격 노드 안에서 통합 모션 축을 대상으로 직선 보간 제어 기능을 제공합니다. 당사의 모션 모듈이 제공하는 보간 제어의 사양 표는 다음과 같습니다.

보간 제어 방식	축 구성 제한 사항
직선 보간 제어	단일 원격 노드 내의 모든 모션 축
원호 보간 제어	단일 모션 모듈 내의 2 축

㈜커미조아의 모션 모듈은 해당 원격 노드 안에서 축 구성에 제약이 없는 직선보간, 2 축 원호 보간 이송 작업을 자동으로 수행하는 기능을 제공합니다. 직선 보간과 원호 보간은 “기본 보간 제어” 기능으로 분류됩니다.

15.3.1 직선 보간

ceSDK 에서는 동일 원격 노드에 속하는 모션 모듈에 대하여 축들간의 “직선 보간” 구동에 아무런 제약 없이 사용하실 수 있습니다. 예를 들어 동일 원격 노드의 1 번 모션 모듈과 9 번 모션 모듈의 축간에 “직선 보간” 구동이 가능합니다.

15.3.2 원호 보간

ceSDK 에서는 동일 모션 모듈에 대하여 “원호 보간” 구동 기능을 사용하실 수 있습니다. 동일 원격 노드에 속하는 서로 다른 모션 모듈에 대해서 “원호 보간” 구동은 허용되지 않습니다.

15.3.3 함수 요약

"기본 보간 제어" 에 관련된 함수들은 모두 "Ix..." 의 형식으로 함수 명이 구성되며 그 리스트는 다음과 같습니다.

Summary of Functions
❒ VT_I4 cmxPmIxMapAxes ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex, [in] VT_I4 MapMask) 해당 원격 노드의 보간 대상 축 그룹(Group)을 설정합니다.
❒ VT_I4 cmxPmIxUnMapAxes ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex) 설정된 보간 대상 축 그룹(Group)을 해제합니다.
❒ VT_I4 cmxPmIxSetSpeedPattern ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex, [in] VT_I4 IsVectorSpeed, [in] VT_I4 SpeedMode, [in] VT_R8 Vel, [in] VT_R8 Acc, [in] VT_R8 Dec) 보간 이송 속도를 설정합니다. 보간 이송 속도는 마스터 속도모드와 백터 속도모드를 설정 할 수 있습니다.
❒ VT_I4 cmxPmIxGetSpeedPattern ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex, [out] VT_PI4 IsVectorSpeed, [out] VT_PI4 SpeedMode, [out] VT_PR8 Vel, [out] VT_PR8 Acc, [out] VT_PR8 Dec) 설정된 보간 이송 속도를 반환합니다. 보간 이송 속도는 마스터 속도 혹은 백터 속도모드에 해당하는 속도모드를 반환합니다.
❒ VT_I4 cmxPmIxSetSpeedPattern_T ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex, [in] VT_I4 IsVectorSpeed, [in] VT_I4 SpeedMode, [in] VT_R8 Vel, [in] VT_R8 AccTime, [in] VT_R8 DecTime) 보간 이송 속도를 설정합니다. 보간 이송 속도는 마스터 속도모드와 백터 속도모드를 설정 할 수 있습니다.
❒ VT_I4 cmxPmIxGetSpeedPattern_T ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex, [out] VT_PI4 IsVectorSpeed, [out] VT_PI4 SpeedMode, [out] VT_PR8 Vel, [out] VT_PR8 AccTime, [out] VT_PR8 DecTime) 설정된 보간 이송 속도를 반환합니다. 보간 이송 속도는 마스터 속도 혹은 백터 속도모드에 해당하는 속도모드를 반환합니다.
❒ VT_I4 cmxPmIxSetVelCorrMode ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex, [in] VT_I4 VelCorrOpt1, [in] VT_I4 VelCorrOpt2) 보간 속도 보정 모드를 설정합니다.
❒ VT_I4 cmxPmIxGetSpeedPattern ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex, [out] VT_PI4 VelCorrOpt1, [out] VT_PI4 VelCorrOpt2) 설정된 보간 속도 보정 모드를 반환합니다.
❒ VT_I4 cmxPmIxLineStart ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex, [in] VT_PR8 DistList) 보간 제어에 있어 직선 보간을 수행하며, 상대 좌표 이송을 수행합니다. 이 함수는 모션 시작 후 바로 반환됩니다.
❒ VT_I4 cmxPmIxLineToStart ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex, [in] VT_PR8 PosList) 보간 제어에 있어 직선 보간을 수행하며, 절대 좌표 이송을 수행합니다. 이 함수는 모션 시작 후 바로 반환됩니다.
❒ VT_I4 cmxPmIxArcAStart ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex, [in] VT_R8 XCentOffset, [in] VT_R8 YCentOffset, [in] VT_R8 EndAngle) 보간 제어에 있어 원호 보간을 수행하며, 상대적 중심 좌표와 각도를 통해 원호 보간을 수행합니다. 이 함수는 모션 시작 후 바로 반환됩니다.
❒ VT_I4 cmxPmIxArcAToStart ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex, [in] VT_R8 XCent, [in] VT_R8 YCent, [in] VT_R8 EndAngle) 보간 제어에 있어 원호 보간을 수행하며, 절대적 중심 좌표와 각도를 통해 원호 보간을 수행합니다. 이 함수는 모션 시작 후 바로 반환됩니다.
❒ VT_I4 cmxPmIxArcPStart ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex, [in] VT_R8 XCentOffset, [in] VT_R8 YCentOffset, [in] VT_R8 XEndPointDist, [in] VT_R8 YEndPointDist, [in] VT_I4 Direction) 보간 제어에 있어 원호 보간을 수행하며, 상대적 중심 좌표와 종점 좌표를 통해 원호 보간을 수행합니다. 이 함수는 모션 시작 후 바로 반환됩니다.
❒ VT_I4 cmxPmIxArcPToStart ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex, [in] VT_R8 XCent, [in] VT_R8 YCent, [in] VT_R8 XEndPos, [in] VT_R8 YEndPos, [in] VT_I4 Direction) 보간 제어에 있어 원호 보간을 수행하며, 절대적 중심 좌표와 종점 좌표를 통해 원호 보간을 수행합니다. 이 함수는 모션 시작 후 바로 반환됩니다.
❒ VT_I4 cmxPmIxArc3PStart ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex, [in] VT_R8 P2X, [in] VT_R8 P2Y, [in] VT_R8 P3X, [in] VT_R8 P3Y, [in] VT_R8 EndAngle) 보간 제어에 있어 원호 보간을 수행하며, 현재 좌표와 상대적 두 좌표를 통해 원호 보간을 수행합니다. 이 함수는 모션 시작 후 바로 반환됩니다.
❒ VT_I4 cmxPmIxArc3PToStart ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex, [in] VT_R8 P2X, [in] VT_R8 P2Y, [in] VT_R8 P3X, [in] VT_R8 P3Y, [in] VT_R8 EndAngle) 보간 제어에 있어 원호 보간을 수행하며, 현재 좌표와 절대적 두 좌표를 통해 원호 보간을 수행합니다. 이 함수는 모션 시작 후 바로 반환됩니다.

❒ VT_I4 cmxPmIxStop ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex) 보간 제어 구동 이송을 감속 후 정지합니다.
❒ VT_I4 cmxPmIxStopEmg ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex) 보간 제어 구동 이송을 비상 정지합니다.
❒ VT_I4 cmxPmIxIsDone ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex, [out] VT_PI4 IsDone) 보간 제어 구동 이송의 보간 완료를 확인합니다.
❒ VT_I4 cmxPmIxWaitDone ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex) 보간 제어 구동 이송의 보간 완료 시점까지 대기합니다.

15.3.4 함수 설명

NAME

cmxPmIxMapAxes

- 보간 대상 축 그룹(Group) 설정

INFORMATION

Interpolation Motion

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 3

다소 주의

보간 대상 축 그룹은 모든 보간 이송 작업 전에 선행되어야 합니다.

SYNOPSIS

❒ VT_I4 cmxPmIxMapAxes

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex, [in] VT_I4 MapMask)

DESCRIPTION

cmxPmIxMapAxes() 함수는 보간작업을 수행할 축들을 맵번호(Map index)로 맵핑(Mapping) 합니다. 맵번호는 다른 "기본보간제어"에 관련된 함수들의 첫 번째 매개 변수로 전달됨으로써 각 함수들이 제어해야 할 축들에 대한 정보가 간편하게 전달됩니다. 따라서 다른 "기본보간제어"에 관련된 함수들을 사용하기 전에 가장 먼저 이 함수를 사용하여 "기본보간제어"에 사용할 축들을 맵핑 하여야 합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***MapIndex*** : 맵 번호(Map index), 맵 번호는 반드시 0 ~ 15 안에서 설정하여야 합니다.

▶ ***MapMask*** : 축 맵에 포함할 축들을 지정할 마스크 값(BIT0 ~ BIT31).

주의	보간 제어는 대상 원격 노드 내의 축들로 제한되며, 축 그룹에 포함할 축 번호는 원격 노드안에서의 제한된 통합 축 번호로 설정하여야 합니다. 즉, 대상 원격 노드의 첫번째 모션 모듈에 속한 첫번째 축이 0 번 축이 됩니다.

이 값의 BIT0~BIT31 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. 이 매개 변수의 각 비트 별 정보는 아래 표와 같습니다.

예) 8 축을 사용하는 경우

Bit Number	Meaning
BIT0 (ccmxX1_MASK)	0 번 축의 포함여부 : 0 => 포함안함, 1 => 포함
BIT1 (ccmxY1_MASK)	1 번 축의 포함여부 : 0 => 포함안함, 1 => 포함
BIT2 (ccmxZ1_MASK)	2 번 축의 포함여부 : 0 => 포함안함, 1 => 포함
BIT3 (ccmxU1_MASK)	3 번 축의 포함여부 : 0 => 포함안함, 1 => 포함
BIT4 (ccmxX2_MASK)	4 번 축의 포함여부 : 0 => 포함안함, 1 => 포함
BIT5 (ccmxY2_MASK)	5 번 축의 포함여부 : 0 => 포함안함, 1 => 포함
BIT6 (ccmxZ2_MASK)	6 번 축의 포함여부 : 0 => 포함안함, 1 => 포함
BIT7 (ccmxU2_MASK)	7 번 축의 포함여부 : 0 => 포함안함, 1 => 포함

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

SEE ALSO

cmxPmIxUnMapAxes

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

#define MAP0 0              // 보간 맵 번호

// BoardID 는 0 이라 지정합니다.
/* 다음과 같은 2 개의 원격 노드가 존재한다고 가정합니다.
Node Master 1 : NodeID = 1, 16 Axes
Node Master 2 : NodeID = 2, 8 Axes

19 번, 20 번 축을 대상으로 보간 맵 설정을 수행합니다.
보간 제어는 원격 노드안의 제한된 통합 축 번호로 관리되므로, 2 번째 원격 노드에서의 통합 축 번호인 2 번, 3 번 축을 MapMask 로
설정하여야 합니다. */

//2 번 노드에 있는 0 번 축과 1 번 축을 하나의 맵으로 묶습니다.
cmxPmIxMapAxes(BoardID, 3, MAP0,  ccmxX1_MASK | ccmxY1_MASK );

//MAP0 로 묶여 있는 축들을 해제 합니다.
cmxPmIxUnMapAxes(BoardID, 3, MAP0 );
```

NAME

cmxPmIxUnMapAxes

- 보간 대상 축 그룹(Group) 해제

INFORMATION

Interpolation Motion

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 3

다소 주의

대상 축 그룹에 대하여 보간 이송 작업 완료 후, 보간 제어 사용 종료를 하고자 하는 경우에 해제합니다.

SYNOPSIS

❒ VT_I4 cmxPmIxUnMapAxes ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex)

DESCRIPTION

cmxPmIxUnMapAxes() 함수는 보간작업을 수행할 보간 대상 축 그룹을 해제합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***MapIndex*** : 보간 대상 축 그룹을 해제할 맵 번호(Map index).

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

SEE ALSO

cmxPmIxMapAxes

NAME

cmxPmIxSetSpeedPattern /

cmxPmIxGetSpeedPattern

- 보간 이송 속도 설정 및 반환

INFORMATION

- Interpolation Motion
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 3
- 다소 주의

SYNOPSIS

❒ VT_I4 cmxPmIxSetSpeedPattern

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex, [in] VT_I4 IsVectorSpeed, [in] VT_I4 SpeedMode, [in] VT_R8 Vel, [in] VT_R8 Acc, [in] VT_R8 Dec)

❒ VT_I4 cmxPmIxGetSpeedPattern

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex, [out] VT_PI4 IsVectorSpeed, [out] VT_PI4 SpeedMode, [out] VT_PR8 Vel, [out] VT_PR8 Acc, [out] VT_PR8 Dec)

DESCRIPTION

cmxPmIxSetSpeedPattern() 함수는 "기본보간제어"의 이송 속도에 대한 환경설정을 정의합니다. 사용자가 지정한 작업 속도는 "IsVectorSpeed"의 설정 값이 'cmxTRUE'이면 벡터 속도(Vector Speed), 'cmxFALSE'이면 마스터 속도(Master Speed)가 적용됩니다. "벡터 속도"에 대한 자세한 내용은 아래의 "REFERENCE" 항목을 참조하십시오.

보간 작업 속도를 벡터 속도로 설정해야만 하는 특별한 경우를 제외하고는 보간 작업 속도를 마스터 속도로 설정하는 것이 모터의 최대속도를 활용하는데 있어서 편리합니다.

cmxPmIxGetSpeedPattern() 함수는 "기본보간제어" 의 이송 속도에 대한 설정된 값을 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***MapIndex*** : 맵 번호(Map index), 이 맵번호를 사용하기 전에 먼저 cmxPmIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑 되어 있어야 합니다.

▶ ***IsVectorSpeed*** : 벡터 속도 또는 마스터 속도 모드를 설정 혹은 반환합니다.

Value	Meaning
0 (cmxFALSE)	마스터 속도 모드. (Master Speed Mode)
1 (cmxTRUE)	벡터 속도 모드. (Vector Speed Mode)

▶ ***SpeedMode*** : 가감속 모드를 설정 혹은 반환합니다. 설정 값은 다음과 같습니다.

Value	Meaning
0 (cmxMODE_C)	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 (cmxMODE_T)	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 (cmxMODE_S)	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.

▶ ***Vel*** : 마스터 속도 모드 일 때는 작업속도 비율(%)을 설정 혹은 반환합니다. 벡터 속도 모드 일 때는 PPS 단위를 사용하여 설정 혹은 반환합니다.

▶ ***Acc*** : 마스터 속도 모드 일 때는 가속도 비율(%)을 설정 혹은 반환합니다. 벡터 속도 모드 일 때는 PPS 단위를 사용하여 가속도를 설정 혹은 반환합니다.

▶ ***Dec*** : 마스터 속도 모드 일 때는 감속도 비율(%)을 설정 혹은 반환합니다. 벡터 속도 모드 일 때는 PPS 단위를 사용하여 감속도를 설정 혹은 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

REFERENCE

보간 제어의 속도에는 마스터 속도 모드와 벡터 속도 모드가 존재합니다. 본 함수의 설명을 잘 읽고, 보간 속도 설정에 주의를 기울여 주시기 바랍니다. 특히 서로 다른 속성을 가지고 있는 서보 드라이버나 스텝 드라이버에서는 보간 속도 설정에 반드시 주의를 요합니다.

□ 직선 보간 이송시 작업속도의 적용

마스터 속도 모드(Master Speed Mode)로 보간 작업 시에는 각 축의 속도가 각 축의 이송거리에 비례하여 자동으로 설정됩니다. 이때 cmxPmIxSetSpeedPattern() 함수의 WorkSpeed 매개 변수를 통하여 지정되는 보간 작업속도는 마스터 속도로 적용됩니다.
각 보간 이송시에 이송거리가 가장 큰 축을 "마스터 축" 이라고 하며 마스터 축의 속도를 "마스터 속도"라 합니다. 각 보간 이송시에 마스터 축의 속도는 사용자가 지정한 보간 작업속도로 설정되며, 마스터 축 이외의 다른 축의 속도는 마스터 축과 해당 축의 이송 거리 비에 따라서 자동으로 설정됩니다.

□ 보간 작업속도의 적용 예

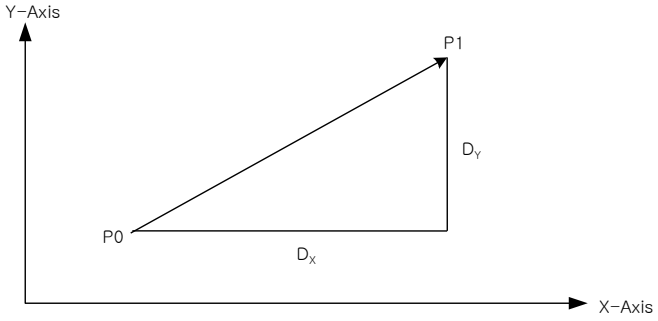
cmxPmIxSetSpeedPattern() 함수의 WorkSpeed 를 10000 으로 설정하고 X, Y, Z 축의 보간 작업을 수행하는 경우에 이송 거리에 따른 각 축의 속도 관계는 아래와 같습니다. (표에서 배경이 회색으로 되어 있는 것은 마스터 축임을 의미하는 것입니다.)

	이동 거리			각 축의 이동 속도		
	X	Y	Z	Vx	Vy	Vz
보간이동 1	1000	2000	5000	2000	4000	10000
보간이동 2	5000	1000	2000	10000	2000	4000
보간이동 3	2000	20000	10000	1000	10000	5000
보간이동 4	10000	0	0	10000	0	0

[표 10] 마스터 속도 모드에 대한 실제 속도 적용 예시 표

□ 직선 보간 이동 시의 벡터 속도

[[그림 15-1] 은 2 축(편의상 X, Y 축으로 가정) 직선 보간 이송을 그래프로 나타낸 것입니다.



[그림 15-1] 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 속도에 대한 관계식은 다음과 같습니다.

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

4 축의 경우에는 각축의 속도에 대한 관계식은 다음과 같습니다.

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

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

#define MAP0 0        // 보간 맵 0
#define NODE_ID 1 // 노드 ID 1

long BoardID = 0;
cmxPmIxMapAxes(BoardID, 3, MAP0, ccmxX1_MASK | ccmxY1_MASK );

/*MAP0 로 설정된 축들을 벡터 스피드로
가속도 10000, 감속도 10000, 등속도 2000 으로 설정합니다.*/
cmxPmIxSetSpeedPattern(BoardID, 3, MAP0, cmxTRUE, cmxMODE_T, 2000, 10000, 10000 );

long nSpeedMode = 0, nIsVectorSpeed = 0;
double fVel = 0.0f, fAcc = 0.0f, fDec = 0.0f;

//MAP0 에 설정된 속도 패턴 값들을 반환합니다.
cmxPmIxGetSpeedPattern (BoardID, 3, MAP0, &nIsVectorSpeed, &nSpeedMode, &fVel, &fAcc, &fDec );
```

NAME

cmxPmIxSetSpeedPattern_T /

cmxPmIxGetSpeedPattern_T

- 보간 이송 속도 설정 및 반환

INFORMATION

Interpolation Motion

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 3

다소 주의

SYNOPSIS

❒ VT_I4 cmxPmIxSetSpeedPattern_T

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex, [in] VT_I4 IsVectorSpeed, [in] VT_I4 SpeedMode, [in] VT_R8 Vel, [in] VT_R8 AccTime, [in] VT_R8 DecTime)

❒ VT_I4 cmxPmIxGetSpeedPattern_T

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex, [out] VT_PI4 IsVectorSpeed, [out] VT_PI4 SpeedMode, [out] VT_PR8 Vel, [out] VT_PR8 Acc_T, [out] VT_PR8 Dec_T)

DESCRIPTION

cmxPmIxSetSpeedPattern_T() 함수는 "기본보간제어"의 이송 속도에 대한 환경설정을 정의합니다. 사용자가 지정한 작업 속도는 "IsVectorSpeed"의 설정 값이 'cmxTRUE'이면 벡터 속도(Vector Speed), 'cmxFALSE'이면 마스터 속도(Master Speed)가 적용됩니다. "벡터 속도"에 대한 자세한 내용은 아래의 "REFERENCE" 항목을 참조하십시오.

보간 작업 속도를 벡터 속도로 설정해야만 하는 특별한 경우를 제외하고는 보간 작업 속도를 마스터 속도로 설정하는 것이 모터의 최대속도를 활용하는데 있어서 편리합니다.

cmxPmIxGetSpeedPattern_T() 함수는 "기본보간제어" 의 이송 속도에 대한 설정된 값을 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***MapIndex*** : 맵 번호(Map index), 이 맵번호를 사용하기 전에 먼저 cmxPmIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑 되어 있어야 합니다.

▶ ***IsVectorSpeed*** : 벡터 속도 또는 마스터 속도 모드를 설정 혹은 반환합니다.

Value	Meaning
0 (cmxFALSE)	마스터 속도 모드. (Master Speed Mode)
1 (cmxTRUE)	벡터 속도 모드. (Vector Speed Mode)

▶ ***SpeedMode*** : 가감속 모드를 설정 혹은 반환합니다. 설정 값은 다음과 같습니다.

Value	Meaning
0 (cmxMODE_C)	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 (cmxMODE_T)	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 (cmxMODE_S)	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.

▶ ***Vel*** : 마스터 속도 모드 일 때는 작업속도 비율(%)을 설정 혹은 반환합니다. 벡터 속도 모드 일 때는 PPS 단위를 사용하여 설정 혹은 반환합니다.

▶ ***AccTime*** : 가속 시간을 설정 혹은 반환합니다.

▶ ***DecTime*** : 감속 시간을 설정 혹은 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

REFERENCE

보간 제어의 속도에는 마스터 속도 모드와 벡터 속도 모드가 존재합니다. 본 함수의 설명을 잘 읽고, 보간 속도 설정에 주의를 기울여 주시기 바랍니다. 특히 서로 다른 속성을 가지고 있는 서보 드라이버나 스텝 드라이버에서는 보간 속도 설정에 반드시 주의를 요합니다.

□ 직선 보간 이송시 작업속도의 적용

마스터 속도 모드(Master Speed Mode)로 보간 작업 시에는 각 축의 속도가 각 축의 이송거리에 비례하여 자동으로 설정됩니다. 이때 cmxPmIxSetSpeedPattern_T() 함수의 WorkSpeed 매개 변수를 통하여 지정되는 보간 작업속도는 마스터 속도로 적용됩니다.
각 보간 이송시에 이송거리가 가장 큰 축을 "마스터 축" 이라고 하며 마스터 축의 속도를 "마스터 속도"라 합니다. 각 보간 이송시에 마스터 축의 속도는 사용자가 지정한 보간 작업속도로 설정되며, 마스터 축 이외의 다른 축의 속도는 마스터 축과 해당 축의 이송 거리 비에 따라서 자동으로 설정됩니다.

□ 보간 작업속도의 적용 예

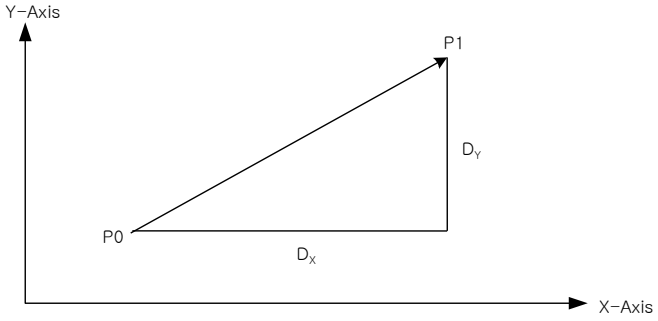
cmxPmIxSetSpeedPattern_T() 함수의 WorkSpeed 를 10000 으로 설정하고 X, Y, Z 축의 보간 작업을 수행하는 경우에 이송 거리에 따른 각 축의 속도 관계는 아래와 같습니다. (표에서 배경이 회색으로 되어 있는 것은 마스터 축임을 의미하는 것입니다.)

	이동 거리			각 축의 이동 속도		
	X	Y	Z	Vx	Vy	Vz
보간이동 1	1000	2000	5000	2000	4000	10000
보간이동 2	5000	1000	2000	10000	2000	4000
보간이동 3	2000	20000	10000	1000	10000	5000
보간이동 4	10000	0	0	10000	0	0

[표 11] 마스터 속도 모드에 대한 실제 속도 적용 예시 표

□ 직선 보간 이동 시의 벡터 속도

[[그림 15-1] 은 2 축(편의상 X, Y 축으로 가정) 직선 보간 이송을 그래프로 나타낸 것입니다.



[그림 15-2] 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 속도에 대한 관계식은 다음과 같습니다.

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

4 축의 경우에는 각축의 속도에 대한 관계식은 다음과 같습니다.

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

NAME

cmxPmIxSetVelCorrMode /

cmxPmIxGetVelCorrMode

- 직선 보간 속도 보정 모드 설정 및 반환

INFORMATION

- Interpolation Motion
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 3
- 다소 주의

SYNOPSIS

❒ VT_I4 cmxPmIxSetVelCorrMode

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex, [in] VT_I4 VelCorrOpt1, [in] VT_I4 VelCorrOpt2)

❒ VT_I4 cmxPmIxGetVelCorrMode

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex, [out] VT_PI4 VelCorrOpt1, [out] VT_PI4 VelCorrOpt2)

DESCRIPTION

cmxPmIxSetVelCorrMode() 함수는 직선 보간 속도 보정 모드를 설정합니다.

직선보간 이송 시에 각 축의 속도는 각 축의 이송거리 에 비례하게 됩니다. 따라서 이송거리가 가장 긴 축의 속도가 가장 빠르게 되며, 이 축을 마스터축(Master axis)이라 하고 나머지 축들을 슬레이브축(Slave axis)이라 합니다.

기본적으로 직선보간 시에 마스터축의 속도는 cmxPmIxSetVelCorrMode() 함수에 의해서 지정됩니다. 그리고 슬레이브축들의 속도는 마스터축과의 이송 거리비에 비례하여 이송속도가 계산되게 됩니다. 그런데 이렇게 계산된 슬레이브축의 이송속도가 정격속도를 초과하게 되면 문제가 발생할 수 있습니다. 예를 들어, 스텝모터를 사용하는 경우에는 탈조현상이 일어날 수 있습니다.
따라서 이러한 경우에는 슬레이브축의 속도를 정격속도 이하로 낮추는 것이 바람직한데 슬레이브축의 속도만을 낮추면 보간 경로를 유지하지 못하므로 마스터축의 속도를 조정하여 결과적으로 슬레이브축의 속도가 낮아지도록 해야합니다. ㈜커미조아의 CEIP 모션제어 모듈은 이러한 보정을 자동으로 수행해줍니다.

cmxPmIxGetVelCorrMode() 함수는 설정된 직선 보간 속도 보정 모드를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId***: 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***MapIndex*** : 맵 번호(Map index), 이 맵번호를 사용하기 전에 먼저 cmxPmIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑 되어 있어야 합니다.

▶ ***VelCorrOpt1***: 직선보간 속도 보정을 할 것인지를 설정 혹은 설정 상태를 반환합니다.

Value	Meaning
0	속도 보정을 하지 않습니다.
1	작업 속도만 보정합니다.
2 [Default]	작업 속도 및 가속도, 감속도를 모두 보정합니다. (스텝모터의 경우 작업속도가 크지 않아도 가속도, 감속도가 정격속도를 넘어서면 탈조현상이 일어날 수 있으므로 가속도, 감속도 모두 보정을 해주는 것을 권장합니다.)

▶ ***VelCorrOpt2*** : 직선 보간 속도 보정 시 슬레이브 축의 작업 속도 및 가속, 감속 한계 값을 정격 속도만을 사용할 지, 아니면 정격 속도에 보간 속도비(%) 를 곱한 값을 사용할 지를 설정 혹은 설정 상태를 반환합니다.

Value	Meaning
0	작업 속도 및 가속, 감속 모두에 대하여 정격 값 설정을 이송 속도의 한계 값으로 적용합니다.
1 [Default]	작업속도의 한계 값을 정격 속도에 보간 속도비(%)를 곱한 값을 한계 값으로 적용하고, 가속, 감속의 한계 값은 정격 값을 적용합니다 .
2	작업 속도와 가속, 감속 모두에 대하여 정격 속도에 보간 속도비(%)를 곱한 값을 한계 값으로 적용합니다.

이 옵션은 보간 속도 모드가 마스터 속도 모드로 설정된 경우에만 적용되며 벡터 속도 모드에서는 무시됩니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

#define MAP0 0
#define NODE_ID 1

long BoardID = 0;

cmxPmIxMapAxes(BoardID, 3, MAP0, ccmxX1_MASK | ccmxY1_MASK );

/*MAP0 로 설정된 축들에 대하여 직선 보간 속도 보정을 작업 속도 및 가감속에 적용하고,
정격 속도에 보간 속도비(%)를 곱한 값을 한계 값으로 적용 되도록 설정합니다 .*/
Long nVelCorrOpt1 = 2, nVelCorrOpt2 = 2;
cmxPmIxSetVelCorrMode(BoardID, 3, MAP0, nVelCorrOpt1, VelCorrOpt2 );

// 설정된 직선 보간 속도 보정 모드를 반환합니다.
cmxPmIxGetSpeedPattern(BoardID, 3, MAP0, &nVelCorrOpt1, &VelCorrOpt2 );
```

NAME

cmxPmIxLineStart

- 직선보간 상대좌표 이송

INFORMATION

Interpolation Motion

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 3

이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인합니다.

SYNOPSIS

❒ VT_I4 cmxPmIxLineStart ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex, [in] VT_PR4 DistList)

DESCRIPTION

cmxPmIxLineStart() 함수는 현재 위치로부터의 상대 좌표 직선 보간 이송을 수행합니다. cmxPmIxLineStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***MapIndex*** : 맵 번호(Map index), 이 맵번호를 사용하기 전에 먼저 cmxPmIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑 되어 있어야 합니다.

▶ ***DistList*** : 현재 위치로부터의 상대적인 이송 좌표 값(각 축의 이송 거리 값)의 배열 주소. 이 배열의 크기는 cmxPmIxMapAxes() 함수를 통하여 맵핑된 축의 수와 일치하여야 합니다. 거리의 단위는 "Unit distance"에 의해 정의되는 논리적 거리를 적용합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

SEE ALSO

cmxPmIxLineToStart

REFERENCE

□ 논리적 단위 거리는 cmxPmCfgSetUnitDist() 함수에 의해 결정됩니다.

□ cmxPmIxLineStart() 함수를 사용하는 경우에는 cmxPmIxIsDone() 함수나 cmxPmIxWaitDone() 함수를 사용하여 모션의 완료를 확인할 수 있습니다.

윈도우 이벤트라는 것은 무엇입니까?

윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용 프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

/* 원격 노드(Node ID : 1)의 0, 1 번 축에 대하여 마스터 속도 모드로 상대 좌표 직선 보간 이송을 수행하는 예제입니다. */

#define MAP0 0
#define NODE_ID 1

long BoardID = 0;

cmxPmIxMapAxes(BoardID, 3, MAP0, ccmxX1_MASK | ccmxY1_MASK );

/*보간 속도 패턴 설정에서 벡터 모드를 사용하지 않을 경우에는
각 축에 대한 속도를 미리 설정해 주어야 합니다.*/
cmxPmCfgSetSpeedPattern(BoardID, 3, ccmxX1, cmxMODE_T, 4000, 20000, 20000 );
cmxPmCfgSetSpeedPattern(BoardID, 3, ccmxY1, cmxMODE_T, 5000, 20000, 20000 );

/*마스터 속도 모드(벡터 모드를 사용하지 않음)로 사용할 경우에는 속도를
미리 설정한 속도의 비율로 설정해 줍니다.*/
cmxPmIxSetSpeedPattern(BoardID, 3, MAP0, cmxFALSE, cmxMODE_T, 100, 80, 80 );

//두 축에 대한 이동 거리를 설정합니다.
double fDistList[2] = { 13000, 90000 };

//보간 구동을 시작합니다.
cmxPmIxLineStart(BoardID, 3, MAP0, fDistList );
cmxPmIxWaitDone(BoardID, 3, MAP0, cmxFALSE );
```

NAME

cmxPmIxLineToStart

- 직선보간 절대좌표 이송

INFORMATION

- Interpolation Motion
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 3
- 이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인합니다.

SYNOPSIS

❒ VT_I4 cmxPmIxLineToStart ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex, [in] VT_PR8 PosList)

DESCRIPTION

cmxPmIxLineToStart() 함수는 절대 좌표 직선 보간 이송을 수행합니다. cmxPmIxLineToStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***MapIndex*** : 맵 번호(Map index), 이 맵번호를 사용하기 전에 먼저 cmxPmIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑 되어 있어야 합니다.

▶ ***PosList*** : 이동할 목표 절대좌표 값(각 축의 절대좌표 값)의 배열 주소. 이 배열의 크기는 cmxPmIxMapAxes() 함수를 통하여 맵핑된 축의 수와 일치하여야 합니다. 거리의 단위는 “Unit distance”에 의해 정의되는 논리적 거리를 적용합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 ‘에러처리’ 편을 참고합니다
0 (ERR_NONE)	수행 성공

SEE ALSO

cmxPmIxLineStart

REFERENCE

☐ 논리적 단위 거리는 cmxPmCfgSetUnitDist() 함수에 의해 결정됩니다.

☐ cmxPmIxLineToStart() 함수를 사용하는 경우에는 cmxPmIxIsDone() 함수나 cmxPmIxWaitDone() 함수를 사용하여 모션의 완료를 확인할 수 있습니다.

윈도우 이벤트라는 것은 무엇입니까?
윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용 프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

/* 원격 노드(Node ID : 1)의 0, 1 번 축에 대하여 마스터 속도 모드로 절대 좌표 직선 보간 이송을 수행하는 예제입니다. */

#define MAP0 0
#define NODE_ID 1
long BoardID = 0;

cmxPmIxMapAxes(BoardID, 3, MAP0, ccmxX1_MASK | ccmxY1_MASK );

/*보간 속도 패턴 설정에서 벡터 모드를 사용하지 않을 경우에는
각 축에 대한 속도를 미리 설정해 주어야 합니다.*/
cmxPmCfgSetSpeedPattern(BoardID, 3, ccmxX1, cmxMODE_T, 4000, 20000, 20000 );
cmxPmCfgSetSpeedPattern(BoardID, 3, ccmxY1, cmxMODE_T, 5000, 20000, 20000 );

/*마스터 속도 모드(벡터 모드를 사용하지 않음)로 사용할 경우에는 속도를
미리 설정한 속도의 비율로 설정해 줍니다.*/
cmxPmIxSetSpeedPattern(BoardID, 3, MAP0, cmxFALSE, cmxMODE_T, 100, 80, 80 );

//두 축에 대한 이동 거리를 설정합니다.
double fPosList[2] = { 13000, 90000 };

//보간 구동을 시작합니다.
cmxPmIxLineToStart(BoardID, 3, MAP0, fDistList );
cmxPmIxWaitDone(BoardID, 3, MAP0, cmxFALSE );
```

NAME

cmxPmIxArcAToStart

- 원호 보간 절대 좌표 이송

(절대적 중심 좌표와 각도)

INFORMATION

Interpolation Motion

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 3

이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인합니다

SYNOPSIS

❒ VT_I4 cmxPmIxArcAToStart

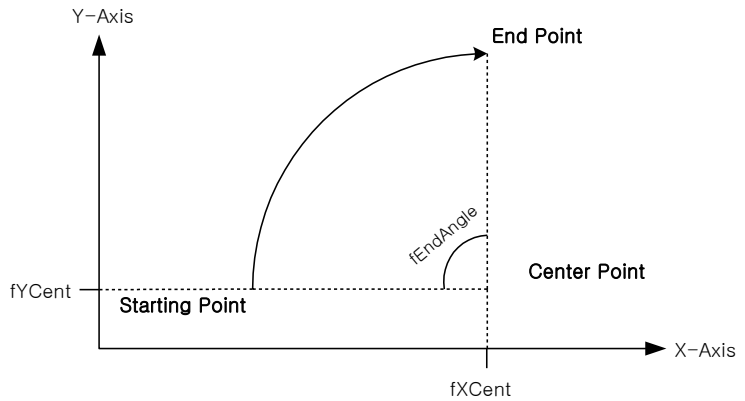
([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex, [in] VT_R8 XCent, [in] VT_R8 YCent, [in] VT_R8 EndAngle)

DESCRIPTION

cmxPmIxArcAToStart() 함수는 중심좌표와 원호의 각도를 매개 변수로 하여 원호보간 이동을 수행합니다. 이때 중심좌표는 절대좌표로 표현됩니다. cmxPmIxArcAToStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.

원호보간은 임의의 두 축에 대해서 적용됩니다. 아래 설명에서는 맵핑된 두 축을 X, Y 축으로 간주하여 설명합니다. 여기서 X 축이라 함은 맵핑된 두 축 중에서 축 번호가 낮은 축을 의미하며 Y 축은 축 번호가 높은 축을 의미합니다. 예를 들어 Z 축과 U 축이 맵핑된 두 축이라면 Z 축이 X 축에 해당하며 U 축이 Y 축에 해당합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.



PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***MapIndex*** : 맵번호(Map index), 이 맵번호를 사용하기 전에 먼저 cmxPmIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑 되어 있어야 합니다.

▶ ***XCent*** : 중심점의 X 축 절대좌표.

▶ ***YCent*** : 중심점의 Y 축 절대좌표.

▶ ***EndAngle*** : 원호보간 이동을 완료할 목표 지점의 현재 위치에 대한 각도 값을 Degree(°)값으로 지정합니다. 각도의 부호가 (+)이면 반 시계방향, (-)이면 시계방향으로의 이동을 의미합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

SEE ALSO

cmxPmIxArcAStart

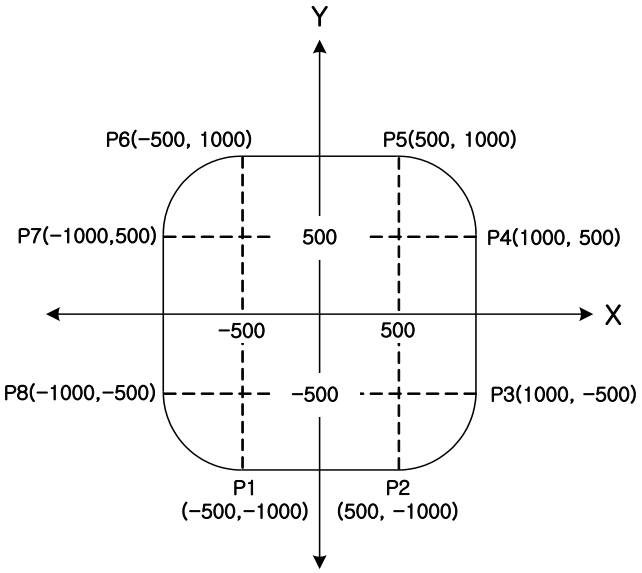
REFERENCE

☐ cmxPmIxArcAToStart() 함수를 사용하는 경우에는 cmxPmIxIsDone() 함수나 cmxPmIxWaitDone() 함수를 사용하여 모션의 완료를 확인할 수 있습니다.

윈도우 이벤트라는 것은 무엇입니까?
윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용 프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

EXAMPLE

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.



C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

#define MAP0 0
#define NODE_ID 1

long BoardID = 0;
//0 번 축과 1 번 축을 MAP0 로 설정합니다.
cmxPmIxMapAxes(BoardID, 3, MAP0, ccmxX1_MASK | ccmxY1_MASK );
//각 축의 속도를 설정합니다.
cmxPmCfgSetSpeedPattern(BoardID, 3, nAxis0, cmxMODE_T, 1000, 5000, 5000 );
cmxPmCfgSetSpeedPattern(BoardID, 3, nAxis1, cmxMODE_T, 1000, 5000, 5000 );

//맵에 포함된 축의 속도 비율을 설정합니다.
cmxPmIxSetSpeedPattern(BoardID, 3, MAP0, cmxFALSE, cmxMODE_T, 100, 70, 70 );

double fPosList[2] = {0.0f, 0.0f};

//P1 에서 P2 로 이동합니다.
fPosList[0] = 500; fPosList[1] = -1000;
cmxPmIxLineToStart(BoardID, 3, MAP0, fPosList );

//P2 에서 P3 로 이동합니다.
cmxPmIxArcAToStart(BoardID, 3, MAP1, 500, -500, 90 );

//P3 에서 P4 로 이동합니다.
fPosList[0] = 1000; fPosList[1] = 500;
cmxPmIxLineToStart(BoardID, 3, MAP0, fPosList );

//P4 에서 P5 로 이동합니다.
cmxPmIxArcAToStart(BoardID, 3, MAP1, 500, 500, 90);
```

```
//P5 에서 P6 로 이동합니다.
fPosList[0] = -500; fPosList[1] = 1000;
cmxPmIxLineToStart(BoardID,  MAP0, fPosList);

//P6 에서 P7 로 이동합니다.
cmxPmIxArcAToStart(BoardID,  MAP1, -500, 500, 90);

//P7 에서 P8 로 이동합니다.
fPosList[0] = -1000; fPosList[1] = -500;
cmxPmIxLineToStart(BoardID,  MAP0, fPosList);

//P8 에서 P1 로 이동합니다.
cmxPmIxArcAToStart(BoardID, MAP1, -500, -500, 90);
```

NAME

cmxPmIxArcPStart

- 원호 보간 상대 좌표 이송

(상대적 중심 좌표와 종점 좌표)

INFORMATION

Interpolation Motion

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 3

이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인합니다.

SYNOPSIS

❒ VT_I4 cmxPmIxArcPStart

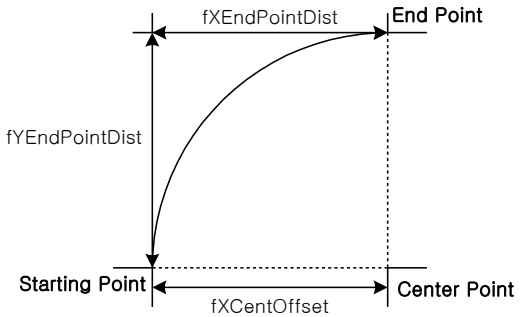
([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex, [in] VT_R8 XCentOffset, [in] VT_R8 YCentOffset, [in] VT_R8 XEndPointDist, [in] VT_R8 YEndPointDist, [in] VT_I4 Direction)

DESCRIPTION

cmxPmIxArcPStart() 함수는 중심좌표와 종점좌표를 매개 변수로 하여 원호보간 이송을 수행합니다. 이때 각 좌표는 상대좌표로 표현됩니다. cmxPmIxArcPStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.

원호보간은 임의의 두 축에 대해서 적용됩니다. 아래 설명에서는 맵핑된 두 축을 X, Y 축으로 간주하여 설명합니다. 여기서 X 축이라 함은 맵핑된 두 축 중에서 축 번호가 낮은 축을 의미하며 Y 축은 축 번호가 높은 축을 의미합니다. 예를 들어 Z 축과 U 축이 맵핑된 두 축이라면 Z 축이 X 축에 해당하며 U 축이 Y 축에 해당합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.



PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId***: 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***MapIndex*** : 맵번호(Map index), 이 맵번호를 사용하기 전에 먼저 cmxPmIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑 되어 있어야 합니다.

▶ ***XCentOffset***: 현재 위치(시작 위치)로부터 원의 중심까지 X 축상의 거리를 설정합니다.

▶ ***YCentOffset***: 현재 위치(시작 위치)로부터 원의 중심까지 Y 축상의 거리를 설정합니다.

▶ ***XEndPointDist***: 원호보간 이송을 완료할 목표 지점의 현재 위치로부터 X 축상의 거리를 설정합니다.

▶ ***YEndPointDist***: 원호보간 이송을 완료할 목표 지점의 현재 위치로부터 Y 축상의 거리를 설정합니다.

▶ ***Direction***: 회전 방향을 지정합니다.

Value	Meaning
0 (ccmxARC_CW)	시계 방향(CW)으로 회전
1 (ccmxARC_CCW)	반 시계 방향(CCW)으로 회전

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

SEE ALSO

cmxPmIxArcPToStart

REFERENCE

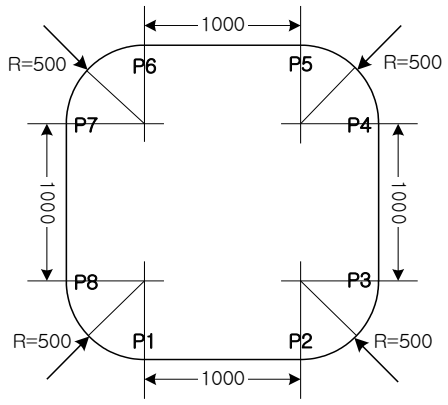
☐ cmxPmIxArcPStart() 함수를 사용하는 경우에는 cmxPmIxIsDone() 함수나 cmxPmIxWaitDone() 함수를 사용하여 모션의 완료를 확인할 수 있습니다.

윈도우 이벤트라는 것은 무엇입니까?

윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용 프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

EXAMPLE

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



C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

#define MAP0 0
#define NODE_ID 1

long BoardID = 0;

//0 번 축과 1 번 축을 MAP0 로 설정합니다.
cmxPmIxMapAxes(BoardID, 3, MAP0, ccmxX1_MASK | ccmxY1_MASK );
//각 축의 속도를 설정합니다.
cmxPmCfgSetSpeedPattern(BoardID, 3, ccmxX1, cmxMODE_T, 1000, 5000, 5000 );
cmxPmCfgSetSpeedPattern(BoardID, 3, ccmxY1, cmxMODE_T, 1000, 5000, 5000 );

//맵에 포함된 축의 속도 비율을 설정합니다.
cmxPmIxSetSpeedPattern(BoardID, 3, MAP0, cmxFALSE, cmxMODE_T, 100, 70, 70 );

double fDistList[2] = {0.0f, 0.0f};

//P1 에서 P2 로 이송합니다.
fDistList[0] = 1000; fDistList[1] = 0;
cmxPmIxLineStart(BoardID, 3, MAP0, fDistList);

//P2 에서 P3 로 이송합니다.
cmxPmIxArcPStart(BoardID, 3, MAP1, 0, 500, 500, 500, ccmxARC_CCW);

//P3 에서 P4 로 이송합니다.
fDistList[0] = 0; fDistList[1] = 1000;
cmxPmIxLineStart(BoardID, 3, MAP0, fDistList);

//P4 에서 P5 로 이송합니다.
cmxPmIxArcPStart(BoardID, 3, MAP1, -500, 0, -500, 500, ccmxARC_CCW);
```

```
//P5 에서 P6 로 이송합니다.
fDistList[0] = -1000; fDistList[1] = 0;
cmxPmIxLineStart(BoardID, 3, MAP0, fDistList);

//P6 에서 P7 로 이송합니다.
cmxPmIxArcPStrat(BoardID, 3, MAP1, 0, -500, -500, -500, ccmxARC_CCW);

//P7 에서 P8 로 이송합니다.
fDistList[0] = 0; fDistList[1] = -1000;
cmxPmIxLineStart(BoardID, 3, MAP0, fDistList);

//P8 에서 P1 로 이송합니다.
cmxPmIxArcPStart(BoardID, 3, MAP1, 500, 0, 500, -500, ccmxARC_CCW);
```

NAME

cmxPmIxArcPToStart

- 원호 보간 절대 좌표 이송

(절대적 중심 좌표와 종점 좌표)

INFORMATION

Interpolation Motion

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 3

이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인합니다.

SYNOPSIS

❒ VT_I4 cmxPmIxArcPToStart

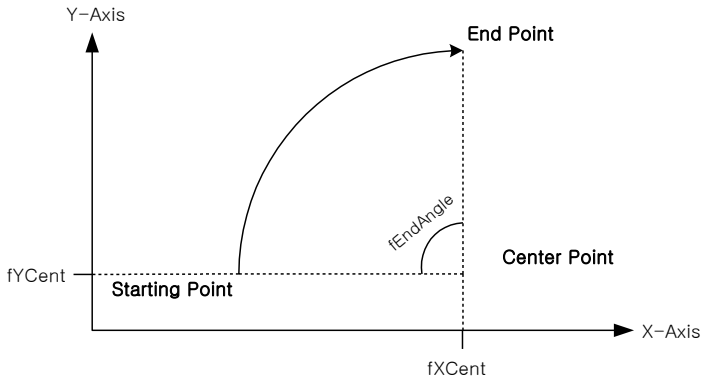
([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex, [in] VT_R8 XCent, [in] VT_R8 YCent, [in] VT_R8 XEndPos, [in] VT_R8 YEndPos, [in] VT_I4 Direction)

DESCRIPTION

cmxPmIxArcPToStart() 함수는 중심좌표와 종점좌표를 매개 변수로 하여 원호보간 이송을 수행합니다. 이때 각 좌표는 절대좌표로 표현됩니다. cmxPmIxArcPToStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.

원호보간은 임의의 두 축에 대해서 적용됩니다. 아래 설명에서는 맵핑된 두 축을 X, Y 축으로 간주하여 설명합니다. 여기서 X 축이라 함은 맵핑된 두 축 중에서 축 번호가 낮은 축을 의미하며 Y 축은 축 번호가 높은 축을 의미합니다. 예를 들어 Z 축과 U 축이 맵핑된 두 축이라면 Z 축이 X 축에 해당하며 U 축이 Y 축에 해당합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.



PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***MapIndex*** : 맵번호(Map index), 이 맵번호를 사용하기 전에 먼저 cmxPmIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑 되어 있어야 합니다.

▶ ***XCent*** : 중심점의 X 축 절대좌표 값을 설정합니다.

▶ ***YCent*** : 중심점의 Y 축 절대좌표 값을 설정합니다.

▶ ***XEndPos*** : 원호보간 이송을 완료할 목표 지점(End point)의 X 축 절대좌표 값을 설정합니다.

▶ ***YEndPos*** : 원호보간 이송을 완료할 목표 지점(End point)의 Y 축 절대좌표 값을 설정합니다.

▶ ***Direction*** : 회전 방향을 지정합니다.

Value	Meaning
0 (ccmxARC_CW)	시계 방향(CW)으로 회전
1 (ccmxARC_CCW)	반 시계 방향(CCW)으로 회전

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

SEE ALSO

cmxPmIxArcPStart

REFERENCE

☐ cmxPmIxArcPToStart() 함수를 사용하는 경우에는 cmxPmIxIsDone() 함수나 cmxPmIxWaitDone() 함수를 사용하여 모션의 완료를 확인할 수 있습니다.

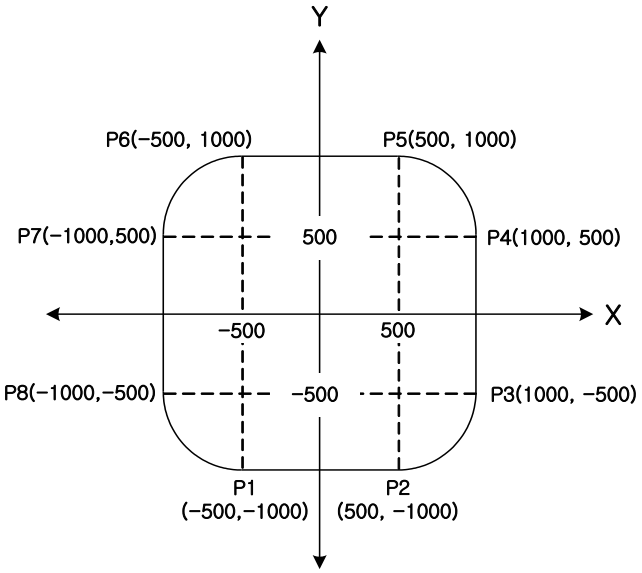
고객 여러분들께서는 스텝 드라이브 사용시에 이점을 주의해 주시기를 부탁 드립니다.

윈도우 이벤트라는 것은 무엇입니까?

윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용 프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

EXAMPLE

본 예제는 아래 그림과 같이 직선보간 이송과 원호보간 이송을 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.



C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

#define MAP0 0
#define NODE_ID 1

long BoardID = 0;
//0 번 축과 1 번 축을 MAP0 로 설정합니다.
cmxPmIxMapAxes(BoardID, 3, MAP0, ccmxX1_MASK | ccmxY1_MASK );
//각 축의 속도를 설정합니다.
cmxPmCfgSetSpeedPattern(BoardID, 3, ccmxX1, cmxMODE_T, 1000, 5000, 5000 );
cmxPmCfgSetSpeedPattern(BoardID, 3, ccmxY1, cmxMODE_T, 1000, 5000, 5000 );

//맵에 포함된 축의 속도 비율을 설정합니다.
cmxPmIxSetSpeedPattern(BoardID, 3, MAP0, cmxFALSE, cmxMODE_T, 100, 70, 70 );

double fPosList[2] = {0.0f, 0.0f};

//P1 에서 P2 로 이동합니다.
fPosList[0] = 500; fPosList[1] = -1000;
cmxPmIxLineToStart(BoardID, 3, MAP0, fPosList );

//P2 에서 P3 로 이송합니다.
cmxPmIxArcPToStart(BoardID, 3, MAP1, 500, -500, 1000, -500, ccmxARC_CCW );

//P3 에서 P4 로 이송합니다.
fPosList[0] = 1000; fPosList[1] = 500;
cmxPmIxLineToStart(BoardID, 3, MAP0, fPosList);
```

```
//P4 에서 P5 로 이송합니다.
cmxPmIxArcPToStart(BoardID, 3, MAP1, 500, 500, 500, 1000, ccmxARC_CCW);

//P5 에서 P6 로 이송합니다.
fPosList[0] = -500; fPosList[1] = 1000;
cmxPmIxLineToStart(BoardID, 3, MAP0, fPosList );

//P6 에서 P7 로 이송합니다.
cmxPmIxArcPToStart(BoardID, 3, MAP1, -500, 500, -1000, 500, ccmxARC_CCW);

//P7 에서 P8 로 이송합니다.
fPosList[0] = -1000; fPosList[1] = -500;
cmxPmIxLineToStart (BoardID, 3, MAP0, fPosList);

//P8 에서 P1 로 이송합니다.
cmxPmIxArcPToStart (BoardID, 3, MAP1, -500, -500, -500, -1000, ccmxARC_CCW);
```

NAME

cmxPmIxArc3PStart

- 3 점(Point) 원호 보간 상대 좌표 이송

(현재 좌표와 상대적 두 좌표)

INFORMATION

Interpolation Motion

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 3

이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인합니다.

SYNOPSIS

❒ VT_I4 cmxPmIxArc3PStart

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex, [in] VT_R8 P2X, [in] VT_R8 P2Y, [in] VT_R8 P3X, [in] VT_R8 P3Y, [in] VT_R8 EndAngle)

DESCRIPTION

cmxPmIxArc3PStart() 함수는 현재좌표와 상대적 두 좌표를 매개변수로 하여 원호보간 이송을 수행합니다. 이때 각 좌표는 상대좌표로 표현됩니다. cmxPmIxArc3PStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.

원호보간은 임의의 두 축에 대해서 적용됩니다. 아래 설명에서는 맵핑된 두 축을 X, Y 축으로 간주하여 설명합니다. 여기서 X 축이라 함은 맵핑된 두 축 중에서 축 번호가 낮은 축을 의미하며 Y 축은 축 번호가 높은 축을 의미합니다. 예를 들어 Z 축과 U 축이 맵핑된 두 축이라면 Z 축이 X 축에 해당하며 U 축이 Y 축에 해당합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***MapIndex*** : 맵번호(Map index), 이 맵번호를 사용하기 전에 먼저 cmxPmIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑 되어 있어야 합니다.

▶ ***P2X*** : 현재 위치(시작 위치)로부터 두 번째 점까지 X 축상의 거리를 설정합니다.

▶ ***P2Y*** : 현재 위치(시작 위치)로부터 두 번째 점까지 Y 축상의 거리를 설정합니다.

▶ ***P3X*** : 두 번째 점부터 세 번째 점까지 X 축상의 거리를 설정합니다.

▶ ***P3Y***: 두 번째 점부터 세 번째 점까지 Y 축상의 거리를 설정합니다.

▶ ***EndAngle*** : 원호보간 이송을 완료할 목표지점의 현재 위치에 대한 각도 값을 Degree(°)값으로 지정합니다. 각도의 부호가 (+)이면 반 시계방향, (-)이면 시계방향으로의 이동을 의미합니다. 단, EndAngle 값을 '0'으로 설정하시면 세번째 점까지 원호보간 이송을 수행합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

SEE ALSO

cmxPmIxArc3PToStart

REFERENCE

☐ cmxPmIxArc3PStart() 함수를 사용하는 경우에는 cmxPmIxIsDone() 함수나 cmxPmIxWaitDone() 함수를 사용하여 모션의 완료를 확인할 수 있습니다.

윈도우 이벤트라는 것은 무엇입니까?

윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용 프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

#define MAP0 0
#define NODE_ID 1

long BoardID = 0;

//0 번 축과 1 번 축을 MAP0 로 설정합니다.
cmxPmIxMapAxes(BoardID, 3, MAP0, ccmxX1_MASK | ccmxY1_MASK );
//각 축의 속도를 설정합니다.
cmxPmCfgSetSpeedPattern(BoardID, 3, ccmxX1, cmxMODE_T, 1000, 5000, 5000 );
cmxPmCfgSetSpeedPattern(BoardID, 3, ccmxY1, cmxMODE_T, 1000, 5000, 5000 );

//맵에 포함된 축의 속도 비율을 설정합니다.
cmxPmIxSetSpeedPattern(BoardID, 3, MAP0, cmxFALSE, cmxMODE_T, 100, 70, 70 );

double fX2 = 1000.0f, fY2 = 3000.0f;
double fX3 = 3000.0f, fY3 = 2000.0f;

/*현재 위치, ( 현재 위치 + fX2, 현재 위치 + fY2 ),
( 현재 위치 + fX3, 현재 위치 + fY3 ) 세점을
지나는 원을 그립니다.*/
cmxPmIxArc3PStart(BoardID, 3, MAP0, fX2, fY2, fX3, fY3, 360 );
```

```
cmxPmIxWaitDone(BoardID, 3, MAP0, cmxFALSE );
```

NAME

cmxPmIxArc3PToStart

- 3 점(Point) 원호 보간 절대 좌표 이송

(현재 좌표와 절대적 두 좌표)

INFORMATION

Interpolation Motion

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 3

이송 함수

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인합니다.

SYNOPSIS

❒ VT_I4 cmxPmIxArc3PToStart

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex, [in] VT_R8 P2X, [in] VT_R8 P2Y, [in] VT_R8 P3X, [in] VT_R8 P3Y, [in] VT_R8 EndAngle)

DESCRIPTION

cmxPmIxArc3PToStart() 함수는 현재좌표와 절대적 두 좌표를 매개변수로 하여 원호보간 이동을 수행합니다. 이때 각 좌표는 절대좌표로 표현됩니다. cmxPmIxArc3PToStart() 함수는 모션을 시작시킨 후에 바로 반환됩니다.

원호보간은 임의의 두 축에 대해서 적용됩니다. 아래 설명에서는 맵핑된 두 축을 X, Y 축으로 간주하여 설명합니다. 여기서 X 축이라 함은 맵핑된 두 축 중에서 축 번호가 낮은 축을 의미하며 Y 축은 축 번호가 높은 축을 의미합니다. 예를 들어 Z 축과 U 축이 맵핑된 두 축이라면 Z 축이 X 축에 해당하며 U 축이 Y 축에 해당합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***MapIndex*** : 맵번호(Map index), 이 맵번호를 사용하기 전에 먼저 cmxPmIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑 되어 있어야 합니다.

▶ ***P2X***: 두 번째 점의 X 축 절대좌표 값을 설정합니다.

▶ ***P2Y***: 두 번째 점의 Y 축 절대좌표 값을 설정합니다.

▶ ***P3X***: 세 번째 점의 X 축 절대좌표 값을 설정합니다.

▶ ***P3Y***: 세 번째 점의 Y 축 절대좌표 값을 설정합니다.

▶ ***EndAngle*** : 원호보간 이동을 완료할 목표지점의 현재 위치에 대한 각도 값을 Degree(°)값으로 지정합니다. 각도의 부호가 (+)이면 반 시계방향, (-)이면 시계방향으로의 이동을 의미합니다. 단, EndAngle 값을 '0'으로 설정하시면 세 번째 점까지 원호보간 이송을 수행합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

SEE ALSO

cmxPmIxArc3PStart

REFERENCE

□ cmxPmIxArc3PToStart() 함수를 사용하는 경우에는 cmxPmIxIsDone() 함수나 cmxPmIxWaitDone() 함수를 사용하여 모션의 완료를 확인할 수 있습니다.

윈도우 이벤트라는 것은 무엇입니까?

윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용 프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

#define MAP0 0
#define NODE_ID 1

long BoardID = 0;

//0 번 축과 1 번 축을 MAP0 로 설정합니다.
cmxPmIxMapAxes(BoardID, 3, MAP0, ccmxX1_MASK | ccmxY1_MASK );
//각 축의 속도를 설정합니다.
cmxPmCfgSetSpeedPattern(BoardID, 3, ccmxX1, cmxMODE_T, 1000, 5000, 5000 );
cmxPmCfgSetSpeedPattern(BoardID, 3, ccmxY1, cmxMODE_T, 1000, 5000, 5000 );

//맵에 포함된 축의 속도 비율을 설정합니다.
cmxPmIxSetSpeedPattern(BoardID, 3, MAP0, cmxFALSE, cmxMODE_T, 100, 70, 70 );

double fX2 = 0.0f, fY2 = 3000.0f;
double fX3 = 3000.0f, fY3 = 0.0f;

//현재 위치, (fX2, fY2), (fX3, fY3) 세점을 지나는 원을 그립니다.
cmxPmIxArc3PToStart(BoardID, 3, MAP0, fX2, fY2, fX3, fY3, 360 );
```

```
cmxPmIxWaitDone(BoardID, 3, MAP0, cmxFALSE );
```

NAME

cmxPmIxStop / cmxPmIxStopEmg

- 보간 이송 정지

- 보간 비상 정지

INFORMATION

Interpolation Motion

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 3

정지 함수

고속 이송시에 급 정지, 비상 정지를 주의하십시오. 기구물의 손상이나 안전사고에 원인이 될 수 있습니다.

SYNOPSIS

- VT_I4 cmxPmIxStop ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex)
- VT_I4 cmxPmIxStopEmg ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex)

DESCRIPTION

cmxPmIxStop()/cmxPmIxStopEmg() 함수는 지정한 보간 맵에 대한 보간작업을 정지합니다. 정지 시에 cmxPmIxStop() 함수를 사용하면 감속 후 정지하며, cmxPmIxStopEmg()를 사용하면 감속 없이 즉시정지를 수행합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***MapIndex*** : 맵 번호(Map index), 이 맵 번호를 사용하기 전에 먼저 cmxPmIxMapAxes() 함수를 통하여 해당 맵 번호에 유효한 축들이 맵핑 되어 있어야 합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

#define MAP0 0
#define NODE_ID 1

long BoardID =0;

cmxPmIxMapAxes(BoardID, 3, MAP0, ccmxX1_MASK | ccmxY1_MASK );

/*MAP0 로 설정된 축들을 벡터 스피드로
가속도 10000, 감속도 10000, 등속도 2000 으로 설정합니다.*/
cmxPmIxSetSpeedPattern(BoardID, 3, MAP0, cmxTRUE, cmxMODE_T, 2000, 10000, 10000 );

double fDistList[2] = { 10000.0f, 20000.0f };

//MAP0 를 직선 보간으로 구동합니다.
cmxPmIxLineStart(BoardID, 3, MAP0 , fDistList );
Sleep(3000);

//MAP0 에 속한 축들을 감속 후 정지 시킵니다.
cmxPmIxStop(BoardID, 3, MAP0 );

//cmxPmIxStopEmg() 함수를 사용할 경우 감속 없이 즉시 정지합니다.
//cmxPmIxStopEmg(BoardID, 3, MAP0 );
```

NAME

cmxPmIxIsDone

- 보간 모션 완료 확인

INFORMATION

- Interpolation Motion
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 3
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxPmIxIsDone ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex, [out] VT_PI4 IsDone)

DESCRIPTION

cmxPmIxIsDone() 함수는 지정한 보간 맵에 해당하는 보간 작업이 완료됐는지를 확인하는 함수입니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***MapIndex*** : 맵번호(Map index), 이 맵번호를 사용하기 전에 먼저 cmxPmIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑 되어 있어야 합니다.

▶ ***IsDone*** : 이 매개를 통해 모션 작업이 완료되었는지를 판단할 수 있습니다.

Value	Meaning
0 (cmxFALSE)	모션작업이 완료되지 않음
1 (cmxTRUE)	모션작업이 완료됨

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

SEE ALSO

cmxPmIxWaitDone

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

#define MAP0 0
#define NODE_ID 1

long BoardID = 0;

cmxPmIxMapAxes(BoardID, 3, MAP0, ccmxX1_MASK | ccmxY1_MASK );

/*MAP0 로 설정된 축들을 벡터 스피드로
가속도 10000, 감속도 10000, 등속도 2000 으로 설정합니다.*/
cmxPmIxSetSpeedPattern(BoardID, 3, MAP0, cmxTRUE, cmxMODE_T, 2000, 10000, 10000 );

double fDistList[2] = { 10000.0f, 20000.0f };

//MAP0 를 직선 보간으로 구동합니다.
cmxPmIxLineStart(BoardID, 3, MAP0 , fDistList );

long nIsDone = cmxFALSE;

while( !nIsDone )
{
    //MAP0 에 혹한 축들의 모션 완료 여부를 판단합니다.
    cmxPmIxIsDone(BoardID, 3, MAP0, &nIsDone );
}
```

NAME

cmxPmIxWaitDone

- 보간 모션 완료 대기

INFORMATION

- Interpolation Motion
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 3
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxPmIxWaitDone ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 MapIndex)

DESCRIPTION

cmxPmIxWaitDone() 함수는 지정한 보간 맵에 해당하는 보간 작업이 완료될 때까지 기다립니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId***: 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***MapIndex*** : 맵번호(Map index), 이 맵번호를 사용하기 전에 먼저 cmxPmIxMapAxes() 함수를 통하여 해당 맵번호에 유효한 축들이 맵핑 되어 있어야 합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

SEE ALSO

cmxPmIxIsDon

REFERENCE

□ INP 입력신호가 Enable 로 설정되었으면 Command 펄스 출력이 완료되어도 INP 입력이 ON 이 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환되지 않습니다.

□ 스텝 드라이브를 사용 중인 고객님들께서는 다음을 참조해 주십시오.

스텝 드라이브는 INP 출력이 없는 경우가 일반적인데, 고객님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스텝 드라이브를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다. 고객 여러분들께서는 스텝 드라이브 사용시에 이점을 주의해 주시기를 부탁 드립니다.

□ 서보 드라이브의 LSP, LSN 신호를 사용 중인 고객님께서는 다음을 참조해 주십시오.
서보드라이브의 입력 신호 중 하나인 EL(End of Limit) 신호는 저희 ㈜ 커미조아 모션 컨트롤러뿐만 아니라 서보드라이브에도 전달 될 수 있도록 설정할 수 있습니다. 통상적으로 LSP 신호와 LSN 신호로 불리어 지는 이 신호는 실제 기구물에서 양의 방향(Positive Direction) 혹은 음의 방향(Negative Direction) 에 장착되어 있는 EL(End of Limit) 신호를 서보 드라이브 측에 전달하기 위한 용도로 사용됩니다.

그러나, 모션 소프트웨어에서 INP 설정이 되어 있는 경우 EL 신호가 검출 된 후에 일부 서보 드라이브에서는 진행 방향에서 정지 한 후 더 이상 움직이지 않는 상황이 발생하며, 이 상황에서 INP 신호가 출력되지 않아, 모션 이송이 완료되지 못하고, 명시적으로 STOP 명령을 통해 강제적으로 모션 종료가 되는 현상이 발생할 수 있습니다.. 이 현상은 EL 모드를 통해 원점 복귀를 하는 상황에서도 발생할 수 있습니다.

따라서, 이러한 경우에는 반드시 원점 복귀나 EL 검출 시에 인터럽트 이벤트나 타이머를 통해 INP 를 무기한 대기하는 현상에 대해서 적절히 대처하시거나 INP 신호 사용을 배제 해야 합니다.

윈도우 이벤트라는 것은 무엇입니까?

윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용 프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 행선지가 되는 윈도우에 전송되어 처리됩니다.

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

#define MAP0 0
#define NODE_ID 1

long BoardID = 0;

cmxPmIxMapAxes( 0. MAP0, ccmxX1_MASK | ccmxY1_MASK );

/*MAP0 로 설정된 축들을 벡터 스피드로
가속도 10000, 감속도 10000, 등속도 2000 으로 설정합니다.*/
cmxPmIxSetSpeedPattern( 0. MAP0, cmxTRUE, cmxMODE_T, 2000, 10000, 10000 );

double fDistList[2] = { 10000.0f, 20000.0f };

//MAP0 를 직선 보간으로 구동합니다.
if( cmxPmIxLineStart(BoardID, 3, MAP0 , fDistList ) != ERR_NONE )
{
    OutputDebugString( “IxMoveStart fail!” );
    return;
}

//모션이 완료 시까지 기다립니다.
if( cmxPmIxWaitDone(BoardID, 3, MAP0, cmxFALSE ) != ERR_NONE )
{
    OutputDebugString( "IxWaitDone fail!" );
    return;
}
```

15.4 원점복귀(Home Return)

이 단원에서는 원점 복귀(Home Return)에 관련된 함수들을 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(Zero)로 초기화됩니다.

원점 복귀 작업을 수행하기 위해서는 ORG(HOME), EZ 및 EL 신호가 참조되는데 이 신호들의 의미 및 작용은 다음과 같습니다.

□ ORG (HOME) 신호

ORG 신호는 구조물이 원점에 복귀했는지를 센서로부터 입력 받는 신호입니다. 일반적으로는 근접 센서와 같은 센서들을 이용하여 원점 복귀 여부를 감지하게 됩니다. 이 신호는 'HOME'단자를 통하여 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 'EZ+' 단자와 'EZ-' 단자를 통하여 입력되어야 합니다.

□ EL 신호

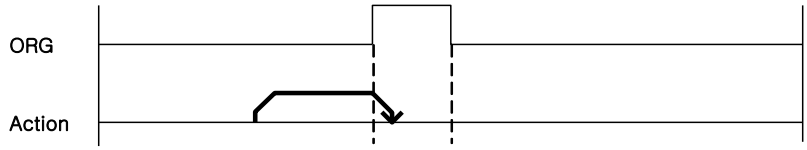
기계적인 리밋(Limit) 신호를 의미합니다. 이 신호는 일반적으로 구조물이 움직일 수 있는 한계점을 감지하기 위해 사용되나 원점 복귀 모드에 따라 ORG 신호의 대용으로도 사용될 수 있습니다. EL 신호는 (+)방향 리밋 신호와 (-)방향 리밋 두 가지 신호가 있습니다. (+)방향 리밋 신호는 '+EL' 단자, 그리고 (-)방향 리밋 신호는 '-EL' 단자를 통하여 입력되어야 합니다.

15.4.1 원점 복귀 모드 안내

㈜커미조아 모션컨트롤러는 다음과 같이 13 가지의 다양한 원점복귀 모드를 제공합니다. 각 원점복귀 모드에 따른 동작 방식은 아래 설명과 같습니다. 아래의 그림은 모두 속도모드를 Trapezoidal 모드로 설정한 상태임을 가정하여 그려 진 것이며, 만일 Constant 속도 모드로 설정된 경우에는 감속이 없이 즉시 정지하게 됩니다.

□ MODE 0 : ORG ON => Stop

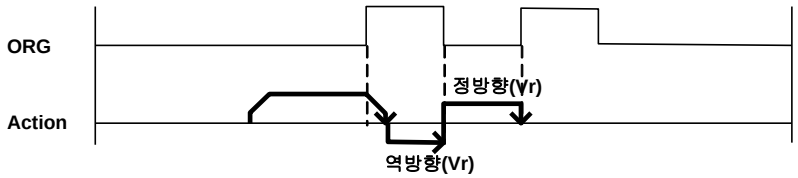
MODE 0 에서는 ORG 신호가 OFF 에서 ON 으로 바뀌는 순간에 모션을 감속 후 정지하고 복귀 작업을 종료합니다.



□ MODE 1 : ORG ON => Stop => Back (Vr) => ORG OFF => Forward(Vr) => ORG ON => Stop

MODE 1 에서는 ORG 신호가 OFF 에서 ON 으로 바뀌는 순간에 모션을 감속 후 정지한 후 ORG 신호가 OFF 가 될 때까지 Vr(Reverse Speed)의 속도로 역방향 회전을 수행합니다. ORG 신호가 OFF 되면 다시 Vr 의 속도로 정방향 회전을 수행하다가 ORG 신호가 다시 ON 되는 순간에 복귀작업을 종료합니다.

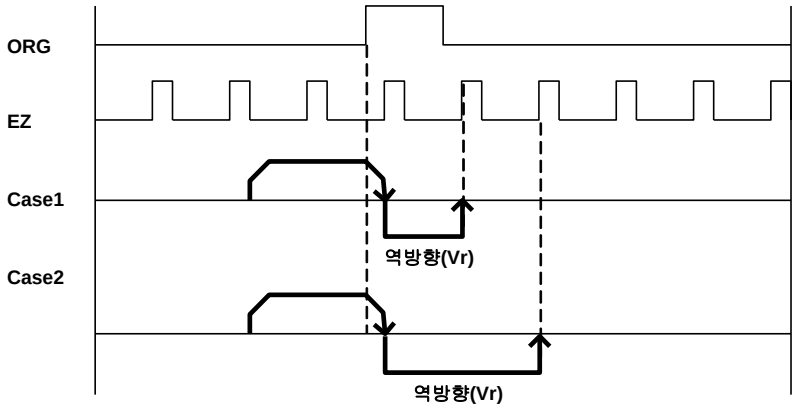
이 모드를 사용할 때 주의할 것은 처음 ORG 신호가 ON 으로 변경되어 감속 후 정지하는 도중에 ORG 센서의 ON 으로 유지되는 시간이 짧아서 이미 OFF 상태로 변경되면 역방향 이동을 생략하고 최종적으로 원점 센서를 찾으려는 단계로 넘어갑니다. 따라서 이러한 경우에는 (-)Limit 위치까지 이동하게 됩니다. 이러한 경우에도 자동으로 다시 탈출하여 정상적인 원점복귀 작업을 다시 수행하지만 의도하지 않게 원점복귀 시간이 길어질 수 있습니다. 이를 방지하는 방법은 구조적으로 ORG 신호가 ON 으로 유지되는 시간을 길게 해주거나 또는 원점복귀 시의 작업속도를 낮추거나 감속도를 크게 해주어 감속 시 이동되는 거리를 짧게 해주면 됩니다.



Vr : Reverse Speed를 의미합니다.

□ MODE 2 : ORG ON => Stop => Back (Vr) => Stop on EZ Count

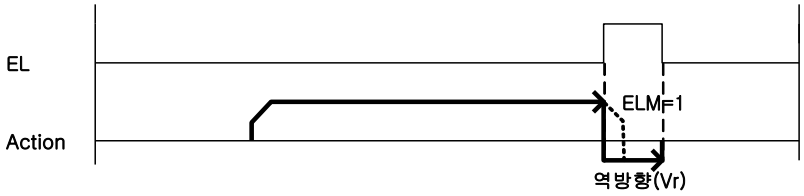
MODE 2 에서는 ORG 신호가 OFF 에서 ON 으로 바뀌는 순간 감속 후 정지합니다. 그리고 Vr 의 속도로 역방향 회전한 후 EZ 신호에 따라 복귀 작업을 종료합니다.



Case1 : EzCount = 1 인 경우
Case2 : EzCount = 2 인 경우
Vr : Reverse Speed

☐ MODE 3 : EL ON => Stop => Back (Vr) => EL OFF => Stop

MODE 3 에서는 EL 신호가 ON 으로 바뀌는 순간 즉시 정지(또는 ELM=1 인 경우에 감속 후 정지합니다. 그리고 반대 방향으로 Vr 속도로 회전하다가 EL 신호가 OFF 되는 순간에 복귀작업을 종료합니다. 여기서 ELM=1 은 EL 의 "Stop mode"가 "감속 후 정지" 모드로 설정됨을 의미합니다.

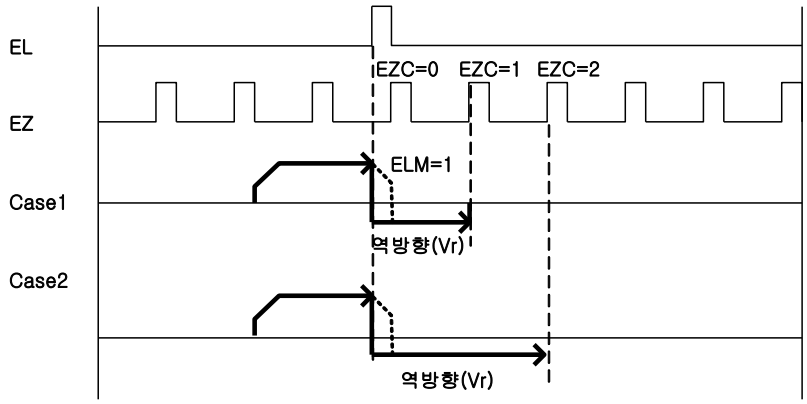


Vr : Reverse Speed

cmxPmCfgSetMioProperty() 함수를 통해, EL 신호가 ON 으로 변경되는 순간 모션을 즉시 정지를 할 것인지 감속 후 정지를 할 것인지를 설정할 수 있습니다. 자세한 설명은 cmxPmCfgSetMioProperty() 함수를 통해 확인해 주시기 바랍니다.

☐ MODE 4 : EL ON => Stop => Back (Vr) => Stop on EZ count

MODE 4 에서는 EL 신호가 ON 으로 바뀌는 순간 즉시 정지(또는 ELM=1 인 경우에 감속 후 정지합니다. 그리고 반대 방향으로 Vr 속도로 회전하다가 EzCount 에 따라 복귀작업을 종료합니다. 여기서 ELM=1 은 EL 의 "Stop mode"가 "감속 후 정지" 모드로 설정됨을 의미합니다.



Case1 : EzCount = 1 인 경우
Case2 : EzCount = 2 인 경우
Vr : Reverse Speed

안내

cmxPmCfgMioSetProperty() 함수를 통해, EL 신호가 ON 으로 변경되는 순간 모션을 즉시 정지를 할 것인지 감속 후 정지를 할 것인지를 설정할 수 있습니다. 자세한 설명은 cmxPmCfgMioSetProperty()함수를 통해 확인해 주시기 바랍니다.

15.4.2 자동 원점 검색 기능에 대하여

일반적으로 기구물의 위치는 원점 센서를 기준으로 (+) 방향의 위치에 있으며, 따라서 (-) 방향으로 원점복귀를 수행하면 됩니다. 하지만 원점복귀를 시작하는 시점에 기구물이 이미 원점 센서가 ON 이 되어 있는 위치에 있거나 원점 센서와 (-)EL 센서 사이에 위치한 경우에는 원점 센서를 기준으로 (+) 방향의 위치까지 탈출한 후에 정상적인 원점복귀 작업을 수행하여야 합니다. 이러한 기능을 "자동 원점 검색" 기능이라 하며, ㈜커미조아의 모션컨트롤러는 이 기능을 자동으로 수행 해주므로 기구물의 위치에 관계없이 원점복귀 작업을 수행할 수 있습니다.

기구물과 각 센서들의 위치에 따른 원점복귀 절차는 다음과 같이 약간씩 다릅니다. 단, 원점복귀 작업의 방향을 음의 방향으로 한 경우에 대한 것입니다. 만일 원점복귀를 양의 방향으로 한 경우에는 -EL 신호 대신 +EL 신호가 사용됩니다.

□ CASE 1. 원점 센서를 기준으로 양의 방향 위치에서 원점복귀를 시작한 경우

정상적인 원점복귀 작업을 수행합니다.

□ CASE 2. 원점 센서가 ON 인 상태에서 원점복귀를 시작한 경우

이러한 경우에는 먼저 원점 탈출 거리(Escape distance) 만큼 양의 방향으로 이동한 후 다시 정상적인 원점복귀 작업을 수행합니다. 원점 탈출 거리만큼 탈출하였어도 원점 센서가 ON 상태이면 원점 탈출 거리만큼 탈출하는 작업을 반복하므로 탈출 거리가 짧아도 원점을 탈출하는 데는 아무 문제가 없습니다.

□ CASE 3. 원점 센서를 기준으로 음의 방향 위치에서 원점복귀를 시작한 경우

이러한 경우에는 먼저 음의 방향으로 이동을 시작합니다. 그리고 -EL(Negative Limit) 센서가 ON 되면 정지 후 양의 방향으로 이동합니다. 원점 센서가 ON 되면 다시 정지 후 CASE 2 에서와 같이 원점 탈출 거리만큼 양의 방향으로 이동한 후 다시 정상적인 원점복귀 작업을 수행합니다.

[그림 15-3]는 원점복귀 모드를 0, 속도 패턴을 사다리꼴 방식으로 하고 원점복귀 작업을 수행할 때 위의 각 경우에 대하여 동작하는 방식을 도식적으로 나타낸 것입니다.

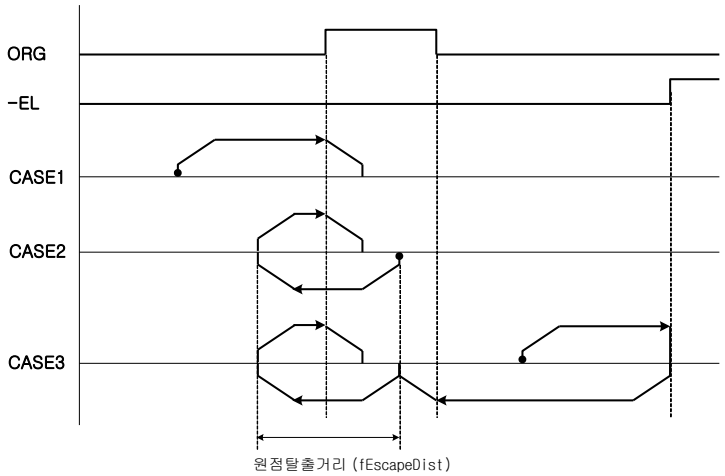


그림 15-3 기구물과 센서의 위치에 따른 원점복귀 작업

15.4.3 함수 요약

"원점복귀"와 관련된 함수들은 다음과 같습니다.

Summary of Functions
❒ VT_I4 cmxPmHomeSetConfig ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 HomeMode, [in] VT_I4 Dir, [in] VT_I4 EzCount, [in] VT_R8 EscDist, [in] VT_R8 Offset) 대상 모션 축에 대해서, 원점복귀에 대한 환경설정을 구성합니다. 원점복귀 모드와 Z 상 검출 횟수, 자동 원점 탈출 거리, 원점복귀 완료 후 추가 이송 거리등을 설정할 수 있습니다.
❒ VT_I4 cmxPmHomeGetConfig ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 HomeMode, [out] VT_PI4 Dir, [out] VT_PI4 EzCount, [out] VT_PR8 EscDist, [out] VT_PR8 Offset) 대상 모션 축에 대해서, 원점복귀에 대해 설정되어 있는 환경을 반환합니다. 원점복귀 모드와 Z 상 검출 횟수, 자동 원점 탈출 거리, 원점복귀 완료 후 추가 이송 거리등이 반환됩니다.
❒ VT_I4 cmxPmHomeSetPosClrMode ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 PosClrMode) 대상 모션 축에 대해서, 원점복귀 완료 후 발생하는 모션 컨트롤러와 서보 드라이브간의 제어 편차에 의해 발생한 입력 펄스(Feedback Pulse) 처리에 대한 환경을 구성합니다.
❒ VT_I4 cmxPmHomeGetPosClrMode ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 PosClrMode) 대상 모션 축에 대해서, 원점복귀 완료 후 발생하는 모션 컨트롤러와 서보 드라이브간의 제어 편차에 의해 발생한 입력 펄스(Feedback Pulse) 처리에 대한 설정 상태를 반환합니다.
❒ VT_I4 cmxPmHomeSetSpeedPattern ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 SpeedMode, [in] VT_R8 Vel, [in] VT_R8 Accel, [in] VT_R8 Decel, [in] VT_R8 RevVel) 대상 모션 축에 대해서, 원점복귀 속도를 설정합니다. 이 속도는 원점복귀 시에만 사용되는 속도이며, 일반 모션 속도와 개별적인 원점복귀 속도를 지정할 수 있습니다.
❒ VT_I4 cmxPmHomeGetSpeedPattern ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 SpeedMode, [out] VT_PR8 Vel, [out] VT_PR8 Accel, [out] VT_PR8 Decel, [out] VT_PR8 RevVel) 대상 모션 축에 대해서, 원점복귀 속도를 반환합니다. 이 속도는 원점복귀 시에만 사용되는 속도이며, 일반 모션 속도와 개별적인 원점복귀 속도를 반환합니다.
❒ VT_I4 cmxPmHomeSetSpeedPattern_T ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 SpeedMode, [in] VT_R8 Vel, [in] VT_R8 AccelTime, [in] VT_R8 DecelTime, [in] VT_R8 RevVel) 대상 모션 축에 대해서, 원점복귀 속도를 설정합니다. 이 속도는 원점복귀 시에만 사용되는 속도이며, 일반 모션 속도와 개별적인 원점복귀 속도를 지정할 수 있습니다.
❒ VT_I4 cmxPmHomeGetSpeedPattern_T ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 SpeedMode, [out] VT_PR8 Vel, [out] VT_PR8 AccelTime, [out] VT_PR8 DecelTime, [out] VT_PR8 RevVel) 대상 모션 축에 대해서, 원점복귀 속도를 반환합니다. 이 속도는 원점복귀 시에만 사용되는 속도이며, 일반 모션 속도와 개별적인 원점복귀 속도를 반환합니다.
❒ VT_I4 cmxPmHomeMoveStart ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel) 대상 모션 축에 대해서, 환경 설정을 바탕으로 원점복귀를 구동합니다. 이 함수는 원점 복귀 이송 시작 후 바로 반환됩니다.
❒ VT_I4 cmxPmHomeMoveAllStart ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 NumChannel , [in] VT_I4 ChannelMask) 모든 모션 축에 대해서, 환경 설정을 바탕으로 원점복귀를 구동합니다. 이 함수는 원점 복귀 이송 시작 후 바로 반환됩니다.
VT_I4 cmxPmHomeGetSuccess ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 IsSuccess) 대상 모션 축에 대해서, 이전에 실행된 원점복귀 완료 상태를 확인할 수 있습니다. 원점 복귀 완료 상태는 이전에 실행된 원점복귀 상태이며, 모션 운영 시스템의 하드웨어적인 전원이 차단되지 않는 다는 조건 하에서 영구히 보존됩니다.
VT_I4 cmxPmHomeSetSuccess ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 IsSuccess) 대상 모션 축에 대해서 , 이전에 실행된 원점복귀 구동 완료 상태를 설정할 수 있습니다. 원점 복귀 완료 상태는 이전에 실행된 원점복귀 상태이며, 모션 운영 시스템의 하드웨어적인 전원이 차단되지 않는 다는 조건 하에서는 영구히 보존되며, 이 함수를 통해 이전의 원점복귀 완료 상태를 설정할 수 있습니다.

15.4.4 함수 설명

NAME

cmxPmHomeSetConfig /

cmxPmHomeGetConfig

- 원점 복귀 환경 설정 및 설정 값 확인

INFORMATION

Home Return

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 4

다소 주의

원점 복귀 함수의 전체 환경 설정을 하는 함수입니다. 매개변수 및 관련 내용을 반드시 숙지합니다.

SYNOPSIS

❒ VT_I4 cmxPmHomeSetConfig

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 HomeMode, [in] VT_I4 Dir, [in] VT_I4 EzCount, [in] VT_R8 EscDist, [in] VT_R8 Offset)

❒ VT_I4 cmxPmHomeGetConfig

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 HomeMode, [out] VT_PI4 Dir, [out] VT_PI4 EzCount, [out] VT_PR8 EscDist, [out] VT_PR8 Offset)

DESCRIPTION

cmxPmHomeSetConfig()/cmxPmHomeGetConfig() 함수는 원점복귀에 관련된 여러 가지 환경을 설정 혹은 설정된 환경을 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***HomeMode*** : 원점복귀 모드 번호를 설정 혹은 반환합니다. 앞서 설명한 바와 같이 ㈜커미조아 모션 컨트롤러는 13 가지(0 ~ 12)의 다양한 원점복귀 모드를 제공합니다.

▶ ***Dir*** : 원점복귀 진행 방향을 설정 혹은 반환합니다.

Value	Meaning
0 (cmDIR_N)	(-) 방향 => Negative direction
1 (cmDIR_P)	(+) 방향 => Positive direction

▶ ***EzCount*** : 이 값은 ORG 신호 또는 EL 신호가 ON 이 된 후 실제로 복귀 작업을 완료하는데 필요한 EZ Count 값을 0 ~ 15 사이의 값으로 설정 혹은 반환합니다. 이 값의 참조 여부는 원점복귀 모드에 따라서 다릅니다.

▶ ***EscDist*** : 원점 탈출 거리를 설정 혹은 반환합니다. 거리의 단위는 논리적 단위 거리를 사용합니다.

▶ ***Offset*** : 원점복귀 완료 위치에서 일정 거리 이상을 상대 이동할 필요가 있을 경우, 그 값을 설정 혹은 반환합니다. 이것은 원점 복귀 완료 위치를 기준으로 추가 모션 이동을 의미합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

long BoardID = 0;
long nHomeMode = 0, nHmDir = 0, nEzCount = 0;
double fEscDist = 0.0f, fOffset = 0.0f;

/*0 번 축을 원점 복귀 모드 = 0, (-)방향, Ez Count = 1, 원점 탈출 거리 = 3000, offset = 0 으로
원점 복귀 환경을 설정합니다.*/
cmxPmHomeSetConfig (BoardID, 3, ccmxX1, 0, cmDIR_N, 1, 3000, 0 );

//0 번 축에 설정되어 있는 원점 복귀 환경 설정 정보를 반환합니다.
cmxPmHomeGetConfig (BoardID, 3, ccmxX1, &nHomeMode, &nHmDir, &nEzCount, &fEscDist, &fOffset );
```

NAME

cmxPmHomeSetPosClrMode /

cmxPmHomeGetPosClrMode

- 원점 복귀 완료 후 위치 소거 모드 설정 및 설정 값 확인

INFORMATION

Home Return

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 4

다소 주의

원점 복귀 후 발생할 수 있는 일정량의 위치 편차는 정밀한 모션 제어에서 매우 중요한 부분입니다.

SYNOPSIS

❒ VT_I4 cmxPmHomeSetPosClrMode ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 PosClrMode)

❒ VT_I4 cmxPmHomeGetPosClrMode ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 PosClrMode)

DESCRIPTION

cmxPmHomeSetPosClrMode()/cmxPmHomeGetPosClrMode() 함수는 원점 복귀 이송 완료 후에 발생하는 모션 컨트롤러와 서보 드라이브간의 제어 편차에 의해 발생하는 입력 펄스(Feedback pulse) 처리 방법을 설정 및 설정 상태를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***PosClrMode*** : 원점복귀가 완료된 후에 Command 및 Feedback 위치가 클리어 되는 모드를 설정 혹은 반환합니다. PosClrMode 는 다음과 같습니다.

Value	Meaning
-1 (ccmxHPCM_DISABLE)	원점 복귀 클리어 모드를 사용하지 않습니다.
0 (ccmxHPCM_M0)	최종 완료 조건에 해당하는 외부 하드웨어 신호(ORG/EL/EZ)가 입력되는 순간에 Command 및 Feedback 의 위치를 '0'으로 소거합니다. 일정량의 Feedback 위치 편차를 보입니다.

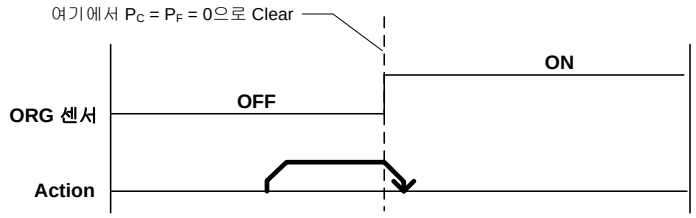
1 (ccmxHPCM_M1)	원점복귀 모드에 상관없이 하드웨어 신호의 입력 상태와 더불어 소프트웨어 적인 모션 완료 확인 동작을 포함한, 전체적인 원점복귀 작업이 모두 완료된 후에 Command 와 Feedback 위치를 모두 '0'으로 소거합니다. 일정량의 Feedback 위치 편차를 보입니다.
2 (ccmxHPCM_M2)	1 차적으로는 ccmxHPCM_M0 일 때와 동일하게 동작합니다. 단, 원점복귀가 완료된 후에 Feedback 위치와 동일한 값으로 Command 위치를 설정합니다. 이를 통해 서보 드라이브에서 실제 이송한 위치에 대한 양을 Command 위치에 반영시켜서, 다음 모션 동작을 수행할 수 있도록 합니다.

RETURN VALUE

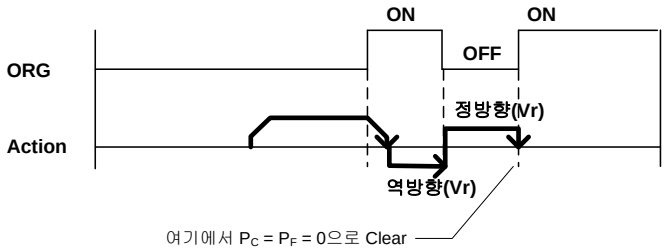
Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

REFERENCE

□ PosClrMode 가 0(ccmxHPCM_M0) 또는 2(ccmxHPCM_M2)로 하면 최종 단계에서 감속 없이 정지하는 원점복귀 모드(1, 2, 3, 4)에서는 Command 위치가 0 인 상태에서 원점복귀가 완료되고, 나머지 모드에서는 감속을 시작 할 때 위치가 0 으로 클리어 되며, 감속 하는 동안에 이송한 양만큼 위치가 증가 또는 감소하게 됩니다. 아래의 두 그림은 최종 단계에서 감속을 하는 원점복귀 모드 0 번과 즉시 정지를 하는 모드 1 번에서의 동작을 예로 든 것입니다.



[원점복귀 모드 0에서의 동작]



[원점복귀 모드 1에서의 동작]

□ 서보모터를 사용하는 경우에는 서보드라이버의 제어 지연 시간 때문에 원점복귀 완료 후 모터의 실제 위치는 약간의 편차가 발생할 수 있으며, 이는 원점복귀의 오차를 유발하게 됩니다. 그런데 PosClrMode 가 0(ccmxHPCM_M0) 또는 1(ccmxHPCM_M1)인 경우에는 모션제어기의 Feedback 카운터에 해당 편차가 그대로 반영됩니다. 따라서 Command 위치를 Feedback 위치와 일치시킨 후에 절대좌표 이송을 수행하면

이러한 편차에 의한 제어 오차를 제거할 수 있습니다. 이의 원리를 적용한 것이 PosClrMode 2(ccmxHPCM_M2)입니다.

□ 스텝모터를 사용하는 경우에는 ccmxHPCM_M2 를 사용하면 안됩니다.

서보모터를 사용하는 경우에는 ccmxHPCM_M2 로 하였을 때 가장 정확한 원점복귀 작업 결과를 얻을 수 있습니다. 단, 이때 다음과 같은 사항에 주의하여야 합니다.

• Command 방향과 Feedback 방향이 반드시 일치하여야 합니다.

• Command 분해능과 Feedback 분해능이 반드시 일치하여야 합니다.

• 원점복귀가 완료된 후에 Command 와 Feedback 좌표는 일치하지만 '0'이 되지는 않습니다. 따라서 경우에 따라서는 원점복귀 완료 후에 절대좌표 '0'으로 이송하는 명령이 필요할 수도 있습니다.

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

long BoardID = 0;
long nPosClrMode = 0;

//0 번 축을 원점복귀 완료 시 Command, Feedback 일치 모드로 사용합니다.
cmxPmHomeSetPosClrMode (BoardID, 3, ccmxX1, ccmxHPCM_M2 );

//0 번 축에 설정된 Position Clear Mode 를 반환합니다.
cmxPmHomeGetPosClrMode (BoardID, 3, ccmxX1, &nPosClrMode );
```

NAME

cmxPmHomeSetSpeedPattern /

cmxPmHomeGetSpeedPattern

- 원점 복귀 속도 설정 및 반환

INFORMATION

Home Return

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 4

다소 주의

속도설정은 논리적인 속도 단위가 적용됩니다.
기본적으로는 PPS(Pulse per second) 단위를 적용하며, 비율로 설정되는 단위가 아닙니다.

SYNOPSIS

❒ VT_I4 cmxPmHomeSetSpeedPattern
([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 SpeedMode, [in] VT_R8 Vel, [in] VT_R8 Accel, [in] VT_R8 Decel, [in] VT_R8 RevVel)

❒ VT_I4 cmxPmHomeGetSpeedPattern
([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 SpeedMode, [out] VT_PR8 Vel, [out] VT_PR8 Accel, [out] VT_PR8 Decel, [out] VT_PR8 RevVel)

DESCRIPTION

cmxPmHomeSetSpeedPattern()/cmxPmHomeGetSpeedPattern() 함수는 설정된 축의 원점복귀 속도 모드와 가속/감속도 및 작업속도, 역방향속도를 설정 및 설정 상태를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***SpeedMode*** : cmxPmHomeSetSpeedPattern() 함수의 인자이며, 원점복귀 시의 S-Curve 형 또는 선형 가감속, 가감속이 없는 모드등, 원점복귀 속도모드를 설정 혹은 반환합니다.

Value	Meaning
0 (cmxMODE_C)	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.

1 (cmxMODE_T)	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 (cmxMODE_S)	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.
-1 (ccmxSMODE_KEEP)	이전 속도 모드를 유지합니다.

▶ ***SpeedMode*** : cmxPmHomeGetSpeedPattern() 함수의 인자이며, 설정된 속도모드를 반환합니다.

Value	Meaning
0 (cmxMODE_C)	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 (cmxMODE_T)	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 (cmxMODE_S)	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.

▶ ***Vel*** : 원점 복귀 작업 속도를 설정 혹은 반환합니다. 기호로는 Vwork 로 표기합니다.

▶ ***Accel*** : 원점 복귀 가속도를 설정 혹은 반환합니다.

▶ ***Decel*** : 원점 복귀 감속도를 설정 혹은 반환합니다.

▶ ***RevVel*** : Reverse Speed 를 설정 혹은 반환합니다. 원점 복귀 모드에 따라 Reverse Speed 를 필요로 하는 모드가 있습니다. 앞의 원점 복귀 모드 설명에서 Reverse Speed 는 Vr 로 표기되었습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

long nHmSpdMode = 0;
double fVel = 0.0f, fAcc = 0.0f, fDec = 0.0f, fRevVel = 0.0f;
long BoardID = 0;

//0 번 축의 원점 복귀 속도 옵션을 설정합니다.
cmxPmHomeSetSpeedPattern (BoardID, 3, ccmxX1, cmxMODE_S, 5000, 1000, 1000, 1000 );

//0 번 축에 설정된 홈 복귀 옵션을 반환합니다.
cmxPmHomeGetSpeedPattern (BoardID, 3, ccmxX1, &nHmSpdMode, &fVel, &fAcc, &fDec, &fRevVel );
```

NAME

cmxPmHomeSetSpeedPattern_T /

cmxPmHomeGetSpeedPattern_T

- 원점 복귀 속도 설정 및 반환

INFORMATION

Home Return

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 4

다소 주의

속도설정은 논리적인 속도 단위가 적용됩니다.
기본적으로는 PPS(Pulse per second) 단위를 적용하며, 비율로 설정되는 단위가 아닙니다.

SYNOPSIS

❒ VT_I4 cmxPmHomeSetSpeedPattern_T
([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 SpeedMode, [in] VT_R8 Vel, [in] VT_R8 AccelTime, [in] VT_R8 DecelTime, [in] VT_R8 RevVel)

❒ VT_I4 cmxPmHomeGetSpeedPattern_T
([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 SpeedMode, [out] VT_PR8 Vel, [out] VT_PR8 AccelTime, [out] VT_PR8 DecelTime, [out] VT_PR8 RevVel)

DESCRIPTION

cmxPmHomeSetSpeedPattern_T()/cmxPmHomeGetSpeedPattern_T() 함수는 설정된 축의 원점복귀 속도 모드와 가속/감속 시간 및 작업속도, 역방향속도를 설정 및 설정 상태를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***SpeedMode*** : cmxPmHomeSetSpeedPattern_T() 함수의 인자이며, 원점복귀 시의 S-Curve 형 또는 선형 가감속, 가감속이 없는 모드등, 원점복귀 속도모드를 설정 혹은 반환합니다.

Value	Meaning
0 (cmxMODE_C)	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.

1 (cmxMODE_T)	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 (cmxMODE_S)	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.
-1 (ccmxSMODE_KEEP)	이전 속도 모드를 유지합니다.

▶ ***SpeedMode*** : cmxPmHomeGetSpeedPattern_T() 함수의 인자이며, 설정된 속도모드를 반환합니다.

Value	Meaning
0 (cmxMODE_C)	CONSTANT 속도모드 => 가감속을 수행하지 않습니다.
1 (cmxMODE_T)	TRAPEZOIDAL 속도모드 => 사다리꼴 가감속을 수행합니다.
2 (cmxMODE_S)	S-CURVE 속도모드 => S-CURVE 가감속을 수행합니다.

▶ ***Vel*** : 원점 복귀 작업 속도를 설정 혹은 반환합니다. 기호로는 Vwork 로 표기합니다.

▶ ***AccelTime*** : 원점 복귀 가속시간을 설정 혹은 반환합니다.

▶ ***DecelTime*** : 원점 복귀 감속 시간을 설정 혹은 반환합니다.

▶ ***RevVel*** : Reverse Speed 를 설정 혹은 반환합니다. 원점 복귀 모드에 따라 Reverse Speed 를 필요로 하는 모드가 있습니다. 앞의 원점 복귀 모드 설명에서 Reverse Speed 는 Vr 로 표기되었습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

NAME

cmxPmHomeMoveStart

- 단축 원점 복귀 이송

INFORMATION

Home Return

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 4

다소 위험

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인합니다.

SYNOPSIS

❒ VT_I4 cmxPmHomeMoveStart ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel)

DESCRIPTION

cmxPmHomeMoveStart() 함수는 원점 복귀 작업을 수행합니다. cmxPmHomeMoveStart() 함수는 원점 복귀 이송을 시작시킨 후에 바로 반환됩니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ERR_NONE)	수행 성공

REFERENCE

□ cmxPmHomeMoveStart() 함수를 사용하는 경우에는 cmxPmSxIsDone(), cmxPmSxWaitDone() 함수를 사용하여 모션의 완료를 확인할 수 있습니다.

보충	윈도우 이벤트라는 것은 무엇입니까?
	윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용 프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 핸선지가 되는 윈도우에 전송되어 처리됩니다.

□ INP 입력신호가 Enable 로 설정되었으면 Command 펄스 출력이 완료되어도 INP 입력이 ON 이 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환되지 않습니다.

□ 스텝 드라이브를 사용 중인 고객님들께서는 다음을 참조해 주십시오.
스텝 드라이브는 INP 출력이 없는 경우가 일반적인데, 고객님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스텝 드라이브를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다. 고객 여러분들께서는 스텝 드라이브 사용시에 이점을 주의해 주시기를 부탁 드립니다.

□ 서보 드라이브의 LSP , LSN 신호를 사용 중인 고객님께서는 다음을 참조해 주십시오.
서보드라이브의 입력 신호 중 하나인 EL(End of Limit) 신호는 저희 ㈜ 커미조아 모션 컨트롤러뿐만 아니라 서보드라이브에도 전달 될 수 있도록 설정할 수 있습니다. 통상적으로 LSP 신호와 LSN 신호로 불리어 지는 이 신호는 실제 기구물에서 양의 방향(Positive Direction) 혹은 음의 방향(Negative Direction) 에 장착되어 있는 EL(End of Limit) 신호를 서보 드라이브 측에 전달하기 위한 용도로 사용됩니다.

그러나, 모션 소프트웨어에서 INP 설정이 되어 있는 경우 EL 신호가 검출 된 후에 일부 서보 드라이브에서는 진행 방향에서 정지 한 후 더 이상 움직이지 않는 상황이 발생하며, 이 상황에서 INP 신호가 출력되지 않아, 모션 이송이 완료되지 못하고, 명시적으로 STOP 명령을 통해 강제적으로 모션 종료가 되는 현상이 발생할 수 있습니다.. 이 현상은 EL 모드를 통해 원점 복귀를 하는 상황에서도 발생할 수 있습니다.

따라서, 이러한 경우에는 반드시 원점 복귀나 EL 검출 시에 인터럽트 이벤트나 타이머를 통해 INP 를 무기한 대기하는 현상에 대해서 적절히 대처하시거나 INP 신호 사용을 배제 해야 합니다.

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

long BoardID = 0;
/* 0 번 축의 원점 복귀 환경을 원점 복귀 모드 = 0, 방향 = (-), Ez Count = 0, 원점 탈출 거리 = 3000, Offset = 0 으로 설정합니다. */
cmxPmHomeGetConfig(BoardID, 3, ccmxX1, 0, cmDIR_N, 0, 3000, 0 );

// 0 번 축의 원점 복귀 속도 패턴을 설정합니다.
cmxPmHomeSetSpeedPattern(BoardID, 3, ccmxX1, cmxMODE_S, 5000, 1000, 1000, 1000 );

// 원점 복귀를 시작합니다.
if( cmxPmHomeMove(BoardID, 3, ccmxX1, cmxFALSE ) != ERR_NONE )
{
    OutputDebugString( "Home move start fail!" );
    return;
}
```

```
long nIsSuccess = 0;
//0 번 축의 원점 복귀 성공 여부를 확인합니다.
cmxPmHomeGetSuccess(BoardID, 3, ccmxX1, &nIsSuccess );

// 원점 복귀 성공 메시지를 표시합니다.
if( nIsSuccess )
{
    MessageBox( "Home return success", "Message", MB_OK );
}
```

NAME

cmxPmHomeMoveAllStart

- 다축 원점 복귀 이송

INFORMATION

Home Return

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 4

다소 위험

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인합니다.

SYNOPSIS

❒ VT_I4 cmxPmHomeMoveAllStart ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 NumChannel , [in] VT_I4 ChannelMask)

DESCRIPTION

cmxPmHomeMoveAllStart() 함수는 원점 복귀 작업을 수행합니다. cmxPmHomeMoveAllStart() 함수는 원점 복귀 이송을 시작시킨 후에 바로 반환됩니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***NumChannel*** : 원점 복귀할 Channel 의 수.

▶ ***ChannelMask*** : 동시에 작업을 수행할 대상 축의 마스크

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다.
0 (ERR_NONE)	수행 성공

REFERENCE

□ cmxPmHomeMoveStart() 함수를 사용하는 경우에는 cmxPmSxIsDone(), cmxPmSxWaitDone() 함수를 사용하여 모션의 완료를 확인할 수 있습니다.

보충	윈도우 이벤트라는 것은 무엇입니까?
	윈도우 운영체제는 Event Driven 혹은 Message Driven 방식의 구조로 되어 있습니다. 각 응용 프로그램은 메시지 큐(Queue)를 가지고 있으며, 정확히 말하면, 메시지를 사용해 이벤트를 통지하는 방식으로 설계되어 있습니다. 윈도우 메시지를 처리한다는 것은 메시지 큐에서 메시지를 하나씩 꺼내서 윈도우 프로시저에 전송한다는 것을 의미하며, 이것은 그 핸선지가 되는 윈도우에 전송되어 처리됩니다.

□ INP 입력신호가 Enable 로 설정되었으면 Command 펄스 출력이 완료되어도 INP 입력이 ON 이 되기 전까지는 모션이 완료되지 않은 것으로 간주되어 반환되지 않습니다.

□ 스텝 드라이브를 사용 중인 고객님들께서는 다음을 참조해 주십시오.
스텝 드라이브는 INP 출력이 없는 경우가 일반적인데, 고객님의 부주의나 잘못된 설정으로 INP 입력에 대한 설정이 Enable 로 되어 있을 경우 INP 입력이 스텝 드라이브를 통해 발생하지 않는 이유 때문에 모션 완료가 되지 않는 경우가 발생할 수 있습니다. 고객 여러분들께서는 스텝 드라이브 사용시에 이점을 주의해 주시기를 부탁 드립니다.

□ 서보 드라이브의 LSP , LSN 신호를 사용 중인 고객님께서는 다음을 참조해 주십시오.
서보드라이브의 입력 신호 중 하나인 EL(End of Limit) 신호는 저희 ㈜ 커미조아 모션 컨트롤러뿐만 아니라 서보드라이브에도 전달 될 수 있도록 설정할 수 있습니다. 통상적으로 LSP 신호와 LSN 신호로 불리어 지는 이 신호는 실제 기구물에서 양의 방향(Positive Direction) 혹은 음의 방향(Negative Direction) 에 장착되어 있는 EL(End of Limit) 신호를 서보 드라이브 측에 전달하기 위한 용도로 사용됩니다.

그러나, 모션 소프트웨어에서 INP 설정이 되어 있는 경우 EL 신호가 검출 된 후에 일부 서보 드라이브에서는 진행 방향에서 정지 한 후 더 이상 움직이지 않는 상황이 발생하며, 이 상황에서 INP 신호가 출력되지 않아, 모션 이송이 완료되지 못하고, 명시적으로 STOP 명령을 통해 강제적으로 모션 종료가 되는 현상이 발생할 수 있습니다.. 이 현상은 EL 모드를 통해 원점 복귀를 하는 상황에서도 발생할 수 있습니다.

따라서, 이러한 경우에는 반드시 원점 복귀나 EL 검출 시에 인터럽트 이벤트나 타이머를 통해 INP 를 무기한 대기하는 현상에 대해서 적절히 대처하시거나 INP 신호 사용을 배제 해야 합니다.

NAME

cmxPmHomeGetSuccess /

cmxPmHomeSetSuccess

- 이전 원점 복귀 완료 확인 및 설정

INFORMATION

Home Return

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 4

다소 주의

원점 복귀 성공 여부는 응용 프로그램의 종료와 관계없이 유지됩니다.

SYNOPSIS

VT_I4 cmxPmHomeGetSuccess ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 IsSuccess)

VT_I4 cmxPmHomeSetSuccess ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 IsSuccess)

DESCRIPTION

cmxPmHomeGetSuccess() 함수는 이 함수가 호출되기 이전에 원점 복귀 이송이 성공적으로 완료되었는지 확인하는 함수입니다.

cmxPmHomeSetSuccess() 함수는 원점 복귀의 성공 여부에 대한 상태를 강제로 설정하는 함수입니다. 일반적으로는 원점 복귀 완료 상태는 원점 복귀의 실제 수행에 의해서 설정됩니다. 그러나 필요한 경우에 강제로 설정할 수 있습니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***IsSuccess*** : cmxPmHomeGetSuccess() 함수의 인자이며, 이 함수가 호출된 시점을 기준으로 이전에 원점복귀가 성공적으로 완료된 상태인지를 알려 주는 매개 변수입니다.

Value	Meaning
0 (cmxFALSE)	지정한 축의 원점 복귀 이송이 진행 중이거나 혹은 비정상적으로 완료 되었음을 나타냅니다.
1 (cmxTRUE)	지정한 축의 원점 복귀 이송이 정상적으로 완료된 상태입니다.

▶ ***IsSuccess*** : cmxPmHomeSetSuccess() 함수의 인자이며, 원점 복귀의 성공 여부에 대한 상태를 강제로 설정합니다.

Value	Meaning
0 (cmxFALSE)	지정한 축을 원점 복귀 이송이 진행 중인 상태 혹은 비정상적으로 완료된 상태로 설정합니다.
1 (cmxTRUE)	지정한 축을 원점 복귀 이송이 정상적으로 완료된 상태로 설정합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

REFERENCE

□ 원점복귀의 성공 여부에 대한 상태는 응용 프로그램이 종료되어도 그대로 유지됩니다. 따라서 다시 응용 프로그램이 시작되면 이전에 원점복귀를 정상적으로 수행했었는지를 알 수가 있습니다. 단, 제어 시스템의 하드웨어적인 전원이 차단되거나 재 시작(Rebooting) 되면 원점 복귀가 완료되지 않은 상태로 설정됩니다. 따라서 cmxPmHomeGetSuccess() 함수의 이러한 특성을 활용하면 프로그램이 종료되었다가 다시 실행될 때 이전의 원점복귀 수행 여부를 확인할 수가 없어서 매번 원점복귀를 수행해야 했던 불편을 보완할 수 있습니다.

□ IsSuccess 매개 변수가 FALSE 인 경우는 원점복귀가 진행 중인 경우를 의미할 수도 있고 비정상적으로 종료되었음을 의미할 수도 있습니다. 따라서 cmxPmHomeMoveStart() 함수를 사용한 경우에는 먼저 cmxPmSxWaitDone() 함수를 선행하여 완료를 확인한 후에 cmxPmHomeGetSuccess()를 사용하여 성공 여부를 확인하는 것이 정석입니다.

□ 이전에 원점복귀가 성공적으로 수행되었더라도 해당 축의 원점복귀를 다시 시작하면 원점복귀가 완료되지 않은 상태로 설정됩니다.

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

long BoardID = 0;
/* 0 번 축에 대하여 원점 복귀 이송을 시작하고, 진행 상태를 확인한 후 완료 여부를 확인합니다. */
long nIsHomming = cmxTRUE;

cmxPmHomeMoveStart(BoardID, 3, ccmxX1 );

while( nIsHomming )
{
    //원점 복귀 진행 여부를 확인합니다.
    cmxPmSxWaitDone(BoardID, 3, ccmxX1, &nIsHomming );
}

long nIsSuccess = cmxFALSE;

//원점 복귀 성공 여부를 확인합니다.
cmxPmHomeGetSuccess(BoardID, 3, ccmxX1, &nIsSuccess );
```

```
if( nIsSuccess )
{cmxMODE_SC
    MessageBox( "Home return success", "Message", MB_OK );
}
else
{
    OutputDebugString( "Home return fail!" );
}
```

Chapter

16

Advanced Pulse Motion Control Functions

커미조아의 모션 세계는 비단 기본 모션제어에서만 국한되지 않습니다. 더 높은 기능과 더 안정적인 기능이 커미조아의 제품을 대변해 주고 있습니다. 고급 모션제어에서는 그 기능에 있어 다양한 기능을 표현하고 있지만, 기능의 응용에서 비로소 진정한 고급 모션제어를 구현하실 수 있습니다. ㈜커미조아의 고급 모션제어서는 1 나노미터의 오차도 허용하지 않는 저희 ㈜커미조아의 모션제품이 고객(顧客) 여러분들을 만족시켜드립니다.

이 단원에서는 속도(速度) 및 위치(位置) 오버라이딩 함수들을 소개합니다. 속도(速度) 오버라이딩은 모션이 진행되고 있는 중에 작업 속도를 변경하는 것을 의미합니다. 위치(位置) 오버라이딩은 상대좌표 모션 이송이나 절대좌표 모션이송과 같이 목적좌표를 향해 추종 모션을 수행하고 있는 중에 최종 목표 거리 또는 최종 목표 좌표를 수정하는 것을 의미 합니다.

16 고급 모션 제어 편

16.1 속도 및 위치 오버라이딩(Overriding)

이 단원에서는 속도 및 위치 오버라이딩 함수들을 소개합니다.

속도 오버라이딩은 모션이 진행되고 있는 중에 작업 속도를 변경하는 것을 의미합니다.

위치 오버라이딩은 Move 나 MoveTo 와 같이 In-Position 모션을 수행하고 있는 중에 목표 거리 또는 목표 좌표를 수정하는 것을 의미합니다.

일반적인 모션 구동에서 많이 요구되는 기능은 아니지만, 그 기능면에 있어 타사의 다른 오버라이드 기능보다 월등한 기능과 성능, 그리고 정확성을 제공하고 있습니다.

16.1.1 함수 요약

㈜ 커미조아의 속도 및 위치 오버라이딩에 관련된 함수는 다음과 같습니다.

Summary of Functions
❒ VT_I4 cmxPmOverrideSpeedSet ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel) 단축 모션 작업이 진행되고 있는 중에 속도를 변경합니다.
❒ VT_I4 cmxPmOverrideSpeedSetAll ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 ChannelMask) 다축 모션 작업이 진행되고 있는 중에 속도를 변경합니다.
❒ VT_I4 cmxPmOverrideMove ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_R8 NewDistance, [in] VT_I4 IsHardApply, [out] VT_PI4 AppliedState) 단축 구동 함수를 통해서 구동되는 단축 상대좌표 이송 모션에 대하여, 상대좌표상의 목표 논리 거리 값을 수정합니다.
❒ VT_I4 cmxPmOverrideMoveTo ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_R8 NewPosition, [in] VT_I4 IsHardApply) 단축 구동 함수를 통해서 구동되는 단축 절대좌표 이송 모션에 대하여, 절대좌표상의 목표 논리 거리 값을 수정합니다.

참고적으로, 속도 및 위치 오버라이딩 함수는 모션이 종료된 시점에서 수행되는 함수가 아닌, 모션의 이송 중에 적용되는 함수이기 때문에, 이전에 수행된 모션 명령이 cmxPmSxMove() 나 cmxPmSxMoveTo() 같은 모션 이송의 목표 위치에 대한 완료를 동반하여 반환되는 함수에는 적용되지 않습니다. 따라서 오버라이딩 함수의 이전 함수는 cmxPmSxMoveStart() 나 cmxPmSxMoveToStart() 함수와 같이 이송 명령이 설정된 후에 응용 프로그램의 제어를 즉시 반환 받고, 오버라이딩을 수행하게 됩니다.

16.1.2 함수 설명

NAME

cmxPmOverrideSpeedSet

- 단축 속도 오버라이딩

INFORMATION

Overriding

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 5

다소 주의

함수의 호출 전에 속도 지령 함수를 통해 변경하고자 하는 속도를 설정합니다.

SYNOPSIS

❒ VT_I4 cmxPmSetOverrideSpeedSet ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel)

DESCRIPTION

cmxPmOverrideSpeedSet() 함수는 단축 모션이 진행되고 있는 중에 속도를 오버라이딩 하고자 할 때 사용하는 함수입니다. 속도를 오버라이딩 하기 위해서는 먼저 cmxPmCfgSpeedPatternSet() 속도 패턴 설정 함수를 통하여 변경하고자 하는 속도 또는 가속도 값을 설정하고 나서 이 함수를 수행해야 합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

REFERENCE

□ 보간 작업을 수행하는 경우에는 속도 오버라이딩을 사용할 수 없습니다.

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

long BoardID = 0;
//0 번 축의 속도를 설정합니다.
cmxPmCfgSetSpeedPattern(BoardID, 3, ccmxX1, cmxMODE_S, 5000, 30000, 30000 );
cmxPmSxSetSpeedRatio(BoardID, 3, ccmxX1, ccmxSMODE_KEEP, 100, 100, 100 );

//0 번 축을 구동 시킵니다.
cmxPmSxMoveStart(BoardID, 3, ccmxX1, 15000 );

//오버라이드 할 축 및 속도를 지정합니다.
cmxPmCfgSetSpeedPattern(BoardID, 3, ccmxX1, cmxMODE_S, 1000, 10000, 10000 );

/*cmxPmSxSetSpeedRatio()함수를 이용하여
오버라이드 할 속도를 설정 할 수 있습니다.*/
//cmxPmSxSetSpeedRatio(BoardID, 3, ccmxX1, ccmxSMODE_KEEP, 70, 70, 70 );

//속도를 오버라이드 합니다.
cmxPmOverrideSpeedSet(BoardID, 3, ccmxX1 );
```

NAME

cmxPmOverrideSpeedSetAll

- 다축 속도 오버라이딩

INFORMATION

Overriding

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 5

다소 주의

함수의 호출 전에 속도 지령 함수를 통해 변경하고자 하는 속도를 설정합니다.

SYNOPSIS

❒ VT_I4 cmxPmSetOverrideSpeedSetAll ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 ChannelMask)

DESCRIPTION

cmxPmOverrideSpeedSet() 함수는 단축 모션이 진행되고 있는 중에 속도를 오버라이딩 하고자 할 때 사용하는 함수입니다. 속도를 오버라이딩 하기 위해서는 먼저 cmxPmCfgSpeedPatternSet() 속도 패턴 설정 함수를 통하여 변경하고자 하는 속도 또는 가속도 값을 설정하고 나서 이 함수를 수행해야 합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***ChannelMask*** : 동시에 작업을 수행할 대상 축의 마스크

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

REFERENCE

□ 보간 작업을 수행하는 경우에는 속도 오버라이딩을 사용할 수 없습니다.

NAME

cmxPmOverrideMove

- 단축 상대 위치 오버라이드 이송

INFORMATION

Overriding

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 5

다소 위험

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인합니다.

SYNOPSIS

❒ VT_I4 cmxPmOverrideMove

([in] VT_I4 BoardID, [in] VT_I4 Node, [in] VT_I4 Channel, [in] VT_R8 NewDistance, [in] VT_I4 IsHardApply, [out] VT_PI4 AppliedState)

DESCRIPTION

이 함수는 cmxPmSxMoveStart() 이송 함수를 통하여 수행되는 상대좌표 In-position 모션에 대하여 상대좌표 값, 즉 목표 거리 값을 오버라이딩 하는 함수입니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***NewDistance*** : 새로운 목표 거리 값을 지정합니다. 이 값의 기준 위치는 오버라이드 하고자 하는 대상이 되는 cmxPmSxMoveStart() 작업에서 사용한 기준점과 같습니다. 즉, 새로운 목표 거리는 cmxPmSxMoveStart() 함수를 실행하기 바로 직전의 위치를 기준으로 계산하여야 합니다.

▶ ***IsHardApply*** : Override 불가 시점에서도 Override 설정을 강제로 적용.

▶ ***AppliedState*** : cmxPmOverrideMove() 함수의 적용 성공/실패 여부를 반환 합니다.

Value	Meaning
0 (cmxFALSE)	모션 에러가 발생하였거나 이미 이송이 완료되어 위치 오버라이드가 적용되지 않음.
1 (cmxTRUE)	위치 오버라이드가 적용됨.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

SEE ALSO

cmxPmSxMoveStart

REFERENCE

□ 위치 오버라이드를 수행하려는 시점에 이미 이송이 완료되어 버린 경우에는 위치 오버라이드는 무시되고 반환값을 0 으로 반환합니다. 따라서 사용자는 반환값이 0 인 경우에는 이미 이송이 완료되어 오버라이드가 적용되지 않은 것으로 인지하여야 하며, 그럼에도 불구하고 목표 좌표를 수정해야 하는 경우에는 cmxPmSxMove() 또는 cmxPmSxMoveTo() 함수를 추가적으로 수행해야 합니다. 이러한 경우에는 오버라이드라는 개념 보다는 추가 이송의 개념을 의미합니다.

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

long BoardID = 0;
//0 번 축의 속도를 설정합니다.
cmxPmCfgSetSpeedPattern(BoardID, 3, ccmxX1, cmxMODE_S, 5000, 30000, 30000 );
cmxPmSxSetSpeedRatio(BoardID, 3, ccmxX1, ccmxSMODE_KEEP, 100, 100, 100 );

//0 번 축을 구동 시킵니다.
cmxPmSxMoveStart(BoardID, 3, ccmxX1, 15000 );

long nAppliedState = 0;
/*이동 할 거리를 20000 으로 오버라이드 합니다.
IsHardApply 인자를 cmxTRUE 로 할 경우에는 모션이
종료된 상태에서도 강제로 이동시킵니다.*/
cmxPmOverrideMove(BoardID, 3, ccmxX1, 20000, cmxTRUE, &nAppliedState );
```

NAME

cmxPmOverrideMoveTo

- 단축 절대 위치 오버라이드 이송

INFORMATION

Overriding

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 5

다소 위험

실제 이송이 진행되는 함수이므로, 사전에 반드시 안전을 확인합니다.

SYNOPSIS

❒ VT_I4 cmxPmOverrideMoveTo

([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_R8 NewPosition, [in] VT_I4 IsHardApply)

DESCRIPTION

이 함수는 cmxPmSxMoveToStart() 함수를 통하여 수행되는 절대좌표 In-position 모션에 대하여 목표 절대좌표 값을 오버라이딩 하는 함수입니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***NewPosition*** : 새로운 목표 절대좌표 값을 지정합니다.

▶ ***IsHardApply*** : Override 불가 시점에서도 Override 설정을 강제로 적용합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

SEE ALSO

cmxPmSxMoveToStart

REFERENCE

□ 위치 오버라이드를 수행하려는 시점에 이미 이송이 완료되어 버린 경우에는 위치 오버라이드는 무시되고 반환값을 0 으로 반환합니다. 따라서 사용자는 반환값이 0 인 경우에는 이미 이송이 완료되어 오버라이드가 적용되지 않은 것으로 인지하여야 하며, 그럼에도 불구하고 목표 좌표를 수정해야 하는 경우에는 cmxPmSxMove() 또는 cmxPmSxMoveTo() 함수를 추가적으로 수행해야 합니다.

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

long BoardID = 0;
//0 번 축의 속도를 설정합니다.
cmxPmCfgSetSpeedPattern(BoardID, 3, ccmxX1, cmxMODE_S, 5000, 30000, 30000 );
cmxPmSxSetSpeedRatio(BoardID, 3, ccmxX1, ccmxSMODE_KEEP, 100, 100, 100 );

//0 번 축을 구동 시킵니다.
cmxPmSxMoveToStart(BoardID, 3, ccmxX1, 13000 );

long nAppliedState = 0;
/*이동 할 위치를 10000 으로 오버라이드 합니다.
IsHardApply 인자를 cmxTRUE 로 할 경우에는 모션이
종료된 상태에서도 강제로 이동시킵니다.*/
cmxPmOverrideMoveTo(BoardID, 3, ccmxX1, 10000, cmxTRUE, &nAppliedState );
```

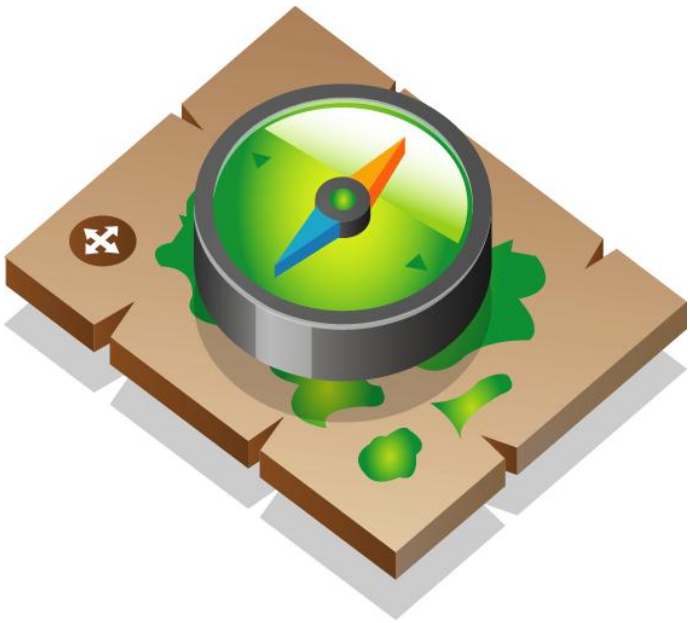
Chapter

17

Advanced Pulse Motion Control Functions

모션제어의 상태를 확인(確認)하는 역할은 고객(顧客)여러분들께서 개발하시는 응용프로그램의 필수 요건 중에 하나입니다. ComiRTEX 에서는 매우 자세하고 효율적인 상태 관리 매커니즘을 가지고 있습니다. ㈜커미조아의 많은 장점 중에 하나인 전문적인 모션 정보 제공 인터페이스를 통해 보다 자세하고 신속한 모션 프로그램의 상태를 구현하시기 바랍니다.

이 단원에서는 모션제어의 상태(狀態) 감시에 관련된 함수들에 대하여 설명합니다. 상태(狀態) 감시 함수들은 모션의 상태를 감시하는데 필요한 함수들을 그룹화한 것입니다. 상태(狀態) 감시에는 모션의 속도, 위치 등을 확인(確認)하는 것을 포함하며 이외에도 현재 모션이 진행되고 있는지를 확인(確認)하고, 현재 진행되고 있는 모션의 단계 및 각 I/O 상태들을 점검하는 기능도 포함합니다.



17 상태 감시 편

17.1 모션제어 상태(Status) 감시 및 설정

이 단원에서는 모션제어의 상태 감시에 관련된 함수들에 대하여 설명합니다.

상태 감시 함수들은 모션의 상태를 감시하는데 필요한 함수들을 그룹화한 것입니다.

상태 감시에는 모션의 속도, 위치 등을 확인하는 것을 포함하며, 이외에도 현재 모션이 진행되고 있는지를 확인하고, 현재 진행되고 있는 모션의 단계 및 각 I/O 상태들을 확인하는 기능도 포함합니다.

17.1.1 함수 요약

상태 감시 및 제어 함수에 관련된 함수들은 다음과 같습니다.

Summary of Functions
❒ VT_I4 cmxPmStSetCount ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 Target, [in] VT_I4 Count) 대상 모션 축의 지정한 카운터(Counter)의 값을 전달된 매개변수를 통해 설정합니다. 단, 이때 지정하는 카운터 값의 단위는 펄스 수(PPS) 입니다.
❒ VT_I4 cmxPmStGetCount ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 Source, [out] VT_PI4 Count) 대상 모션 축의 지정한 카운터(Counter)의 값을 전달된 매개변수를 통해 반환합니다. 단, 이때 반환되는 카운터 값의 단위는 펄스 수(PPS) 입니다.
❒ VT_I4 cmxPmStSetPosition ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 Target, [in] VT_R8 Position) 대상 모션 축의 지정한 카운터(Counter)의 값을 전달된 매개변수를 통해 설정합니다. 단, 이때 지정하는 카운터 값의 단위는 논리적인 단위 거리(Unit Distance) 입니다.
❒ VT_I4 cmxPmStGetPosition ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 Source, [out] VT_PR8 Position) 대상 모션 축의 지정한 카운터(Counter)의 값을 전달된 매개변수를 통해 반환합니다. 단, 이때 반환되는 카운터 값의 단위는 논리적인 단위 거리(Unit Distance) 입니다.
❒ VT_I4 cmxPmStGetSpeed ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 Source, [out] VT_PR8 Speed) 대상 모션 축의 Command 또는 Feedback 속도를 확인하여, 전달된 매개 변수를 통해 논리적 속도 단위로 반환합니다.
❒ VT_I4 cmxPmStReadMotionState ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 MotStates) 대상 모션 축에 대해서, 현재 모션 동작 상태를 반환합니다.
❒ VT_I4 cmxPmStReadMIOStatuses ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 MioStates) 대상 모션 축에 대해서, 현재 모션 관련 I/O 신호 및 주변 신호(Machine I/O) 상태를 반환합니다.

17.1.2 함수 설명

NAME

cmxPmStSetCount / cmxPmStGetCount

- 사용자 정의 하드웨어 카운트 값 설정 및 반환

INFORMATION

Motion Status

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 7

위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxPmStSetCount ([in] VT_I4 BoardID, [in] VT_I4 Axis, [in] VT_I4 Target, [in] VT_I4 Count)

❒ VT_I4 cmxPmStGetCount ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 Source, [out] VT_PI4 Count)

DESCRIPTION

cmxPmStSetCount() 함수는 지정한 축의 지정한 카운터 값을 새로이 설정합니다. 단, 이때 지정하는 카운터 값의 단위는 펄스 수입니다. 이 함수는 카운터의 값을 지정하는 매개 변수의 단위가 펄스 수라는 것을 제외하고는 cmxPmStGetCount() 함수와 동일합니다.

cmxPmStGetCount()함수는 지정한 축의 지정한 카운터의 값을 읽어서 반환합니다. 단, 이때 반환되는 값의 단위는 펄스 수입니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***Target*** : 설정할 카운터 번호. cmxPmStSetCount() 함수의 인자이며, 다음의 4 가지 값 중의 하나이어야 합니다.

Value	Meaning
0 (cmxCNT_COMM)	Command Counter
1 (cmxCNT_FEED)	Feedback Counter
2 (ccmxCNT_DEV)	Deviation Counter : Command 와 Feedback counter 의 편차 카운터

3 (ccmxCNT_GEN)	General Counter : 사용자의 정의에 따라 여러 가지 용도로 사용될 수 있는 카운터

▶ ***Source*** : 대상 카운터 번호. cmxPmStGetCount() 함수의 인자이며, 이 값은 다음의 4 가지 값 중의 하나이어야 합니다.

Value	Meaning
0 (cmxCNT_COMM)	Command Counter
1 (cmxCNT_FEED)	Feedback Counter
2 (ccmxCNT_DEV)	Deviation Counter
3 (ccmxCNT_GEN)	General Counter

▶ ***Count*** : 지정한 값으로 대상 카운터의 값을 설정 혹은 반환합니다. 이 값은 이 값은 논리 단위가 아닌 실제 펄스 카운트 값입니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

SEE ALSO

cmxPmStSetPosition, cmxPmStGetPosition

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

long BoardID = 0;
//0 번 축의 Command Counter 를 0 으로 설정합니다.
cmxPmStSetCount(BoardID, 3, ccmxX1, cmxCNT_COMM, 0 );

//0 번 축의 Feedback Counter 를 0 으로 설정합니다.
cmxPmStSetCount(BoardID, 3, ccmxX1, cmxCNT_FEED, 0 );

long nCmdCounter = 0;
long nFdbCounter = 0;

//0 번 축의 Command Counter 값을 반환합니다.
cmxPmStGetCount(BoardID, 3, ccmxX1, cmxCNT_COMM, &nCmdCounter );

//0 번 축의 Feedback Counter 값을 반환합니다.
cmxPmStGetCount(BoardID, 3, ccmxX1, cmxCNT_FEED, &nFdbCounter );
```

NAME

cmxPmStSetPosition/ cmxPmStGetPosition

사용자 정의 논리적 카운트 값 설정 및 반환

INFORMATION

- Motion Status
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 7
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxPmStSetPosition ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 Target, [in] VT_R8 Position)

❒ VT_I4 cmxPmStGetPosition ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 Source, [out] VT_PR8 Position)

DESCRIPTION

cmxPmStSetPosition () 함수는 지정한 축의 지정한 카운터의 값을 새로이 설정합니다. 단, 이때 지정하는 카운터 값의 단위는 "Unit distance" 에 의해 정의되는 논리적 단위 거리입니다. 이 함수는 카운터의 값을 지정하는 매개 변수가 논리적 단위 거리라는 것을 제외하고는 cmxPmStSetCount() 함수와 동일합니다.

cmxPmStGetPosition()함수는 지정한 축의 지정한 카운터의 값을 읽어서 반환합니다. 단, 이때 반환되는 값의 단위는 "Unit distance" 에 의해 정의되는 논리적 거리입니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***Target*** : 설정할 카운터 번호. cmxPmStSetPosition() 함수의 인자이며, 다음의 4 가지 값 중의 하나이어야 합니다.

Value	Meaning
0 (cmxCNT_COMM)	Command Counter
1 (cmxCNT_FEED)	Feedback Counter
2 (ccmxCNT_DEV)	Deviation Counter : Command 와 Feedback counter 의 편차 카운터
3 (ccmxCNT_GEN)	General Counter : 사용자의 정의에 따라 여러 가지 용도로 사용될 수 있는 카운터

▶ ***Source*** : 대상 카운터 번호. cmxPmStGetPosition() 함수의 인자이며, 이 값은 다음의 4 가지 값 중의 하나이어야 합니다.

Value	Meaning
0 (cmxCNT_COMM)	Command Counter
1 (cmxCNT_FEED)	Feedback Counter
2 (ccmxCNT_DEV)	Deviation Counter
3 (ccmxCNT_GEN)	General Counter

▶ ***Position*** : 대상 카운터에 설정 혹은 반환될 값. 단, 이 값은 논리적 단위 거리 입니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

SEE ALSO

cmxPmStSetCount, cmxPmStGetCount

REFERENCE

□ 논리적 단위 거리는 cmxPmCfgSetUnitDist() 함수에 의해 결정됩니다

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

long BoardID = 0;

//0 번 축의 Command Position 을 0 으로 설정합니다.
cmxPmStSetPosition(BoardID, 3, ccmxX1, cmxCNT_COMM, 0 );

//0 번 축의 Feedback Position 을 0 으로 설정합니다.
cmxPmStSetPosition(BoardID, 3, ccmxX1, cmxCNT_FEED, 0 );

long nCmdPosition = 0;
long nFdbPosition = 0;

//0 번 축의 Command Position 값을 반환합니다.
cmxPmStGetPosition(BoardID, 3, ccmxX1, cmxCNT_COMM, &nCmdPosition );

//0 번 축의 Feedback Position 값을 반환합니다.
cmxPmStGetPosition(BoardID, 3, ccmxX1, cmxCNT_FEED, &nFdbPosition );
```

NAME

cmxPmStGetSpeed

- 논리적 속도 반환

INFORMATION

Motion Status

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 7

위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxPmStGetSpeed ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 Source, [out] VT_PR8 Speed)

DESCRIPTION

이 함수는 Command 또는 Feedback 속도를 읽어서 논리적 속도 단위로 반환합니다. Source 매개 변수에 따라서 Command 속도 혹은 Feedback 속도 중 해당하는 속도에 대해서 반환 대상이 결정됩니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***Source*** : 속도 반환 대상이 되는 카운터 번호. 이 값은 다음의 2 가지 값 중의 하나이어야 합니다.

Value	Meaning
0 (cmxCNT_COMM)	Command Counter
1 (cmxCNT_FEED)	Feedback Counter

▶ ***Speed*** : 전달된 변수를 통해 지정한 카운터의 속도를 읽어서 논리적 속도 단위로 반환합니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

REFERENCE

□ 논리적 속도 단위는 cmxPmCfgSetUnitSpeed() 함수에 의해 결정됩니다.

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

long BoardID = 0;
long nCmdSpeed = 0;
long nFdbSpeed = 0;

//0 번 축의 Command Speed 값을 반환합니다.
cmxPmStGetSpeed(BoardID, 3, ccmxX1, cmxCNT_COMM, &nCmdSpeed );

//0 번 축의 Feedback Speed 값을 반환합니다.
cmxPmStGetSpeed(BoardID, 3, ccmxX1, cmxCNT_FEED, &nFdbSpeed );
```

NAME

cmxPmStReadMotionState

- 모션 동작 상태 반환

INFORMATION

Motion Status
VC++ (6, 7, 8)/VB
BCB/Delphi
Level 7
위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxPmStReadMotionState ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 MotStates)

DESCRIPTION

cmxPmStReadMotionState() 함수는 현재 모션의 동작 상태를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 0 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***MotStates*** : 모션 상태.

Value	Meaning
음수	모션작업 도중 에러 발생 => 에러코드 참조.
0 (cmxMST_STOP)	Stop
1 (cmxMST_WAIT_DR)	Waiting for DR input signal
2 (cmxMST_WAIT_STA)	Waiting for STA input signal
3 (cmxMST_WAIT_INSYNC)	Waiting for internal sync. signal
4 (cmxMST_WAIT_OTHER)	Waiting other axis
5 (cmxMST_WAIT_ERC)	Waiting for ERC output finished
6 (cmxMST_WAIT_DIR)	Waiting for DIR change (DIR 은 외부 입력 신호가 아닌 내부적인 신호임)
7 (cmxMST_RESERVED1)	Reserved
8 (cmxMST_WAIT_PLSR)	Waiting for PA/PB input signal
9 (cmxMST_IN_RVSSPD)	In home special speed (reverse speed)
10(cmxMST_IN_INISPD)	In start velocity motion
11(cmxMST_IN_ACC)	In acceleration
12 (cmxMST_IN_WORKSPD)	In working velocity

13 (cmxMST_IN_DEC)	In deceleration
14 (cmxMST_WAIT_INP)	Waiting for INP input signal
15 (cmxMST_SPARE0)	Reserved
16 (cmxMST_HOMMING)	In Homming

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

SEE ALSO

cmxPmStGetMstString

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

long BoardID = 0;

char szMstList[18][20] = {
    "Stop",
    "Wait DR",
    "Wait STA",
    "Wait INSYNC",
    "Wait Other Axis",
    "Wait ERC",
    "Wait DIR",
    "Reserved1",
    "Wait PA/PB",
    "On Reverse Speed",
    "On Initial Speed",
    "On Acceleration",
    "On Work Speed",
    "On Deceleration",
    "Wait INP",
    "Reserved",
    “In Homming”,
    “”
};

long nMST = 0;

//0 번 축의 모션 상태를 반환합니다.
cmxPmStReadMotionState(BoardID, 3, ccmxX1, &nMST );

if( nMST < 0 )
{
    OutputDebugString( "ReadMotionState Error!" );
}

//화면에 표시할 문자열을 조합합니다.
CString sMsg = "Current Motion State : " + szMstList[nMST];

//DisplayStatus() 함수는 화면에 상태를 표시하는 가상의 함수입니다.
DisplayStatus( sMsg );
```

NAME

cmxPmStReadMIOStatuses

- 모션 관련 I/O (Motion I/O) 상태 반환

INFORMATION

Motion Status

VC++ (6, 7, 8)/VB

BCB/Delphi

Level 7

위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxPmStReadMioStatuses ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 MioStates)

DESCRIPTION

이 함수는 현재 모션과 관련된 여러 가지 MIO 상태를 반환합니다. 각 비트별로 할당된 MIO 의 상태를 표시하므로 사용자는 비트 마스크를 수행하여 원하는 I/O 의 상태를 확인하여야 합니다.

범용적인 모션 응용 프로그램에서는 MIO(Machine I/O) 상태를 표현하기 위한 용도로 본 함수의 사용 빈도가 매우 높습니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***MioStates*** : Machine I/O 상태

Bit No.	Name	Meaning
0 (cmxIOST_RDY)	RDY	Servo ready signal input status(1=ON)
1 (cmxIOST_ALM)	ALM	Alarm signal status(1=ON)
2 (cmxIOST_ELN)	-EL	Negative limit switch status(1=ON)
3 (cmxIOST_ELP)	+EL	Positive limit switch status(1=ON)
4 (cmxIOST_ORG)	ORG	Origin switch status(1=ON)
5 (cmxIOST_DIR)	DIR	Operating direction status(1=ON)
6 (cmxIOST_EZ)	EZ	Index signal status(1=ON)
7 (cmxIOST_LTC)	LTC	Latch signal input status(1=ON)
8 (cmxIOST_SD)	SD	Slow Down signal input status(1=ON)
9 (cmxIOST_INP)	INP	In-Position signal input status(1=ON)
10 (cmxIOST_DRN)	DRN	-DR input signal status(1=ON)
11 (cmxIOST_DRP)	DRP	+DR input signal status(1=ON)
12 (cmxIOST_STA)	STA	STA input signal status(1=ON)
13 (cmxIOST_STP)	STP	STP input signal status(1=ON)

14 (cmxIOST_ALMR)	ALMR	Alarm Reset output signal status(1=ON)
15 (cmxIOST_EMG)	EMG	Emergency output signal status(1=ON)
16 (cmxIOST_SVON)	SVON	Servo-ON output signal status(1=ON)
17 ~31		Reserved

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

EXAMPLE

C/C++

```
#include "ComiRTEX_SDK.h"
#include "ComiRTEX_SDK_Def.h"

long BoardID = 0;
long nMioState = 0;

//0 번 축의 MioState Bit 가 설정된 32Bit 값을 반환합니다.
cmxPmStReadMioStatuses(BoardID, 3, ccmxX1, &nMioState );

//nMioState 의 값을 오른쪽으로 쉬프트 연산하여, 해당 상태 값을 얻습니다.
bool ALM_State = ( nMioState >> ccmxIOST_ALM ) & 0x01;
bool ELP_State = ( nMioState >> ccmxIOST_ELP ) & 0x01;
bool ELN_State = ( nMioState >> ccmxIOST_ELN ) & 0x01;
bool ORG_State = ( nMioState >> ccmxIOST_ORG ) & 0x01;
```

17.2 위치 값 래치(Position Latch)

Position Latch 는 특정 순간에 Motion 의 위치 관련 카운터 값을 래치(Latch)하여 읽을 수 있도록 하는 기능입니다.

Position Latch 는 LTC 입력 핀에 LATCH 신호가 입력되면 그 순간의 각 카운트 값을 래치(Latch) 합니다.

이 기능은 LTC 핀에 입력되는 하드웨어 신호에 동기되어서 각 위치값이 래치 되므로 시간 지연이 발생하지 않습니다.

17.2.1 함수 요약

Position Latch 함수에 관련된 함수들은 다음과 같습니다.

Summary of Functions
❒ VT_I4 cmxPmLtcIsLatched ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 IsLatched) 대상 모션 축의 래치 카운터(Latch Counter) 가 활성화 되었음을 확인하고, 결과를 반환합니다.
❒ VT_I4 cmxPmLtcReadLatch ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 Counter, [out] VT_PR8 LatchedPos) 대상 모션 축의 래치 카운터(Latch Counter) 에 저장된 카운터(Counter) 값을 반환합니다.

NAME

cmxPmLtcIsLatched

- 해당 축의 래치 카운터 (Latch Counter) 활성화 여부 확인

INFORMATION

- Latch
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 7
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxPmLtcIsLatched ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [out] VT_PI4 IsLatched)

DESCRIPTION

해당 축의 래치 카운터가 활성화 되었음을 확인하고, 결과를 반환합니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***IsLatched*** : 해당 축의 래치 카운터의 활성화 여부를 반환합니다.

Value	Meaning
0 (cmxFALSE)	래치 카운터 비활성
1 (cmxTRUE)	래치 카운터 활성

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

SEE ALSO

cmxPmLtcReadLatch

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

long BoardID = 0;

// 0 번 축의 래치 카운터 활성화 여부를 반환합니다.
long nIsLatched = cmxFALSE;
cmxPmLtcIsLatched(BoardID, 3, ccmxX1, &nIsLatched );
```

NAME

cmxPmLtcReadLatch

- 해당 축의 래치 카운터 (Latch Counter) 값 확인

INFORMATION

- Latch
- VC++ (6, 7, 8)/VB
- BCB/Delphi
- Level 7
- 위험 요소 없음

SYNOPSIS

❒ VT_I4 cmxPmLtcReadLatch ([in] VT_I4 BoardID, [in] VT_I4 NodeId, [in] VT_I4 Channel, [in] VT_I4 Counter, [out] VT_PR8 LatchedPos)

DESCRIPTION

지정한 축의 현재 래치 된 카운트 값을 반환합니다. 이때 반환되는 위치 값의 단위는 논리적 거리 단위가 적용됩니다.

이 함수의 사용과 호출에 있어, 제공된 ㈜커미조아의 함수 헤더 Visual Basic 에서는 함수의 첨두어 cmx 가 붙지 않습니다.

PARAMETER

▶ ***BoardID*** : 사용자가 설정한 디바이스(보드) ID.

▶ ***NodeId*** : 노드 번호. 노드 번호는 3 부터 시작합니다.

▶ ***Channel*** : 축 번호. 통합 축으로 관리되는 축 번호를 의미하며, 상수 값으로 0 (Zero Based) 이상, 최대 통합 축 개수 - 1 이하의 값을 축 번호로 설정할 수 있습니다.

▶ ***Counter*** : 읽을 래치 카운터를 지정합니다. 이 값은 다음과 같습니다.

Value	Meaning
0 (cmxCNT_COMM)	명령 위치 카운터(Command position counter)
1 (cmxCNT_FEED)	실제 위치 카운터(Feedback position counter)
2 (ccmxCNT_DEV)	Deviation 또는 펄스 출력 속도
3 (ccmxCNT_GEN)	General Counter

▶ ***LatchedPos*** : 지정한 축의 래치 된 카운트 값을 반환합니다. 이때 반환되는 위치 값의 단위는 논리적 거리 단위가 적용됩니다.

RETURN VALUE

Value	Meaning
음수	수행 실패. 자세한 내용은 '에러처리' 편을 참고합니다
0 (ERR_NONE)	수행 성공

SEE ALSO

cmxPmLtcIsLatched

EXAMPLE

C/C++

```
#include “ComiRTEX_SDK.h”
#include “ComiRTEX_SDK_Def.h”

long BoardID = 0;
// 0 번 축의 래치 카운트를 Feedback position counter 로 설정하고 래치된 카운트 값을 반환합니다.
long nLtcCounter = cmxCNT_FEED;
double fLatchedPos = 0.0f;
cmxPmLtcReadLatche(BoardID, 3, ccmxX1, nLtcCounter, &nLatchedPos );
```

Chapter

18

18 Motion Default Parameter

어떤 응용프로그램이나 장비에도 모든 설정에는 초기값을 보유하고 있습니다. 여기서는 보다 자세하고 정확한, 그리고 적절한 모션 환경설정을 위해 고객(顧客) 여러분들께 제공하는 모션 초기화 매개변수(媒介變數)를 소개합니다. 모션 환경 설정 이전에도 세심한 배려와 더불어 고객(顧客)님께 안정적인 환경설정을 안내해드리기 위해 준비하였습니다.

모션 시스템 초기화시에 설정되는 장치 초기값(Default) 에 대해서 안내합니다. 여기서 말하는 장치 초기화는 ComiRTEX 의 함수를 통해 초기화된 상태 혹은 그 값(Value)을 의미하며, 고객(顧客) 여러분들께서 해당 부분에 대해서 설정하지 않으셨거나, 초기화 설정값을 알기 위한 용도로 다양하게 사용될 수 있습니다.

I 모션장치초기화시의 초기(Default) 값

I.I Command & Feedback

항목구분		기본(초기) 값		관련 함수
		값	의미	
Command	Unit distance	1	거리의 단위를 펄스로 사용	cmxCfgSetUnitDist
	Unit speed	1	속도의 단위를 PPS 로 사용	cmxCfgSetUnitSpeed

I.II INP , EL, ORG

항목구분		기본(초기) 값		관련 함수
		값	의미	
Inposition (INP)	Enable INP	0	INP 신호를 모션에 적용하지 않음	cmxCfgSetMioProperty (cmxMIO_INP_EN)
External Limit (-EL, +EL)	Input Logic (+EL)	0	Normal Open(A 접점)	cmxCfgSetMioProperty (cmxMIO_PEL_LOGIC)
	Input Logic (-EL)	0	Normal Open(A 접점)	cmxCfgSetMioProperty (cmxMIO_NEL_LOGIC)
	Stop Mode	0	즉시 정지(停止)	cmxCfgSetMioProperty (cmxMIO_EL_MODE)
Origin (ORG)	Input Logic (ORG)	0	Normal Open(A 접점)	cmxCfgSetMioProperty (cmxMIO_ORG_LOGIC)

I.III Software Limit

항목구분		기본(초기) 값		관련 함수
		값	의미	
Software Limit	Enable	0	소프트웨어 Limit 해제	cmxCfgSetSoftLimit
	N-Limit value	0	음의 방향의 최대값	
	P-Limit value	0	양의 방향의 최대값	

I.IV 원점복귀 환경

항목구분	기본(초기) 값		관련 함수
	값	의미	
Home mode	1	원점복귀 모드 1 번	cmxHomeSetConfig
Home Direction	0	음의 방향	
Offset Distance	0	원점 복귀 완료 시 추가 이송 거리	cmxHomeSetOffset
Speed Mode	2	S-CURVE 모드	cmxHomeSetSpeedPattern
Work Speed	400000	1 단계 원점복귀 정속도	
Acceleration	400000	1 단계 원점복귀 가속도	
Deceleration	400000	1 단계 원점복귀 감속도	
Speed Mode	2	S-CURVE 모드	
Work Speed	20000	2 단계 원점복귀 정속도	
Acceleration	20000	2 단계 원점복귀 가속도	
Deceleration	20000	2 단계 원점복귀 감속도	
Speed Mode	2	S-CURVE 모드	
Work Speed	2000	3 단계 원점복귀 정속도	
Acceleration	2000	3 단계 원점복귀 가속도	
Deceleration	2000	3 단계 원점복귀 감속도	

I.V 모션 정격 속도 환경

항목구분	기본(초기) 값		관련 함수
	값	의미	
Speed Mode	2	S-CURVE 모드	cmxCfgSetSpeedPattern
Work Speed	400000	정속도	
Acceleration	400000	가속도	
Deceleration	400000	감속도	
Initial Speed	0	초기 속도	
End Speed	0	종료 속도	

I.VI PM Module – Command & Feedback

항목구분		기본(초기) 값		관련 함수
		값	의미	
Command	Output mode	4	CW & CCW 1	cmxPmCfgSetOutMode
	Unit distance	1	거리의 단위를 펄스로 사용	cmxPmCfgSetUnitDist
	Unit speed	1	속도의 단위를 PPS 로 사용	cmxPmCfgSetUnitSpeed
Feedback	Input Mode	0	1x A/B Phase	cmxPmCfgSetInMode
	Inverse direction	0	방향을 바꾸지 않음	cmxPmCfgGetInMode
In/Out Ratio		1	FeedBack / Command Ratio	cmxPmCfgSetInOutRatio

I.VII PM Module - INP, ALM , EL

항목구분		기본(초기) 값		관련 함수
		값	의미	
Inposition (INP)	Enable INP	0	INP 신호를 모션에 적용하지 않음	cmxPmCfgSetMioProperty (cmxPM_INP_EN)
	Input Logic	0	Normal Open(A 접점)	cmxCfgSetMioProperty (cmxPM_INP_LOGIC)
Alarm(ALM)	Input Logic	0	Normal Open(A 접점)	cmxCfgSetMioProperty (cmxPM_ALM_LOGIC)
	Stop Mode	0	즉시 정지(停止)	cmxCfgSetMioProperty (cmxPM_ALM_MODE)
External Limit (-EL, +EL)	Input Logic	0	Normal Open(A 접점)	cmxCfgSetMioProperty (cmxPM_EL_LOGIC)
	Stop Mode	0	즉시 정지(停止)	cmxCfgSetMioProperty (cmxPM_EL_MODE)

I.VIII PM Module - Software Limit

항목구분		기본(초기) 값		관련 함수
		값	의미	
Software Limit	Enable	0	소프트웨어 Limit 해제	cmxPmCfgSetSoftLimit
	N-Limit value	-99999999	음의 방향의 최대값	
	P-Limit value	99999999	양의 방향의 최대값	

I.IX PM Module - Servo ON Input Logic

항목구분	기본(초기) 값		관련 함수
	값	의미	
Servo ON Input Logic	0	Normal open (A 접점)	cmxPmCfgSetMioProperty (cmxPM_SVON_LOGIC)

I.X PM Module - 원점복귀 환경

항목구분	기본(초기) 값		관련 함수
	값	의미	
Home mode	0	원점복귀 모드 0 번	cmxPmHomeSetConfig
ORG Input Logic	0	Normal Open(A 접점)	
Ez Input Logic	0	Normal Open(A 접점)	
Ez Count	0	원점 복귀시에 EZ 상의 계수 값	
Escape Distance	10	원점 탈출 거리	
Offset Distance	0	원점 복귀 완료 시 추가 이송 거리	
ERC out	0	원점복귀 완료 후에 ERC 출력하지 않음	cmxPmCfgSetMioProperty (cmxPM_ERC_OUT)
Speed Mode	2	S-CURVE 모드	cmxPmHomeSetSpeedPattern
Work Speed	5000	원점복귀 정속도	
Acceleration	100000	원점복귀 가속도	
Deceleration	100000	원점복귀 감속도	
Reverse Speed	1000	원점복귀 역방향 이송 속도	

I.XI PM Module – 모션 정격 속도 환경

항목구분	기본(초기) 값		관련 함수
	값	의미	
Max Speed	655350	655350 (PPS)	cmxPmCfgSetSpeedRange
Initial Speed	0	초기 속도	cmxPmSxOptSetIniSpeed
Speed Mode	2	S-CURVE 모드	cmxPmCfgSetSpeedPattern
Work Speed	10000	정속도	
Acceleration	100000	가속도	
Deceleration	100000	감속도	

Appendix

B

Frequently Asked Questions

주로 자주 발생하는 질문이나, 고객(顧客) 기술 지원에 대한 내용을은 고객(顧客)님의 제품 개발 및 모션 라이브러리 운용시에 많은 도움이 될 수 있습니다. 다양한 개발 환경에서 발생할 수 있는 여러가지 문제들을 세심하고 주의깊게 다루어, 고객(顧客)님들께 진정으로 도움이 될 수 있는 내용을 준비하였습니다.

본 부록에서는 ㈜커미조아의 ComiRTEX 사용에 있어, 문의 사항이나 궁금하신 점을 FAQ 로 안내하였습니다. 각 FAQ 는 ㈜커미조아의 웹 사이트(http://www.comizoa.com) 이나 별도로 등록된 고객(顧客)님의 정보를 통해 전달 될 수 있도록 노력하겠사오며, 본 FAQ 장을 고객(顧客) 문의를 하시기전에 미리 확인(確認)하시어, 좋은 참고가 되시기를 간절히 바라겠습니다.

II Frequently Asked Questions (FAQ)

II.I Visual Studio 2005

Q Microsoft ® Visual Studio 2005 에서 MFC WIZARD 선택 단계에서 [Use Unicode Libraries] 항목이 확인(確認)된 상태에서 프로젝트를 시작한 다음, 빌드 하였을 경우 아래와 같은 에러가 발생하게 됩니다. 에러 발생의 이유는 ComiRTEX 는 표준 ANSI Libraries API 함수를 사용하는데, 현재 프로젝트의 설정이 Unicode 로 되어 있어 다음과 같은 에러가 발생할 수 있습니다.

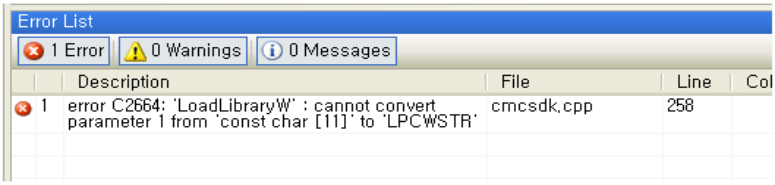


그림 18-1 Unicode 로 되어 있을 경우 발생하는 에러 화면

A 해당 문제를 해결 하는 방법은 다음과 같습니다. MS VC++ 의 Solution Explorer 에서 현재 작업중인 프로젝트를 선택합니다.

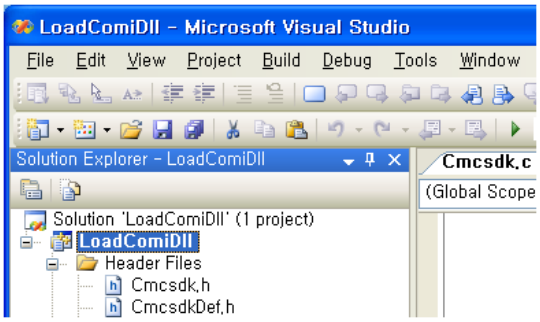


그림 18-2 사용자 생성 프로젝트 선택 화면

메뉴에서 [Project]->[Properties]를 선택하여 [Property Page]창을 엽니다.

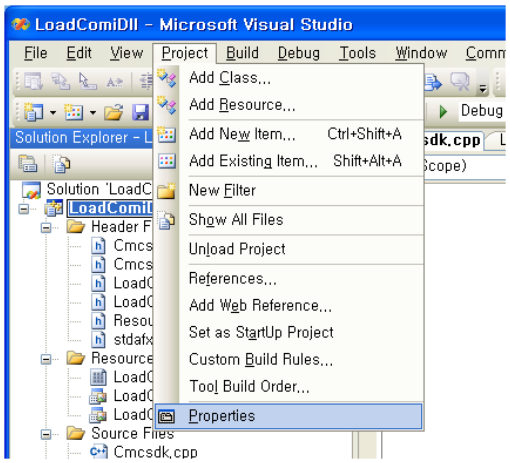


그림 18-3 사용자 생성 프로젝트의 등록정보 선택 화면

[Configuration Properties]아래의 [General]항목을 선택하면 오른쪽에 [Project Defaults]라는 항목이 표시 됩니다. [Project Defaults]항목의 [Character Set]이라는 항목을 [Use Unicode Character set]에서 [Not Set]으로 변경 하여 줍니다.

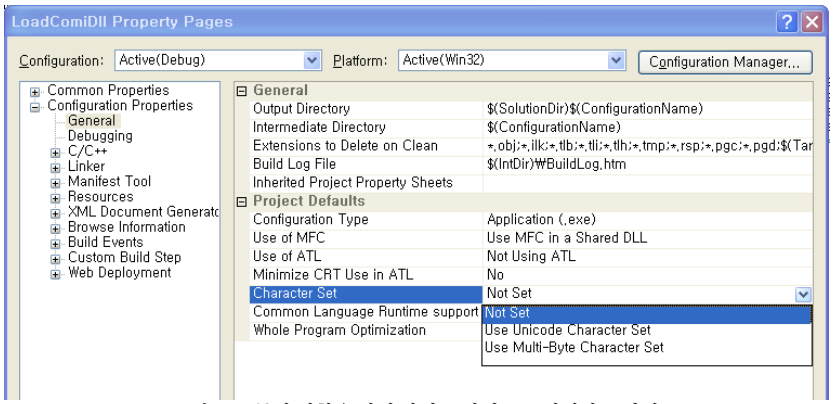


그림 18-4 문자 집합을 단일 바이트 타입으로 변경하는 화면

참고: Release Mode 로 컴파일 할 경우에는 [Property Pages]창의 [Configuration]항목을 [Release]로 변경한 후 [Character Set]의 옵션을 [Not Set]으로 반드시 변경해야 합니다.

II.II Visual Basic

Q Visual Basic 6.0 에서 사용하던 COMIZOA SDK Library 를 Visual Studio .NET 에서 사용하면 오류가 발생한다.

A Microsoft 사가 인정한 Visual Studio 오류로서 디자인 타임 라이센스가 존재하지 않을 경우에 발생합니다. Visual Basic 6.0 으로 작성한 프로젝트를 Visual Studio.NET 으로 업그레이드하여 사용하고자 할 때 발생할 수 있는 오류 사항입니다. COMIZOA SDK Library 에는 라이센스가 없으므로 문제가 발생한다면 프로젝트에 포함되어있는 다른 ActiveX Control 로 인하여 문제가 발생할 수 있는 여지가 있습니다.

http://support.microsoft.com/default.aspx?scid=kb;ko;318597

마이크로 소프트에서 제공하는 위의 주소를 통해 자세한 내용 및 해결방안을 찾아보실 수 있습니다.

II.III Borland C++ Builder

Q C++ Builder 5 버전과 6 버젼에서 COMIZOA OLE Components (ComiSliderCtrl.ocx)가 Import 되어 있는데 보이지 않는다.

A Visual Basic 으로 작성한 OCX 를 C++ Builder 에서 사용할 때 발생하는 문제입니다. ..\SDK\COMIZOA Components\Borland Package\C++ Builder\ 안에 각 버전별로 Component 관련 파일이 들어 있습니다. 이 파일들을 각각 위치에 맞게 복사해 주시면 됩니다.

새로운 환경에서 ComiSliderCtrl.ocx 사용하여 프로그램을 작성하시려면 다음과 같은 방법으로 등록하여 주시면 됩니다.

* Builder 5 에서 ComiSliderCtrl.ocx 사용방법.

1. 윈도우 시스템폴더에 OCX 를 복사한 후 regsvr32 명령으로 OCX 를 등록시킵니다.
   (시작->실행 메뉴에서 : regsvr32 c:\windows\systemx32\ComiSliderCtrl.ocx)

2. ComiSliderCtrl.ocx 를 [..\CBuilder5\Bin\] 폴더로 복사합니다.
   tlibimp.exe 프로그램을 이용하여 Import Type Library File 을 생성합니다.

   ```
   C:\Program Files\Borland\CBuilder5\BIN> tlibimp.exe -Yu -Ya ComiSliderCtrl.ocx
   ComiSliderCtrl_TLB.h
   ComiSliderCtrl_TLB.cpp
   ComiSliderCtrl_OCX.h
   ComiSliderCtrl_OCX.cpp
   ComiSliderCtrl_OCX.dcr
   ```

   총 5 가지 파일이 생성됩니다.

3. 계속하여, 시작-> 실행 메뉴을 통해 'cmd' 를 입력하여, 명령프롬프트를 실행합니다.

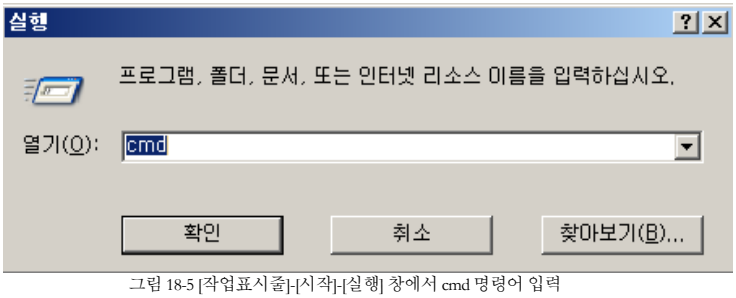


그림 18-5 [작업표시줄]-[시작]-[실행] 창에서 cmd 명령어 입력

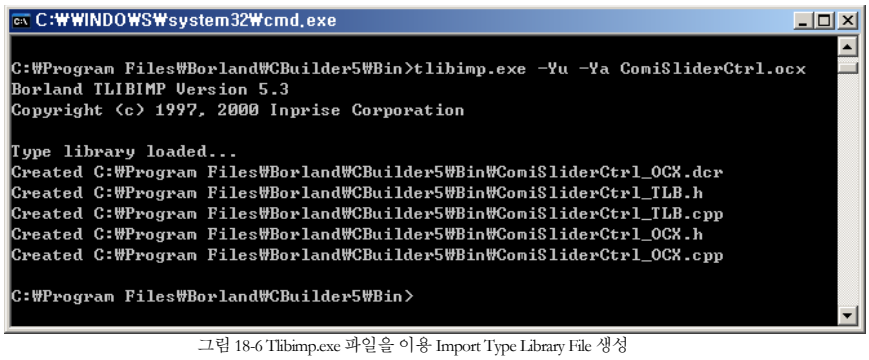


그림 18-6 Tlibimp.exe 파일을 이용 Import Type Library File 생성

4. Builder 5 에서 OCX 를 등록합니다.

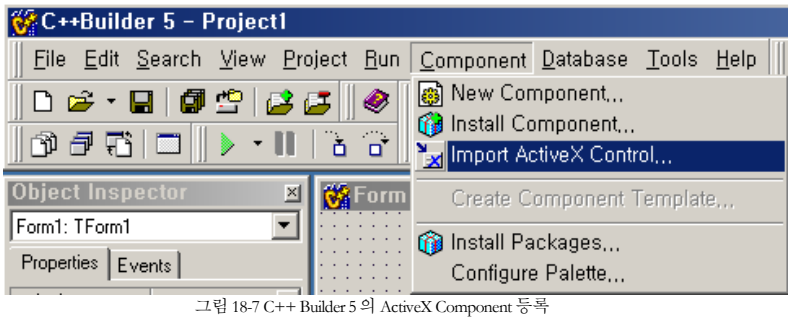


그림 18-7 C++ Builder 5 의 ActiveX Component 등록

5. Import ActiveX 윈도우가 화면에 나타나면, ComiSliderCtrl 을 선택하고 원하는 Palette Page 를 선택한뒤 Install 버튼을 눌려 설치를 시작합니다.

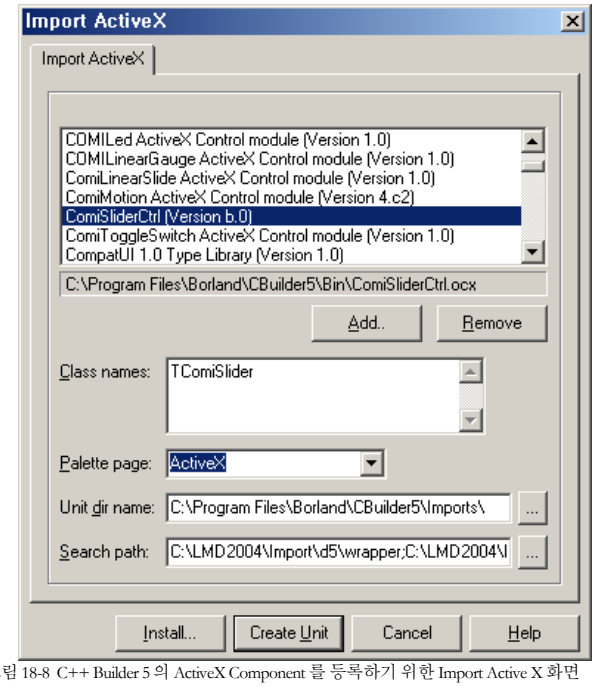


그림 18-8 C++ Builder 5 의 ActiveX Component 를 등록하기 위한 Import Active X 화면

6. Install 창이 화면에 나타나면 Into new package 탭에서 생성을 원하는 Package 파일 이름 및 설명을 입력하여 새로운 bpk 를 생성합니다.

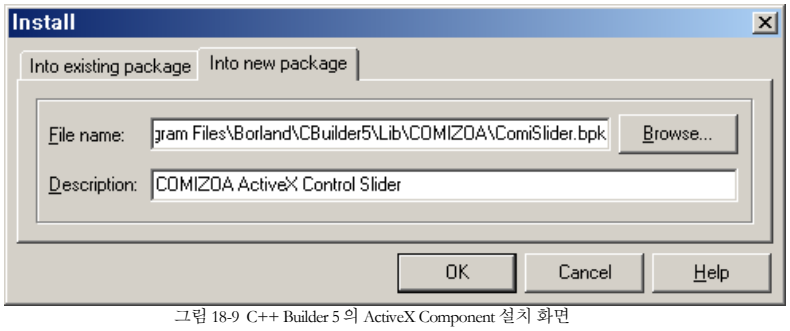


그림 18-9 C++ Builder 5 의 ActiveX Component 설치 화면

7. Package 를 Install 한다는 확인(確認) 창이 화면에 표시되면, "No" 를 선택합니다.

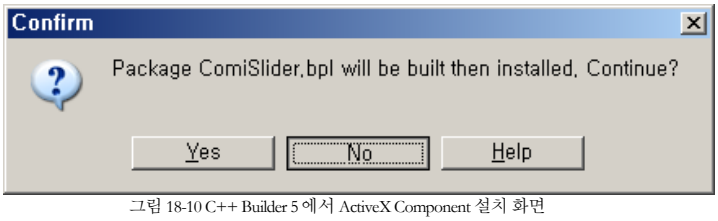


그림 18-10 C++ Builder 5 에서 ActiveX Component 설치 화면

8. Tlibimp.exe 파일을 이용해 생성한 5 개의 파일을 ..\Borland\CBuilder5\Imports 에 덮어쓰기(Overwrite) 합니다

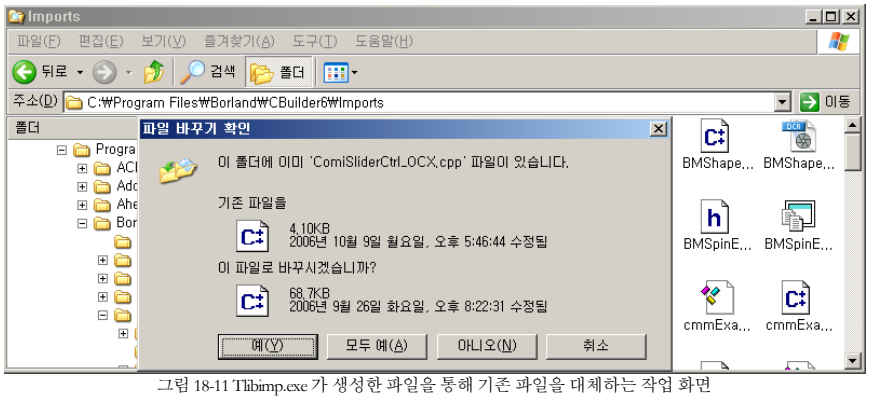


그림 18-11 Tlibimp.exe 가 생성한 파일을 통해 기존 파일을 대체하는 작업 화면

9. 덮어쓰기가 끝나면 생성한 Package 파일을 Install 합니다.

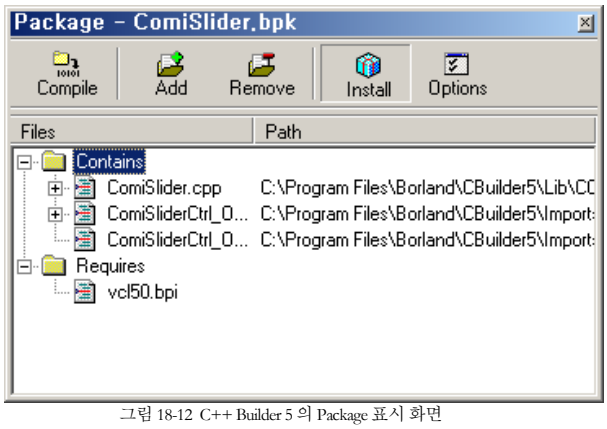


그림 18-12 C++ Builder 5 의 Package 표시 화면

10. Package 가 Install 되었다는 메시지와 함께 툴 팔레트에 OCX 가 정상적으로 등록된 것을 확인(確認)할 수 있습니다.

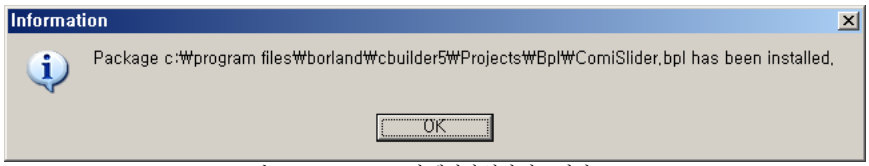


그림 18-13 C++ Builder 5 의 패키지 설치 완료 화면

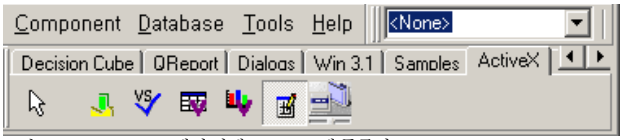


그림 18-14 C++ Builder 5 에서 팔레트(Pallette) 에 등록된 ComiSlider Active X Control

* Builder 6 에서 ComiSliderCtrl.ocx 사용방법.

11. [Tool]-[Environment Options..]를 선택합니다..

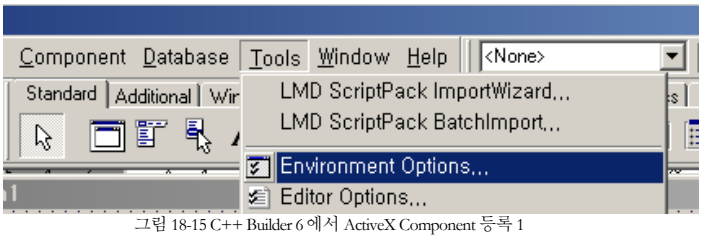


그림 18-15 C++ Builder 6 에서 ActiveX Component 등록 1

12. Enviroment Options 창의 Type Library 탭에서 "Ignore special CoClass Flags when importing", "Can Create" - Check 한뒤 OK 버튼을 클릭합니다.

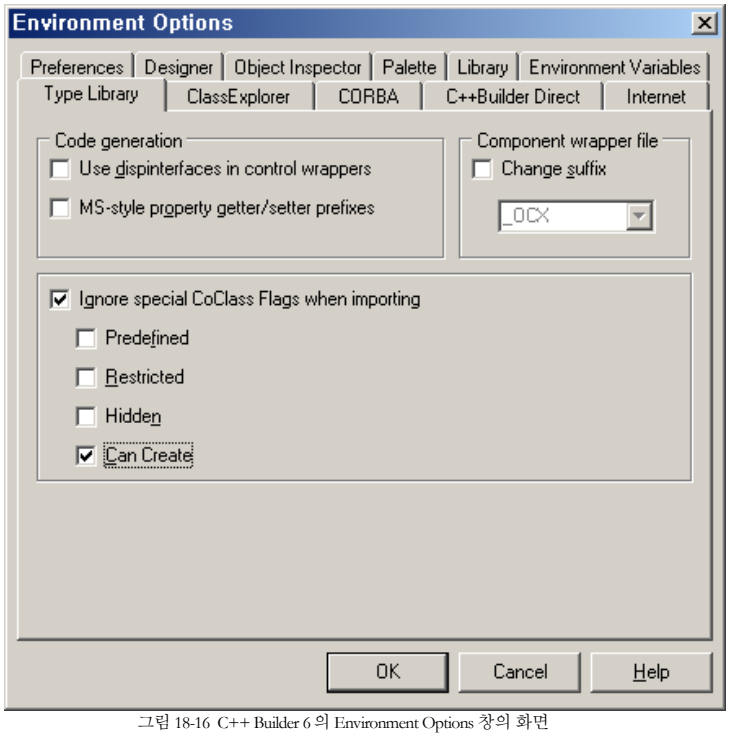


그림 18-16 C++ Builder 6 의 Environment Options 창의 화면

13. [Component]-[Import ActiveX Control..]를 선택한뒤 절차에 따라 OCX 를 등록하면 됩니다.

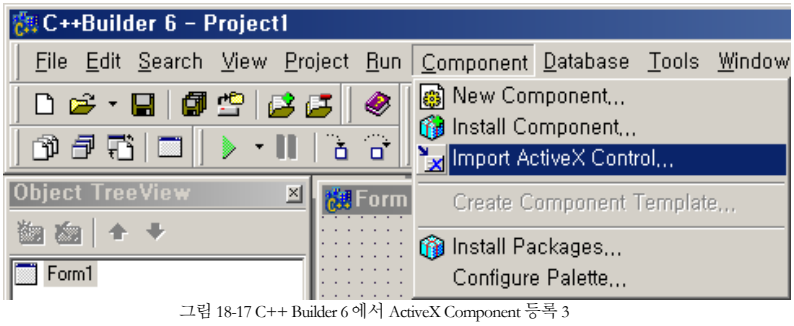


그림 18-17 C++ Builder 6 에서 ActiveX Component 등록 3

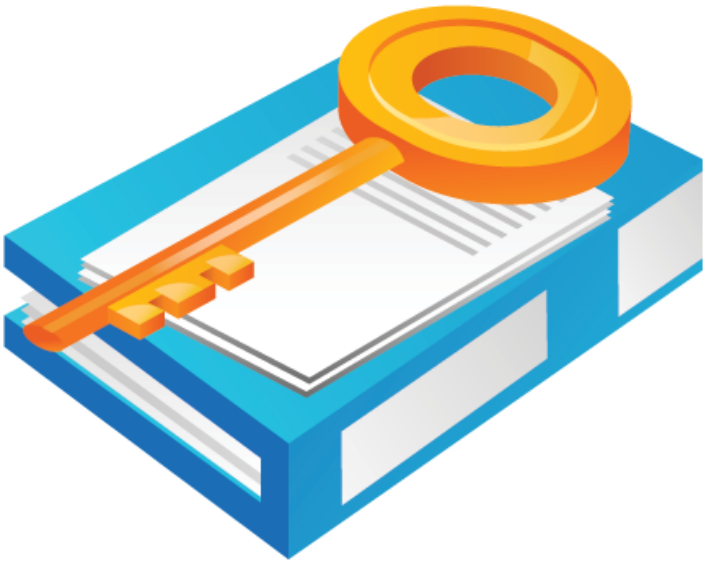
Chapter

C

Index of ComiRTEX functions

필요한 함수(函數)를 가장 빠르고 쉽게 찾으십시요. ComiRTEX 매뉴얼에서는 고객(顧客)님들께서 원하시는 함수들을 일목 요연하게 정리하였습니다. 필요한 함수는 온라인 문서에서 하이퍼 링크 기능으로 찾을 수 있도록 구성하였습니다.

하이퍼 링크 기능(機能)을 통해 본장에서는 빠르고 정확하게 고객(顧客)님들께서 원하시는 함수(函數)를 찾으실 수 있도록 구성(構成)하였습니다. Adobe 社의 Acrobat Reader 와 같은 전자 문서 뷰어(Viewer) 를 통해 최단 시간내에 원하시는 함수(函數)를 찾을 수 있습니다.



III Index of ComiRTEX Functions

III.I Quick Reference to ComiRTEX Functions

cmxAdvMemoryRead _____ *오류! 책갈피가 정의되어 있지 않습니다.*

cmxAdvMemoryWrite _____ *오류! 책갈피가 정의되어 있지 않습니다.*

cmxAdvCmxmdCtrl _____ *오류! 책갈피가 정의되어 있지 않습니다.*

cmxAdvCmxmdStat _____ *오류! 책갈피가 정의되어 있지 않습니다.*

cmxAdvSvCmdCtrl _____ *오류! 책갈피가 정의되어 있지 않습니다.*

cmxAdvSvCmdStat _____ *오류! 책갈피가 정의되어 있지 않습니다.*

cmxAdvSetSvCmdOutput _____ *오류! 책갈피가 정의되어 있지 않습니다.*

cmxAdvGetSvCmdInput _____ *오류! 책갈피가 정의되어 있지 않습니다.*

cmxAdvPositionSet _____ *오류! 책갈피가 정의되어 있지 않습니다.*

cmxAdvSetBreak _____ *오류! 책갈피가 정의되어 있지 않습니다.*

cmxAdvSetSensor _____ *오류! 책갈피가 정의되어 있지 않습니다.*

cmxAdvStateMonitor _____ *오류! 책갈피가 정의되어 있지 않습니다.*

cmxAdvSetServo _____ *오류! 책갈피가 정의되어 있지 않습니다.*

cmxAdvInterpolate _____ *오류! 책갈피가 정의되어 있지 않습니다.*

cmxAdvPositioning _____ *오류! 책갈피가 정의되어 있지 않습니다.*

cmxAdvFeed _____ *오류! 책갈피가 정의되어 있지 않습니다.*

cmxAdvExFeed _____ *오류! 책갈피가 정의되어 있지 않습니다.*

cmxAdvExPositioning _____ *오류! 책갈피가 정의되어 있지 않습니다.*

cmxAdvZeroReturn _____ *오류! 책갈피가 정의되어 있지 않습니다.*

cmxAdvVelCtrl _____ *오류! 책갈피가 정의되어 있지 않습니다.*

cmxAdvTrqCtrl _____ *오류! 책갈피가 정의되어 있지 않습니다.*

cmxAdvSvPrmRead _____ *오류! 책갈피가 정의되어 있지 않습니다.*

cmxAdvSvPrmWrite _____ *오류! 책갈피가 정의되어 있지 않습니다.*

cmxAdvGetSelMon _____ *오류! 책갈피가 정의되어 있지 않습니다.*

TEST & MEASURCMXENT & AUTOMATION / COMIZOA

ComiRTEX Manual

2014 년 02 월 21 일 1 판 인쇄
매뉴얼 자료 번호 : 5.0.1

㈜커미조아
http://www.comizoa.com
Tel) 042 – 936 – 6500~6
Fax) 042 – 936 – 6507